2018

Accelerate: State of DevOps

Strategies for a New Economy

Presented by

DORA
DEVOPS RESEARCH & ASSESSMENT

Diamond Sponsor

Google Cloud

Gold Sponsors

Deloitte.
XebiaLabs Enterprise DevOps
pagerduty
redgate
IT REV
cloudbees
Datical
sumologic
TRICENTIS
GitLab
Electric Cloud
Microsoft Azure
Pivotal
aws
ca technologies

CONTENTS

EXECUTIVE SUMMARY

The Accelerate State of DevOps Report is the largest and longest-running research of its kind. It represents five years of work surveying over 30,000 technical professionals worldwide. The results allow us to better understand the practices that lead to higher software delivery performance that can generate powerful business outcomes. This is the only DevOps report that includes cluster analysis to help teams benchmark themselves against the industry as high, medium, or low performers and predictive analysis that identifies specific strategies that teams can use to improve their performance.

We've found that these benefits and results apply equally to all organizations, regardless of their industry vertical.

This year we examine the impact that cloud adoption, use of open source software, organizational practices (including outsourcing), and culture all have on software delivery performance.

Our research continues to examine the role of software delivery metrics—throughput and stability—on organizational performance and finds evidence of tradeoffs for organizations that conventionally optimize for stability.

For the first time, our research expands our model of software delivery performance to include availability. This addition improves our ability to explain and predict organizational outcomes and forms a more comprehensive view of developing, delivering, and operating software. We call this new construct **software delivery and operational performance, or SDO performance**. This new analysis allows us to offer even deeper insight into DevOps transformations.

KEY FINDINGS INCLUDE:

p21

SDO performance unlocks competitive advantages.
Those include increased profitability, productivity, market share, customer satisfaction, and the ability to achieve organization and mission goals.

p37

How you implement cloud infrastructure matters.
The cloud improves software delivery performance and teams that leverage all of cloud computing's essential characteristics are 23 times more likely to be high performers.

p42

Open source software improves performance.
Open source software is 1.75 times more likely to be extensively used by the highest performers, who are also 1.5 times more likely to expand open source usage in the future.

p43

Outsourcing by function is rarely adopted by elite performers and hurts performance.
While outsourcing can save money and provide a flexible labor pool, low-performing teams are almost 4 times as likely to outsource whole functions such as testing or operations than their highest-performing counterparts.

p52

Key technical practices drive high performance.
These include monitoring and observability, continuous testing, database change management, and integrating security earlier in the software development process.

p11

Industry doesn't matter when it comes to achieving high performance for software delivery.
We find high performers in both non-regulated and highly regulated industries alike.

WHO TOOK THE SURVEY?

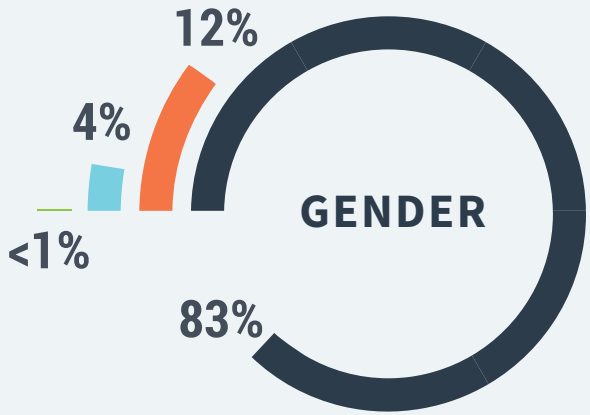
Our research provides the most comprehensive view of the growing DevOps industry, with scientific studies that span five years and more than 30,000 survey responses. Nearly 1,900 professionals worldwide participated in this year's study, which reflects increasing diversity as measured by the proportion of women and underrepresented minorities, although the industry is still far from parity. This year we also report on respondents with disabilities for the first time.

DEMOGRAPHICS & FIRMOGRAPHICS

This year, we see a significant increase in the percentage of female respondents compared to previous years. We believe this increase—which reflects widely reported industry gender distribution[1]—can be explained by our focus on diversifying our survey sample. To get a more complete picture of how many women are working on technical teams, even if we might not have their responses in our study, we asked respondents and heard that 25 percent of respondents' teams are women. Note that this change in demographics does not invalidate prior findings; we were able to confirm and revalidate prior years' results and we call these out throughout the report.

[1] A 2018 Report by HackerRank puts women in tech at 14.2% (https://www.hpcwire.com/2018/03/05/2018-hackerrank-report-shows-progress-challenge-women-coders/), while a NCWIT study reports 25% women in tech (https://www.ncwit.org/sites/default/files/resources/womenintech_facts_fullreport_05132016.pdf)

DEMOGRAPHICS

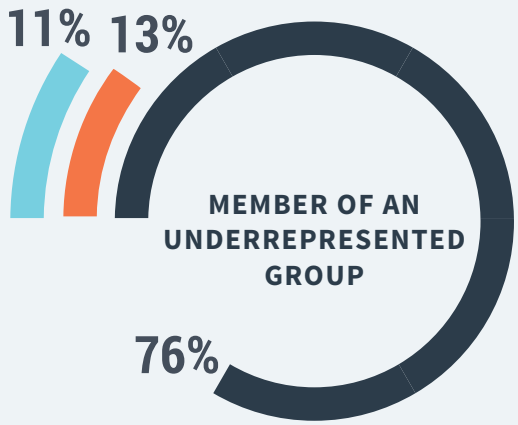


Non-Binary | Did not specify | Female | Male

Respondents stated 25% of their teams are women

Preferred not to respond | Identify | Do not identify

This is the first year we have asked about disability, which is identified along six dimensions that follow guidance from the Washington Group Short Set.[2]



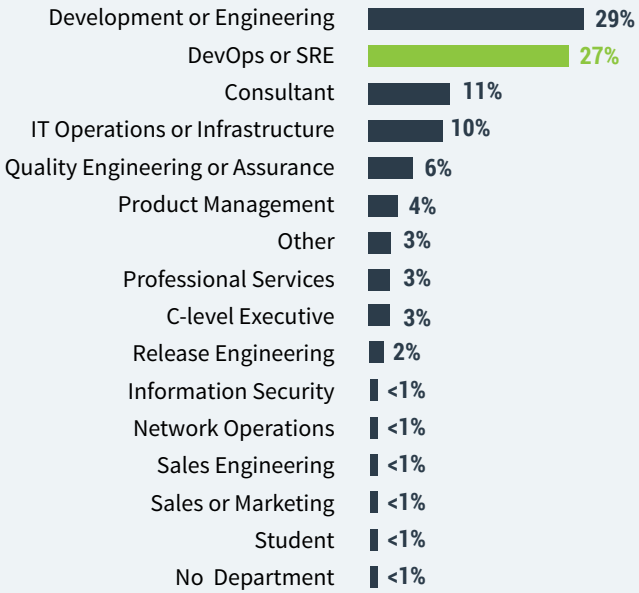
Did not specify | Identify | Do not identify

Identifying as a member of an underrepresented minority can refer to race, gender, or another characteristic. This is the second year we have captured this data and it has slightly increased, up from 12% in 2017 to 13% in 2018.

[2] http://www.washingtongroup-disability.com/wp-content/uploads/2016/01/The-Washington-Group-Short-Set-of-Questions-on-Disability.pdf

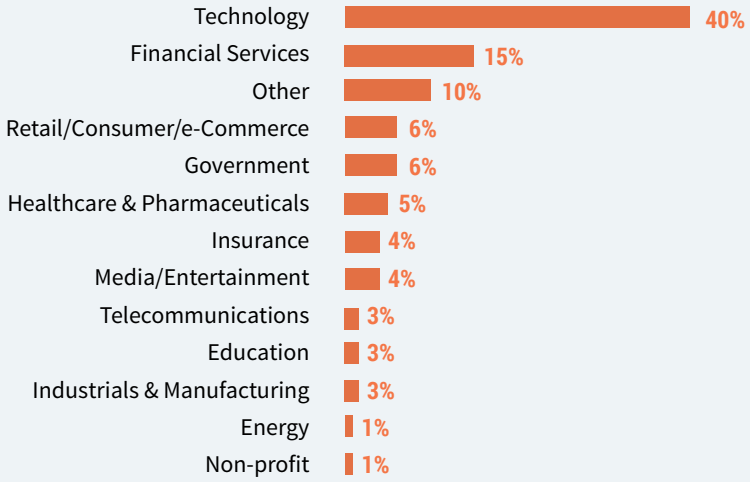
FIRMOGRAPHICS

DEPARTMENT

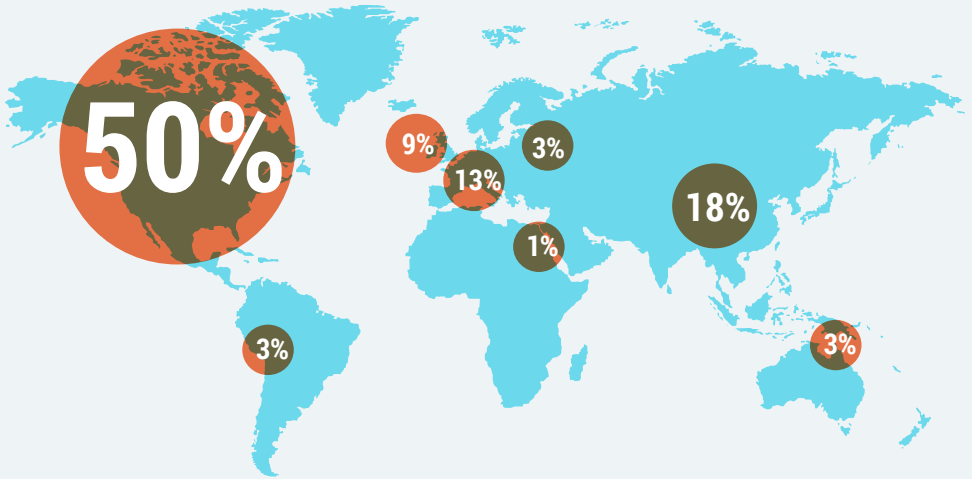


Participants who work in a DevOps team have increased since we began our study, reporting 16% in 2014, 19% in 2015, and 22% in 2016, and holding steady at 27% in 2017 and 2018.

INDUSTRY

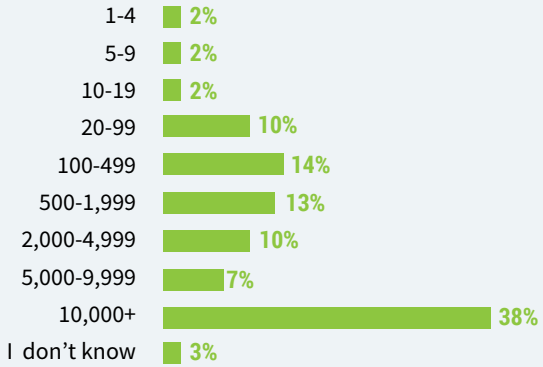


REGION

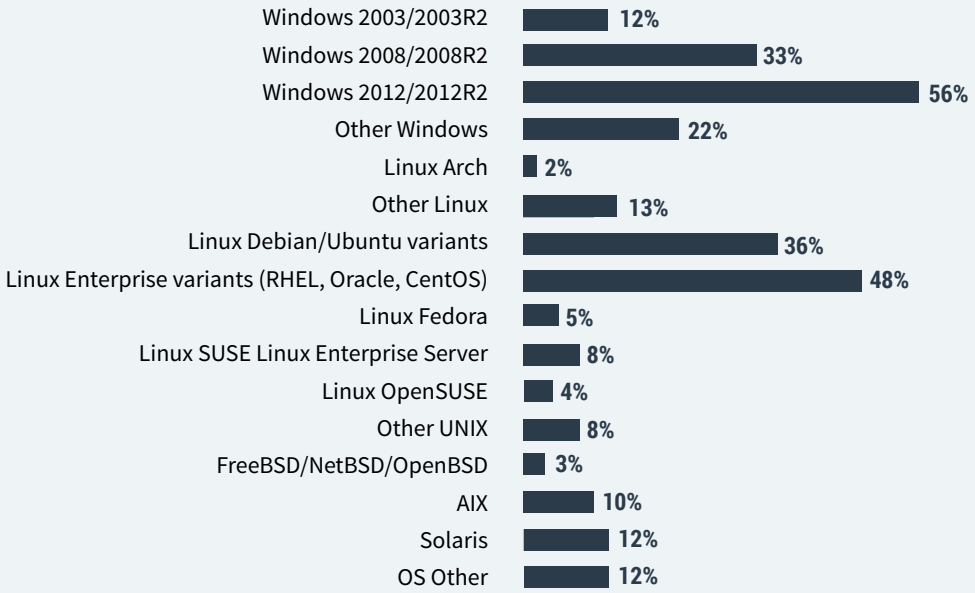


FIRMOGRAPHICS

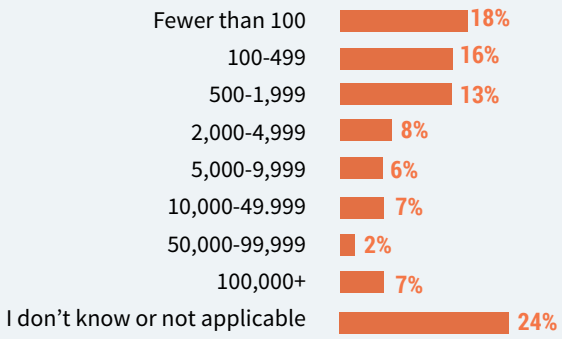
NUMBER OF EMPLOYEES



OPERATING SYSTEMS



NUMBER OF SERVERS



HOW DO WE COMPARE?

Consider this section your DevOps benchmark assessment. We examined teams to understand—in statistically meaningful ways—how they are developing, delivering, and operating software systems. Benchmarks for high, medium, and low performers show where you are in the context of multiple important analyses throughout the report. We also identify trends year over year.

