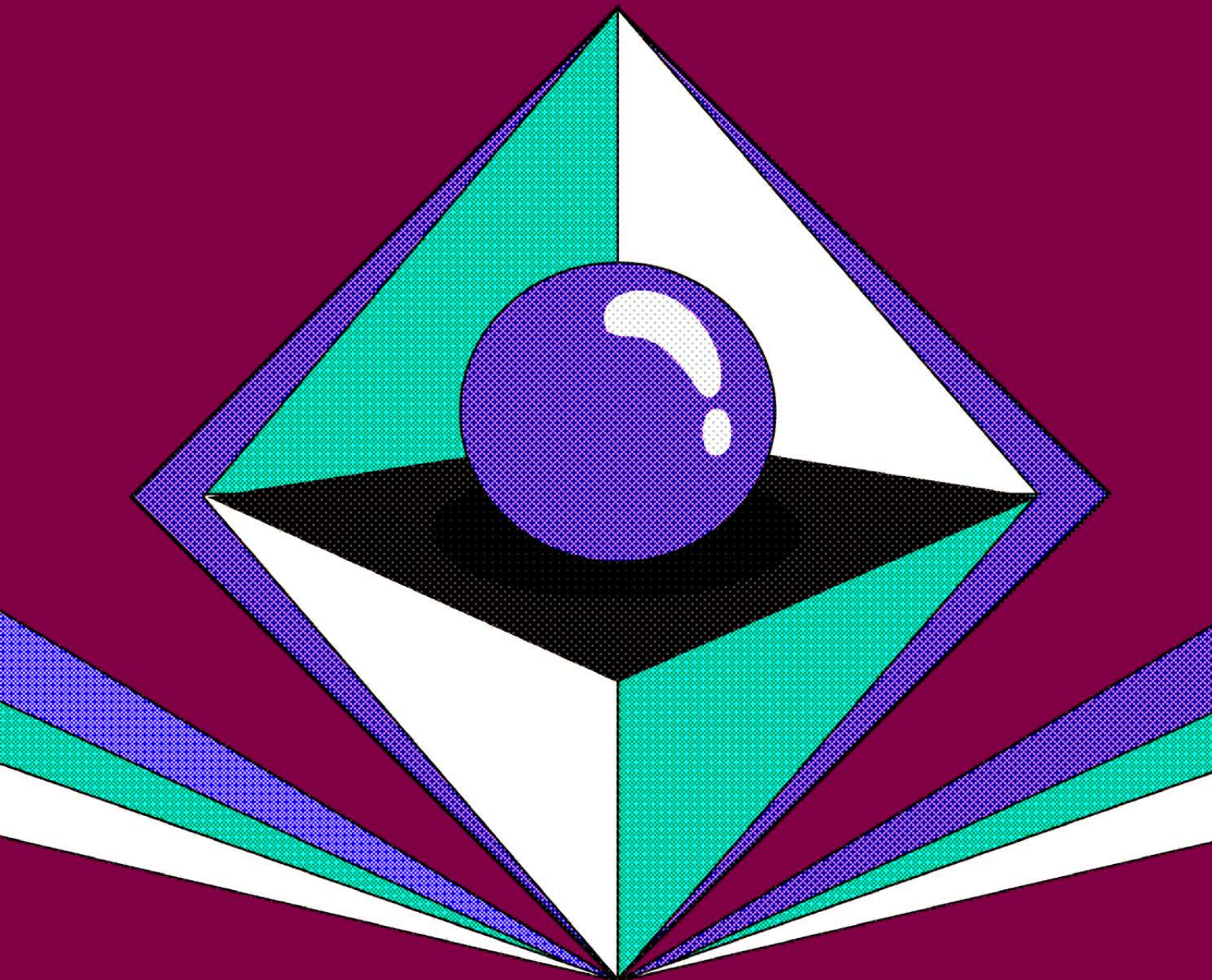


# State *of* AI in Platform Engineering



# Table of contents

About our partners	03
--------------------	----

Executive summary	04
-------------------	----

Defining AI in the context of platform engineering	08
--	----

What is the state of AI in platform engineering?	11
--	----

Is AI over-hyped or appropriately hyped?	13
--	----

How are platform engineers using AI?	15
--------------------------------------	----

What do platform engineers want to use AI for?	16
--	----

From top-down mandates to bottom-up innovation	18
--	----

The platform as an AI enabler: A new mandate	19
--	----

From cloud-native to AI-native infrastructure	21
---	----

Distinct demands of AI workloads	22
----------------------------------	----

AI-native and the dual mandate of platform engineering	23
--	----

The potential of agentic AI and autonomous systems	27
--	----

Key challenges and barriers	30
-----------------------------	----

People-centric hurdles	31
------------------------	----

Technical and operational hurdles	33
-----------------------------------	----

Security and governance challenges and solutions	35
--	----

The evolving platform team	37
----------------------------	----

Addressing the skill gap: Training and enablement strategies	39
--	----

Is fear of AI holding us back?	41
--------------------------------	----

Five key recommendations to guide platform teams toward effective AI integration into AI-powered platforms	43
--	----

Build foundations before intelligence	44
---------------------------------------	----

Start with clear intent and measurable outcomes	44
---	----

Embrace Platform as Product thinking for AI	46
---	----

Implement gradual, pragmatic governance	47
---	----

Invest in your team's evolution	48
---------------------------------	----

Final outlook: What will the future bring?	49
--	----

Appendix	52
----------	----

Survey methodology snapshot	52
-----------------------------	----

Glossary of key AI terms in platform engineering	53
--	----

References	58
------------	----



# About *our* partners



Vultr is on a mission to make high-performance cloud infrastructure easy to use, affordable, and locally accessible for enterprises and AI innovators around the world. Vultr is trusted by hundreds of thousands of active customers across 185 countries for its flexible, scalable, global Cloud Compute, Cloud GPU, Bare Metal, and Cloud Storage solutions. Founded by David Aninowsky and self-funded for over a decade, Vultr has grown to become the world's largest privately held cloud infrastructure company. Learn more at [Vultr.com](https://vultr.com).



ONA is mission control for software projects and software engineering agents. Ona transforms how software is built by moving beyond the IDE to enable developers and agents to work together in secure, disposable environments across the entire development lifecycle. Ona Agents handle complex tasks like refactoring and migrations in parallel while developers focus on higher-value work. Ona Environments are consistent and pre-configured to eliminate “works on my machine” issues. And finally, Ona Guardrails ensure enterprise-grade security, audit trails, and VPC deployment so source code and secrets never leave your network. Ona is trusted by over 2 million developers and deployed at America's largest banks and Europe's leading pharmaceutical companies. Learn more at [Ona.com](https://ona.com).



Coder is the AI software development company leading the future of autonomous coding. Coder helps teams build fast, stay secure, and scale with control by combining AI coding agents and human developers in one trusted workspace. Coder's award-winning self-hosted Cloud Development Environment (CDE) gives enterprises the power to govern, audit, and accelerate software development without trade-offs. Learn more at [coder.com](https://coder.com).





# Executive *summary*

Platform engineering is eating the world. As platforms grow and absorb everything from security and compliance to data and infrastructure, it is no surprise that AI, the key trend of our times, is a defining player in the platform engineering story. Our State of AI in Platform Engineering report hopes to summarise this immense topic. It analyses the survey responses of 242 platform engineering professionals from across the industry, alongside interviews with platform engineering and AI experts from the industry. Their responses reveal where AI currently stands, and how it is pulsating through every facet of the platform engineering story.



**89%** of respondents reported daily AI usage

AI is clearly everywhere, but the data gathered reveals a fundamental tension. Daily AI usage has become standard, with 89% of respondents reporting regular use, for code generation (75%) and documentation (70%). Yet this widespread adoption hides a critical gap: most usage remains tactical, driven by individual experimentation rather than strategic value. Tools like Cursor contribute almost 1 billion lines of code a day, roughly one-fifth of global code production, but many organizations are struggling to measure any meaningful ROI (beyond a few productivity metrics).

The report confirms the facts we all know in our gut. 84.5% of those who responded report that AI plays a large role in org goals and 90% believe AI will have a huge impact on us in the future. Platform teams are pressured from both C-suite mandates demanding ROI and developers experiencing “prompt fatigue” from uncoordinated tool proliferation. AI is everywhere, and we are all using it. Every executive is demanding it. We anticipate the rewards to be immense, and thus the pressures are immense too. Driven by quick wins and early

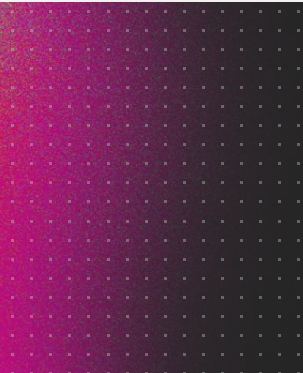
excitement the hype builds, but the long term work doesn’t deliver fast enough on what we want, and the “AI implementation plateau”, the gap between potential and realized value, is reached.

It is our belief that platform engineering teams are uniquely positioned to break through this plateau. Already, 75% of respondents are hosting or preparing to host AI workloads. Thus the platform engineer transforms from merely a user of AI to a driver of AI. Their cross-functional expertise enables them to curate toolchains, ensure security compliance, and transform fragmented experiments into cohesive solutions that actually deliver on the aspirations of our AI initiatives.

At the same time, platform engineers are well suited to face the challenges of AI, since as always, the key challenges remain human-centric. Skill gaps affect 57% of teams, while 56% struggle with AI hallucinations. Security teams, though 69.7% have AI policies, create friction through overly restrictive approaches (16.9% report blocking). Success requires balancing innovation with governance, experimentation with standardization, and understanding the socio-technical challenges of org transformation - an operating model that non-platform engineering teams are not used to existing within.







This is reinforced by data from [Google Cloud](#), where a staggering 94% of organizations identify AI as either ‘Critical’ or ‘Important’ to the future of platform engineering. And more, 86% believe that platform engineering is essential to realizing the full business value of AI.

This research by Google reflects a symbiotic relationship: AI-powered platforms, where AI empowers Internal Developer Platforms (IDPs), and Platforms for AI, where purpose-built foundations enable the efficient deployment and scaling of AI/ML workloads.

