The Control Gap Report

AI-GENERATED CODE

LaunchDarkly

2026

AI delivered on the promise of faster code generation, but it also exposed a critical gap.

While teams can now write code more quickly, they often lack the necessary control mechanisms to manage code after it's live. We partnered with independent research firm UserEvidence to survey Engineering and DevOps professionals about this control gap to better understand what separates high performers from everyone else.

Here's what we found from respondents.

Three challenges keep most teams from reliable and confident delivery:

1. **Teams automate code creation but not control.**

AI increased code velocity for

94% 

but decreased confidence for nearly as many

91% 

2. **Pressure to deliver quickly forces compromise.**

81%

shipped with unresolved risks within the past six months, creating a cycle where 38% spend more than 25% of their time resolving incidents.

3. **Having capabilities doesn't mean using them effectively.**

99%

have runtime guardrails in place, but approximately 7 in 10 still roll back at least weekly.

AI promised to make software development faster.

And it did; just not in the way engineers hoped it would. Code velocity is increasing, but so are production incidents, emergency rollbacks, and recovery times.

Any engineering team will tell you the same thing: **Shipping has never been faster, and it's never been higher stakes.**

Automation stopped at code generation.

Organizations have failed to extend it to the control mechanisms necessary to release code safely to their users: progressive rollouts, instant rollbacks, and targeted releases. Even teams that have fully integrated AI tools report strain.

77%

still struggle with the pace and risk of AI-accelerated development

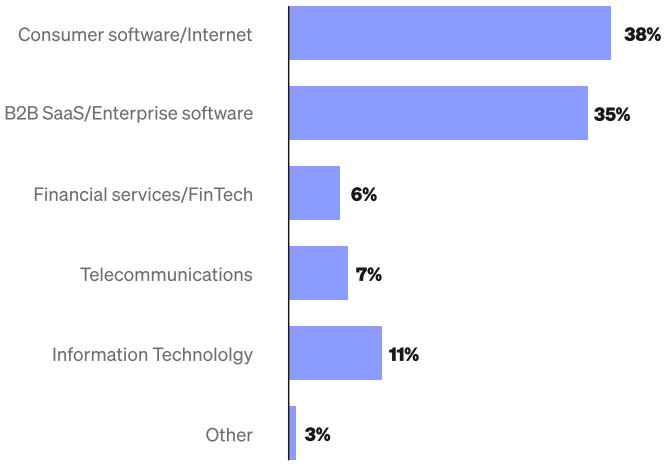
Teams can generate features in minutes, but still need hours to recover when something breaks.

This report explores the reasons for this control gap, investigating how teams deploy AI-generated code, where existing tools fall short, and what sets apart the teams that have found a solution.

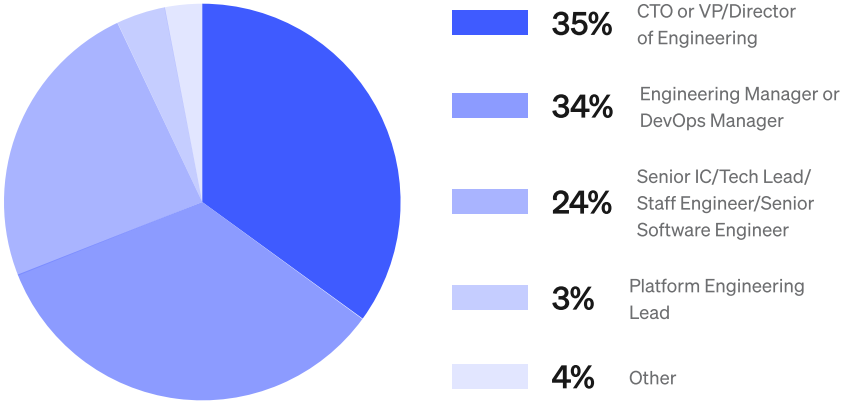
Demographics

To understand the current state of AI coding adoption vs. code release realities, we partnered with independent research firm UserEvidence to survey **767 engineering and DevOps leaders** from mid-sized to enterprise organizations across various industries.

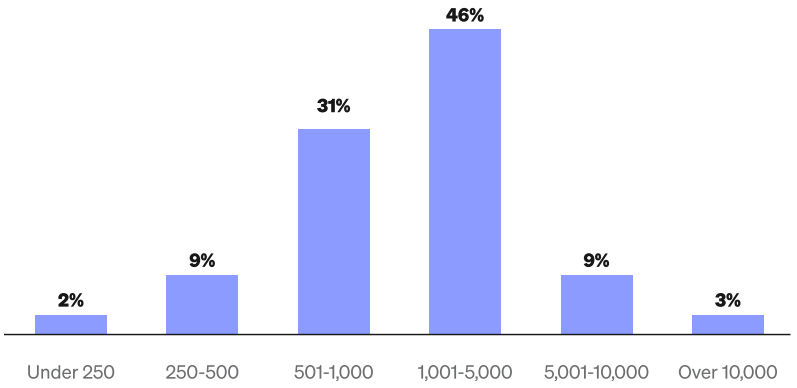
Which industry best describes your organization?