SOFTWARE DELIVERY PERFORMANCE

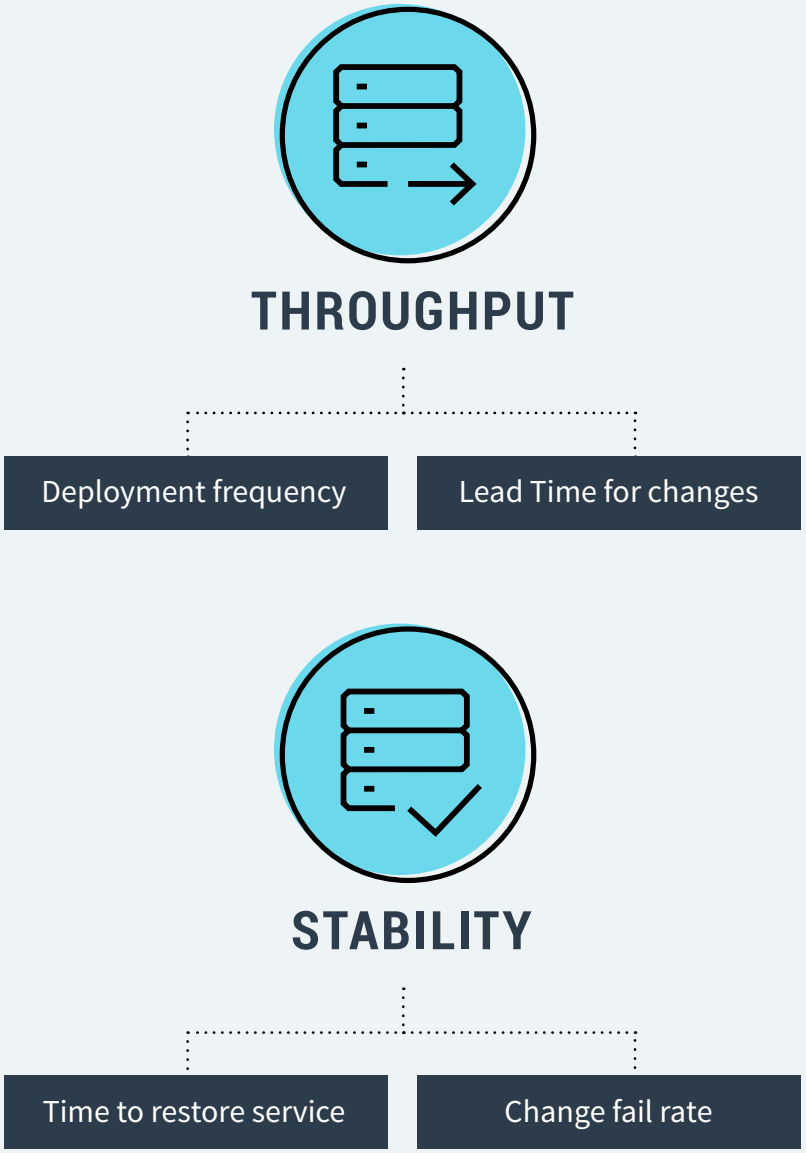
This year's report reflects a fundamental industry change in how software and technology are viewed: What we referred to as IT performance in earlier research is now referred to as software delivery performance to differentiate this work from IT helpdesk and other support functions. The ability to leverage this high-value performance is a key differentiator for organizations. Those that develop and deliver quickly are better able to experiment with ways to increase customer adoption and satisfaction, pivot when necessary, and keep up with compliance and regulatory demands.

Our analysis shows that any team in any industry, whether subject to a high degree of regulatory compliance or not—across all industry verticals—has the ability to achieve a high degree of software delivery performance. We classify teams into high, medium, and low performers and find that they exist in all organization types and industry verticals. You'll find these classifications referenced throughout the report.

To capture software delivery performance, we use measures that focus on global outcomes instead of output to ensure that teams aren't pitted against each other. As in previous years, we use four measures of software delivery performance that capture two aspects of delivery, as shown to the right.

A common industry practice, especially in government or highly regulated fields, is to approach throughput and stability as a trade-off. But our research consistently finds that the highest performers are achieving excellence in both throughput and stability without making tradeoffs. In fact, throughput and stability enable one another. For the fifth year in a row, cluster analysis shows statistically significant differences in both throughput and stability measures among our performance profiles, with the highest performers excelling at all aspects of throughput and stability, and medium and low performers falling behind.

SOFTWARE DELIVERY PERFORMANCE

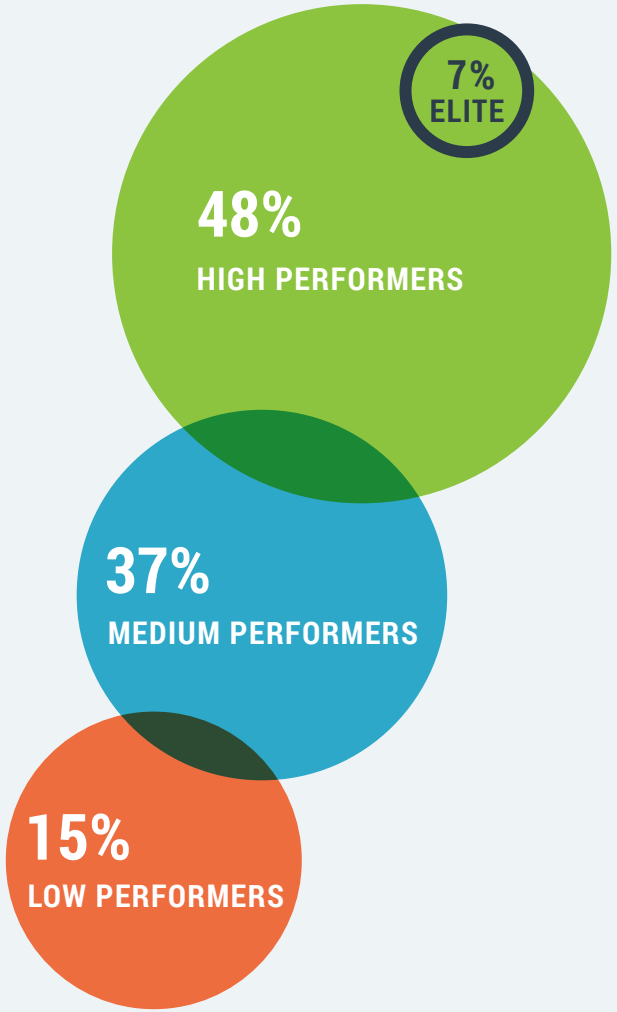


However, this year we find interesting behavior among medium performers that suggests evidence of tradeoffs. Medium performers are doing well in terms of stability (on par with the high performers), but they fall behind when it comes to speed. As expected, and consistent with previous years, low performers again trail all groups in all aspects of performance.

New elite performers raise the bar

This year, the data shows a fourth high-performance group: elite performers. This new category exists for two reasons. The first is that we see the high-performing group growing and expanding, suggesting the overall industry is improving its software development and delivery practices. This trend signals that high performance is attainable for many teams in the industry and is not something reserved for a an exclusive group of teams with unique characteristics. The second is that the elite group demonstrates that the bar for excellence is evolving across the industry, with the highest performers still optimizing for throughput and stability.

PERFORMANCE PROFILES



The proportion of high performers has grown year over year, showing that the industry is continuing to improve. We still see the highest performers (as seen in the "elite performers," a subset of our high-performing group) developing and delivering software at the highest levels, just as we've observed in years past. We also see low performers are struggling to keep up, widening the gap.

Aspect of Software Delivery Performance	Elite[a]	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code?	On-demand (multiple deploys per day)	Between once per hour and once per day	Between once per week and once per month	Between once per week and once per month
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code commit to code successfully running in production)?	Less than one hour	Between one day and one week	Between one week and one month[b]	Between one month and six months[b]
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident occurs (e.g., unplanned outage, service impairment)?	Less than one hour	Less than one day	Less than one day	Between one week and one month
Change failure rate For the primary application or service you work on, what percentage of changes results either in degraded service or subsequently requires remediation (e.g., leads to service impairment, service outage, requires a hotfix, rollback, fix forward, patch)?	0-15%	0-15%	0-15%	46-60%

Medians reported because distributions are not normal.
All differences are significantly different based on Tukey's post hoc analysis except where otherwise noted.
[a] The elite performance group is a subset of the high performance group.
[b] Means are not significantly different based on Tukey's post hoc analysis; medians exhibit differences because of underlying distribution. Typical low performers have a lead time for changes between one month and six months, and typical medium performers have a lead time for changes between one week and one month; however, tests for significant differences show that overall, these two groups are not statistically different when including all group members' variance in behavior.

COMPARING THE ELITE GROUP AGAINST THE LOW PERFORMERS, WE FIND THAT ELITE PERFORMERS HAVE...

46 TIMES MORE
frequent code deployments

2,555 TIMES FASTER
lead time from commit to deploy

7 TIMES LOWER
change failure rate
(changes are 1/7 as likely to fail)

2,604 TIMES FASTER
time to recover from incidents

MISGUIDED PERFORMERS SUFFER FROM CAUTIOUS APPROACH

We often hear from organizations that prefer to take a cautious approach to software development and delivery. They assure us—and their stakeholders—that releasing code infrequently can be an effective strategy as they use the extra time between deployments for testing and quality checks to minimize the likelihood of failure. For the most part, they achieve that goal.

Our cluster analysis this year shows a performance profile that matches this misguided approach: relatively low speed (deployment frequency and lead time for changes) and a better change fail rate than low performers. However, this cluster also reports the longest time to restore service.

Developing software in increasingly complex systems is difficult and failure is inevitable. Making large-batch and infrequent changes

MISGUIDED PERFORMERS

Deployment frequency	Between once per month and once every six months
Lead time for changes	Between one month and six months
Time to restore service	Between one month and six months
Change failure rate	16-30%

introduces risk to the deployment process. When failures occur, it can be difficult to understand what caused the problem and then restore service. Worse, deployments can cause cascading failures throughout the system. Those failures take a remarkably long time to fully recover from. While many organizations insist this common failure scenario won't happen to them, when we look at the data, we see five percent of teams doing exactly this—and suffering the consequences.

At first glance, taking one to six months to recover from a system failure seems preposterous. But consider scenarios where a system outage causes cascading failures and data corruption, or when multiple unknown systems are compromised by intruders. Several months suddenly seems like a plausible timeline for full recovery.

Customers may be able to use the system within hours or days, but engineers and operations professionals are likely investigating all the contributing factors for the incident, checking for and remediating data loss or inconsistencies, and restoring the system to a fully operational state. (The crash of healthcare.gov in October 2013 is a particularly high-profile example of this scenario.) When systems are compromised by intruders, investigation and remediation can take even longer. These periods of incident resolution are intensive and exhausting. When they happen multiple times a year, they can quickly take over the work of the team so that unplanned work becomes the norm, leading to burnout, an important consideration for teams and leaders.

Throughput

Deployment frequency

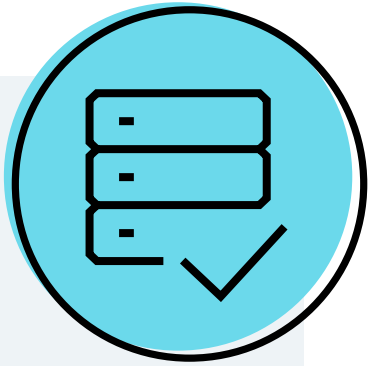
The elite group reported that it routinely deploys on-demand and performs multiple deployments per day. By comparison, low performers reported deploying between once per week and once per month; an improvement from last year. Consistent with last year, the normalized annual deployment numbers range from 1,460 deploys per year (calculated as four deploys per day x 365 days) for the highest performers to 32 deploys per year for low performers. Extending this analysis shows that elite performers deploy code 46 times more frequently than low performers.

Change lead time

Similarly, elite performers are optimizing lead times, reporting that the time from committing code to having that code successfully deployed in production is less than one hour, whereas low performers required lead times between one month and six months. With lead times of 60 minutes for elite performers (a conservative estimate at the high end of "less than one hour") and 26,940 minutes for low performers (the mean of 43,800 minutes per month and 262,800 minutes over six months), the elite group has 2,555 times faster change lead times than low performers.

It's worth noting that four deploys per day is a conservative estimate when comparing against companies such as CapitalOne that report deploying 50 times per day,[3] or companies such as Google and Netflix that deploy thousands of times per day (aggregated over the hundreds of services that comprise their production environments).

[3] Banking on Capital One: https://youtu.be/_DnYSQEUTfo



Stability

Time to restore service

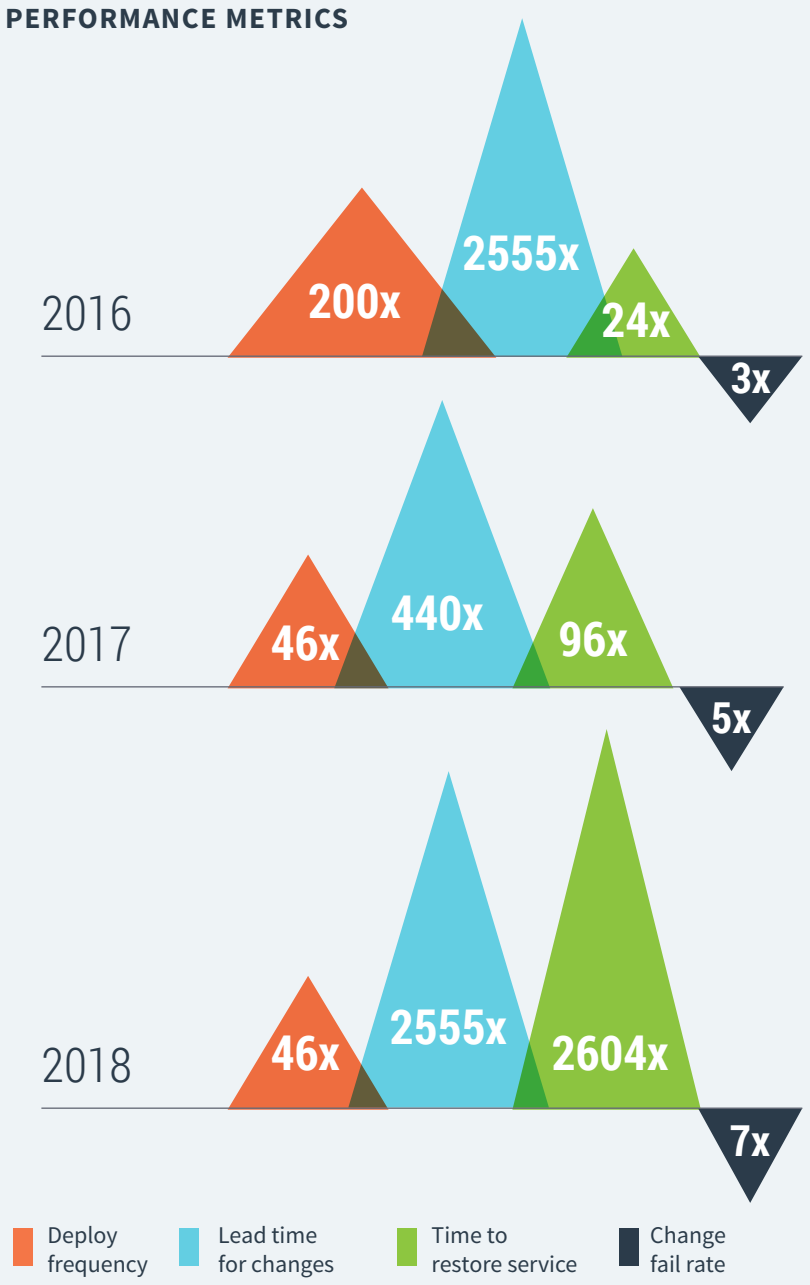
The elite group reported that its time to restore service is less than one hour, while low performers reported between one week and one month. For this calculation, we chose conservative time ranges: one hour for the highest performers and the mean of one week (168 hours) and one month (5,040 hours) for low performers. Based on these numbers, elites have 2,604 times faster time to restore service than low performers. As previously noted, time to restore service worsened for low performers compared to last year.