This duality underscores the opportunity. As developers lean heavily on AI, whether to auto-generate infrastructure as code, anticipate bottlenecks, or author CI/CD pipelines, platform engineering teams adapt swiftly to maintain control, reliability, and security. And as the most powerful AI demands platforms built specifically to better enable their deployment and use, platform engineers must embrace a new customer, the data scientist or AI engineer. They must dive into the challenge of building highly specialized ecosystems designed to support the world of AI and machine learning. They must enable data scientists and ML engineers to develop, train, and deploy AI models efficiently.

Just as the cloud-native era reshaped software delivery, we now stand on the edge of an

‘AI-native’ era where infrastructure, pipelines, and governance must evolve to handle GPU-accelerated workloads, agentic systems, and new operational patterns. Platform engineers must rise to the challenge of AI and help drive its adoption not purely based on individual usage, but system wide as a driver of AI-powered IDPs, and IDPs that better power AI. At the same time, platform engineers must ensure that their platform teams have robust product mindsets, clear and effective golden paths and automations, and goals driven by the needs of their customers (whether they be developers or data scientists) not by the whims and demands of the all powerful tidal wave of hype.

Platform engineers must understand their role, and their importance in this paradigm shift. They must not forget that the power of platform engineering will lie with its core principles, which will be enhanced by AI, not replaced by it.

## KEY TAKEAWAYS FROM REPORT

01

— AI adoption is near-universal

89% of platform engineers use it daily, yet most usage remains tactical, with orgs struggling to achieve measurable strategic ROI.

02

— Platform engineers are becoming AI enablers

Shifting from individual users to orchestrators of secure, scalable AI adoption that turns fragmented experiments into enterprise-wide value. This includes embracing new platform customers like data scientists and AI engineers.

03

— Challenges are multifaceted

Skill gaps, tool sprawl, hallucinations, governance friction, and cost concerns create a more complex and challenging transformation than many expect.

04

— Future platforms will be self-evolving

Blending AI-powered operations, agentic autonomy, and sustainability practices to deliver adaptive, intelligent infrastructure at scale.

05

— Success demands strong platform foundations evolving

Measurable outcomes, pragmatic governance, and continuous investment in people to ensure AI enhances platforms rather than amplifies chaos.



# Defining AI in the context *of* platform engineering

It is important that as we discuss the intersection of AI and platform engineering, we speak the same language, and define terms the same way. This section highlights how we interpret this terminology, and their importance for understanding the relationship between AI and platform engineering.





## AI-powered platforms

As stated, these platforms leverage AI to enhance traditional platform capabilities, focusing on making platform teams more efficient and developers more productive. They achieve this through AI-assisted infrastructure provisioning and configuration, intelligent troubleshooting and root cause analysis, automated code reviews and security scanning, natural language interfaces for platform interactions, and predictive scaling and resource optimization.

This perspective transforms how platform teams, and internal developer platforms work, enabling them to handle greater complexity with less manual effort.

## Platforms for AI

Platforms for AI provide the infrastructure and tooling necessary for AI/ML workloads, addressing unique challenges such as managing GPU resources and specialized compute, implementing MLOps pipelines for the model lifecycle, providing data versioning and feature stores, ensuring model governance and compliance, and supporting experimentation and model serving at scale.

This perspective expands platform teams' responsibilities to serve new customers: data scientists, ML engineers, and AI researchers.

## Generative AI (GenAI)

GenAI uses Large Language Models (LLMs) to understand and produce human-like text and code. For platform teams, this means automating repetitive tasks, generating boilerplate code, and creating documentation. The technology also democratizes development by enabling natural language interactions with platform capabilities.



## AI Agents

These software systems take task requests and use LLMs to plan and execute steps autonomously. Unlike traditional automation, agents adapt to changing conditions and learn from experience. They perceive their environment, make decisions, and take actions with minimal human oversight, a capability that promises to transform platform operations.



## Agentic AI

This broader capability combines LLMs, traditional ML, and enterprise automation to create truly autonomous systems. Agentic AI operates probabilistically, making decisions based on patterns and likelihoods rather than deterministic rules. This adaptability makes it ideal for complex platform engineering challenges.



It is crucial to understand that LLMs and autonomous agents don't eliminate the need for platforms, they elevate the abstraction layer. As AI automates low-level tasks, platforms shift from managing granular complexity to providing intelligent orchestration. Instead of engineers crafting individual CI/CD pipelines, they build platforms enabling AI agents to manage releases autonomously.

Kaspar von Grünberg

CEO OF HUMANITEC





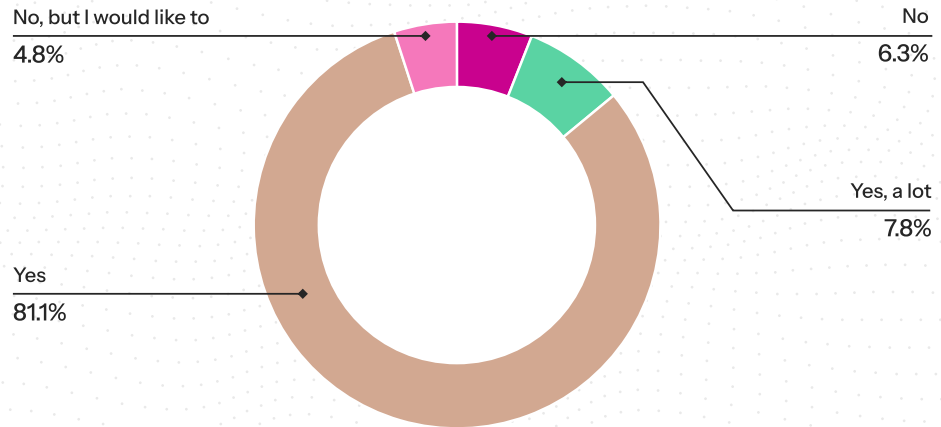
# What is the state *of* AI in platform engineering?

AI has clearly crossed the adoption chasm in platform engineering. What began as experimentation has become daily practice, fundamentally changing how we all work. However, AI usage for platform engineers is still immature, with code and documentation generation dominating, and advanced usage (the kind that brings step change ROI) lagging behind.





## DO YOU USE AI IN YOUR DAY-TO-DAY WORK?



The numbers tell a clear story: 89% of platform professionals use AI daily, with 81% confirming regular use and 8% reporting intensive usage.

These are more incredible adoption rates. Not since the Dotcom era in the 1990s has a new technology seen such rapid adoption, clear potential and excitement.

Yet these impressive statistics mask important nuances. Most current usage stems from individual exploration rather than any coordinated strategy. Engineers experiment with GitHub Copilot, Cursor, and ChatGPT, discovering individual productivity gains through trial and error. This grassroots adoption, while valuable for building familiarity, doesn't drive the org level ROI that leaders are hunting for, and

at the same time creates "Shadow AI", uncoordinated tool usage that bypasses organizational governance.

The ease of accessing AI tools is what drives this. Unlike previous technology waves requiring infrastructure investment, AI tools offer immediate value through simple browser interfaces or IDE plugins. This accessibility accelerates adoption certainly, but at the same time complicates governance massively.

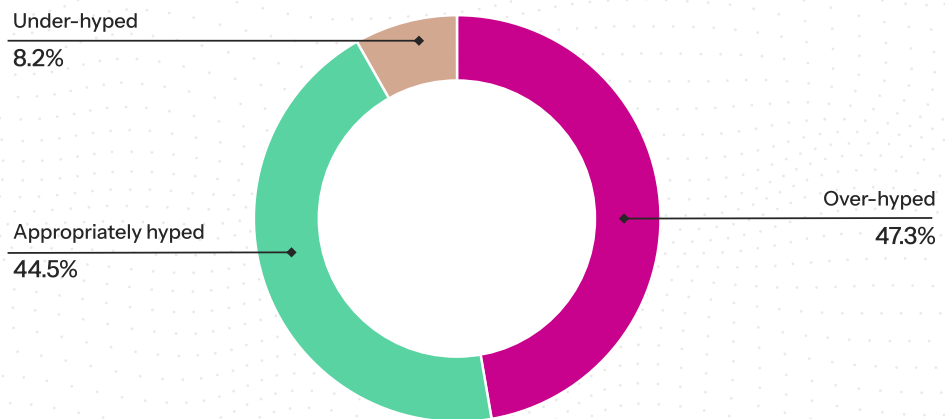
It's no surprise that in many organizations, as individual productivity gains don't translate to enterprise value, the disconnect between AI's perceived capabilities and actual outcomes ends up frustrating both executives seeking ROI and developers experiencing ever increasing pressure and expectations from leadership and AI mandates.



# Is AI over-hyped or *appropriately* hyped?

It will thus come as no shock that platform engineers are completely divided on the appropriate level of hype about AI: 47% consider it “over-hyped” while 45% see it as “appropriately hyped”. This split reflects the gap between promise and current reality.

WOULD YOU DESCRIBE AI AS BEING OVER-HYPED, APPROPRIATELY HYPED, OR UNDER-HYPED?



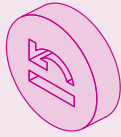
This division in hype perception is perfectly explained by the “AI implementation plateau”, the stage where organizations stop seeing rapid gains from AI adoption, and face slower ROI, integration hurdles, and the need for deeper cultural or process change. Despite widespread usage, organizations struggle to move beyond basic use

cases as foundation models and coding assistants dominate, while specialized platform engineering AI tools remain barely utilised.

Beyond the data, [across all AI talks and panels](#) at PlatformCon 2025, the recurring point was clear, “We are very far from utilising the full potential of AI”.

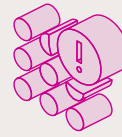


# Several factors at organizations are contributing to this plateau:



## Optimising for the easy

Orgs stick to quick wins, avoiding harder integrations.



## Tool proliferation

The overwhelming variety of AI tools creates decision paralysis.



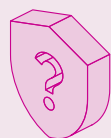
## Unclear ROI

Productivity gains prove difficult to quantify in business terms.



## Integration challenges

Fitting AI into existing workflows requires significant effort.



## Quality concerns

AI-generated outputs require extensive validation.