Which best describes your role?



How many employees does your organization have globally?



Key takeaways

The data reveals a clear problem: AI has accelerated the pace of development, but it has also exposed a control gap most teams are struggling to close.

AI has accelerated development but eroded confidence.

94% report faster velocity, yet 91% are more cautious about deploying to production.

Incidents persist in spite of runtime management capabilities.

99% have runtime guardrails or safety mechanisms in place, but 69% still roll back or perform hotfixes on a weekly basis.

Pressure subverts best practices.

Within the past six months, 82% of teams have shipped changes with unresolved risks because they were under pressure to deliver.

Mid-size orgs struggle most.

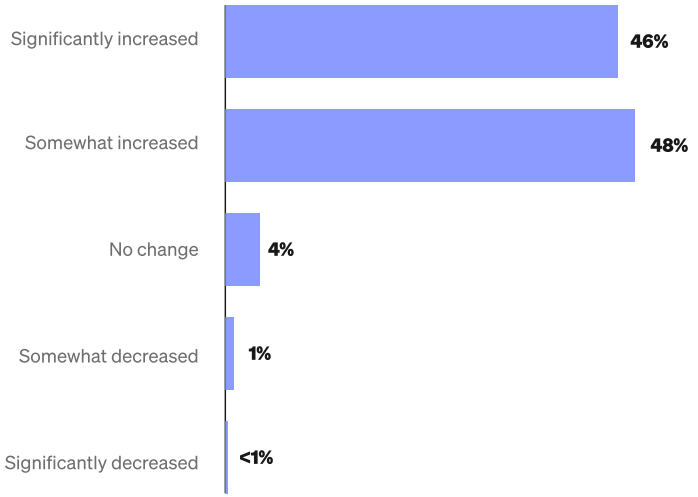
Organizations with 11-20 teams deploy fastest (75% daily), but only 22% experience incidents monthly or less. Scale without coordination amplifies chaos; the fastest teams are also the most exposed.

Only 15% have figured it out.

Only 15% of dev teams deploy daily while keeping incidents to monthly or less. LaunchDarkly users are 2.2x more likely to achieve this balance.

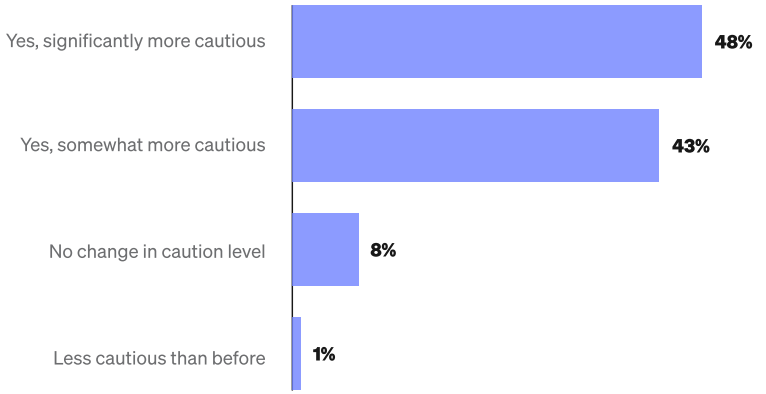
Faster code, less confidence

Has AI accelerated the amount of code or change velocity?



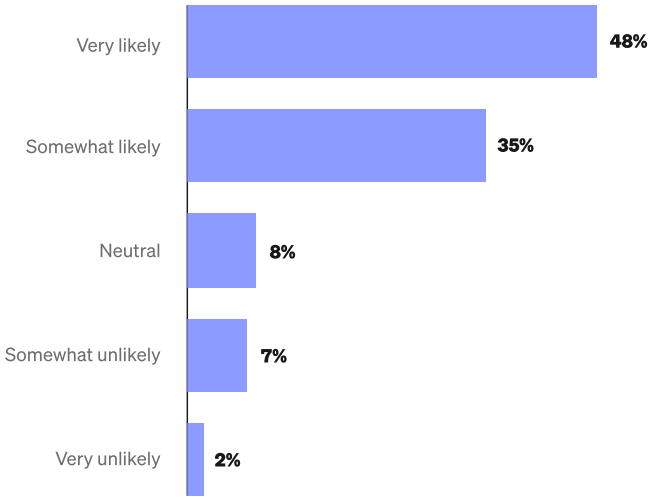
There's no question that **AI has accelerated code** velocity, as reported by 94% of respondents.