Change failure rate

Elite performers reported a change failure rate between zero and 15 percent, while low performers reported change failure rates of 46 to 60 percent. The mean between these two ranges shows a 7.5 percent change failure rate for elite performers and 53 percent for low performers. This represents change failure rates for elite performers that are seven times better than low performers. As noted earlier, change failure rates worsened for low performers when compared to the previous year.

All of the measures reported in this graphic are relative; that is, they compare the highest and the lowest performers each year. As we look at performance data over the past few years, some of the gaps in performance widen while others narrow. From 2017 to 2018, the gap for all performance metrics between the lowest and highest performers increased or stayed the same. The increased gap indicates a slip in performance among low performers, which may be due to growing complexity in environments and therefore difficulty in delivering software. We do note a trend in change fail rate over the past few years: the highest performers continue to see low change fail rates, while the low performers are increasingly likely to have changes impact their systems. This suggests that building resilient systems, or systems that we expect to fail (as Dr. Richard Cook says),[4] is increasingly important.

[4] http://web.mit.edu/2.75/resources/random/How%20Complex%20Systems%20Fail.pdf



SDO PERFORMANCE: ADDING AVAILABILITY

This year, we captured an additional measure of software performance important to organizations: availability. At a high level, availability represents an ability for technology teams and organizations to make and keep promises and assertions about the software product or service they are operating. Notably, availability is about ensuring a product or service is also available to and can be accessed by your end users. Our measure of availability also captures how well teams define their availability targets and learn from any outages, making sure their feedback loops are complete. The items used to measure availability form a valid and reliable measurement construct.

Analysis showed the availability measures are significantly correlated with software delivery performance profiles, and elite and high performers consistently reported superior availability, with elite performers being 3.55 times more likely to have strong availability practices.

This analysis, in addition to observations in industry and research, suggested we add availability to our model of software delivery performance.[5]

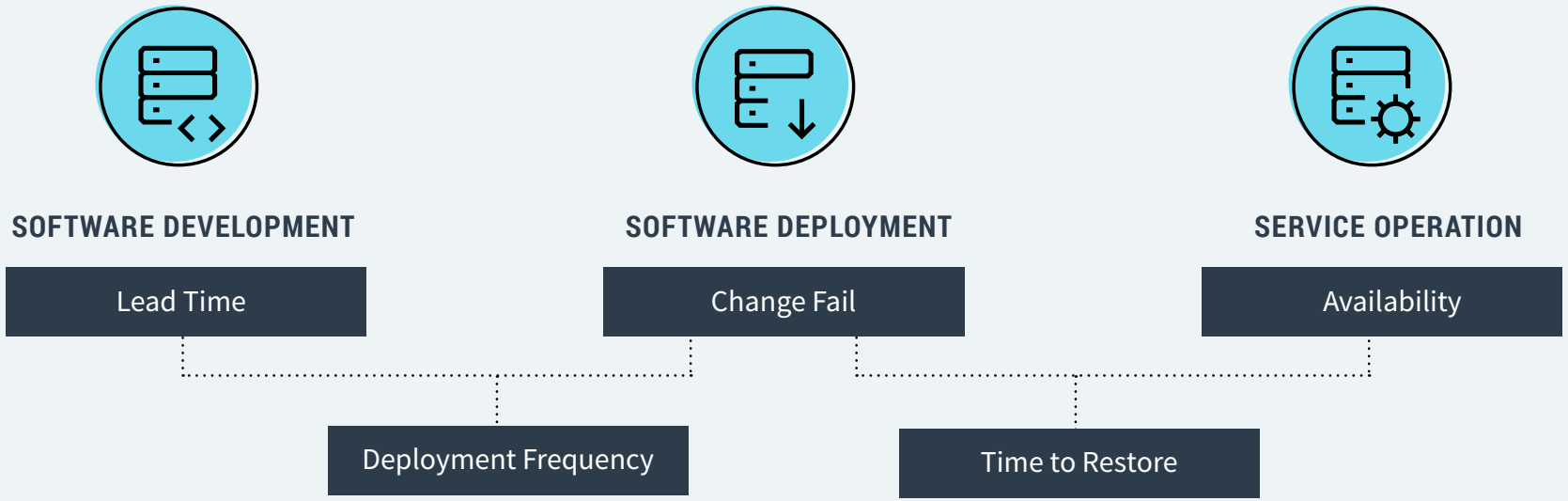
[5] We include availability in our model of software delivery as it relates to predicting organizational performance (including profitability, productivity, and customer satisfaction), but not as a classifier of performance (that is, it is not included in our cluster analysis). This is because availability measures do not apply the same way for software solutions that are not services, such as packaged software or firmware.

We show in the diagram below how our software development and delivery metrics combine with availability to form a more comprehensive view of developing, delivering, and operating software today. We also find support for this from NIST,[6] which defines availability as "ensuring timely and reliable access to and use of information." We call this new construct **software delivery and operational performance, or SDO performance**, and we find that it contributes to organizational performance.[7]

[6] NIST Special Publication 800-12r1: "An Introduction to Information Security"

[7] We note that teams can think about this in terms of software or services, and so the "s" in SDO performance can be interpreted to mean software or service.

PERFORMANCE METRICS



DOES DEVOPS MATTER?

Intuition tells us that DevOps matters: that technology transformations drive business outcomes and quality improvements. We hear stories from organizations about how they are leveraging technology to realize improved outcomes in efficiency, profit, and customer satisfaction. But stories and intuition aren't enough to support continuing investments; we need evidence and data. Our analysis shows that implementing DevOps practices and capabilities during technology transformations pays off in terms of organizational performance as well as quality outcomes.

ORGANIZATIONAL PERFORMANCE

SDO performance is a key value driver and differentiator for teams and organizations in any industry because it enables organizations to leverage software to deliver improved outcomes. These outcomes are measured by many factors, including productivity, profitability, and market share as well as non-commercial measures such as effectiveness, efficiency, and customer satisfaction. Our analysis shows that elite performers are 1.53 times more likely to meet or exceed their goals for organizational performance, and high performers are 1.38 times more likely to meet or exceed their goals.

For the fifth year in a row, our research finds that software delivery performance is an important component of organizational performance. Our measure of organizational performance references academic literature and captures two aspects of achieving or exceeding mission goals for organizations: commercial goals[8] and non-commercial goals.[9]

[8] Widener, S. K. (2007). An empirical analysis of the levers of control framework. *Accounting, organizations and society*, 32(7-8), 757-788.

[9] Cavalluzzo, K. S., & Ittner, C. D. (2004). Implementing performance measurement innovations: evidence from government. *Accounting, organizations and society*, 29(3-4), 243-267.

Combined, commercial and non-commercial goals include:

- Profitability
- Productivity
- Market share
- Number of customers
- Quantity of products or services
- Operating efficiency
- Customer satisfaction
- Quality of products or services provided
- Achieving organization or mission goals

Analysis shows that software delivery performance is an important factor in understanding organizational performance. Adding availability to the model this year created a second-order construct for predicting organizational performance. Our new second-order construct of software delivery and operational performance predicts organizational performance better than software delivery performance or availability do alone.

WHY FOCUS ON ORGANIZATIONAL GOALS RATHER THAN ABSOLUTE NUMBERS?

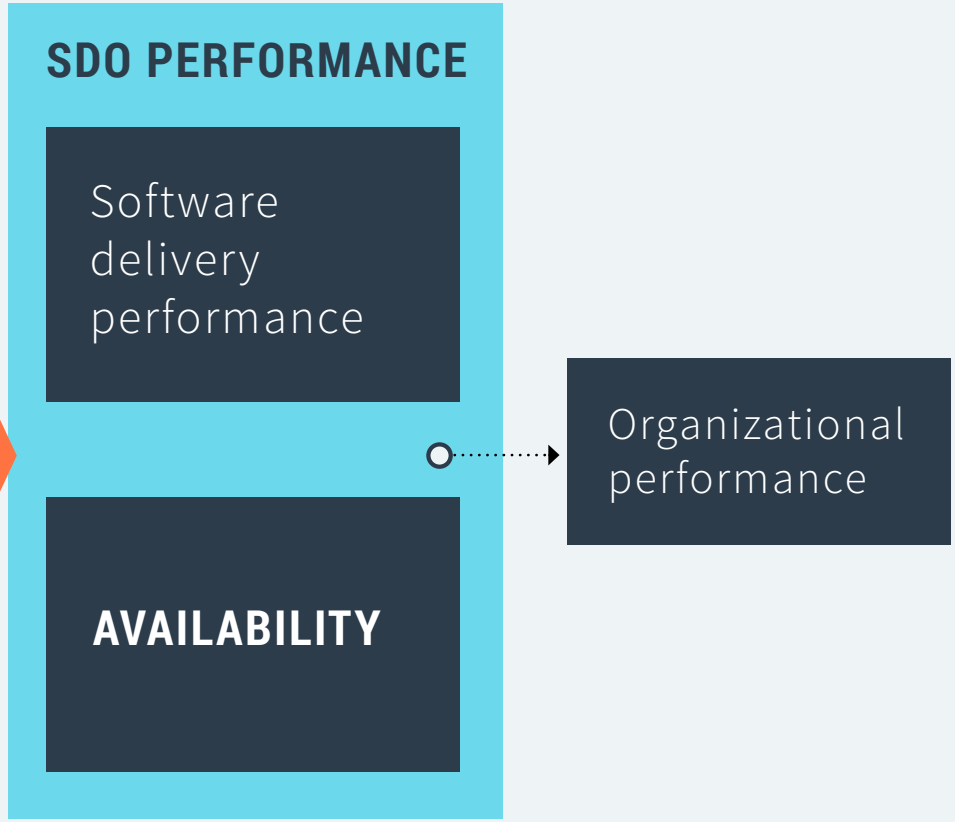
These measures allow us to collect data from companies of all sizes, across industries.
Absolute numbers make comparing a small startup to a large conglomerate nonsense, with widely different revenue, profit, and income levels. Sophisticated financial ratios may pick up on differences, but good ratios differ by industry. Measuring against organizational goals provides comparative responses that can be used across industries and organization sizes.

Respondents may not know absolute numbers for profit or revenue information.
But knowing if sales or customer satisfaction targets are being met is often general knowledge in companies of all sizes.

How closely an organization meets targets indicates how well leaders know the market and can run their business.
The stock market rewards public companies for meeting or exceeding earnings targets but punishes them if they over-exceed earnings goals because that indicates leaders didn't understand their market or business well. Measurement using absolute numbers doesn't provide these kinds of insights.

Our structural equation model (SEM) is used throughout the rest of the report. It is a predictive model used to test relationships. Each box represents a construct we measured in our research, and each arrow represents relationships between the constructs. A larger box that contains boxes (constructs) is a second-order construct.

To interpret the model, all arrows can be read using the words *predicts*, *affects*, *drives*, or *impacts*. In this example, the second-order construct SDO performance is comprised of the constructs software delivery performance and availability, and these together drive organizational performance. We indicate that availability is a newly investigated construct this year by marking it in **bold.**



QUALITY OUTCOMES

Teams and organizations embarking on technology transformations also have goals of improving quality. However, measuring quality is challenging because it is context-dependent and many measures vary by industry and even by company.[10]

Despite the challenges of identifying quality metrics that apply to all organizations, we can identify good proxies for quality that work across companies and industries. These include how time is spent, because it can tell us if we are working on value-add work or non-value-add work. In this research, we used measures such as the proportion of time spent on manual work, unplanned work or rework, security remediations, and customer-support work, and the results were revealing. Our analysis shows that high performers do significantly less manual work across all vectors, spend more time doing new work, and spend less time remediating security issues or defects than their low-performing counterparts. Because they build quality in, they spend less time fixing problems downstream, freeing up more time to do value-add work.

[10] This concept is discussed by software quality expert Jerry Weinberg in his book *Quality Software Management. Volume 1: Systems Thinking*. New York: Dorset House Publishing, 1992.

Manual Work

By leveraging automation for repetitive tasks or tasks we can parallelize (and therefore speed up), teams and organizations can improve work quality, repeatability, and consistency, and free workers from spending time on low-value tasks. With more work automated, high performers free their technical staff to do innovative work that adds real value to their organizations.

However, we've learned that estimating the level of automation in our work is difficult; it is much easier to estimate the percentage of work that is still done manually. That's not surprising. Manual work is painful and so people are highly aware of it. Once work is automated, it's no longer painful and it tends to disappear from people's attention.

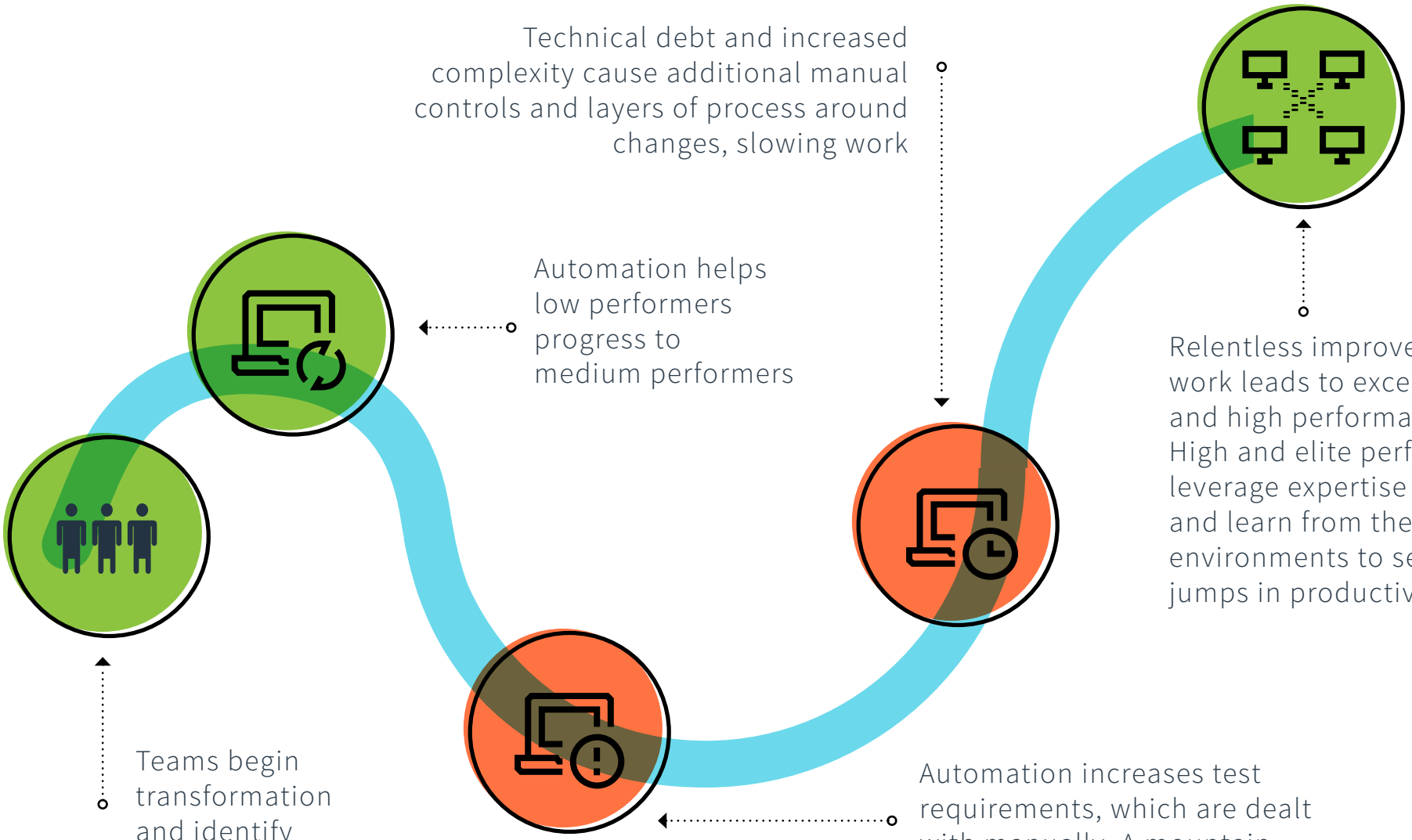
When we compare high performers to their lower-performing peers, we find that elite and high performers are doing less manual work than their lower-performing peers at statistically significant levels on all dimensions, while medium performers have the highest amount of manual work on all dimensions.