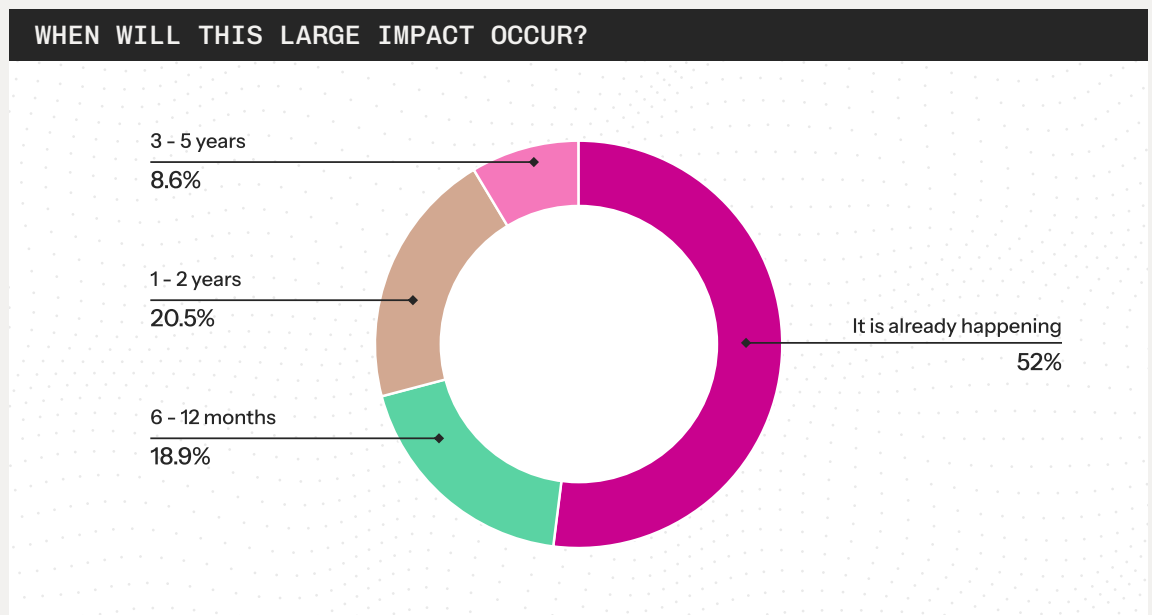
Breaking through demands strategic focus. Orgs need to move beyond counting lines of code generated to measuring real business impact i.e. measuring “usefulness”. This requires connecting AI adoption to specific outcomes: faster delivery, fewer defects, improved developer satisfaction. This kind of thinking should be a platform team’s bread and butter.





# How are platform engineers *using* AI?

The platform engineering community does show remarkable consensus in one area however: 71% expect AI's major impact within 12 months, and half of all platform engineers believing it's already underway. While overall, 90% of professionals anticipate a large impact in their day-to-day work in the future.



However, as highlighted above, current AI usage still concentrates on immediate productivity improvements rather than on large strategic changes. Two use cases dominate.



## Quality concerns

AI accelerates coding through boilerplate generation, function completion, and syntax assistance. Developers report significant time savings on repetitive tasks, allowing focus on complex problem-solving.

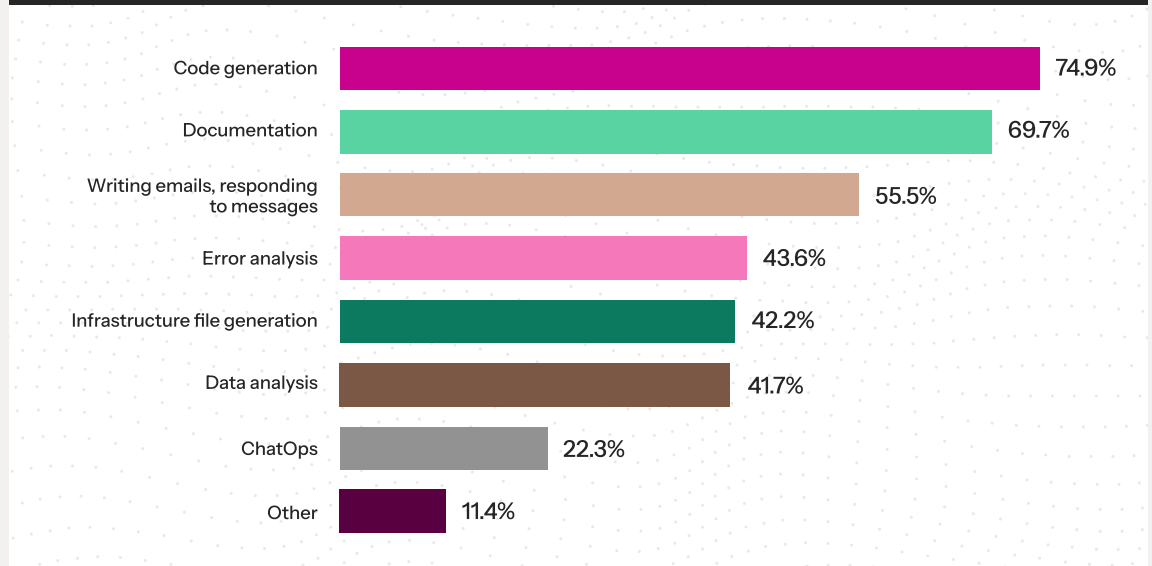


## Documentation

AI transforms documentation from burden to byproduct. Tools generate technical documentation, summarize code behavior, and create user guides automatically, addressing a perennial engineering pain point.



## WHAT DO YOU USE AI FOR?



Beyond these primary uses, AI enhances daily operations through email composition (56%), error analysis (44%), and infrastructure file generation (42%). A smaller but growing segment (22%) uses AI for ChatOps, integrating conversational interfaces into operational workflows.

# What do platform engineers want to use *AI for*?

We asked all of those surveyed to write their own answer to the question, “What would you like to use AI for?”. This generated a host of interesting, and unsurprising answers (49% of people simply wrote some form of “Everything, make my life easier, do my job” etc).

“ Everything ie. AI to replace me

“ Except for mental modelling...everything

“ Anything and everything

“ Everything

“ Everything

“ Everything

“ Everything

“ Everything

# Through the many answers, recurring themes appeared.

25%

## Software Development Life Cycle (SDLC) enhancement

Teams want intelligent code review that understands architectural patterns, automated testing that generates scenarios that are actually meaningful, and CI/CD optimization that can learn from deployment history and improve itself more effectively.

Unrepresented in the survey data, but highlighted by experts in the community, three specialized use cases deserve particular attention. These are test case generation, where the AI analyzes requirements and historical data to produce comprehensive test suites that improve coverage and reduce manual effort; legacy code comprehension, where it decodes undocumented systems, interprets functionality, and identifies vulnerabilities to support modernization of decades-old codebases. And, architectural refactoring, where AI provides real-time feedback on design decisions, surfacing technical debt and prioritizing improvements to prevent the accumulation of complexity that can cripple system evolution.

These use cases address fundamental engineering challenges, rather than individual productivity hacks.

See examples of successful AI implementations in one of the [30+ AI talks from PlatformCon 2025](#).

16%

## Data analysis and observability

Deeper operational intelligence. AI should identify patterns humans miss, predict failures before they occur, and recommend optimizations based on system behavior.

10%

## Platform management and architecture

Strategic interest centers on AI managing infrastructure complexity. This includes capacity planning, cost optimization, and architectural recommendations based on usage patterns.







Platform teams want AI that understands context, learns from patterns, and makes intelligent recommendations. They seek partners in platform evolution, not just productivity tools. We have enough productivity tools...

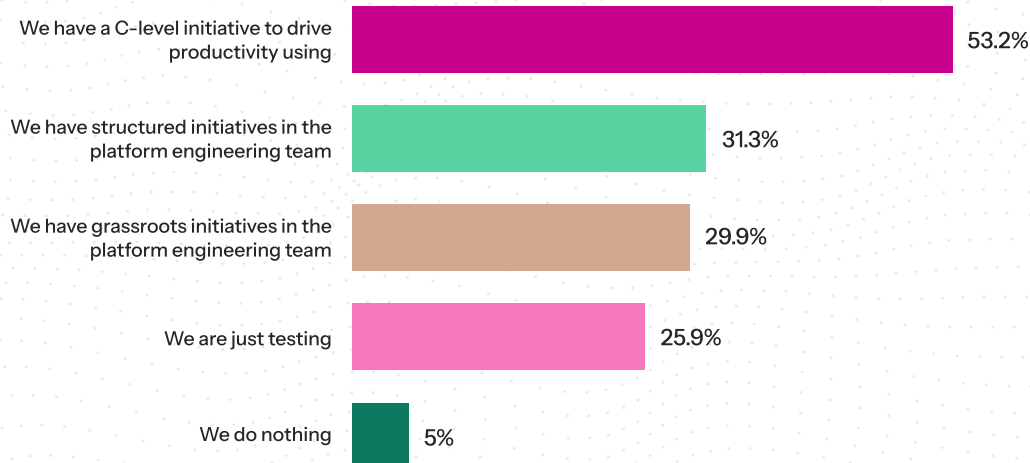
Luca Galante

CORE CONTRIBUTOR TO PLATFORM ENGINEERING COMMUNITY

# From top-down mandates to bottom-up innovation

AI adoption in platform engineering is emerging from two converging forces. Top-down mandates meet bottom-up innovation. This combination drives immense usage but also creates a tension that shapes the implementation patterns for orgs.

## WHAT IS YOUR ORGANIZATION'S ATTITUDE TOWARDS AI?



Executive leadership tries to drive strategic intent, with 84.5% of respondents confirming AI plays a large role in organizational goals. AI mandates from C-suite typically focus on broad efficiency gains and competitive advantage, often without specific guidance for platform teams beyond generic “improve ROI for Copilot adoption” directives.



While at the same time, the grassroots adoption (30%) drives momentum where leadership is slow. Local champions demonstrate practical value, showing colleagues how AI enhances real workflows. This peer-driven adoption proves more effective at overcoming developer skepticism than top-down mandates. Engineers trust demonstrations from peers over executive proclamations. However, usage often focuses on individual productivity or quality of life changes, not strategic values.

Platform teams must occupy the crucial middle ground. By translating executive vision of “Use AI, cut costs” into practical tools, and specific use cases, while at the same time supporting and providing governance frameworks for grassroots innovation. This way, platform engineers can drive org level AI success by making the right path the easy path and providing secure, integrated AI capabilities that developers choose voluntarily, and serve org wide objectives.