Has AI-accelerated development made your team more cautious about pushing changes to production?



91% say their teams have become more cautious about pushing these AI-accelerated changes to production. **So where's the disconnect?**

How likely is it that a release containing AI-generated code will cause production issues?



This caution appears justified: 91% expect that a release containing AI-generated code will cause production issues as much or more often than hand-generated code.

The cost of this imbalance between AI-accelerated code generation and effective control methods often shows up in production—in late nights, emergency patches, and more engineering time spent resolving incidents than building features.

What is the control gap?

It's the distance between how quickly AI enables us to generate and ship code, and how safely we can deploy it in production.

Speed — Safety

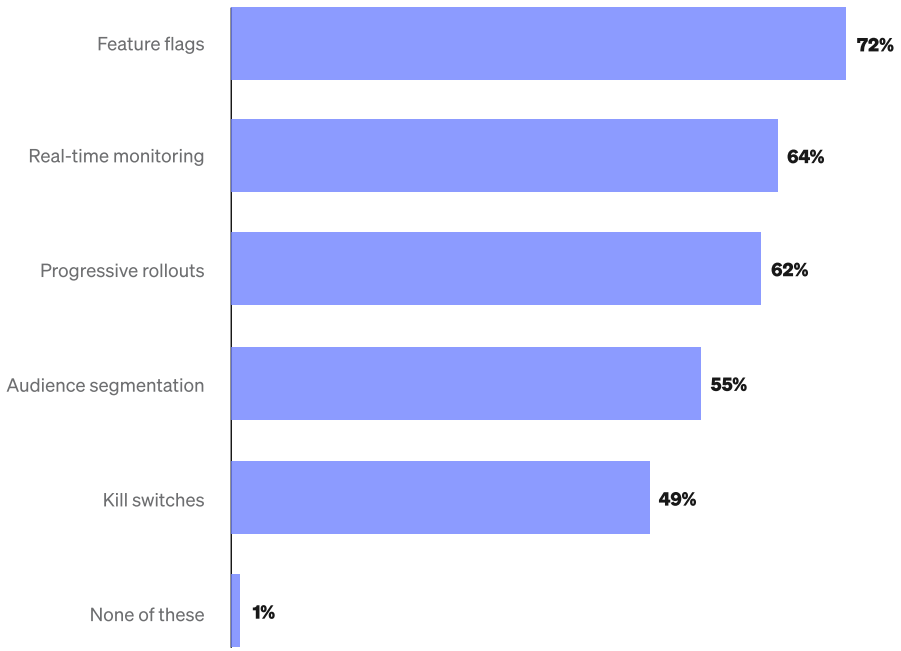
Capabilities ≠ outcomes

On paper, teams look well-equipped. In practice, they're still in reactive mode.

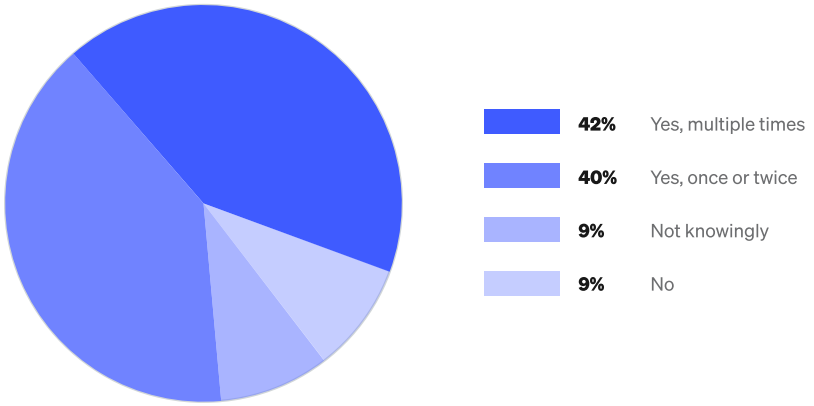
Most teams have deployed runtime guardrails, safety mechanisms, and monitoring tools.

Theoretically, teams have the resources necessary to maintain quality and prevent incidents. **But their day-to-day reality suggests otherwise.**

Which runtime guardrails or safety mechanisms exist post-deploy in your team?