Readers may be surprised to see that medium performers are doing more manual work than low performers when it comes to testing and change-approval processes, and these differences are statistically significant. However, we also found a similar pattern in the data last year, where medium performers reported more manual work than low performers in deployment and change-approval processes than low performers (again, at statistically significant levels). We have heard and seen this story several times with teams undergoing transformations. The j-curve diagram on the next page illustrates this experience.

Manual work	Elite	High	Medium	Low
Configuration Management	5%	10%	30%[a]	30%[a]
Testing	10%	20%	50%	30%
Deployments	5%	10%	30%[b]	30%[b]
Change Approvals	10%	30%	75%	40%

Medians reported because distributions are not normal
[a,b] Not significantly different when testing for differences using Tukey's post hoc analysis

J-CURVE OF TRANSFORMATION



How Time Is Spent

Another place to measure value and quality of work is how teams spend their time. That is, when teams are doing work, are they able to focus their time devoting effort and energy on developing new features and supporting infrastructure? Or do teams spend most of their time correcting problems, remediating issues, and responding to defects and customer-support work (that is, fixing issues that arise because quality was not built in up front)? We conceptualize this time into two categories.

The first category is proactive or new work, in which we are able to design, create, and work on features, tests, and infrastructure in a structured and productive way to create value for our organizations.

The second category is called reactive unplanned work, or rework. Interruptions, errors, and reactions drive this work, making it difficult to focus and get things done. In doing work, the goal is to build quality in,[11] but measuring this is difficult. Therefore, we looked for evidence of missed quality; that is, where did errors slip through? This time spent on rework, remediations, and customer support is an indication of poor quality because we are having to spend time fixing quality we did not build in initially. In most cases, we want to spend more time on the first category and less time on the second.

[11] Deming, W. Edwards. *Out of the Crisis*. Cambridge, MA: MIT Press, 2000.

We asked our respondents how they spend their time and found that across the board, elite performers are getting the most value-add time out of their days and are spending the least amount of time doing non-value-add work of all groups, followed by high performers and medium performers. Low performers are doing the worst on all dimensions in terms of value-add vs. non-value-add time.

Time Spent	Elite	High	Medium	Low
NEW WORK	**50%**	**50%**	**40%**	**30%**
Unplanned work and rework	19.5%	20%[a]	20%[a]	20%[a]
Remediating security issues	5%	5%[b]	5%[b]	10%
Working on defects identified by end users	10%	10%[c]	10%[c]	20%
Customer support work	5%	10%	10%	15%

Medians reported because distributions are not normal.

[a] Significantly different when testing for differences using Tukey's post hoc analysis

[b, c] Not significantly different when testing for differences using Tukey's post hoc analysis

HOW DO WE IMPROVE?

Once you understand how you compare to your peers and the impact of improved performance, the next step is to apply that understanding to improve. Our analysis identifies capabilities that are statistically shown to improve software delivery and operational performance. You can leverage this information to drive conversations and initiatives to progress to higher-performing categories.

CLOUD, PLATFORM & OPEN SOURCE

Forrester predicts[12] that the total global public cloud market will be $178B in 2018, up 22 percent from 2017, and Forbes reports[13] that 83 percent of enterprise workloads will be in the cloud by 2020. In our survey, 67 percent of respondents said the primary application or service they were working on was hosted on some kind of cloud platform. This year's report looks at the impact of common cloud usage patterns on SDO performance, and finds that what really matters is how teams use cloud services, not just that they use them.

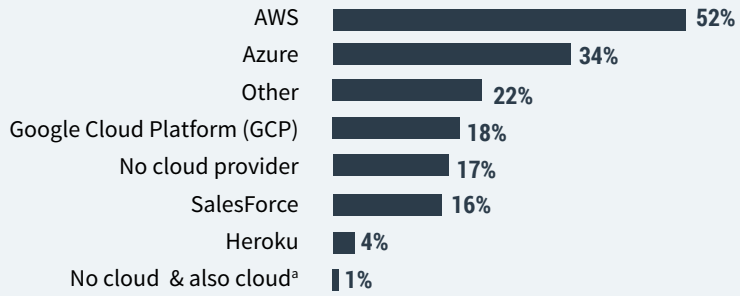
As you'll notice, the percentages add up to over 100%. We asked our respondents if their teams were using multiple cloud providers and their reasons why.[14]

[12] https://www.forrester.com/report/Predictions+2018+Cloud+Computing+Accelerates+Enterprise+Transformation+Everywhere/-/E-RES139611

[13] https://www.forbes.com/sites/louiscolumbus/2018/01/07/83-of-enterprise-workloads-will-be-in-the-cloud-by-2020/#173157906261

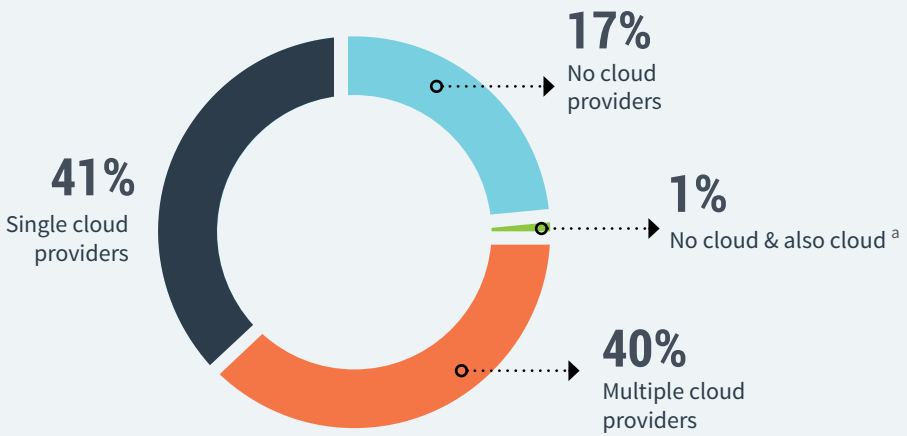
[14] We note that our respondents report cloud usage in proportions similar to that in other reports, such as the 2018 Right Scale State of the Cloud Report https://www.rightscale.com/lp/state-of-the-cloud?campaign=7010g0000016JiA and the Clutch Amazon Web Services vs. Google Cloud Platform vs. Microsoft Azure Survey, supporting the external validity of our data. (https://www.rightscale.com/lp/state-of-the-cloud?campaign=7010g0000016Ji)

CLOUD PROVIDER USAGE



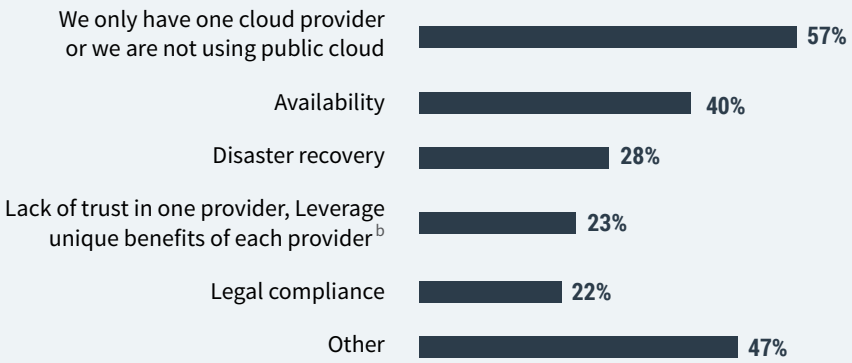
[a] Respondents indicated they use no cloud, but also selected a cloud provider.

USAGE OF MULTIPLE CLOUD PROVIDERS



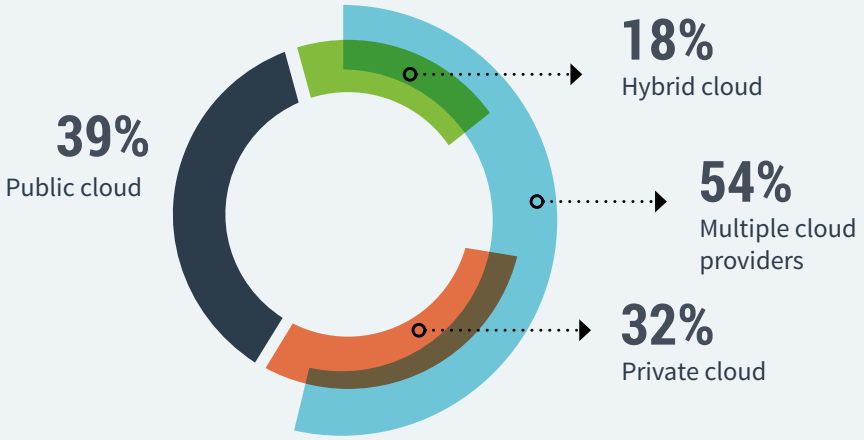
[a] Respondents indicated they use no cloud, but also selected a cloud provider.

REASON FOR USING MULTIPLE CLOUD PROVIDERS



[b] An error in survey setup meant that two options ("Lack of trust in one provider" and " Leverage unique benefits of each provider") were presented as a single option and therefore collected as one option. Our apologies.

PRIMARY PRODUCT OR SERVICE[c]



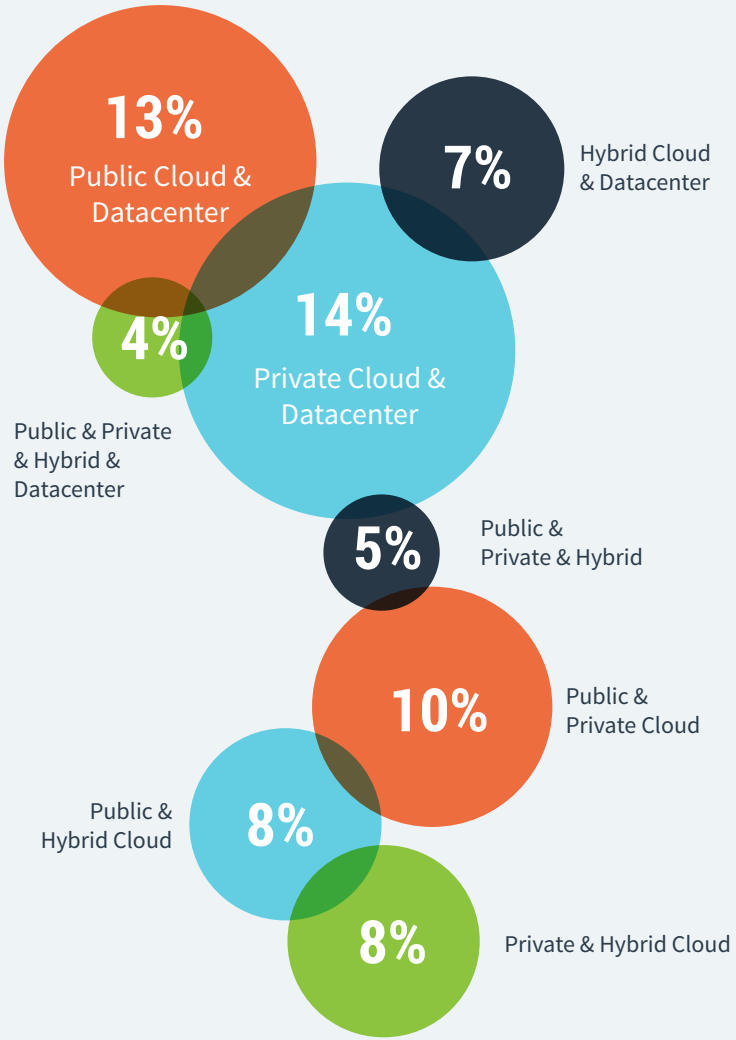
[c] Sum totals exceed 100 percent; reflecting that some products or services are deployed to multiple environments, for example 13 percent of respondents indicate the primary product or service they support runs on both a public cloud and a traditional datacenter.

Readers may note some inconsistency here: How can applications be categorized as Public & Private & Hybrid by our respondents? One challenge is that hybrid is often self-defined: If respondents say they're hybrid (using both private and public cloud), then they are. These answers may also reflect respondents working on several apps, one of which is hosted publicly, one privately, and another in a hybrid environment.

We also note that a datacenter isn't necessarily a private cloud. An organization can build a private cloud in its datacenter, but conversely, a datacenter can be managed in a traditional way that doesn't meet any of the essential cloud characteristics we describe below.

The fact that we as an industry are unclear when using these definitions may explain why some reports don't seem to reflect our experience. That is, our experience may differ due to definitions and measurement. We address this specifically in the next section where we talk about cloud computing.

OVERLAPS OF CLOUD TYPE USAGE



How You Implement Cloud Infrastructure Matters

Respondents who agreed or strongly agreed that they met all essential cloud characteristics were 23 times more likely to be in the elite group than those in the low performing group. Similarly, users adopting platform as a service are 1.5 times more likely to be elite performers and users adopting cloud-native design practices are 1.8 times more likely to be elite performers. Users of infrastructure as code to manage their cloud deployments are 1.8 times more likely to be elite and users of containers are 1.5 times more likely to be elite performers.

However, many respondents that say they are using cloud computing haven't actually adopted the essential patterns that matter—and this could be holding them back. NIST defines five essential characteristics of cloud computing (see next page), but only 22 percent of respondents that said they were using cloud infrastructure agreed or strongly agreed that they met all of these characteristics.[15]

23 TIMES

Teams that adopt essential cloud characteristics are 23 times more likely to be elite performers.

[15] NIST Special Publication 800-145: "The NIST Definition of Cloud Computing."

These characteristics matter when defining what it really means to adopt cloud computing, and our research shows they impact software delivery performance. Respondents who agreed or strongly agreed that they met all characteristics were 23 times more likely to be in the elite group than those in the low performing group.

Broad network access and on-demand self-service are often overlooked and are especially important because they directly affect performance outcomes for consumers. For example, some cloud implementations still require that users raise tickets in order to access critical resources to accomplish their work or they cannot access cloud systems easily from their devices. From the consumer perspective, they may as well be using a traditional datacenter. This is a huge barrier to realizing the efficiency improvements in delivery process that lead to higher-performance teams.