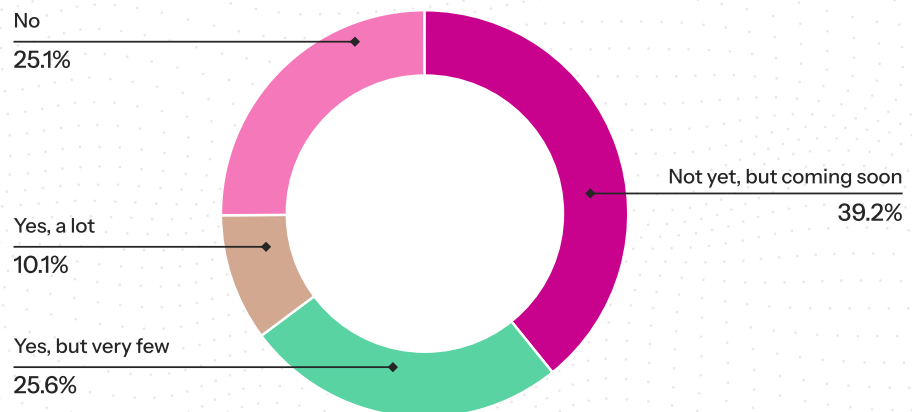
This does come with new challenges for platform teams, as it means you must no longer serve only developers; data scientists, ML engineers, and business analysts must also become platform customers, with each group bringing different expectations and requirements, demanding flexible yet standardized approaches.



# The platform as an AI enabler: A new mandate

Platform teams face a fundamental shift in purpose. No longer just infrastructure providers, and AI users but AI enablers at foundation level, a transformation already underway with 75% hosting or preparing to host AI workloads.

AS A PLATFORM TEAM, ARE YOU ASKED TO HOST AGENTS OR AI-INFUSED APPLICATIONS



This new mandate encompasses the two critical perspectives highlighted above enabling AI-powered platform engineering and providing Platforms for AI.





# Key capabilities defining this include:

## RAG Pipeline Implementation

Retrieval-Augmented Generation connects LLMs to organizational knowledge. Platform teams build and maintain these pipelines, ensuring AI models access current, relevant information without expensive retraining. RAG reduces hallucinations while providing domain-specific intelligence.

## API standardization

Agentic approaches observe and standardize APIs across organizations. Platform teams create unified interfaces enabling seamless AI service integration while maintaining security boundaries. Techniques such as Multi-Party Computation (MPC) further strengthen cross-organizational security by enabling data sharing and collaboration without exposing sensitive inputs.

## Metadata centralization

AI requires rich context to function effectively. Platform teams aggregate metadata about services, dependencies, and behaviors, creating the knowledge foundation AI agents need for intelligent decisions.

## Governance frameworks

Responsible AI deployment demands robust governance. Platform teams establish prompting guidelines, audit mechanisms, and explainability tools ensuring AI usage aligns with organizational policies.



# From cloud-native to AI-native infrastructure

**There is one missing ingredient in this story of AI's ubiquity and the new mandate for platform engineers: the infrastructure leap required to actually succeed. Think of it as cloud-native's sequel: AI-native.**

The evolution of enterprise architecture in the last two decades saw the emergence of the cloud-native era. It appears now that we are entering the [threshold of the AI-native era](#). Accelerated by large-scale AI adoption, this transformation redefines what platforms must deliver. Where cloud-native championed agility and scalability for web applications, AI-native demands a more sophisticated, composable, and globally distributed foundation built for GPU-accelerated training, real-time inference, and hybrid deployment, and tighter MLOps governance. For platform engineering teams, this shift marks the strategic imperative to design infrastructure tailored for intelligent systems.

Cloud-native infrastructure, pioneered by global cloud infrastructure providers, enabled enterprises to transition from monolithic web applications to highly flexible, containerised, and microservices-based architectures. Cloud-native infrastructure abstracted away much of the underlying hardware. This abstraction, however, primarily focused on CPU-centric workloads, managing virtual machines and containers with relative ease. The rapid emergence of AI introduces new computational demands and architectural patterns that demand another evolution.

The shift from cloud-native to AI-native requires platform engineers to apply core principles of composability, governance and scalability to GPU-accelerated workloads and inference. At Vultr we see platform engineering teams as the ones standardizing AI infrastructure so it is consistent, reliable and ready for production.

**Kevin Cochrane**  
CMO, VULTR



# Distinct demands of AI workloads

AI workloads, particularly those involving machine learning (ML), large language models (LLMs), and real-time inference, massively alter infrastructure requirements. These applications thrive on globally orchestrated GPU + CPU infrastructure, demanding a new level of performance, efficiency, and distributed compute capabilities. Unlike traditional CPU-bound applications, AI models require specialised hardware GPUs for accelerated training and inference. The need for composable architectures at both the application and infrastructure tiers becomes crucial, allowing for the seamless swapping of resource types as innovation speeds up.

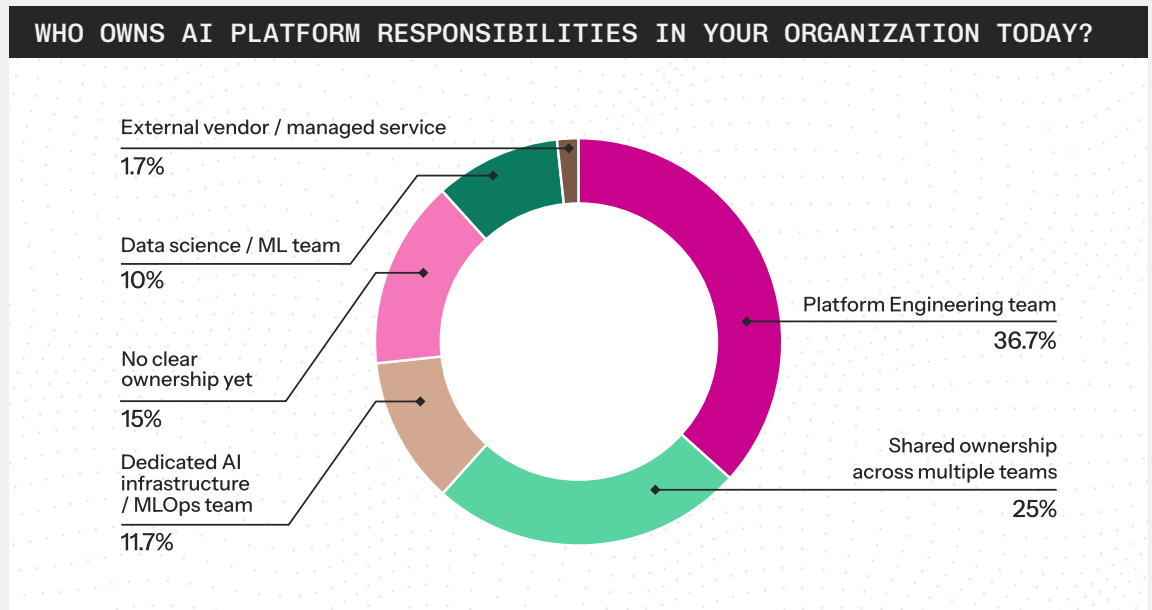
Not to mention the need for AI-native infrastructure to support containerised models and real-time inference across hybrid environments, including public cloud, on-prem systems, and edge locations. Imagine a model trained in the cloud, updated nightly on-prem where sensitive data lives, and served at the edge in hospitals, factories, or retail outlets. This would require a unified, composable cloud approach that integrates CPU and GPU clusters, accelerated storage, and high-speed, standards-based networking, such as those enabled by initiatives like Ultra Ethernet, to handle massive data flows and real-time processing.

Leading infrastructure vendors are developing full-stack, deep-compute architectures where components are highly composable and designed for seamless integration, supporting diverse open-source frameworks and models. This includes robust MLOps pipelines for managing the entire model lifecycle, from data versioning and feature stores to model governance and scalable serving infrastructure.



# AI-native and the dual mandate of platform engineering

To better understand this infrastructure evolution, we did a separate AI survey targeting those in the community who are most actively working building an AI-native future. We spoke to 109. This survey revealed both major gaps of where we need to be and where we are, and the ever present impact of this new “dual mandate” for platform engineering.



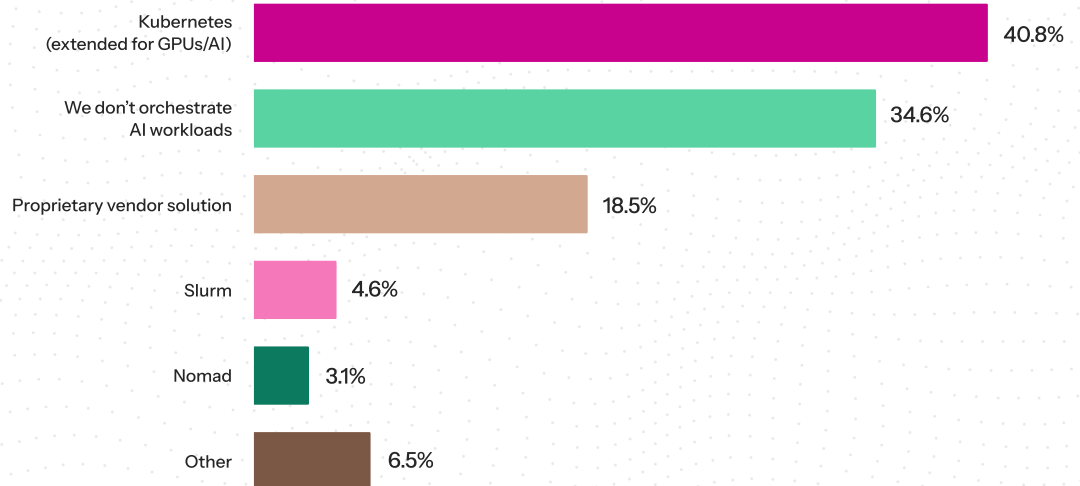
The survey data reported that while a significant 36.7% of organizations assign AI platform responsibilities to platform engineering teams, making them the largest single group owning these initiatives, 25% report shared ownership and dedicated AI infrastructure or MLOps teams account for 11.7%. Significantly, 15% still have no clear ownership, indicating the vast differential in maturity between different enterprises.

