International Journal of Sustainable Development Through AI, ML and IoT
Volume 4 | Issue2 | 2025
https://ijsdai.com/index.php/IJSDAI/index
ISSN (Online): 2584-0827

DevOps: Practices, Metrics, Security, and Emerging Trends

Mallikarjun Bellundagi[1]
Solution Architect, USA
Corresponding author: ***Mallikarjun Bellundagi***

ARTICLE INFO
Received: 31 May 2025
Revised: 20 Nov 2025
Accepted: 31 Dec 2025

ABSTRACT

DevOps, the set of practices and cultural principles that unify software development and IT operations, has matured from a grassroots movement into a formally measured engineering discipline. This paper surveys the current state of DevOps: its continuous lifecycle and toolchain, the DORA (DevOps Research and Assessment) metrics used to benchmark software delivery performance, the rise of platform engineering and internal developer platforms, the adoption of GitOps for declarative infrastructure management, and the integration of security into the delivery pipeline through DevSecOps. It reviews the growing role of artificial intelligence across the DevOps lifecycle, from AI-assisted coding to predictive pipeline automation, and the emerging evidence that AI adoption is decoupling delivery speed from delivery stability. The paper includes three original figures illustrating the DevOps lifecycle, a DevSecOps-integrated CI/CD pipeline, and a comparison of DORA performance tiers, and closes with a discussion of open challenges, including tool sprawl, the platform engineering skills gap, and the measurement difficulties introduced by AI-assisted development

1. Introduction

DevOps emerged in the early 2010s as a response to the friction between software development teams, who are incentivized to ship change quickly, and operations teams, who are incentivized to keep systems stable. Rather than treating these as opposing goals, DevOps reframes them as a single, continuous lifecycle in which development, testing, delivery, and operations are integrated, automated, and measured together. A decade and a half later, DevOps is no longer a niche practice: cloud-native delivery, containerization, and automated pipelines are now the default way software reaches production in most organizations.

What has changed more recently is the degree to which DevOps has become quantified and productized. The DORA research program has turned software delivery performance into a small set of well-validated metrics that are now used to benchmark entire organizations. Platform engineering has turned internal tooling into a formal discipline, with dedicated teams building internal developer platforms (IDPs) as products in their own right. And generative AI has begun reshaping the pipeline itself, both by accelerating individual developer output and by complicating the very metrics used to judge whether that acceleration is translating into more reliable software. This paper surveys each of these developments in turn.

2. The DevOps Lifecycle and Toolchain

The DevOps lifecycle is typically visualized as a continuous loop spanning eight stages: Plan, Code, Build, Test, Release, Deploy, Operate, and Monitor, split roughly between development-facing and operations-facing activities, with

feedback from monitoring flowing back into planning. Figure 1 illustrates this cycle. The loop has no natural start or end point in a mature DevOps organization; instead, telemetry gathered in the Monitor stage becomes input to the next planning cycle, closing the feedback loop that distinguishes DevOps from earlier, more linear software delivery models.