FIVE ESSENTIAL CHARACTERISTICS OF CLOUD COMPUTING

%	AGREED OR STRONGLY AGREED
46%	**On-demand self-service** Consumers can provision computing resources as needed, automatically, without any human interaction required.
46%	**Broad network access** Capabilities are widely available and can be accessed through heterogeneous platforms (e.g., mobile phones, tablets, laptops, and workstations).
43%	**Resource pooling** Provider resources are pooled in a multi-tenant model, with physical and virtual resources dynamically assigned and reassigned on-demand. The customer generally has no direct control over the exact location of provided resources, but may specify location at a higher level of abstraction (e.g., country, state, or datacenter).
45%	**Rapid elasticity** Capabilities can be elastically provisioned and released to rapidly scale outward or inward commensurate with demand. Consumer capabilities available for provisioning appear to be unlimited and can be appropriated in any quantity at any time.
48%	**Measured service** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported for transparency.

Platform as a Service

Another way to provide a better service to application developers is through implementing a platform as a service (PaaS) in which "the consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment."[16] Examples of a PaaS include Heroku, RedHat OpenShift, Azure App Service, Google App Engine, AWS Elastic Beanstalk and Cloud Foundry.

Only 24 percent of respondents report using a PaaS. However, respondents that do most of their work on a PaaS are 1.5 times more likely to be in the elite performance group. These respondents agreed or strongly agreed that their team uses libraries and infrastructure defined by the PaaS as the basis for their applications, can deploy their application into the cloud on demand using a single step, and can perform self-service changes on-demand for databases and other services required by their application.

[16] NIST Special Publication 800-145: "The NIST Definition of Cloud Computing."

Infrastructure as Code

One of the key innovations of the DevOps movement is the idea of infrastructure as code. In this paradigm, we reproduce and change the state of our environments in an automated fashion from information in version control rather than configuring infrastructure manually.

This way of working is a natural fit for cloud infrastructure, where resources can be provisioned and configured through APIs. Tools such as Terraform make it simple to provision and evolve cloud infrastructure using declarative, version-controlled configuration. This, in turn, makes provisioning testing and production environments fast and reliable, improving outcomes both for administrators and users of cloud infrastructure. Similar techniques can be used to deploy applications automatically.

In our study, 44 percent of cloud adopters agreed or strongly agreed that environment configuration and deployments use only scripts and information stored in version control, with no manual steps required (other than approvals). Respondents using infrastructure as code are 1.8 times more likely to be in the elite performance group.

Cloud Native

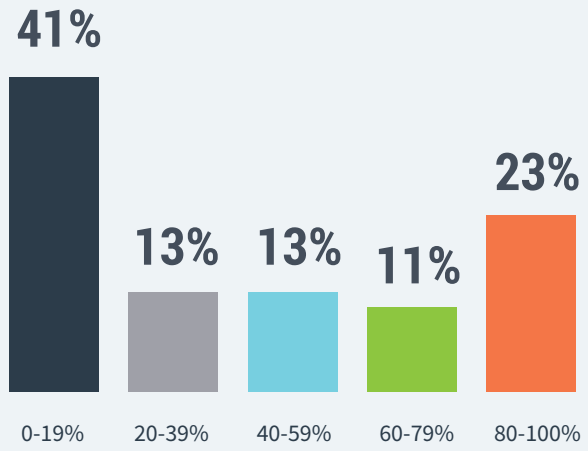
Applications designed specifically around the constraints inherent to cloud systems are referred to as cloud native. These applications differ from those designed for traditional datacenters in several critical ways.[17] Importantly, systems in the cloud are presumed to run on unreliable underlying infrastructure and must be designed to handle failures. This means cloud native applications must be resilient, able to respond dynamically to changes in workload (i.e., elastic), and easy to deploy and manage on-demand.

Of respondents deploying to a cloud, 47 percent agreed or strongly agreed that the application or service they were working on was originally designed and architected to run in the cloud. While **not all high-performing teams run cloud native applications**, teams that do are 1.8 times more likely to be in the elite performing group.

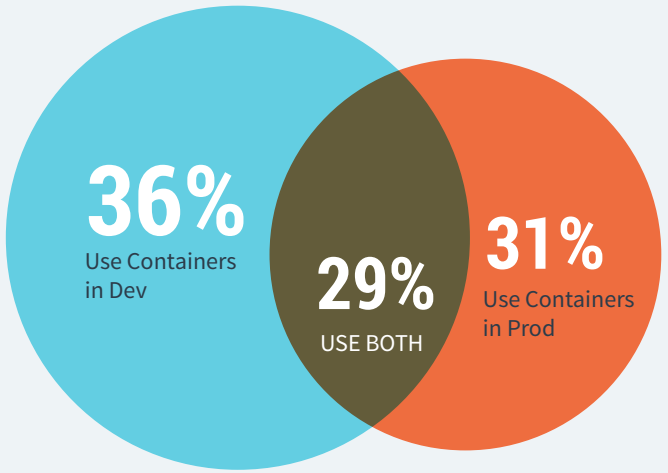
Finally, we asked people whether they were using containers. Respondents using containers in production are 1.3 times more likely to be elite performers.

[17] A popular guide to designing cloud native applications is available at https://12factor.net/

PERCENTAGE OF PRODUCTS/SERVICES RUNNING IN THE CLOUD THAT ARE CLOUD NATIVE



CONTAINER USE



Open Source

Over the last two decades, open source software has become widely adopted. In our survey, 58 percent of respondents agreed that their team made extensive use of open source components, libraries, and platforms, with over 50 percent agreeing that their team planned to expand use of open source software.

Elite performers are 1.75 times more likely to make extensive use of open source components, libraries, and platforms than low performers, and 1.5 times more likely to plan to expand their use of open source software.

OPEN SOURCE AT CAPITAL ONE

Capital One is one of the ten largest banks in the US, and is known for its innovative approach to customized services and offerings. The company embraced Agile and DevOps methods early, and credit that with allowing them to become a much more productive, high-performing organization that empowers technology professionals to do better work and deliver a superior product to their customers.

Dr. Tapabrata Pal, Sr. Engineering Fellow at Capital One, who has worked with engineering teams at the company extensively, says: “At Capital One we have an ‘Open Source First’ philosophy. We build and run our software on Open Source foundations, and we actively contribute to the Open Source community. We also launch our own Open Source projects, most notably our award-winning DevOps dashboard, Hygieia. We believe that it is essential to embrace, adapt and adopt Open Source. For us, it is not just the Open Source software, it is also the culture – the culture of collaboratively building software in our organization. This use of and approach to open source has helped Capital One to deliver software to our customers faster, more reliably, and with higher quality, allowing us to better serve our customers.”

OUTSOURCING

Outsourcing has traditionally been viewed as a quick way to expand capabilities and bandwidth. Aside from the workload benefits for short-term or difficult-to-hire projects, outsourcing can be beneficial in cost; it reduces the number of full-time employees required and provides elasticity for technical labor. A popular outsourcing model is to assign individual organizational functions—for example, application development, testing/QA, or service operation—to external vendors. However this model introduces additional handoffs and potential friction between functional groups. The functional division of responsibilities can also inhibit agility: Once contracts have been signed, changes to specifications are difficult to manage across external silos.

The handoffs and silos created in many outsourcing models have drawn criticism from Agile and DevOps communities because they are perceived as a barrier to high performance. This year, we looked at outsourcing practices and impacts on SDO performance. We asked respondents about the extent to which they outsource application development, testing and QA, and IT operations work.

Analysis shows that low-performing teams are 3.9 times more likely to use functional outsourcing (overall) than elite performance teams, and 3.2 times more likely to use outsourcing of any of the following functions: application development, IT operations work, or testing and QA. This suggests that outsourcing by function is rarely adopted by elite performers.

Let's take a deeper look into the impact of outsourcing by function. Using the information we have about the elite and low-performing profiles, we can quantify and estimate some of the consequences in dollars. First, we know that elite performers typically deliver software multiple times per day, whereas low performers deliver between once every month and once every six months.

IMPACT OF OUTSOURCING ON MISGUIDED PERFORMERS

Recall our "misguided performers" from the Software Delivery Performance section: the group with deployment frequency and lead time for changes slower than or on par with our low performers. While this group has a better change fail rate than low performers, it also reports the longest time to restore service, with downtimes of one to six months.

Misguided performers also report the highest use of outsourcing, which likely contributes to their slower performance and significantly slower recovery from downtime. When working in an outsourcing context, it can take months to implement, test, and deploy fixes for incidents caused by code problems.

Outsourcing tends to lead to batching work—and thus long lead times—because the transaction cost of taking work from development to QA to operations is so high when these are held in outsourced groups. When work is batched into projects or releases, high-value and low-value features get lumped together into each release, meaning that all of the work—whether high or low value—is delivered at the same speed.

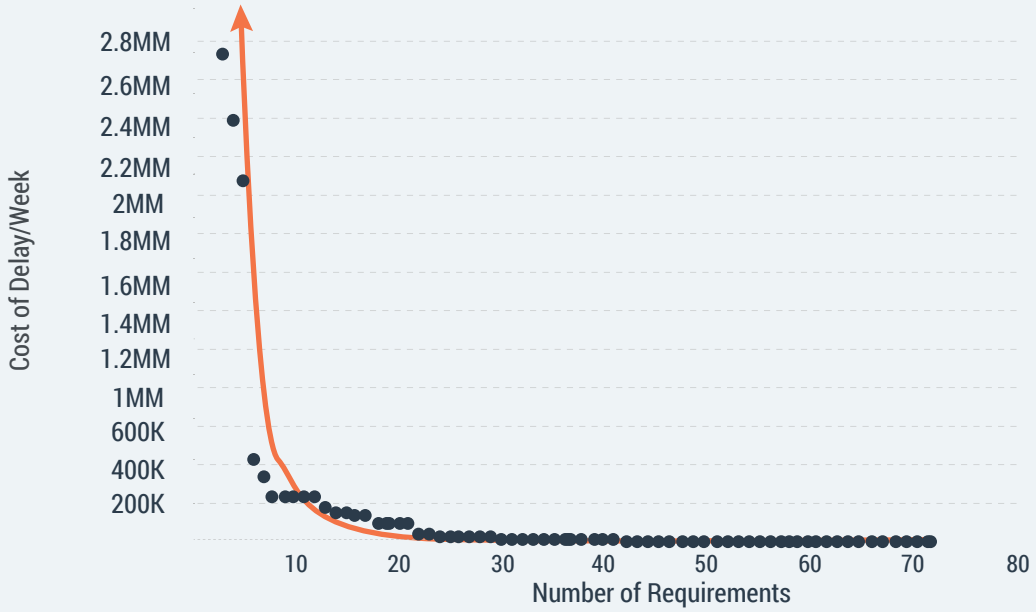
Let us emphasize this point: Important and critical features are forced to wait for low-value work because they are all grouped together into a single release. We're sure many professionals have seen this in practice: in most project backlogs, there are a few features that are disproportionately valuable. The cost of delaying these high-value features (because all features are released together) is often significant. And in many cases, this cost of delay is likely to exceed the amount saved through outsourcing.

Let's take a concrete example from Maersk Line, the world's largest container shipping company.[18] In one project, a team estimated how much it was costing the company per week to not have the features delivered from the backlog.

[18] This example, along with the figure on the next page, is taken from "Black Swan Farming Using Cost of Delay" by Joshua Arnold and Özlem Yüce, https://blackswanfarming.com/experience-report-maersk-line/

We can work through an example based on the cost of delay calculated by Maersk, and assuming we are working in a team with a profile matching elite performers. As a reminder, this means we are able to deliver a completed feature on demand, rather than waiting months for a batch of features to be released. Just the top three features in the graph have a cost of delay of roughly USD $7 million per week, or about $30 million per month. If, as our data shows, outsourcing by function is correlated with lead times of months to get features delivered, it is entirely possible that the costs associated with this delay on your ability to deploy far outweigh the savings of outsourcing.

MAERSK COST OF DELAY POWER LAW CURVE



This power law curve[19] is typical of product backlogs

[19] A power law distribution describes phenomena where a small number of items accounts for 95% of the resources. See http://www.statisticshowto.com/power-law/

Of course, this is not the only benefit of improving SDO performance. High performers are also able to restore service more rapidly in the event of an outage, have more stable releases, have better availability, and produce higher-quality software with fewer defects.

There are some important caveats to our findings on outsourcing. The arguments presented here address wholesale outsourcing by function (development, testing, QA, operations), and these findings confirm many stories we hear in industry about outsourcing leading to poor outcomes. (These arguments can also be applied to organizations with internal functional silos.)

However, it is important to note that our findings do not necessarily extend to other working models involving multiple entities. For example, these arguments don't apply to the use of partners or vendors for the development of whole systems, wherein the entire process is outsourced from design through to operation. In this case, the key to success is that technical dependencies between services are managed properly[20] and do not cause delays to the delivery of high-value features.

---

[20] For more detail, we point you to the importance of a loosely coupled architecture in the 2017 State of DevOps Report. https://devops-research.com/assets/state-of-devops-2017.pdf

Other popular models that are not addressed in this analysis include both geographically distributed teams and so-called “embedded” contractor models. A key difference in these models is that the “other” teams—whether they are in-house distributed teams or the additional staff provided by contracting and consulting firms—operate and behave as part of the primary organization’s cross-functional product or technology teams; if the rhythms of software development and delivery are maintained, outcomes are very likely to be maintained.

In fact, we see support for this in the data: high-performing teams are twice as likely to be developing and delivering software in a single, cross-functional team, a key practice we discuss in the next section.

LEAN AND AGILE PRACTICES

When hoping to improve performance, organizations often focus on buying solutions in the form of tools, vendors, and methodologies. **However what's important is the capabilities these solutions enable, not the solutions themselves**. Over the last four years we've set out to identify which capabilities actually have a statistically significant impact on the outcomes we care about. This year, we confirmed that lean approaches to product management impact software delivery performance and looked at how teams are organized.

The Importance of Cross-Functional Teams

The concept of cross-functional teams is central to many Agile approaches. According to the *Scrum Guide*, "The Scrum Team consists of a Product Owner, the Development Team, and a Scrum Master. Scrum Teams are self-organizing and cross-functional… Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team."[21] In *Extreme Programming Explained: Embrace Change* (Second Edition), Kent Beck and Cynthia Andres write, "Include on the team people with all the skills

[21] https://www.scrumguides.org/scrum-guide.html#team

and perspectives necessary for the project to succeed." (p38) Indeed, we found that low performers were twice as likely to be developing and delivering software in separate, siloed teams than elite performers.

Lean Product Management

In prior years, we looked at the impact of Lean and Agile product management practices on both software delivery performance and organizational performance. These practices have seen success in many contexts, although they are not always implemented. For example, it's still common to see months spent on budgeting, analysis, and requirements-gathering before starting work; to batch work into big projects with infrequent releases; for software delivery teams to have no input over how their work is done; and for customer feedback to be treated as an afterthought.