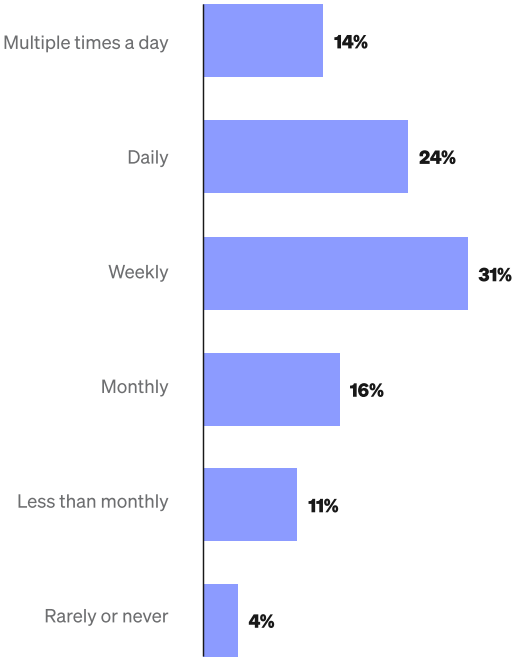
In the past 6 months, has your team knowingly shipped changes with unresolved risks due to deadlines or pressure?



81%

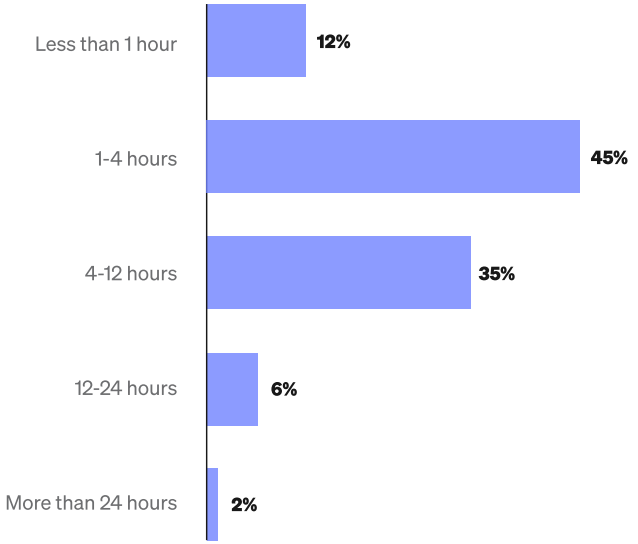
of teams have shipped changes with unresolved risks because they were under pressure to deliver on deadlines; only slightly more than **9% haven't.**

How often are teams rolling back or hotfixing production issues?



Nearly **seven in ten** teams have to roll back or hotfix production issues at least weekly, with only slightly more than 15% keeping it to less than monthly or ever. And these issues take time and effort to fix.

How long does recovery typically take after a production issue?



12%

of respondents can recover from a production issue **within an hour.**

Recent research shows that even low-business-impact outages can cost **large enterprises up to US $1.3 million per hour.**

80%

of incidents take 1–12 hours to fix, meaning even a modest monthly average of two hours per incident can add up to **tens of millions a year in lost productivity.**

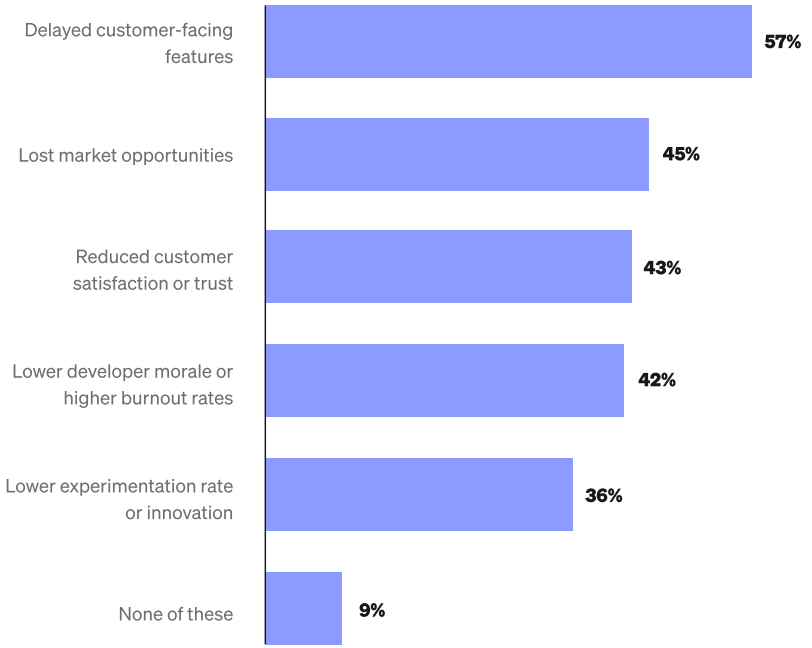
For example, mid-sized enterprises feel the financial strain of outages at an average cost of **upwards of $300,000** per hour.