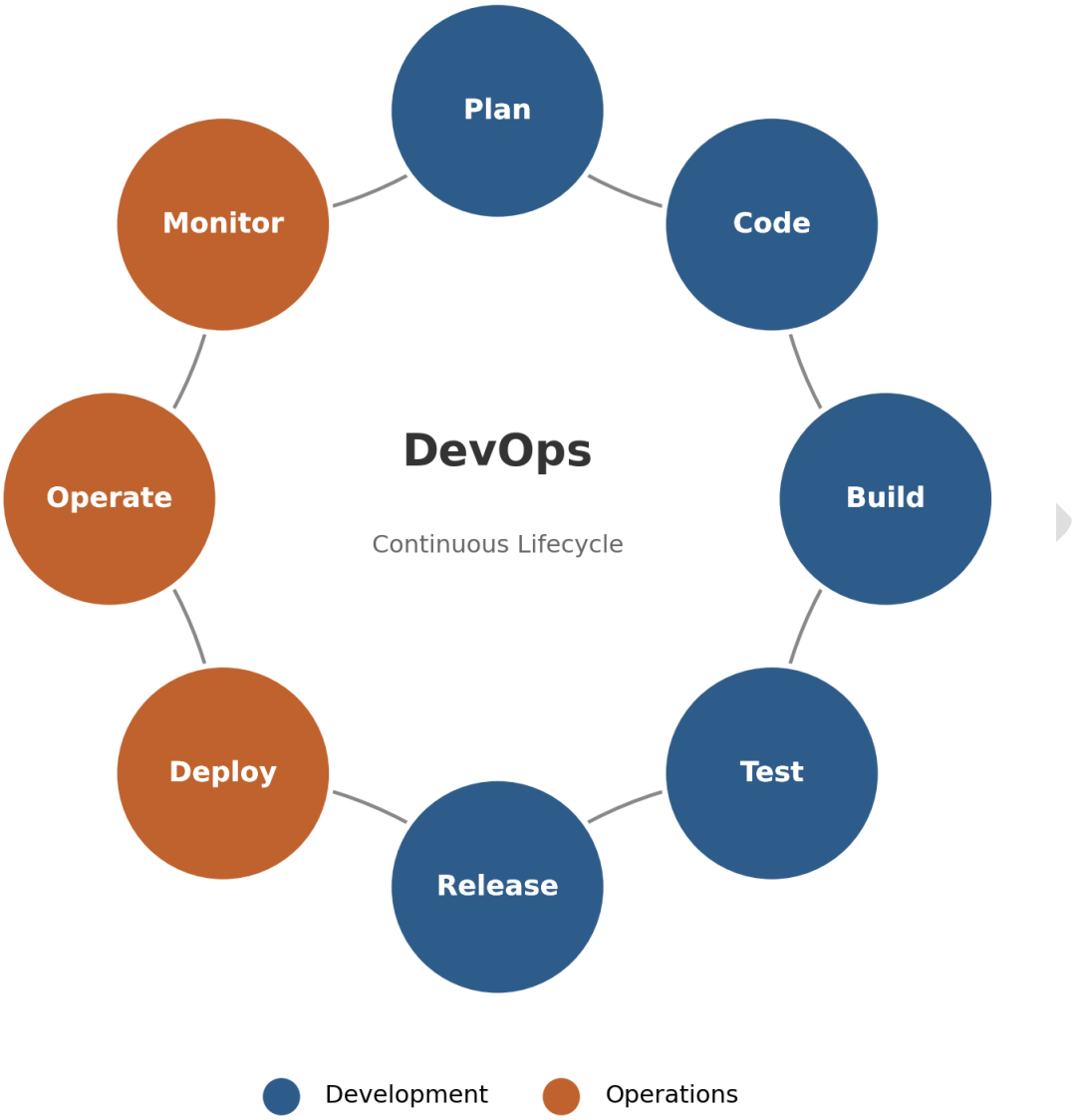

Figure 1. The continuous DevOps lifecycle, spanning development-facing stages (blue) and operations-facing stages (orange).

In practice, this lifecycle is implemented through a toolchain: version control systems, CI/CD orchestration platforms, container runtimes and orchestrators such as Kubernetes, infrastructure-as-code frameworks, and observability platforms. A recurring finding in recent industry research is that more tooling does not straightforwardly improve outcomes: teams using fewer than five DevOps tools have been reported to deploy in under an hour roughly twice as often as teams using more than twenty tools, suggesting that toolchain consolidation, not proliferation, is associated with faster delivery.

3. Measuring Delivery Performance: The DORA Metrics

The DORA (DevOps Research and Assessment) research program, now part of Google Cloud, has spent over a decade studying what distinguishes high-performing software organizations. Its core contribution is a small set of metrics, deployment frequency, lead time for changes, change failure rate, and time to restore service, that together capture both the throughput and the stability of a software delivery process. Historically, DORA has grouped organizations into four performance tiers (Elite, High, Medium, and Low) based on these metrics; the 2025 report introduced a more granular, cluster-based model with seven team archetypes, though the original four-tier framing remains widely cited in industry reporting.

The gap between tiers is substantial. Elite performers have been reported to deploy roughly 182 times more frequently

than low performers, while low-performing teams can take between one week and a full month to recover from a failed deployment, compared to under an hour for elite teams. Yet elite status remains rare: only around 19% of surveyed teams qualified as elite performers in the most recent widely cited benchmark, while the low-performing tier has reportedly grown year over year. Table 1 and Figure 2 summarize these gaps.

Table 1. DORA metric comparison between elite and low-performing teams.