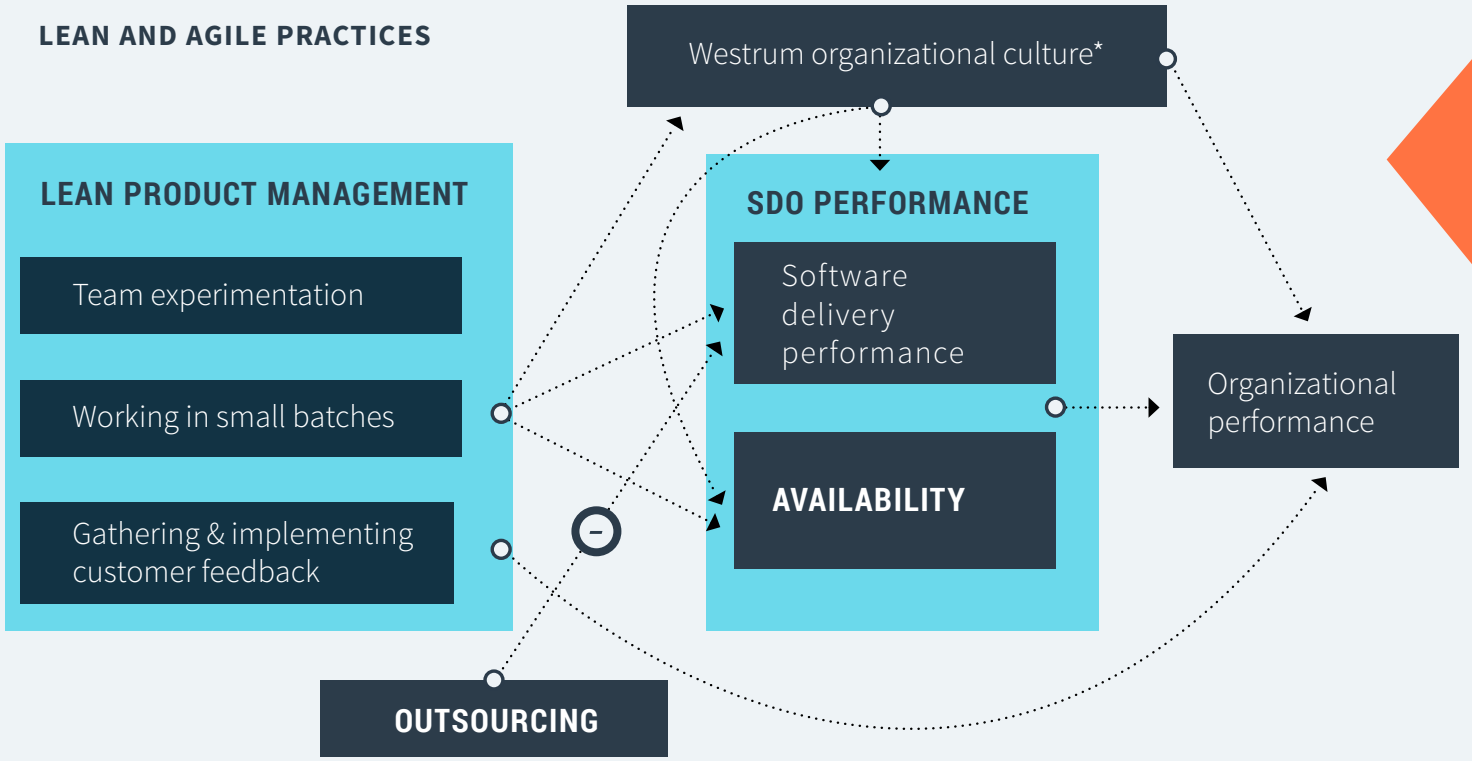
THREE CHARACTERISTICS THAT COMPRISE A LEAN APPROACH TO PRODUCT DEVELOPMENT

1. The extent to which teams slice up products and features into small batches that can be completed in less than a week and released frequently, including the use of minimum viable products (MVPs)

2. Whether organizations actively and regularly seek customer feedback and incorporate this feedback into the design of their products

3. Whether development teams have the authority to create and change specifications as part of the development process without requiring approval

Our research confirms findings from earlier studies: Lean product management capabilities positively impact software delivery performance, organizational culture, and organizational performance. Our research this year also finds new results: outsourcing negatively affects software delivery performance and Lean product management positively affects availability.



* Westrum is a measure of organizational culture particularly well-suited to DevOps. We discuss this in detail on page 62 in our Culture section.

Recall that boxes are constructs and boxes that contain boxes are second-order constructs. Arrows are predictive relationships. Constructs in bold are new this year, while others have been studied in previous years and their relationships are revalidated in this year's research.

TECHNICAL PRACTICES

Our research highlights practices that are essential to successful technology transformations. These include the use of version control, deployment automation, continuous integration, trunk-based development, and a loosely coupled architecture. This year we also found that using monitoring and observability solutions, continuous testing, integrating database changes into the software delivery process, and shifting left on security all positively contribute to continuous delivery.

Continuous Delivery

Technical practices in delivery and deployment that reduce the risk and cost of performing releases—which we collectively refer to as continuous delivery—are key to achieving higher software delivery performance. Note that continuous delivery for the sake of continuous delivery is not enough if you want your organization to succeed.

HOW WE MEASURED CONTINUOUS DELIVERY

1. Teams can deploy on-demand to production or to end users throughout the software delivery lifecycle.

2. Fast feedback on the quality and deployability of the system is available to everyone on the team and acting on this feedback is team members' highest priority.

As in previous years, we confirmed our hypotheses that the following capabilities all positively affect continuous delivery: deployment automation, continuous integration, trunk-based development, loosely coupled architecture, and the use of version control for all production artifacts. We also tested new capabilities this year, which we discuss in greater detail. These all positively affect continuous delivery: monitoring and observability, continuous testing, integrating data and the database into the deployment pipeline, and integrating security into software delivery work. We also confirmed outcomes of continuous delivery this year and found it significantly contributes to reductions in deployment pain and burnout and improvements in SDO performance.

Monitoring and Observability

Good monitoring has been a staple of high-performing teams. In previous years, we found that proactively monitoring applications and infrastructure, and using this information to make business decisions, was strongly related to software delivery performance.

MONITORING

is tooling or a technical solution that allows teams to watch and understand the state of their systems and is based on gathering predefined sets of metrics or logs.

OBSERVABILITY

is tooling or a technical solution that allows teams to actively debug their system and explore properties and patterns they have not defined in advance.

Growing system complexity has driven more discussion about monitoring and observability, so we included these topics in this year's research. We found that a comprehensive monitoring and observability solution positively contributes to continuous delivery and that those who have one were 1.3 times more likely to be in the elite-performing group.

To understand how teams are leveraging monitoring and observability in their work, we created our survey measures with the assumption that monitoring and observability are two distinct capabilities or practices. However, when we statistically validated our data, we found that the survey respondents perceive these practices as the same thing. Therefore, our analysis was conducted with a construct that combines monitoring and observability.

MEASURING CONSTRUCTS

In order to carefully capture and measure capabilities, we followed a rigorous approach that we use when we conduct our research:*

1. Define each capability (or construct)
2. Develop survey questions (or items) based on the carefully written definitions and have them reviewed by subject matter experts
3. Collect data
4. Validate constructs through statistical tests

We use several items whenever possible, because there is always the possibility that an item could be misunderstood and needs to be discarded from further analysis. During the statistical validation process (Step 4), we analyze the survey items to validate they are measuring the constructs they are intended to measure, they are not measuring constructs they are not intended to measure, and that survey respondents understand them consistently.

This process allows us to take a careful and systematic approach to survey research, so that we can measure and investigate new areas and test their impact on outcomes.

*More detail about this process can be found in Part II of *Accelerate: The Science of Lean Software and DevOps*

Are monitoring and observability the same thing? Industry authorities strongly argue that they are not, so allow us to clarify. What we mean when we say that monitoring and observability loaded onto the same construct is that this year's respondents perceive the two sets of practices and capabilities as basically the same thing. This presents a couple of possibilities for interpretation. First, the observability market is still relatively new and hasn't clearly outlined differences in a way that resonates with the entire market, suggesting there is opportunity for differentiation and messaging. Another possibility is that monitoring and observability have a market differentiation that is only noticeable or meaningful for a subset of specialized users while our survey targeted DevOps practitioners working in all facets of technology and not just those that would notice differences between monitoring and observability.

Continuous Testing

In previous years we found that test automation had a significant impact on continuous delivery. This year, we built upon prior years' research and found that continuous testing positively impacts continuous delivery.

In recent years there has been skepticism of continuous delivery from some parts of the testing community so we wanted to investigate if the evolving role of testing contributes to the continuous delivery outcomes presented above.

But what is continuous testing? And how does it differ from automated testing? In previous years, we found that automated testing was important to continuous delivery. Our measure of automated testing includes fast, reliable suites of automated tests that are primarily created and maintained by developers. In addition, automated tests should be easy for developers to reproduce, developers should be able to fix test failures using their own development environments, and technical professionals should have test data available to easily run tests.

Continuous testing includes these practices, with some important additions:

- Continuously reviewing and improving test suites to better find defects and keep complexity and cost under control
- Allowing testers to work alongside developers throughout the software development and delivery process
- Performing manual test activities such as exploratory testing, usability testing, and acceptance testing throughout the delivery process
- Having developers practice test-driven development by writing unit tests before writing production code for all changes to the codebase
- Being able to get feedback from automated tests in less than ten minutes both on local workstations and from a CI server

Continuous testing can be a significant investment, but because teams see testing in much more of the development and delivery lifecycle we do see strong benefits from this approach.

Managing Database Changes

Database changes are often a major source of risk and delay when performing deployments. We wanted to investigate which database-related practices help when implementing continuous delivery to improve both software delivery performance and availability.

Our analysis found that integrating database work into the software delivery process positively contributed to continuous delivery[22] but how can teams improve their database delivery in continuous delivery? There are a few practices that are predictive of performance outcomes. We discovered that good communication and comprehensive configuration management that includes the database matter. Teams that do well at continuous delivery store database changes as scripts in version control and manage these changes in the same way as production application changes. Furthermore, when changes to the application require database changes, these teams discuss them with the people responsible for the production database and ensure the engineering team has visibility into the progress of pending database changes. When teams follow these practices, database changes don't slow them down, or cause problems when they perform code deployments.

---

[22] For a detailed discussion and tips on integrating database work into your software delivery pipeline, we suggest reading the excellent *Database Reliability Engineering* by Laine Campbell and Charity Majors.

Security and Security Performance

Information security is vitally important in an era where threats are ubiquitous, ongoing, and sometimes state-sponsored. Many organizations are also subject to regulations such as PCI DSS, HIPAA, or Sarbanes-Oxley, which require the implementation of information security (infosec) controls as part of the software delivery lifecycle. In our survey, 87 percent of respondents said they were subject to requirements related to regulatory compliance.

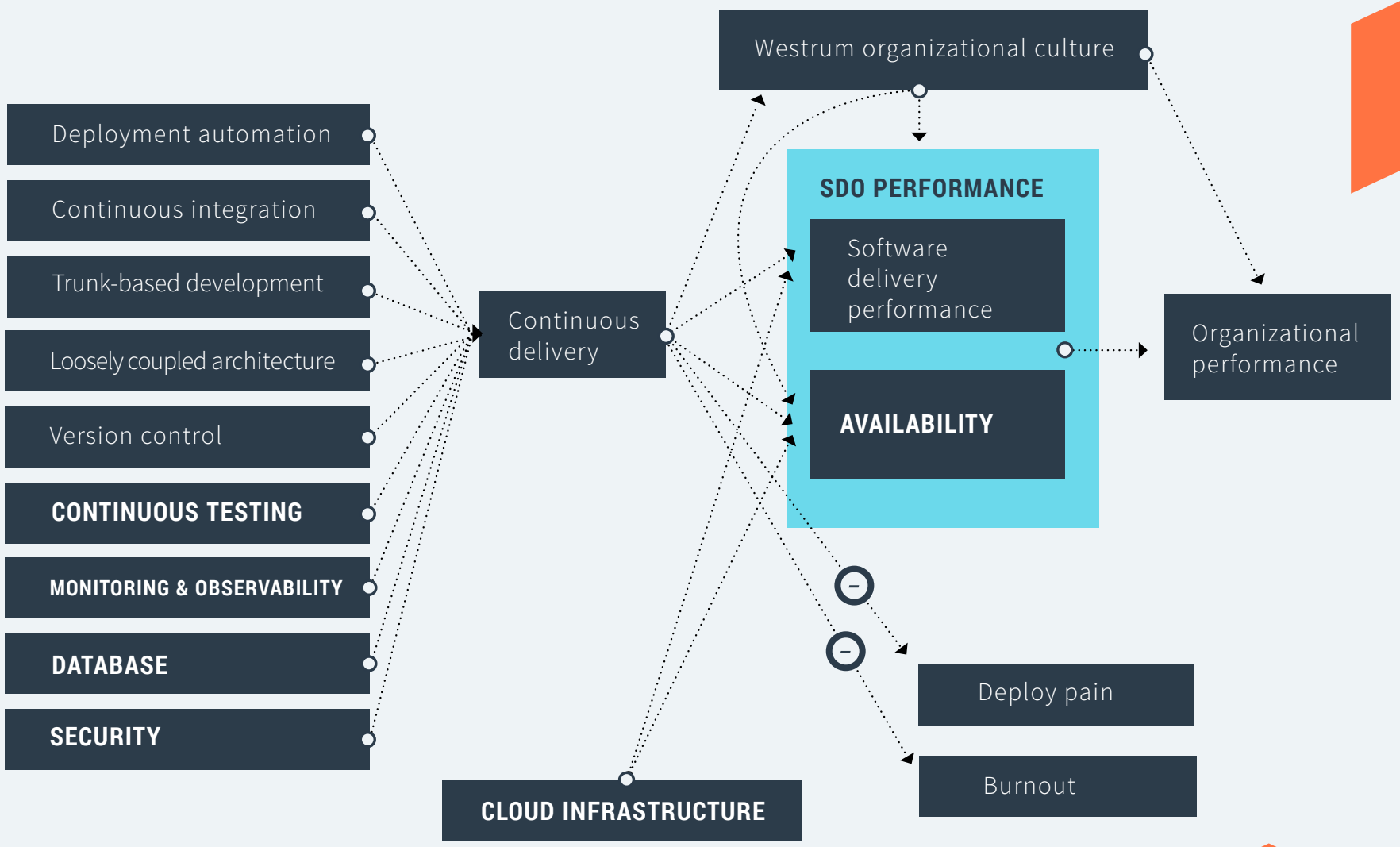
However, infosec teams are often relatively poorly staffed when compared to their technical peers. James Wickett, head of research at Signal Sciences, cites a ratio of one infosec person per 10 infrastructure people per 100 developers in large companies. Furthermore, he points out they are usually only involved at the end of the software delivery lifecycle, when it is often painful and expensive to implement the changes necessary to improve security.