$300k / hr

Most teams have the necessary capabilities (feature flags, rollouts, monitoring), but they're not using those for control. Simply having guardrails doesn't prevent incidents; using them effectively does.

The cost of the control gap

What opportunities is your team missing due to slower deployment or outdated workflows?



The opportunity costs of poorly controlled deployments are steep: More than half of respondents are delaying customer-facing features.

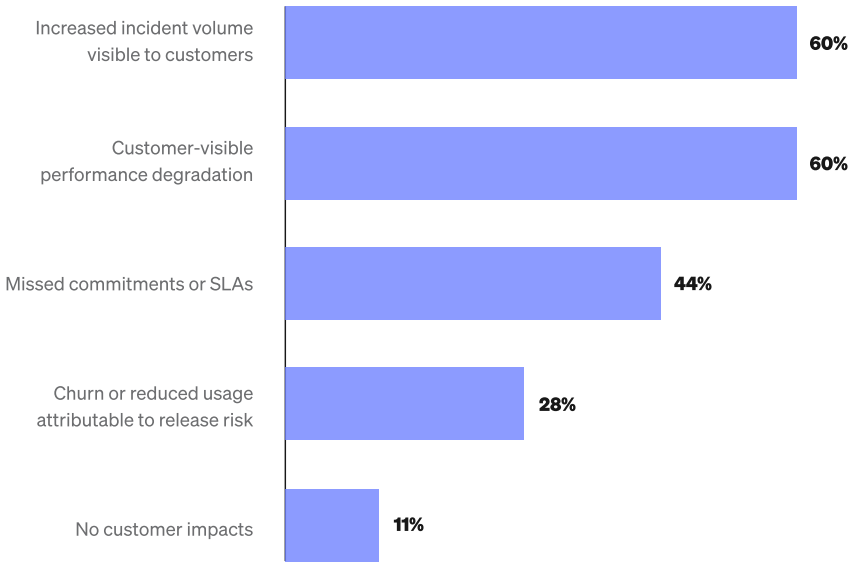
45% report having lost market opportunities

36% report they're not experimenting or innovating as much

42% report lower morale and higher burnout

The impact doesn't stop with engineering teams; customers feel it, too.

In the past 12 months, which customer impacts (if any) have you experienced due to risky or delayed releases?



Nearly nine in ten respondents say release risk has directly impacted customers in at least one way, with

60%

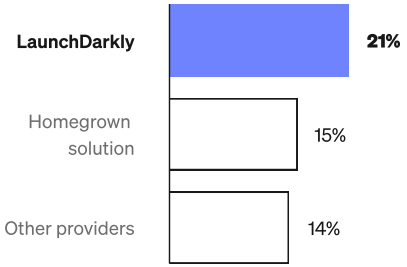
of that impact manifesting as either **customer-visible incidents or performance degradation.**

The tools exist. The control doesn't.

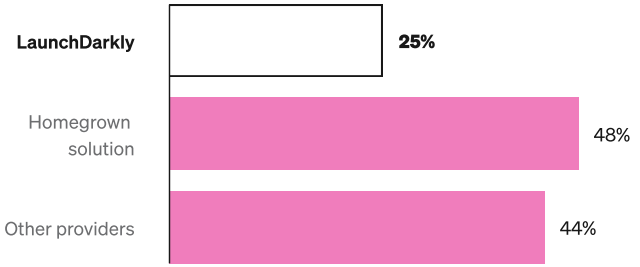
And that gap is costing teams in lost productivity, delayed features, and burned-out engineers. This is where LaunchDarkly helps users diverge, translating capability into consistent outcomes.

The LaunchDarkly difference

Percentage of teams with **fewer production incidents** (monthly or less)



Percentage of teams with **more frequent production incidents** (daily or more)



> **LaunchDarkly allows us to experiment and validate features without putting our customers' experience at risk, and lets our team deliver smaller deploys confidently and more often.**

Tech Lead, Large Enterprise Internet Software & Services Company

LaunchDarkly customers are much less likely to have to roll back or hotfix production issues than those using other providers or a homegrown solution.