Platform engineering may be taking over management of AI responsibilities. The data reveals progress and immaturity simultaneously. While 40.8% leverage Kubernetes extended for GPUs and AI for orchestration, surprisingly 24.6% still run AI workloads manually, often on single servers or loosely connected GPU clusters creating bottlenecks when scaling into production.



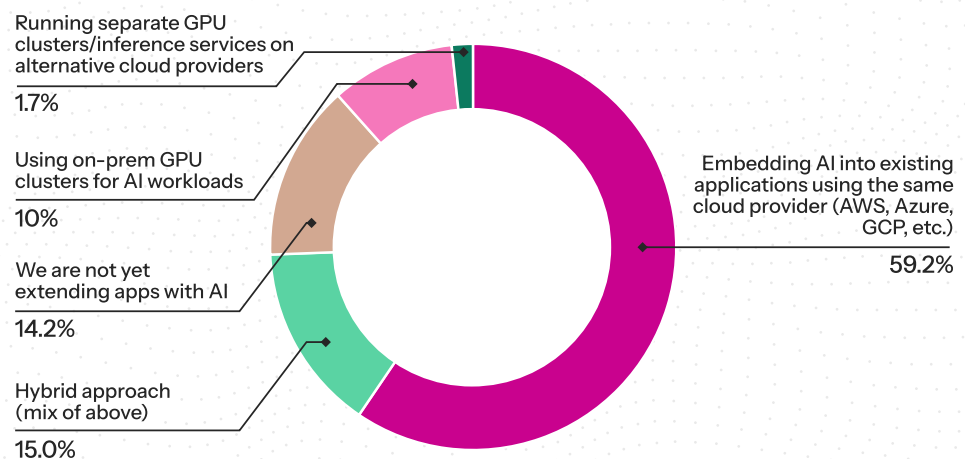


## WHAT ORCHESTRATION LAYER(S) ARE YOU USING TO MANAGE AI TRAINING AND INFERENCE WORKLOADS?



Beyond orchestration, enterprises are already embedding AI into their existing digital fabric. The survey shows that 59.2% extend their cloud-native applications with AI services from their existing cloud provider, while others take hybrid (15%) or on-premises GPU approaches (10%). This integration trend underscores how AI is being treated not as a separate silo but as an expected capability of modern applications.

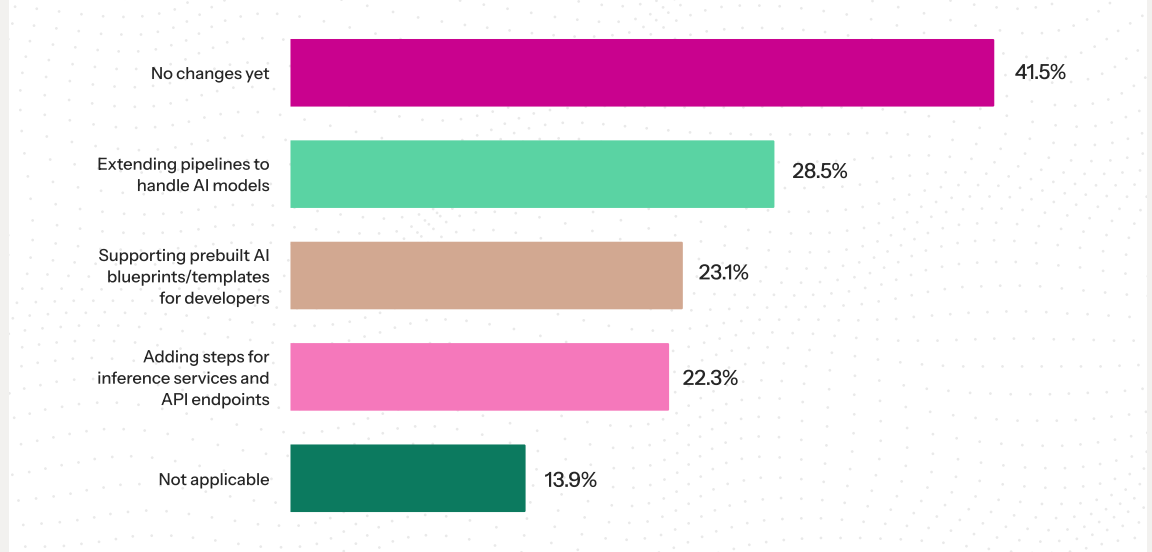
## HOW DOES YOUR ORGANIZATION EXTEND EXISTING CLOUD-NATIVE APPS WITH AI SERVICES?



At the same time, CI/CD and DevSecOps pipelines are unevenly adapting: while 28.5% of organizations are extending pipelines to handle AI models and 22.3% are adding inference service steps, 41.5% haven't touched their CI/CD at all. This means model handoffs remain manual and inference endpoints are often deployed outside the same governance as

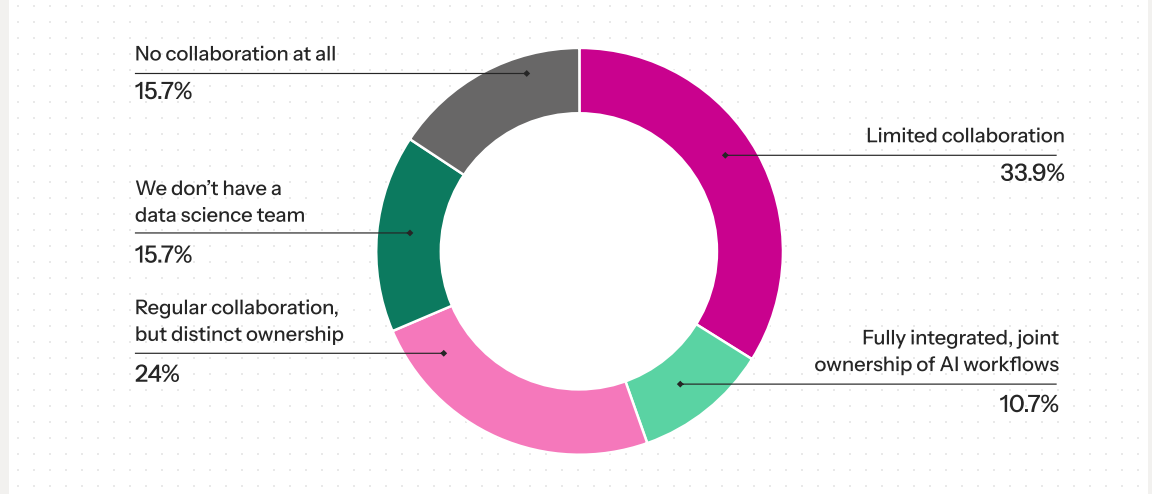
applications. Standardisation is emerging as a top priority, with more than half of respondents rating AI infrastructure templates and blueprints as either critical or very important. Together, these findings signal both strong momentum toward AI-native practices and the work still required to ensure consistency and maturity across enterprises.

#### HOW ARE YOUR CI/CD AND DEVSECOPS PIPELINES EVOLVING TO SUPPORT AI ARTIFACTS?



Collaboration with data science teams, crucial for successful AI initiatives, also remains fragmented, with 33.9% reporting “Limited collaboration” and 15.7% indicating “No collaboration at all”.

#### TO WHAT EXTENT ARE PLATFORM ENGINEERING TEAMS COLLABORATING WITH DATA SCIENCE TEAMS ON AI INITIATIVES?



As Luca Galante of the Platform Engineering Community aptly states, “Platform teams want AI that understands context, learns from patterns, and makes intelligent recommendations. They seek partners in platform evolution, not just productivity tools”. This necessitates platform teams to actively embrace new personas

and build “golden paths” for AI development. Ultimately, AI-native platforms, demanding globally orchestrated GPU resources, composable architectures, and advanced MLOps governance, are becoming the essential foundation for intelligent and autonomous systems.

# The potential of agentic AI and autonomous systems

It is clear that despite the challenges and implementation gaps, the future potential for AI and platform engineering is immense. That potential is best emphasized by Agentic AI, which represents the next frontier in platform evolution. Moving beyond simple automation, these systems demonstrate true autonomy, shaping the trajectory of AI-native infrastructure towards autonomous and self-evolving platforms. In this future, AI becomes a fundamental component of every platform capability, from deployment to monitoring, enabling systems that observe usage patterns, identify optimization opportunities, and implement improvements independently. Agentic AI will not only manage entire platform subsystems but also orchestrate complex deployments, negotiate resource allocation, and even design platform features based on user needs.

This future will redefine human-AI collaboration, shifting platform engineers from reviewers to strategists, setting objectives and auditing outcomes, rather than managing every operational detail. This transformation will also demand a focus on sustainability, with platform engineers bearing responsibility for efficient model serving, intelligent caching, and optimising workloads for renewable energy. Reflecting this forward-looking vision, some organizations foresee their strategy as “Evolving our IDP to an AI PaaS” and “Defined agentic golden paths for developers”. Others anticipate “More open source, more templates, more best practices,” signifying a collaborative and standardised approach to future AI infrastructure.





It is crucial to understand that LLMs and autonomous agents don't eliminate the need for platforms, they elevate the abstraction layer. As AI automates low-level tasks, platforms shift from managing granular complexity to providing intelligent orchestration. Instead of engineers crafting individual CI/CD pipelines, they build platforms enabling AI agents to manage releases autonomously.

Kaspar von Grünberg

CEO OF HUMANITEC

We see examples of the shift highlighted by Kaspar with background agents (also sometimes called 'async' or 'parallel' agents) with platforms like [Ona](#). This is where engineers who previously would be doing labour intensive and low-level work typing code are now able to move to higher-level 'orchestration' of multiple agents. The long adopted tools of developers like their IDEs are being upended as engineers demand conversation-first interfaces stepping in to provide steering and guidance only when necessary.

"But here's the thing: the IDE isn't optimized for the "review changes" task. It's still showing you syntax highlighting and auto-completion suggestions when what you really need are better diff views, easier ways to understand what changed and why, tools for quickly testing whether the changes actually work." — Kent Beck in Beyond the IDE

Against this backdrop, it's important to understand how AI's current role compares to what's emerging next.