Building security into software development improves SDO performance and security quality. **Low performers take weeks to conduct security reviews and complete the changes identified**. In contrast, **elite performers build security in and can conduct security reviews and complete changes in just days.**

Our research shows that infosec personnel should have input into the design of applications and work with teams (including performing security reviews for all major features) throughout the development process. In other words, we should adopt a continuous approach to delivering secure systems. In teams that do well, security reviews do not slow down the development process.

Shifting left on security drives continuous delivery. Teams should build security in by running tests to help discover security problems throughout the software development process, making it easy for teams to consume pre-approved libraries, packages, and toolchains, and having predefined secure processes for teams to reference.

We also asked who is responsible for security. In this word cloud, we see the relative frequency of the possible responses, showing us that among survey respondents, the operations and infrastructure team is often left doing most of the security work—even above infosec professionals.



Recall that boxes are constructs and boxes that contain boxes are second-order constructs, while arrows are predictive relationships. Constructs in **bold** are new this year, while others have been studied in previous years and their relationships are revalidated in this year's research.

CULTURE

Culture has always been a key part of the DevOps, Agile, and Lean movements. However, culture is intangible and not straightforward to measure. In 2014, we operationalized and validated a model of organizational culture proposed by sociologist Ron Westrum and showed that it drives both software delivery performance and organizational performance. Over the last few years we've found a number of management and technical capabilities that influence culture, showing that you can change culture by changing the way work is done in your organization.

This year we revalidated some of our results from previous years, and investigated how to influence culture through leadership practices and learning culture.

Westrum Organizational Culture

For the last several years, our research has confirmed what many in industry have been saying for years: that culture is a key component of DevOps and technology transformations. We find that technical and management practices shape culture and that culture in turn helps to improve performance outcomes.

To measure organizational culture, we reference a typology developed by Ron Westrum, a sociologist who found that organizational culture was predictive of safety and performance outcomes. Westrum's model of organizational cultures includes three types of organizations:[23]

[23] Westrum, Ron. "A Typology of Organisational Cultures." *Quality and Safety in Health Care* 13, no. suppl 2 (2004): ii22–ii27

Pathological (Power-oriented)	Bureaucratic (Rule-oriented)	Generative (Performance-oriented)
Low cooperation	Modest cooperation	High cooperation
Messengers "shot"	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure leads to scapegoating	Failure leads to justice	Failure leads to inquiry
Novelty crushed	Novelty leads to problems	Novelty implemented

His definition of culture references many things we hear when talking about things important to DevOps teams: cooperation, surfacing problems (training messengers to bring us bad news so we can find and fix errors), breaking down silos (bridging encouraged), postmortems (failure leads to inquiry), and continually experimenting to drive improvement (novelty implemented).

This also mirrors other research, which shows that team dynamics are much more important to team effectiveness than a particular set of skills among team members. When researchers at Google studied over 180 engineering teams, they found that the most important factor in predicting a high-performing team is psychological safety, or feeling safe taking risks around your team.[24] This was followed by dependability, structure and clarity of work, meaning, and impact.

When teams have a good dynamic, their work benefits at the technology and organizational level. Our research has confirmed this for several years and we caution organizations not to ignore the importance of their people and their culture in technology transformations.

---

[24] https://rework.withgoogle.com/blog/five-keys-to-a-successful-google-team/

But how do we influence culture? We have seen that our management and technical practices (that is, the way we do our work) influence culture. Is there anything else we can do?

Influencing culture through leadership and autonomy

This year, we investigated the role that leaders have on influencing culture. We found that when leaders give their teams autonomy in their work it leads to feelings of trust and voice. Trust reflects how much a person believes their leader or manager is honest, has good motives and intentions, and treats them fairly. Voice is how strongly someone feels about their ability and their team's ability to speak up, especially during conflict—for example, when team members disagree, when there are system failures or risks, and when suggesting ideas to improve work. Trust and voice, in turn, positively affect organizational culture.

EMPLOYEE NET PROMOTER SCORE (eNPS)

Employee Net Promoter Score (eNPS) is a measure of how engaged employees are and how likely they are to recommend their team or organization to their peers. It is correlated to company growth in many industries,[25] and other research has found that it is correlated with better business outcomes.[26]

Our research found that Westrum organizational culture is highly correlated with (eNPS)[27] and that elite performers are 1.8 times more likely to recommend their team as a great place to work.

---

[25] https://hbr.org/2003/12/the-one-number-you-need-to-grow/ar/1

[26] Azzarello, Domenico, Frédéric Debruyne, and Ludovica Mottura. "The Chemistry of Enthusiasm." Bain.com. May 4, 2012. http://www.bain.com/publications/articles/the-chemistry-of-enthusiasm.aspx

[27] https://en.wikipedia.org/wiki/Net_Promoter

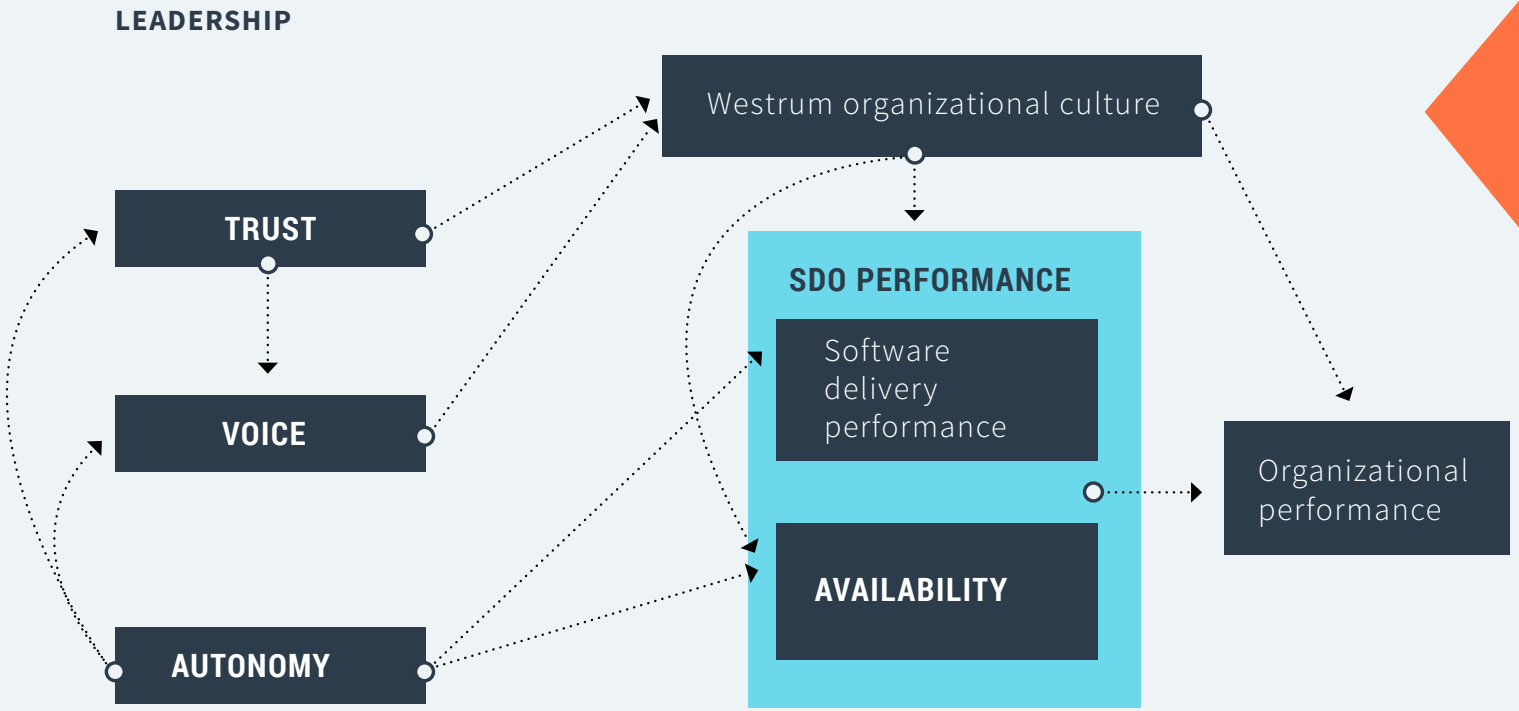
How can leaders most effectively help their teams gain autonomy in their work? Important components are:

- Establishing and communicating goals, but letting the team decide how the work will be done
- Removing roadblocks by keeping rules simple
- Allowing the team to change rules if the rules are obstacles to achieving the goals
- Letting the team prioritize good outcomes for customers, even if it means bending the rules

We can see that this prioritizes strong leadership because clear communication of outcomes and goals to the team is key. And once the team understands the goal, a good leader trusts team members to execute according to their expertise. Indeed, our research finds that more autonomy fosters trust in the leader—that is, the team believes its leader is fair, honest, and trustworthy. This trust in leadership contributes to a stronger organizational culture.

Autonomy has additional benefits. It leads teams to voice their opinions about their work, the team, and suggestions to improve the work. This transparent communication helps improve organizational culture as well.

We show this in the model below.

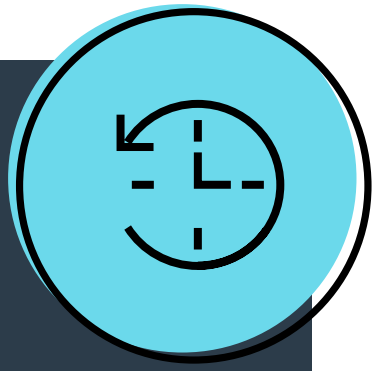


Recall that boxes are constructs and boxes that contain boxes are second-order constructs. Arrows are predictive relationships. Constructs in bold are new this year, while others have been studied in previous years and their relationships are revalidated in this year's research.

Influencing culture through learning

Another way to influence organizational culture is through learning. In DevOps and engineering circles, this is often done through retrospectives, also called learning reviews. In the operations world, learning reviews are often performed after an incident in order to figure out how to improve the system to prevent similar incidents from happening again. In this context they are sometimes known as post-mortems.

The goal is the same, however: learning how to improve. In both the Agile and ops worlds, the importance of making these activities "blameless" is often emphasized.[28] For example, in his 2001 work *Project Retrospectives: A Handbook for Team Reviews*, Norm Kerth presents the "Retrospective Prime Directive," which he suggests participants read at the beginning of a retrospective: "Regardless of what we discover, we must understand and truly believe that everyone did the best job they could, given what was known at the time, their skills and abilities, the resources available, and the situation at hand."



RETROSPECTIVES

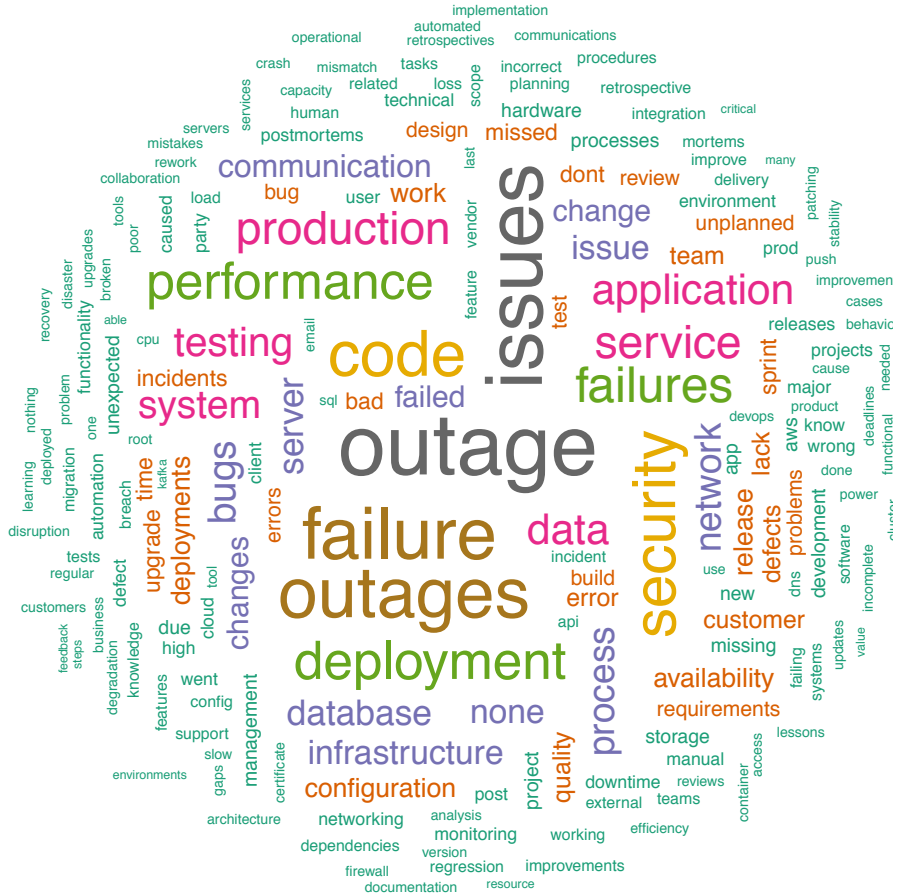
The twelfth principle of the Agile Manifesto states, "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly."

[28] An important addition to blameless post-mortems has been made by J. Paul Reed, with his writeup of "blame-aware post-mortems" (https://techbeacon.com/blameless-postmortems-dont-work-heres-what-does). We suggest readers be aware of a human tendency to blame and structure your retrospectives to deal with this.

We found that learning reviews help contribute to a climate for learning and also impact organizational culture. It is important to note that this is true when teams conduct them to learn from their mistakes and failures and then turn these into opportunities to improve how they work. In particular, teams that leverage findings from retrospectives to implement changes to tooling, processes, or procedures see the strongest impacts.

Our analysis found that elite performers are 1.5 times more likely to consistently hold retrospectives and use them to improve their work. When we asked respondents about their most recent retrospective, we see common themes around outages, failures, performance, and deployment.