LaunchDarkly users are:

48% less likely to experience daily production incidents **compared to homegrown solutions.**

43% less likely **compared to other providers.**

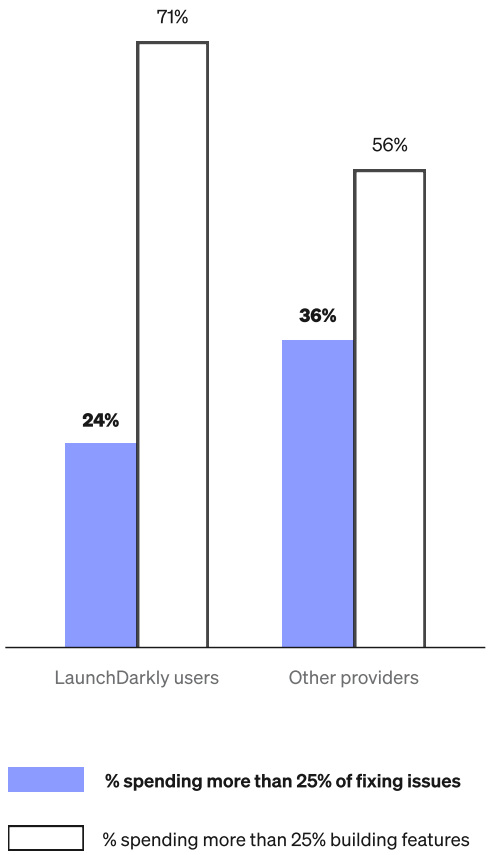
And when they do experience production issues, LaunchDarkly customers recover much more quickly.

66% of LaunchDarkly users **recover from issues in under four hours.**

vs.

54% using other tools—about a 12-percentage-point advantage.

Issue remediation vs. feature building



LaunchDarkly users spend less time resolving issues.

Approximately 24% spend more than a quarter of their time on incidents vs. 36% using other tools. That extra time shows up in feature velocity:

71% of LaunchDarkly users spend more than a quarter of their time building features.

vs.

56% with other providers.

Slicing these numbers a different way, the average LaunchDarkly customer spends about **4 percentage points less time fixing issues** than those using other providers.

With the average developer salary at just over $131,000, that savings comes out to around:

$ 5k per year per dev

x **20** devs per team

that's around

$ 100k

in productivity gains annually.

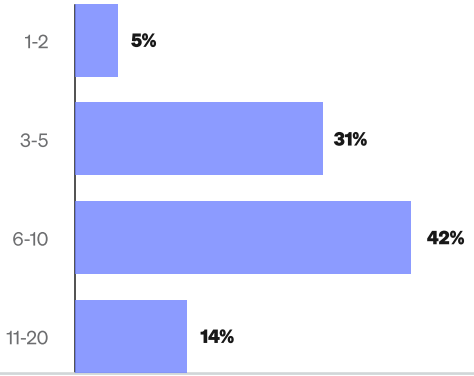
Governing AI-written code at scale

As organizations grow from small to mid-sized, performance often degrades before it improves. The largest organizations rebound, but what's happening in the middle? As organizations scale, speed peaks before safety does.

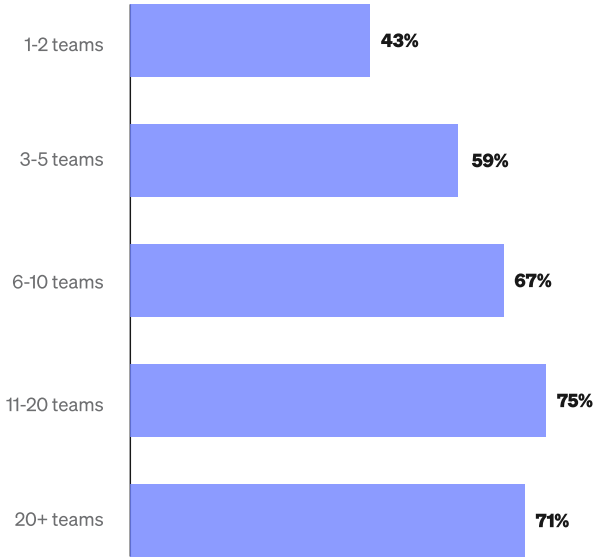
Team size affects deployment frequency.

73% reported having 3–10 teams deploying code to production regularly.

How many engineering teams in your organization regularly deploy code to production?



Daily deployment rates among team sizes

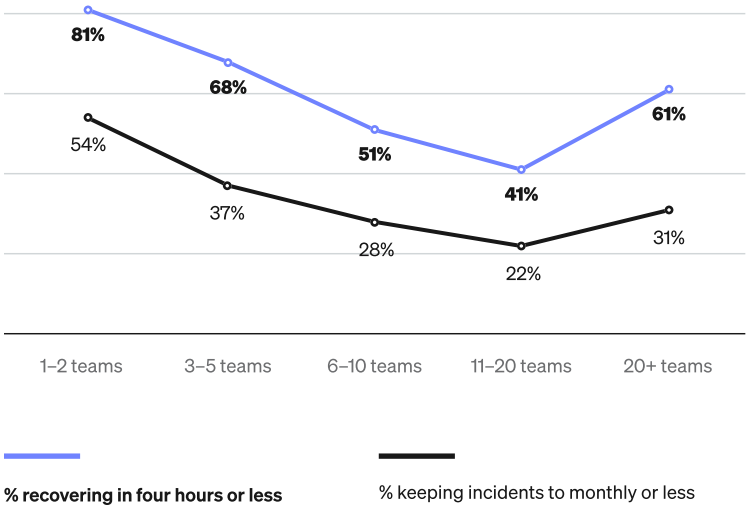


Deployment velocity trends upwards with team count. Organizations with 11–20 teams are nearly twice as likely to deploy daily (75%) compared to those with just one or two teams (43%). **But velocity doesn't mean stability.**