## Current state – AI Agents

Today's AI agents execute defined tasks using LLM planning . They handle specific workflows like reviewing pull requests or generating deployment configurations. While useful, they operate within narrow boundaries.

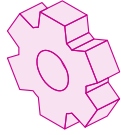
## Emerging reality – Agentic AI

True agentic systems combine multiple AI technologies for dynamic problem-solving. They perceive environmental changes, adapt strategies, and learn from outcomes. Unlike deterministic automation, they operate probabilistically, handling uncertainty and complexity.



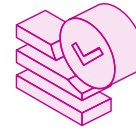


# Platform engineering presents ideal use cases for agentic AI:



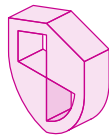
## Failure remediation

Agents diagnose issues, implement fixes, and learn from resolutions



## Capacity management

Systems predict demand, allocate resources, and optimize costs autonomously



## Security response

Agents detect anomalies, investigate threats, and implement countermeasures



## Change management

AI evaluates change risks, suggests deployment strategies, and monitors outcomes

These capabilities transform platform operations from reactive to proactive. Instead of responding to alerts, teams focus on strategic improvements while agents handle routine operations.

The impact extends beyond efficiency. Agentic AI enables platforms to handle complexity exceeding human cognitive capacity. Modern platforms

comprise thousands of services with intricate dependencies which AI agents can effectively navigate making decisions based on patterns humans cannot perceive.

Though the absolute majority of orgs are far from this capability, the fast moving platform teams are already exploring agentic AI capabilities within their platform initiatives.



# Key challenges *and* barriers

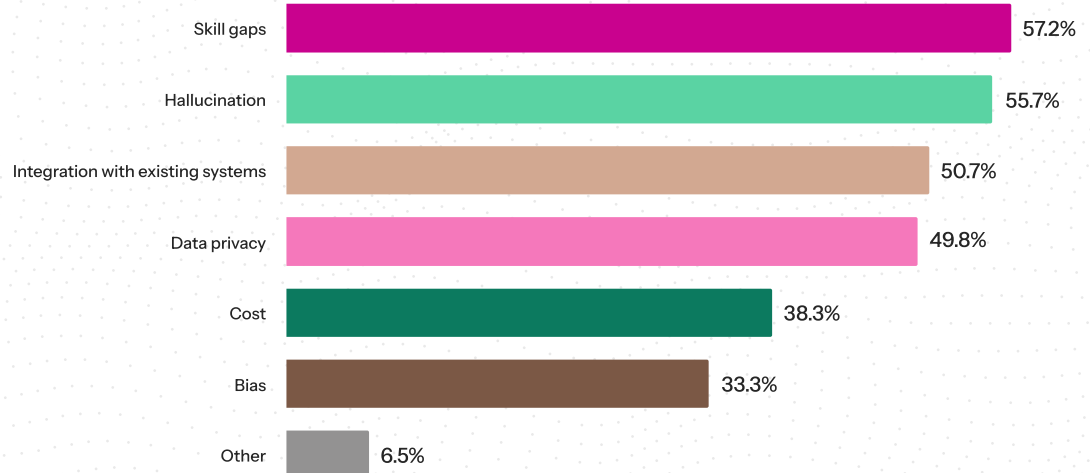
The path to AI-powered platforms, AI platforms and an AI-native future is beset with obstacles. Understanding these challenges and the strategies to overcome them will determine success or failure in your AI journey. Platform teams face a complex landscape of technical, human, and organizational barriers from skill gaps to security and governance.

These challenges reflect AI's fundamental nature, powerful but unpredictable, accessible but complex, transformative but disruptive.





## WHAT AI CHALLENGES HAVE YOU ENCOUNTERED WITHIN PLATFORM ENGINEERING?



# People-centric hurdles

Human challenges dominate the barrier landscape, reflecting AI's profound impact on roles, skills, and working patterns.

## Skill gaps

57%

The most cited challenge reveals a fundamental mismatch between current capabilities and AI demands . Platform engineers need new competencies:

- **Prompt engineering to effectively communicate with AI systems**
- **Understanding of ML fundamentals and model behavior**
- **Data literacy to manage AI inputs and outputs**
- **Soft skills for cross-functional collaboration with data scientists**

The rapid pace of AI evolution means skills become obsolete quickly. Yesterday's best practices fail with today's models. This creates continuous learning pressure that many find overwhelming.



## Tool proliferation and fragmentation

56%

AI hallucinations, plausible but incorrect outputs, pose serious risks. These aren't bugs to fix but inherent model characteristics. Platform teams must implement safeguards:

- **Input validation preventing problematic prompts**
- **Output verification catching incorrect responses**
- **Context management ensuring relevant information**
- **Temperature controls balancing creativity with accuracy**

Hallucinations threaten core platform engineering values: reliability, predictability, and trust. Managing them requires new approaches balancing AI benefits with safety requirements.

## Understanding “Agentic” AI

35%

Many of those surveyed expressed total confusion about agentic AI concepts. This knowledge gap hinders adoption of advanced capabilities. Without clear understanding, teams cannot effectively evaluate or implement autonomous systems, and more so, cannot effectively challenge or embrace executive mandates.

## Fear vs. augmentation

Job security concerns create resistance. While 60% view AI as augmentation, 25% fear automation of their tasks, and 11% see long-term job risk. This fear particularly affects mid-level engineers who've spent decades perfecting now-automatable skills. While Junior engineers face extreme challenges as AI handles tasks traditionally used for helping them learn.

Organizations need to address these fears head





on. As historical precedent shows, technology transforms rather than eliminates jobs and Platform engineers who embrace AI tools enhance their value; while those who resist risk obsolescence.

The key message: AI amplifies capabilities for those who learn to wield it effectively.

---

# Technical and operational hurdles

**Human-centric hurdles might dominate the barrier landscape, but technological challenges are still a challenge facing many teams in their race to achieve enterprise value from AI.**

## Integration complexity



51%

Existing platforms were never designed with AI integration in mind, and monolithic architectures, legacy systems, and technical debt often turn the process into a nightmare. Common issues include data format mismatches between systems and AI models, performance impacts from AI processing overhead, and architectural conflicts between deterministic and probabilistic systems.

New protocols such as the Model Context Protocol (MCP) to standardize LLM inputs, the Agent Communication Protocol (ACP) to enable inter-agent coordination, and Agent2Agent (A2A) to facilitate cross-platform collaboration attempt to address these challenges, but adoption remains fragmented, leaving platform teams to support multiple standards simultaneously.



## Tool proliferation and fragmentation

The explosion of AI tools creates paradoxical problems. Teams struggle choosing among hundreds of options, each claiming revolutionary capabilities. This leads to:

- **“Prompt fatigue” from constant tool switching**
- **Fragmented workflows reducing productivity**
- **Redundant implementations wasting resources**
- **Incompatible tools creating data silos**

## Code stability

27%

You don’t need us to tell you the issues with AI-generated code. While AI produces code quickly, quality varies dramatically, presenting unique (to say the least) maintenance challenges.

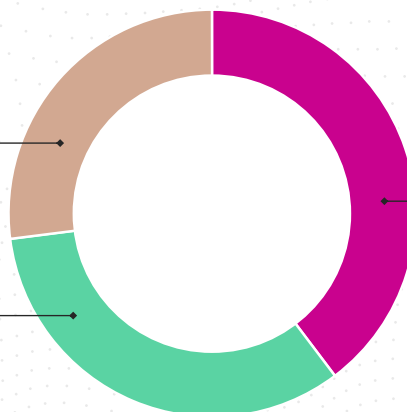
- **Hidden vulnerabilities escape traditional security scans**
- **Inconsistent patterns violate coding standards**
- **Architectural assumptions conflict with existing systems**
- **Documentation gaps complicate future modifications**

AS A PLATFORM TEAM DO YOU HAVE TO DEAL WITH AUTO GENERATED CODE?

Yes and it's hard to keep things stable  
27%

Not really  
33.3%

Yes, but it's not an issue  
39.7%



Platform teams must implement rigorous review processes. AI-generated code requires significantly more scrutiny than human-written code, paradoxically increasing review burden while promising efficiency gains.



## Data privacy

50%

AI's hunger for data often clashes with privacy requirements, as models trained on vast datasets raise concerns around consent for data usage in training, the potential exposure of sensitive information, reliance on third-party AI services that process organizational data, and compliance with rapidly evolving regulations.

## Cost

38%

AI implementation expenses often surprise organizations, as costs extend well beyond the obvious. Beyond GPU infrastructure for model serving and the increased compute required for AI processing, teams must also account for specialized platforms and tools, training and skill development, and the ongoing updates and maintenance that AI systems demand.

# Security and governance challenges and solutions

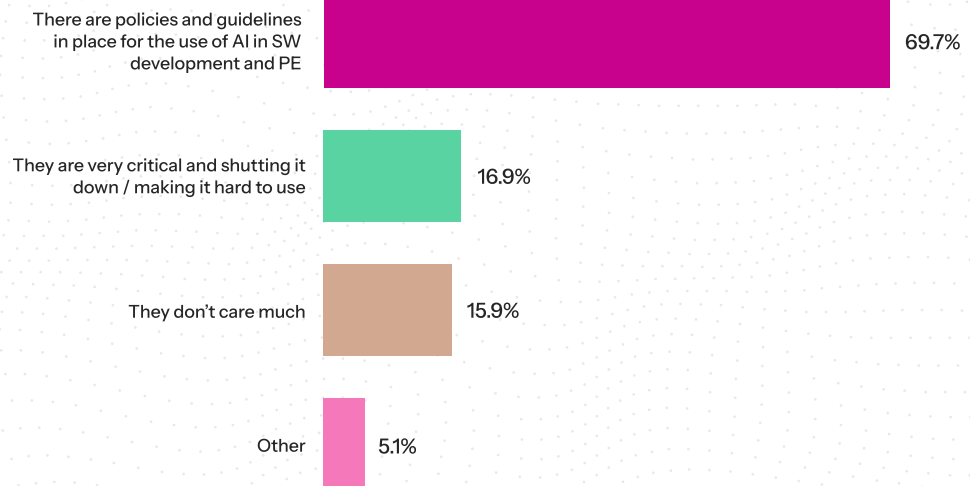
The data reveals clear policy-practice gaps as only 69.7% of organizations have AI policies (despite the aforementioned 89% of platform engineering using AI daily). While 16.9% report security teams blocking simply blocking usage, highlighting how governance struggles to keep pace.