RETROSPECTIVE THEMES



Climate for Learning

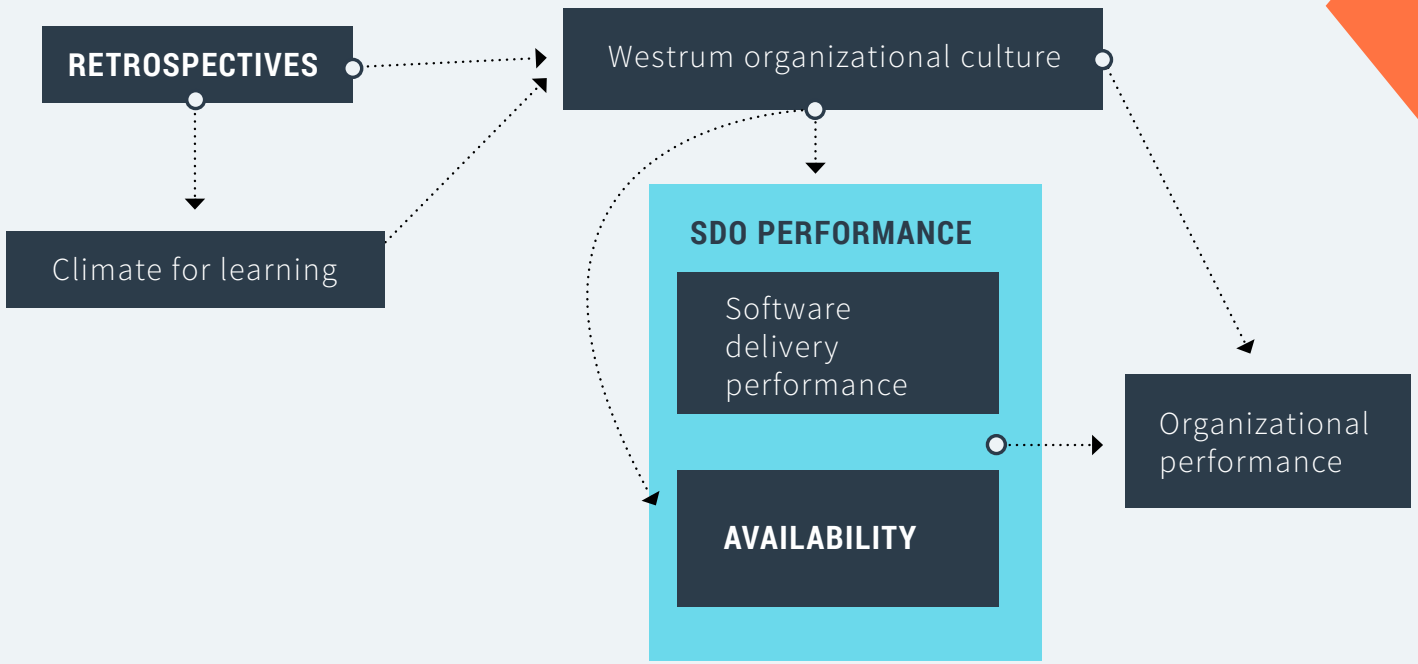
An organization with a climate for learning is one that views learning as an investment that is needed for growth as opposed to a necessary evil, undertaken only when required. Research done in other areas, such as finance, shows that a climate for learning is predictive of performance gains. Our research this year confirmed findings from early in our research program that a climate for learning positively affects organizational culture.

Equally important is where organizations expect learning to happen. Are opportunities for learning and growth provided during work time? In contrast, is learning expected but not supported, so that employees feel pressure to put in time during evenings and weekends? These environments ultimately lead to burnout and have disproportionate and negative effects on underrepresented minorities and people with non-traditional lifestyles. If you have to care for children or ailing parents, for example—the kind of work that disproportionately falls to women—creating time for education is much more difficult than for those who can devote their spare time to work and training.

Having a strong climate for learning can be a strategic advantage for teams and organizations. As we work in increasingly complex environments with changing requirements, a culture that excitedly embraces change and jumps at the opportunity to learn new things will ultimately excel. These teams thrive during change, whether that is a technology transformation, organizational change, or rapidly changing customer demands and markets.

With that in mind, how can organizations support a climate for learning? They can start by looking at the kinds of opportunities and resources that are provided to support learning. For example, hack days, internal meetups, and brown bags can be great options in addition to support and budget for formal training and attending conferences.

CLIMATE FOR LEARNING



Recall that boxes are constructs and boxes that contain boxes are second-order constructs. Arrows are predictive relationships. Constructs in bold are new this year, while others have been studied in previous years and their relationships are revalidated in this year's research.

FINAL THOUGHTS

In our fifth year of rigorous, scientific analysis, we continue to see the importance of software delivery in every kind of organization. Our unique approach delivers data to help you benchmark your own teams along with predictive analysis to help you identify key capabilities that you can apply to your own digital transformations.

We sincerely thank everyone who contributed by taking the survey. We hope our research helps inspire you and your organization as you experiment with new ways of working, and we look forward to hearing your stories.

Methodology

Our rigorous methodology goes beyond reporting raw numbers and looks at the predictive relationships between software delivery and operational performance, organizational performance, technical practices, cultural norms, and management. In this section, we describe our analysis methods, as well as how we enlisted survey respondents and how we designed our questions, models, and constructs. For more detail, we point you to Part II of our book Accelerate: The Science of Lean Software and DevOps.

We welcome questions about our research methodology at data@devops-research.com.

Research design

This study employs a cross-sectional, theory-based design. This theory-based design is known as inferential, or inferential predictive, and is one of the most common types conducted in business and technology research today. Inferential design is used when purely experimental design is not possible and field experiments are preferred—for example, in business, when data collection happens in complex organizations, not in sterile lab environments, and companies won't sacrifice profits to fit into control groups defined by the research team.

Target population and sampling method

Our target population for this survey was practitioners and leaders working in, or closely with, technology work and transformations and especially those familiar with DevOps. Because we don't have a master list of these people—we can describe them, but we don't know exactly where they are, how to find them, or how many of them exist—we used snowball sampling to obtain respondents. This means we promoted the survey via email lists, online promotions, and social media and also asked people to share the survey with their networks, growing the sample like a snowball. Our sample is likely limited to organizations and teams that are familiar with DevOps, and as such, may be doing some of it. A key to overcoming limitations in snowball sampling is to have a diverse initial sample. We accomplished this by leveraging our own contact lists as well as those of our sponsors for our initial sample, resulting in demographics and firmographics that largely match industry trends.

Creating latent constructs

We used previously validated constructs wherever possible. When we needed to create new constructs, we wrote them based on theory, definitions, and expert input. We then took additional steps to clarify intent and wording to ensure that data collected from the final survey would have a high likelihood of being reliable and valid.[29] We used Likert-type[30] questions for construct measurement, which make it possible to perform more advanced analysis.

---

[29] We used Churchill's methodology: Churchill Jr, G. A. "A paradigm for developing better measures of marketing constructs," Journal of Marketing Research 16:1, (1979), 64–73.

[30] McLeod, S. A. (2008). Likert scale. Retrieved from www.simplypsychology.org/likert-scale.html

Statistical analysis methods

Cluster analysis

We use cluster analysis to identify our software delivery performance profiles. In this approach, those in one group are statistically similar to each other and dissimilar from those in other groups, based on our performance behaviors of throughput and stability: deployment frequency, lead time, time to restore service, and change fail rate. A solution using Ward's method[31] was selected based on (a) change in fusion coefficients, (b) number of individuals in each cluster (solutions including clusters with few individuals were excluded), and (c) univariate F-statistics.[32] We used a hierarchical cluster-analysis method because it has strong explanatory power (letting us understand parent-child relationships in the clusters) and because we did not have any industry or theoretical reasons to have a predetermined number of clusters. That is, we wanted the data to determine the number of clusters we should have. Finally, our dataset was not too big (hierarchical clustering is not suitable for extremely large datasets).

Measurement model

Prior to conducting analysis, constructs were confirmed using exploratory factor analysis with principal components analysis using varimax rotation.[33] Statistical tests for convergent and divergent validity[34] and reliability[35] were confirmed using average variance extracted (AVE), correlation, cronbach's alpha,[36] and composite reliability.[37] The constructs passed these tests, therefore exhibiting good psychometric properties.

Structural equation modeling

The structured equation models (SEM) were tested using PLS analysis, which is a correlation-based SEM. We utilize PLS for our analysis for several reasons: it does not require assumptions of normality in the data, it is well suited to exploratory and incremental research, and the analysis optimizes for prediction of the dependent variable (vs. testing for model fit of the data).[38] SmartPLS 3.2.6 was used. When controlling for industry, no significant effect was found. All paths shown in the SEM figures are $p < .001$, except the following, which are $p < 0.05$: Database → Continuous Delivery, Monitoring & Observability → Continuous Delivery, Westrum → SDO Performance (in the technical practices model), and Westrum → organizational performance (in the Lean and Agile practices model).

---

[31] Ward, J. H. "Hierarchical Grouping to Optimize an Objective Function." *Journal of the American Statistical Association* 58 (1963): 236–244.

[32] Ulrich, D., and B. McKelvey. "General Organizational Classification: An Empirical Test Using the United States and Japanese Electronic Industry." *Organization Science* 1, no. 1 (1990): 99–118.

[33] Straub, D., M.-C. Boudreau, and D. Gefen. "Validation Guidelines for IS Positivist Research." *Communications of the AIS* 13 (2004): 380–427.

[34] http://www.socialresearchmethods.net/kb/convdisc.htm

[35] http://www.socialresearchmethods.net/kb/reliable.php

[36] Nunnally, J. C. *Psychometric Theory*. New York: McGraw-Hill, 1978.

[37] Chin, Wynne W. "How to Write Up and Report PLS Analyses." In: V. Esposito Vinzi, W. W. Chin, J. Henseler, and H. Wang (eds.), *Handbook of Partial Least Squares*. Berlin: Springer (2010): 655–690.

[38] These methodology considerations are supported by Chin (1998), Chin, W. W. (1998). Issues and opinions on structural equation modeling. *MIS Quarterly*, 22(2), vii-xvi. Geffen et. al (2011) Gefen, D., Straub, D. W., & Rigdon, E. E. (2011). An update and extension to SEM guidelines for administrative and social science research. *MIS Quarterly*, 35(2), iii-xiv. and Hulland (1999). Hulland, J. (1999). Use of partial least squares (PLS) in strategic management research: A review of four recent studies. *Strategic Management Journal*, 20(2), 195-204.

ACKNOWLEDGEMENTS

The authors would like to thank several people for their input and guidance on the report this year. All acknowledgements are listed alphabetically by type of contribution.

Special thanks to **George Miranda** and **Xavier Velasquez** for providing writing and analysis support for this year's survey and report. The project greatly benefitted from your involvement this year.

Thanks to **Sam Guckenheimer** and **Bryan Kirschner** for overall guidance on the report.

For topic review and input on measurement items, we thank **Silvia Botros**, **Stephanie Herr**, **Liz Fong-Jones**, **Bridget Kromhout**, **Cheryl Kwinn**, **Aruna Ravichandran**, **J. Paul Reed**, **Mary Robbins**, **Robert Reeves, Dustin Smith**, **Claire Taylor**, and **Seth Vargo**.

The authors would like to thank **Cheryl Coupé** for her careful eye and meticulous work editing this year's report.

Report layout and design by **Siobhán Doyle**.

AUTHOR BIOGRAPHIES

Dr. Nicole Forsgren is Co-founder, CEO, and Chief Scientist at DevOps Research and Assessment (DORA) and co-author of the book *Accelerate: The Science of Lean Software and DevOps*. She is best known for her work measuring the technology process and as the lead investigator on the largest DevOps studies to date. She has been a professor, sysadmin, and performance engineer. Nicole's work has been published in several peer-reviewed journals. Nicole earned her PhD in Management Information Systems from the University of Arizona, and is a Research Affiliate at Clemson University and Florida International University.

Jez Humble is co-author of *Accelerate*, *The DevOps Handbook*, *Lean Enterprise*, and the Jolt Award-winning *Continuous Delivery.* He has spent his career tinkering with code, infrastructure, and product development in companies of varying sizes across three continents, most recently working for the U.S. government at 18F. He is currently researching how to build high-performing teams at his startup, DevOps Research and Assessment, and teaching at UC Berkeley.

Gene Kim is a multiple award-winning CTO, researcher, and author. He was founder and CTO of Tripwire for 13 years and is the co-author of *The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win*, *The DevOps Handbook*, and the newly-released *Accelerate*. Since 2014, he has been the organizer of the DevOps Enterprise Summit, studying the technology transformations of large, complex organizations.

About DevOps Research and Assessment

DevOps Research and Assessment (DORA), founded by Dr. Nicole Forsgren, Jez Humble, and Gene Kim, conducts research into understanding high performance in the context of software development and the factors that predict it. DORA's research over four years and more than 30,000 data points serves as the basis for a set of evidence-based tools for evaluating and benchmarking technology organizations and identifying the key capabilities to accelerate their technology transformation journey.

Learn more at devops-research.com.

About Google Cloud

Google Cloud is the premier sponsor of the 2018 Accelerate State of DevOps Report.

Google Cloud's portfolio of products, services, and tools enable customers to modernize their operations for today's digital world. Google Cloud touches every layer of the business and includes: Google Cloud Platform with offerings that span storage, infrastructure, networking, data, analytics, and app development; machine learning tools and APIs; G Suite's collaboration and productivity tools; enterprise Maps APIs; and also Android phones, tablets, and Chromebooks for the enterprise.

Our Sponsors

We would like to thank our sponsors. In addition to Google Cloud, *Accelerate*: State of DevOps 2018: Strategies for a New Economy is made possible by a globally recognized group of companies that jointly support our commitment to publish high-quality and vendor-neutral research. Their support enables industry-standard research to help technology-enabled organizations understand how to accelerate their drive toward high performance.

Amazon Web Services

Amazon Web Services offers over 120 global cloud-based services including compute, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security, Machine Learning and enterprise applications.

CA Technologies

From planning to development to management to security, at CA we create software that fuels transformation for companies in the application economy.

CloudBees

CloudBees is powering the continuous economy by building the world's first end-to-end system that automates continuous delivery and provides the governance, visibility and insights to optimize efficiency and control new risks.

Datical

Datical's mission is to transform the way businesses build software. Our solutions deliver the automation capabilities enterprises need to remove database deployments as a barrier to delivering application innovation.

Deloitte

At Deloitte, we help our clients embrace Lean, Agile, and DevOps capabilities from strategy through operations. Our team of specialists create and deploy contextualized solutions that foster and support agility.

Electric Cloud

Electric Cloud's Adaptive Release Orchestration and DevOps Analytics platform enables organizations like E*TRADE, GM, Intel and Lockheed Martin to confidently release new applications and adapt to change on business demand.

GitLab

GitLab is a single application used by more than 100,000 organizations to enable Product, Development, QA, Security, and Operations teams to achieve a 200% faster DevOps lifecycle.

IT Revolution

IT Revolution assembles technology leaders and practitioners through publishing, events, and research. We seek to elevate the state of work, quantify the costs of suboptimal IT performance, and improve technology professionals' lives.

Microsoft

Our mission is to empower every person and every organization on the planet to achieve more. This includes the Azure cloud, DevOps tools, and Visual Studio and VS Code development environments.

PagerDuty

PagerDuty is the leading digital operations management platform for organizations. Over 10,000 enterprises and small to mid-size organizations around the world trust PagerDuty to improve digital operations.

Pivotal

Pivotal Software, Inc., combines our cloud-native platform, developer tools, and unique methodology to help the world's largest companies transform the way they build and run their most important software applications.

Redgate

The leading Microsoft SQL Server tools vendor, Redgate designs highly usable, reliable software which elegantly solves the problems developers and DBAs face every day, and helps them adopt database DevOps.

Sumo Logic

Sumo Logic is the leading cloud-native, machine data analytics platform, that delivers real-time, continuous intelligence across the application lifecycle and stack.

Tricentis

With the industry's No. 1 Continuous Testing platform, Tricentis is recognized for reinventing software testing for DevOps—transforming testing from a roadblock to a catalyst for innovation.

XebiaLabs

XebiaLabs develops enterprise-scale Continuous Delivery and DevOps software, providing companies with the visibility, automation and control to deliver software faster and with less risk.