Stuck between scrappy and scalable

Incident management follows a U-shaped curve.

Strong incident management by number of teams



Smaller organizations can easily coordinate across one or two teams, which means fewer incidents and faster recovery.

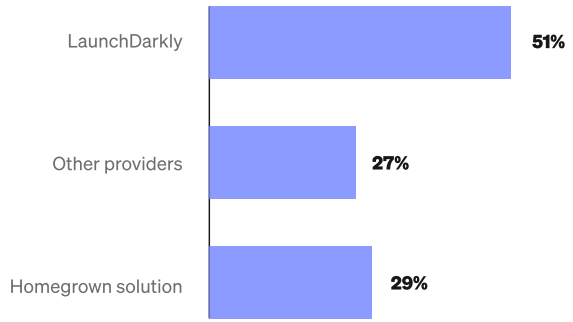
Mid-size orgs with 11–20 teams are struggling. This is often where tooling fragmentation and inconsistent rollout practices emerge. These orgs are too big for informal communication but too small to justify extensive infrastructure. They're deploying at the highest velocity (75% daily) but coordinating incident response across multiple teams with different practices and tools. The result: more incidents and slower recovery.

Scale forces discipline. When organizations have more than 20 teams, informal communication doesn't cut it. Large organizations need clear processes, defined roles, and systematic approaches. That structure yields better outcomes.

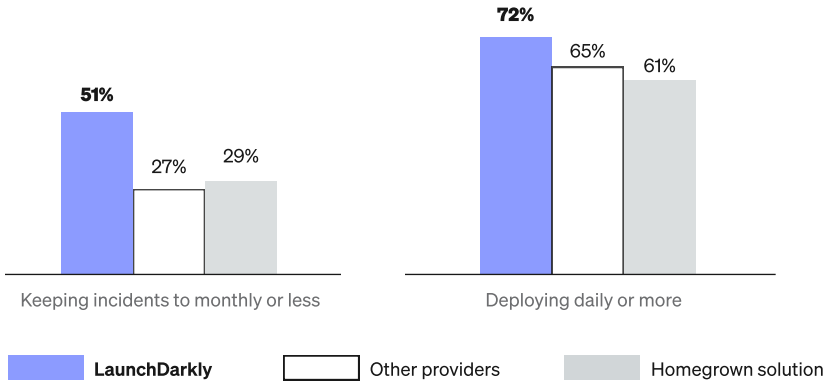
The LaunchDarkly difference

LaunchDarkly users skip the growing pains of mid-sized teams and standardize right from the start. They're also able to ship more frequently and keep incidents lower.

Percentage of those who standardize release patterns across teams:



LaunchDarkly users deploy faster with fewer incidents.



LaunchDarkly provides the coordination and guardrails that usually only come at scale. Teams can deploy quickly while still keeping incident frequency low, with the ability to roll back instantly and catch issues before they escalate.

> **Before fully adopting LaunchDarkly, our engineers were spending a lot more time babysitting releases, making manual changes, and watching metrics. Now, they can focus on building the product rather than constantly monitoring it.**

Adam Kadzban, Principal Engineer, Relativity

Runtime control in the age of AI

What high-performing teams do differently

Only 15% of teams achieve the goal state: deploying daily or more and keeping incidents to monthly or less.

LaunchDarkly users are

2.2x

more likely to reach this elite cohort.

They've solved the AI delivery paradox of maintaining speed without sacrificing safety.

They've embedded runtime control into every release, treating safety as a default.

Where does your team fall?

Fast but fragile	☐ Ship often and hope nothing breaks ☐ Fix often ☐ Rollback means hours lost, large team needed
Slow but safe	☐ Long, time-consuming review cycles ☐ Rare incidents ☐ Rollback takes hours of structured process
Fast and safe	☐ Deploy often and confidently ☐ Control built into every release ☐ Effective tools stop problems instantly

The missing runtime control plane for features

Every part of the stack has a control plane:

Infrastructure

Kubernetes orchestrates containers.

Services

APIs and SLOs control interactions and uptime.