This is corroborated by interviews with platform engineering teams, who revealed a common

challenge: security teams, often trained to handle deterministic systems, find probabilistic AI behaviors difficult to manage. This then often leads to blanket prohibitions that ignore nuanced use cases, approval processes modeled on traditional software, audit requirements that are impossible to apply to black-box models, and compliance frameworks that lack explicit AI provisions.



## WHAT IS YOUR SECURITY TEAM'S ATTITUDE TOWARDS AI?



Some fast moving security teams however are embracing the adaptive governance solutions required to embrace AI in a safe and productive way.

### Policy as Code (PaC)

Machine-readable policies enable automated enforcement. PaC scans AI-generated code, validates model inputs, and enforces constraints without human intervention. This provides flexible, transparent governance scaling with AI adoption.

### AI observability

Requiring comprehensive monitoring to reveal system behavior, including input/output tracking for audit trails, performance metrics that identify degradation, drift detection to catch model decay, and explainability tools that help demystify AI decisions.

### Policy as Code (PaC)

Platform teams implement boundaries controlling AI access. Sensitive data remains isolated while AI processes permitted information. This granular approach balances capability with security. For example, platform teams enforcing AI experimentation within a CDE where each ephemeral workspace ships with pre-scoped credentials, approved RAG indexes, and model access policies, ensuring prompts and outputs stay within project boundaries and are fully auditable.

These capabilities enable governance based on actual behavior rather than theoretical risks.





# The *evolving* platform team

AI catalyzes fundamental changes in platform team composition and focus, blurring traditional boundaries as teams evolve from infrastructure managers to AI enablers. This coming evolution will reshape responsibilities: instead of reacting to developer requests, AI-enabled teams will anticipate needs, with agents identifying optimization opportunities before humans notice problems; instead of providing static tools, teams will deliver adaptive systems that learn from usage patterns, automatically adjusting configurations and suggesting improvements; and instead of serving primarily developers, platform teams will support multiple personas, including data scientists requiring ML pipelines, business analysts needing data platforms, and AI researchers demanding GPU clusters, each bringing distinct requirements that must be balanced and understood.



These modern platforms will include development environments with integrated AI assistance, knowledge management systems feeding RAG pipelines, observability platforms capable of understanding AI behavior patterns, and governance frameworks designed to manage probabilistic systems. The platform team will also be responsible for owning new tooling like the [Cloud Development Environment](#), which can become the meeting point for DevEx, AI security, and MLOps.

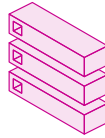
As platform engineering expands beyond its current horizons into new domains, its focus becomes clearer: not merely to provide infrastructure, but to curate intelligent experiences. This demands an understanding of both technical capabilities and human needs, bridging the gap between AI's potential and its practical value for platform customers, whoever they may be.

A RAG pipeline (Retrieval-Augmented Generation) boosts LLMs by first retrieving relevant data from external sources (like vector databases or APIs), then injecting that context into the prompt. The model uses both retrieved facts and its own reasoning to generate accurate, up-to-date, domain-specific answers.

At the same time, AI integration will create new specializations and areas of focus within platform engineering, reflecting the technology's diverse demands.

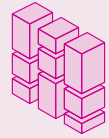


## Data platform engineering



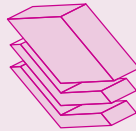
- Designing data pipelines feeding AI models
- Implementing feature stores for ML workflows
- Ensuring data quality and governance
- Optimizing storage and compute for AI workloads

## ML platform engineering



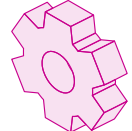
- Building MLOps pipelines for model lifecycle management
- Implementing feature stores for ML workflows
- Ensuring data quality and governance
- Optimizing storage and compute for AI workloads

## AI security engineering



- Developing threat models for AI systems
- Implementing adversarial testing frameworks
- Creating audit trails for AI decisions
- Ensuring model integrity and preventing tampering

## Developer experience (DevEx) for AI



- Integrating AI tools into developer workflows
- Creating abstractions hiding AI complexity
- Building feedback systems improving AI assistance
- Measuring and optimizing AI-enhanced productivity

These specializations must not create silos however, they must deepen expertise while maintaining platform coherence. Successful teams will balance specialized knowledge with collaborative integration.



# Addressing the skill gap: Training and enablement strategies

Skill gaps are one of the biggest barriers to AI adoption in platform engineering, cited by 57% of respondents. This goes beyond missing technical know-how, it reflects the breadth of change AI brings, where infrastructure and DevOps skills alone are no longer enough. Platform engineers now

need a multidimensional skill set that blends technical depth with collaboration and governance, all while keeping pace with AI’s rapid evolution. Success depends on structured training and, above all, an endless mindset of continuous learning and adaptation.

## Essential skill categories for platform engineers

SKILL CATEGORY	DESCRIPTION
Technical AI/ML	Understanding machine learning fundamentals, model development, and lifecycle management (MLOps, GenAIOps, LLMOps).
Cloud/Infrastructure	Expertise in cloud computing, containerization, and infrastructure as code for AI workloads.
Prompt engineering	The ability to craft effective prompts to guide AI tools for desired results, providing context and reducing ambiguity.
Soft skills	Communication, collaboration, problem-solving, adaptability, and business acumen for interdisciplinary AI projects.
AI governance & ethics	Knowledge of responsible AI principles, data privacy, compliance, and ethical considerations in AI deployment.





# Effective training strategies address multiple learning modes

## Formal training

Certifications from the platform engineering community, alongside specialised courses on AI and platform engineering from the platform engineering university. Alongside, cloud providers (Azure AI, AWS ML, Google Cloud AI) which provide foundational knowledge.

## Experiential learning

Sandbox environments like CDEs allow safe experimentation letting engineers practice with production-like AI integrations i.e RAG, vector DBs, model gateways, without risking live data. You can also pair programming with agents inside the CDE to accelerate skill acquisition while keeping guardrails intact.

## Community learning

Internal communities of practice share discoveries. External conferences provide exposure to emerging practices. Open source contributions build real-world experience. Hackathons focused on AI integration build practical skills. Pair programming with AI tools develops intuition.

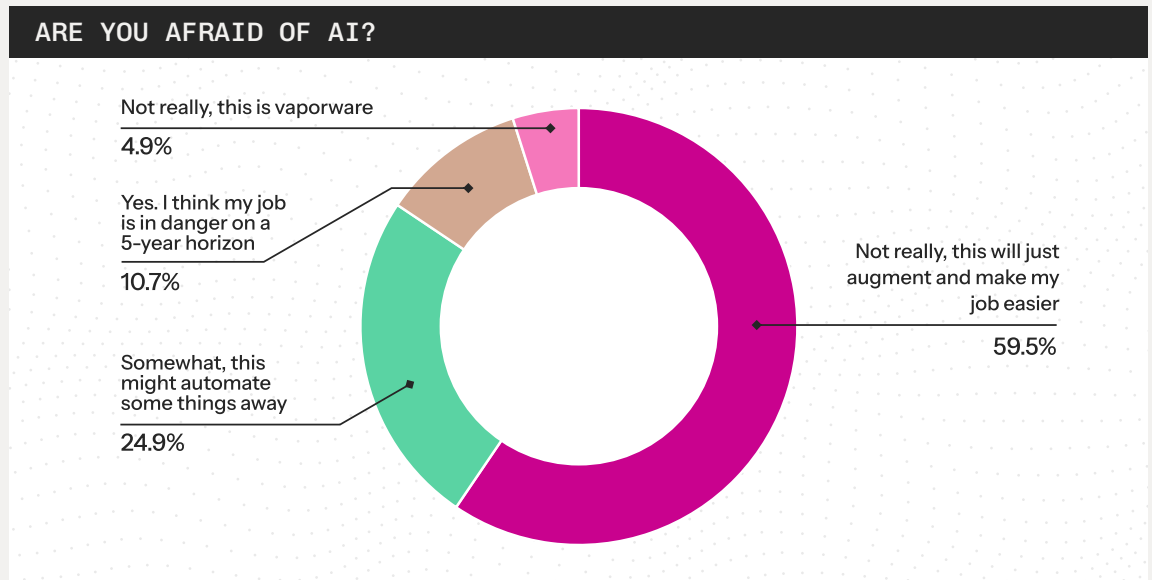
At the same time, critical to this evolution is the platform team's role in abstracting complexity. Not every developer needs deep AI expertise. Platform teams should create interfaces making AI accessible without requiring specialized knowledge. This democratization multiplies AI impact across organizations.

Yet history provides a hopeful perspective. Each technological wave transformed rather than eliminated jobs. Platform engineers who adapted to cloud, containers, and DevOps enhanced their careers. AI presents similar opportunities for those who embrace change.



# Is fear of AI holding us back?

As highlighted above in people centric hurdles, career concerns permeate discussions about AI's impact. The data reveals nuanced perspectives on how AI will impact platform engineers jobs and career prospects.



These fears deserve serious consideration. AI does automate tasks previously requiring human expertise. Code generation, documentation, and basic troubleshooting; all traditional tasks available at an entry-level, now fall within AI capabilities.



# The transformation will affect different experience levels uniquely:

## Junior engineers

AI handles tasks traditionally used for learning basics. This accelerates junior engineers into complex problem-solving but requires intentional skill development strategies. Successful juniors become AI power users, leveraging tools to punch above their weight class. While junior engineers who don't embrace self-learning are at massive risk of being left behind.

## Mid-level engineers

Sandbox environments like CDEs allow safe experimentation letting engineers practice with production-like AI integrations i.e RAG, vector DBs, model gateways, without risking live data. You can also pair programming with agents inside the CDE to accelerate skill acquisition while keeping guardrails intact.

## Senior engineers

Experienced professionals find AI amplifies their impact. Strategic thinking, architectural decisions, and complex problem-solving remain distinctly human. AI handles implementation details, freeing seniors for higher-value work.