Metric	Elite performers	Low performers	Approx. gap
Deployment frequency	Multiple times per day	Fewer than once per month	~182x more frequent
Lead time for changes	Less than one hour	One to six months	Orders of magnitude faster
Change failure rate	~5%	~40%	~8x lower failure rate
Time to restore service	Under one hour	One week to one month	Orders of magnitude faster

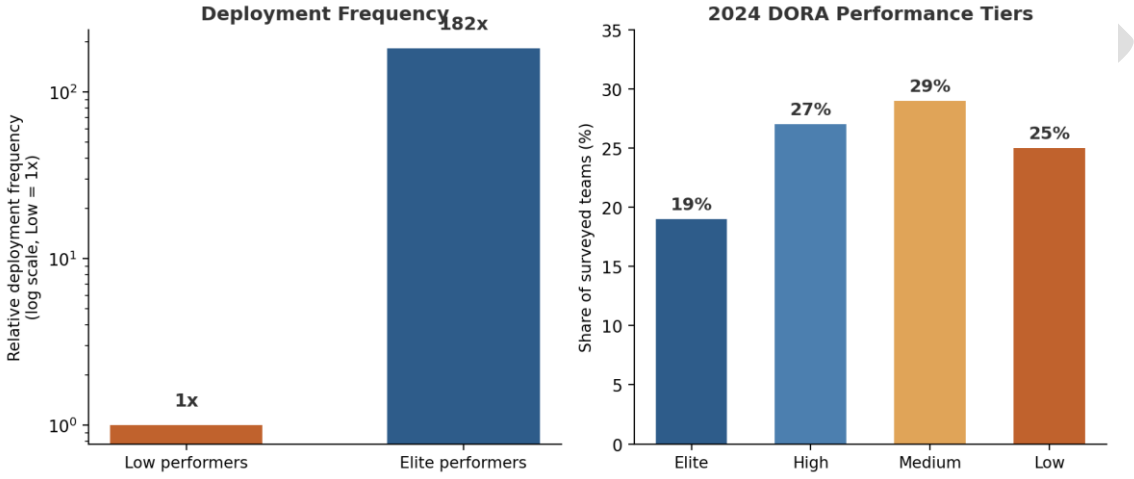

Figure 2. Left: relative deployment frequency of elite versus low performers (log scale). Right: approximate distribution of teams across 2024 DORA performance tiers.

A notable recent finding complicates this picture: as organizations adopt AI coding assistants, throughput and stability, which have historically moved together, appear to be decoupling. Increased AI adoption has been associated with improvements in individual developer effectiveness and code quality, but also with increased software delivery instability, meaning that faster code production does not automatically translate into a lower change failure rate. This has prompted DORA researchers to describe the AI-era relationship between the four metrics as a distinct and expected phase of transition rather than a simple failure signal.

4. Platform Engineering and Internal Developer Platforms

Platform engineering has emerged as a discipline distinct from, but closely related to, DevOps: rather than expecting every product team to independently master Kubernetes, CI/CD configuration, and cloud infrastructure, platform teams build an internal developer platform (IDP) that packages these capabilities as a self-service product for other engineers. Adoption has moved quickly from early-adopter status to near universal practice: by 2025, a large majority of organizations reported using at least one internal developer platform, and a majority had established dedicated platform engineering teams. Analyst projections suggest a large share of major software engineering organizations will have formal platform engineering teams in place by 2026.

The correlation between platform investment and delivery performance appears strong in recent survey data: organizations characterized as cloud-native "innovators" are substantially more likely to use GitOps workflows and to achieve elite DORA performance than organizations earlier in their cloud-native adoption journey, suggesting that platform capability is not merely correlated with, but may be a meaningful driver of, high-performing software delivery.

5. GitOps: Declarative, Git-Centered Operations

GitOps extends infrastructure-as-code by using a Git repository as the single source of truth for both application and

infrastructure state, with automated controllers (such as ArgoCD or Flux) continuously reconciling the live environment to match what is declared in Git. This gives every change a full audit trail, and rollbacks become a matter of reverting a Git commit rather than manually undoing a sequence of imperative operational steps. Adoption has grown quickly: roughly two-thirds of surveyed organizations report having adopted GitOps, with the large majority of adopters reporting improved infrastructure reliability and faster rollback times as a direct result.