Applications

CI/CD and APM control releases and performance.

Features

Nothing.

This is the gap.

Features are what customers actually experience, yet they're often managed as side effects of deployment.

Engineers ship code and hope it works.

That approach breaks down entirely in the era of AI. **AI makes feature-level control essential for shipping safely:**

Progressive rollouts to limit blast radius.

Instant kill switches to stop problems immediately.

Audience targeting to test with specific users.

Real-time monitoring to see what's actually happening.

According to Leonid Igolnik, former EVP Engineering at Clari, very few teams achieve the ability to cleanly roll back after code is live.

When they had that capability at their disposal, they reduced their change failure rate by more than

70%.

Leonid's team also reduced their MTTR drastically by simply being able to turn features off and reduce the blast radius.

Without this layer, teams either slow down (adding gates, extending reviews) or speed up unsafely (hoping nothing breaks).

Closing the gap requires runtime control: The ability to progressively roll out, instantly kill, target users, and observe impact in real time.

Looking ahead: Putting this change into practice

	Before	After
Release velocity	Daily or weekly deployments, semi-automated but inconsistent.	Fully automated and on demand, teams can release at any time.
Recovery cost	Manual, slow recovery (taking 4+ hours) with revenue impact.	Instant, proactive recovery. Failures accelerate innovation.
Release safety	Limited segmentation despite having progressive rollout capabilities.	Real-time adaptive targeting and auto-rollbacks.
Release impact	Post-release metrics are reviewed after deployments.	Continuous feedback shapes roadmap and future releases.

This is how you operationalize AI to stay better aligned with customers, ship more releases at a faster pace, and reduce risk without burning out your team. The organizations that are closing the gap aren't slowing down; they're controlling what happens after code ships.

About LaunchDarkly

LaunchDarkly is the runtime control plane for software features.

The platform gives engineering teams the confidence to innovate quickly without compromising reliability or customer trust by providing visibility and control where it matters most—in the features that reach customers. As AI and automation accelerate software delivery, LaunchDarkly helps ensure that speed doesn't come at the cost of safety or stability. With LaunchDarkly, teams can progressively release new capabilities, observe performance in real time, and remediate issues instantly before users are impacted. They can also learn quickly and iterate continuously to optimize feature performance, enhance user experience, and drive better business outcomes.

More than 5,500 organizations, including a quarter of the Fortune 500, rely on LaunchDarkly to deliver exceptional digital experiences and turn every software release into a competitive advantage.

About UserEvidence

UserEvidence is a software company and independent research partner that helps B2B technology companies produce original research content from practitioners in their industry.

All research completed by UserEvidence is verified and authentic according to their research principles: identity verification, significance and representation, quality and independence, and transparency. All UserEvidence research is based on real user feedback without interference, bias, or spin from our clients.

UserEvidence Research Principles

These principles guide all research efforts at UserEvidence—whether working with a vendor's users for our Customer Evidence offering, or industry practitioners in a specific field for our Research Content offering. The goal of these principles is to give buyers trust and confidence that you are viewing authentic and verified research based on real user feedback, without interference, bias, and spin from the vendor.

1. Identity Verification

In every study we conduct, UserEvidence independently verifies that a participant in our research study is a real user of a vendor (in the case of Customer Evidence) or an industry practitioner (in the case of Research Content). We use a variety of human and algorithmic verification mechanisms, including corporate email domain verification (i.e., so a vendor can't just create 17 Gmail addresses that all give positive reviews), and pattern-based bot and AI deflection.

2. Significance and Representation

UserEvidence believes trust is built by showing an honest and complete representation of the success (or lack thereof) of users. We pursue statistical significance in our research, and substantiate our findings with a large and representative set of user responses to create more confidence in our analysis. We aim to canvas a diverse swatch of users across industries, seniorities, personas—to provide the whole picture of usage, and allow buyers to find relevant data from other users in their segment, not just a handful of vendor-curated happy customers.

3. Quality and Independence

UserEvidence is committed to producing quality and independent research at all times. This starts at the beginning of the research process with survey and questionnaire design to drive accurate and substantive responses. We aim to reduce bias in our study design, and use large sample sizes of respondents where possible. While UserEvidence is compensated by the vendor for conducting the research, trust is our business and our priority, and we do not allow vendors to change, influence, or misrepresent the results (even if they are unfavorable) at any time.

4. Transparency

We believe research should not be done in a black box. For transparency, all UserEvidence research includes the statistical N (number of respondents), and buyers can explore the underlying blinded (de-identified) raw data and responses associated with any statistic, chart, or study. UserEvidence provides clear citation guidelines for clients when leveraging research that includes guidelines on sharing research methodology and sample size.