6. DevSecOps: Integrating Security into the Pipeline

DevSecOps embeds security testing and controls directly into the CI/CD pipeline rather than treating security review as a separate, late-stage gate. The guiding principle, often summarized as shift-left security, is to surface vulnerabilities as early in the development process as possible, ideally in the developer's own editor, so that remediation cost stays low. In practice, this means layering several categories of automated scanning across the pipeline: static application security testing (SAST) and secret scanning during coding, software composition analysis (SCA) and software bill of materials (SBOM) generation during build, dynamic application security testing (DAST) and infrastructure-as-code policy checks during test, and artifact signing and runtime threat detection at release and deployment. Figure 3 illustrates how these security gates map onto the standard CI/CD pipeline.

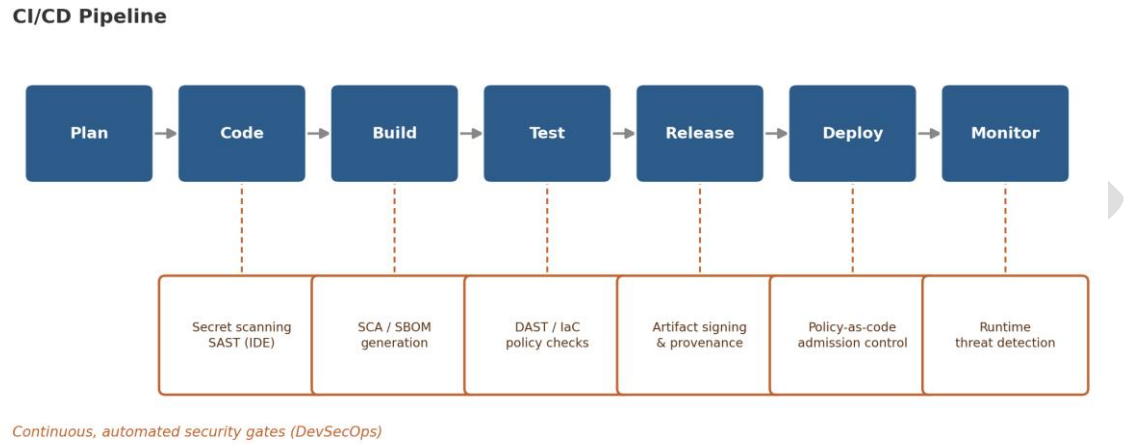

Figure 3. A CI/CD pipeline with DevSecOps security gates layered across each stage.

Despite growing urgency, driven in part by regulation such as the EU Cyber Resilience Act and by a wave of software supply chain attacks, adoption remains uneven. Only a minority of organizations report actively practicing DevSecOps as an integrated discipline today, and even among adopters, security scanning is sometimes treated as a compliance checkbox rather than a genuine engineering practice. Visibility into open-source dependencies remains a particular weak point: a widely cited industry survey found that only about one in five organizations reported full visibility into their own open-source components, even as the large majority planned to expand their use of SBOMs over the following year.

7. Artificial Intelligence Across the DevOps Lifecycle

AI has moved from an experimental add-on to a mainstream component of the DevOps toolchain. A large majority of DevOps teams report having integrated AI into their CI/CD workflows, shifting pipelines from passive dashboards toward more predictive, automated responses, such as AI-assisted anomaly detection, automated root-cause analysis, and AI-suggested remediation for failed builds. AI coding assistance has also been shown to measurably speed up individual developer task completion.

As discussed in Section 3, however, this acceleration has not been uniformly positive for delivery stability, and engineering leaders are increasingly having to reconcile faster code production with unchanged or worsening change failure rates. This has elevated observability and monitoring, historically a supporting capability, into a more central role: mature observability practices have been associated with substantial reductions in mean time to resolve incidents, providing a partial counterbalance to the instability introduced by faster, AI-assisted code production.

8. Open Challenges

8.1 Tool Sprawl and Diminishing Returns

As noted in Section 2, the accumulation of DevOps tooling shows a clear pattern of diminishing, and eventually negative, returns; consolidating toolchains rather than expanding them is increasingly viewed as a precondition for improving delivery speed, not a nice-to-have simplification.

8.2 The Platform Engineering Skills and Adoption Gap

Despite the strong reported correlation between platform engineering and delivery performance, meaningful gaps remain: a notable share of organizations still operate without any internal developer platform, and many DevOps and

DevSecOps teams report insufficient in-house skills to build and operate these platforms effectively, a gap made more acute by the cost and time required for specialized training and certification.

8.3 Measuring Performance in the AI Era

The decoupling of throughput and stability under AI-assisted development poses a genuine measurement problem: the DORA metrics were designed for a world in which faster code production and delivery stability moved together, and engineering leaders now need to interpret rising deployment frequency alongside rising change failure rates as two sides of the same transition rather than treating either metric in isolation.

8.4 Security Adoption Lag

Even as software supply chain attacks and regulatory pressure increase, DevSecOps adoption continues to lag DevOps adoption more broadly, and the gap between organizations that treat security scanning as a genuine engineering discipline and those that treat it as a checkbox exercise remains a key differentiator in real-world security outcomes.

9. Conclusion

DevOps has evolved from a cultural philosophy into a measurable, productized engineering discipline, with DORA metrics providing a common benchmarking language, platform engineering formalizing internal tooling as a product, and GitOps and DevSecOps embedding reliability and security directly into the delivery pipeline. At the same time, the field's center of gravity is shifting again as AI reshapes both how code is written and how delivery performance should be measured. The organizations most likely to benefit from this shift appear to be those that treat platform investment, security integration, and AI adoption as a coordinated operating model rather than as three independent initiatives, since the evidence reviewed here suggests these capabilities compound: strong platforms correlate with better DORA outcomes, and mature observability practices help absorb the instability introduced by faster, AI-assisted code production.

References

Axify. (2025). State of DevOps report in 2025: Lessons for engineering leaders. Retrieved from https://axify.io/blog/state-of-devops

Cortex. (2025). Turning the 2024 State of DevOps into your 2025 playbook for DevOps excellence. Retrieved from https://www.cortex.io/post/2025-playbook-for-devops-excellence

Cybersecurity News. (2025). 7 DevSecOps trends to watch in 2025. Retrieved from https://cybersecuritynews.com/7-devsecops-trends-to-watch-in-2025/amp/

DEVOPSdigest. (2025). 2025 DevSecOps predictions - Part 1. Retrieved from https://www.devopsdigest.com/2025-devsecops-predictions-part-1

DORA. (2025). Capabilities: Platform engineering. Google Cloud. Retrieved from https://dora.dev/capabilities/platform-engineering/

DX. (2025). DORA metrics: The complete guide to measuring DevOps performance in the AI era. Retrieved from https://getdx.com/blog/dora-metrics/

Deviniti. (2025). 40 DevOps stats for 2026: DORA, AI, Atlassian. Retrieved from https://deviniti.com/blog/leadership-teamwork/40-devops-stats-for-2026/

Frugal Testing. (2025). Boost CI/CD security: Best DevSecOps tools for 2025. Retrieved from https://www.frugaltesting.com/blog/boost-ci-cd-security-best-devsecops-tools-for-2025

Fyld. (2025). 2025 complete guide to DevSecOps for agile teams. Retrieved from https://www.fyld.pt/blog/2025-guide-to-devsecops-for-agile-teams/

MotivaLogic. (2025). Shift-left security in 2025: How DevSecOps is evolving. Retrieved from https://www.motivalogic.com/blog/shift-left-security-in-2025-how-devsecops-is-evolving/

Nukala, M. (2025). Platform engineering in 2026: The numbers behind the boom and why it's transforming DevOps. DEV Community. Retrieved from https://dev.to/meena_nukala_1154d49b984d/platform-engineering-in-2026-the-numbers-behind-the-boom-and-why-its-transforming-devops-3811

Oligo Security. (2025). DevSecOps in 2025: Principles, technologies & best practices. Retrieved from https://www.oligo.security/academy/devsecops-in-2025-principles-technologies-best-practices

OpsMx. (2025). How DevSecOps CI/CD pipelines secure the software supply chain. Retrieved from https://www.opsmx.com/blog/how-devsecops-ci-cd-pipeline-secures-the-software-supply-chain/

OpsMx. (2025). The ultimate DevSecOps checklist to secure the software supply chain. Retrieved from https://www.opsmx.com/blog/ultimate-devsecops-checklist-to-secure-ci-cd-pipeline/