### THE CRUCIAL MESSAGE

AI proficiency must become mandatory. As one respondent noted, engineers skilled with AI tools will outpace those without. Platform teams must model this adaptation, demonstrating how AI enhances rather than threatens professional growth.



# Five key recommendations to *guide platform teams* toward effective AI integration into AI-powered platforms

If this report hasn't emphasised it clearly enough, AI adoption alone does not guarantee impact. To unlock real value, organizations need deliberate strategy, disciplined execution, and the right mindset. Based on the survey insights alongside emerging practices from top platform engineering experts and orgs, here are key recommendations to help ensure that platform teams turn AI from fragmented experiments into resilient, enterprise-wide capabilities.





# Build *foundations* before intelligence

The inconvenient truth is that AI amplifies existing problems rather than solving them. Organizations with chaotic deployments, unclear ownership, and technical debt often find that AI makes things worse, not better, as generative AI accelerates the accumulation of technical debt, and autonomous agents making decisions require clear boundaries and well-defined systems.

To prevent this, platform teams must first establish strong foundations, including

clear service ownership and boundaries, an established platform-as-product mindset, robust CI/CD pipelines with quality gates, comprehensive observability and monitoring, well-documented APIs and interfaces, and effective incident management processes.

Only with these fundamentals in place can AI enhance rather than complicate platform operations; teams that rush into adoption without them risk messy expensive failures.

## Start with clear intent and measurable outcomes

Many organizations adopt AI tools hoping for magical improvements. This approach guarantees disappointment. Successful integration begins with specific objectives and success metrics.

### Define clear intentions:

What specific problems will AI solve?

What metrics demonstrate success?

Which workflows will improve and how?

How will ROI be calculated?



# Move beyond vanity metrics

## DORA metrics

Deployment frequency, lead time, MTTR, change failure rate

## Business impact

Feature delivery speed, customer satisfaction, revenue impact

## Developer productivity

Time saved on specific tasks, cognitive load reduction

## AI-specific metrics

Model accuracy, hallucination rates, adoption percentages

Without clear intent and measurement, AI initiatives devolve into expensive experiments with unclear value.

# Embrace Platform as Product thinking for AI

**Though this is most obvious with platforms built for AI. It is important to ensure that product thinking continues to reign supreme whether it's an AI powered platform or a platform for AI.**

Platform engineers must control the flow of AI integration into platforms, ensuring that they are operating from a standpoint of best practice, enterprise value, security and effective governance. They must utilise product thinking to balance the needs, and wants of the platform's customers, and its other stakeholders like security and leadership - who may have a


very different perspective on AI integration.

Each persona brings unique requirements, and success depends on conducting user research to understand those needs, iterating development based on feedback, providing clear documentation and onboarding, defining success



metrics for each user type, and running regular reviews to adapt to changing expectations.

Platform teams must also resist the urge to build everything, instead focusing on capabilities that provide maximum value across user groups, for example, prioritizing RAG infrastructure that benefits all personas rather than specialized tools that serve only a single use case.



The platform-as-product philosophy is especially critical for AI platforms where teams must understand their expanding customer base and evolving needs. New platform customers now include traditional developers using AI-enhanced tools, data scientists requiring ML infrastructure, ML engineers needing model deployment pipelines, business analysts leveraging data platforms, and AI researchers demanding specialized compute.



# Implement gradual, pragmatic governance

The tension between innovation and governance requires a delicate balance. Overly restrictive policies kill innovation; absent governance creates chaos. Pragmatic governance evolves through stages.

## STAGE 01

### Enable experimentation

Create sandboxes for safe AI exploration. Define clear boundaries between experimentation and production. Encourage learning while protecting critical systems.

## STAGE 02

### Establish guidelines

Develop practical guidelines based on actual usage patterns. Focus on principles over prescriptive rules. Emphasize outcomes rather than methods.

## STAGE 03

### Automate enforcement

Implement Policy as Code for consistent application. Build observability revealing actual AI behavior. Create feedback loops improving policies based on reality.

## STAGE 04

### Continuous evolution

Treat governance as a product requiring iteration. Regular reviews adapt to new capabilities. Balance risk management with innovation enablement.

This gradual approach prevents the common trap of either blocking AI entirely or allowing unrestricted usage.





# Invest in your team's evolution

Like most things in platform engineering (and tech in general), it is the human element that will determine AI success more than technology choices. Platform teams must model the transformation they enable for others. Investment strategies include:

## Dedicated learning time

Reserve 20% of time for AI skill development. This isn't optional, it's survival. Skills decay rapidly without continuous learning, especially as AI abilities and best practices change rapidly over time.

## Mentorship networks

Pair AI-experienced members with learners. External mentors provide fresh perspectives. Reverse mentoring lets juniors teach AI tools to seniors, while potentially helping bridge the education gap juniors face with increased AI adoption.

## Rotation programs

Rotate team members through AI-focused projects. Exposure to different AI applications builds versatility. Cross-functional collaboration with data science teams accelerates learning.

## Failure celebration

Create safe spaces for AI experiments to fail. Document learnings from failures publicly. Reward innovation attempts, not just successes.

## Career pathing

Define clear progression incorporating AI skills. Show how AI expertise enhances career opportunities. Provide resources supporting individual growth plans.

### REMEMBER

Someone skilled with AI tools will outpace those without. Make sure your team members are the ones doing the outpacing.





# Final outlook: What will the *future* bring?

Platform engineering's future intertwines inextricably with AI evolution. The next 3-5 years promise fundamental shifts in how platforms operate, teams organize, and value gets delivered on a massive scale.





# Three converging trends will reshape platform engineering by 2028

## Ubiquitous AI integration

Current projections suggest the vast majority of software development will incorporate AI by 2025. For platform engineering, this means AI becomes as fundamental as version control. Every platform capability from deployment to monitoring will include AI enhancement. Platforms without AI will seem as outdated as those without automation today.

## Self-evolving systems

Platforms will progress from self-service to self-evolving. Today's platforms require human configuration. Tomorrow's platforms will observe usage patterns, identify optimization opportunities, and implement improvements autonomously. A platform might notice deployment patterns and automatically adjust resource allocation, or detect security vulnerabilities and patch them before human awareness.

## Human-AI collaboration redefined

The “human in the loop” evolves from reviewer to strategist. Platform engineers won't review every AI decision, the volume makes this impossible. Instead, they'll set objectives, define constraints, and audit outcomes. This shifts focus from operational tasks to strategic platform evolution.

At the same time, Agentic AI will transform platform engineering by moving beyond discrete tasks to managing entire subsystems. Future agents will orchestrate deployments across environments, negotiate resource allocation, design and implement new features, and

collaborate with other agents on cross-cutting concerns. Instead of operating with a narrow, task-specific context, AI will maintain organizational memory tracking historical outcomes, service relationships, business objectives, and even cultural norms. With this understanding, decisions will align



with both technical efficiency and organizational values. It also seems likely that synthetic environments will complete the picture. AI will generate realistic workloads, failure conditions, and user behaviors, enabling testing at a scale and fidelity impossible with manual methods. Together, these capabilities will redefine how platforms are built, validated, and evolved.

However, an uncomfortable truth lurks beneath the enthusiasm for AI: massive energy consumption from training and running large models. Platform engineers, as infrastructure gatekeepers, must balance capability with responsibility by designing efficient model serving, applying intelligent caching, and scheduling workloads to align with renewable energy availability. Transparent reporting on environmental impact will help

organizations track and improve sustainability efforts. Teams that embrace these practices will not only cut ecological costs but also gain a competitive edge as environmental responsibility becomes central to the technology landscape challenges we cannot ignore, even as we push forward.

The opportunity and the challenges are immense. AI-powered platforms can accelerate developers, improve safety, and enable scale, while platforms for AI provide the environments needed to deploy transformative models responsibly. Currently however, at adoption is nearly universal, but maturity lags, and most usage remains tactical, while executives demand strategic impact. This tension between hype and ROI, promise and plateau defines the mission of our time.

AI won't solve fundamental problems automatically, it will amplify them, and as the organizations with strong platform foundations will thrive, those with chaos will find AI accelerates their decline. The future belongs to platform teams that harness AI not just as a tool but as a transformative force balancing innovation with discipline, capability with sustainability.

Those who rise to this mandate will break through the plateau, shaping the next era of intelligent, adaptive platforms. Those who don't will be left behind.





# Appendix

## Survey methodology snapshots

### State of AI in platform engineering survey

- **Participants:** 242 platform engineering professionals globally
- **Roles:** Platform engineers, architects, technical leaders
- **Scope:** 21 questions covering AI usage, challenges, organizational attitudes, and future expectations
- **Purpose:** Understanding current state and future trajectory of AI in platform engineering

### AI infrastructure survey

- **Participants:** 120 platform engineering professionals globally
- **Roles:** Platform engineers, architects, technical leaders
- **Scope:** 9 questions covering AI ownership, orchestration choices, cloud-native integration, CI/CD evolution, cross-team collaboration, infrastructure standardization, and predictions for the future
- **Purpose:** Exploring the maturity of AI infrastructure amongst platform engineering teams



# Glossary of key AI terms in platform engineering

A2A (Agent-to-Agent)	Framework for interactions and workflows between autonomous AI agents across platforms.
ACP (Agent Communication Protocol)	Emerging protocol enabling structured coordination and messaging between AI agents.
AI agents	Autonomous software systems that perceive their environment, make decisions, and act toward goals with minimal human oversight.
AI-native infrastructure	Infrastructure purpose-built for GPU-accelerated training/inference, composability, and advanced MLOps governance.
AI observability	Monitoring and tracing of AI system behavior, including prompts, responses, drift, costs, and explainability.
AI-powered platforms	Platforms that embed AI to enhance developer workflows (e.g., troubleshooting, provisioning, code review).
AI PaaS (Platform-as-a-Service)	Managed service layer abstracting AI runtimes, models, and governance for developers.
Background / async agents	AI agents that run in parallel, progressing tasks autonomously without constant user prompting.
ChatOps	Operating workflows through conversational interfaces integrated with platform automation.
CDE (Cloud Development Environment)	Ephemeral, policy-scoped cloud environments for development with pre-baked tooling and guardrails.
Composable infrastructure	Disaggregated compute, storage, networking, and GPU resources assembled dynamically on demand.



Context isolation	Restricting AI systems' access to specific datasets/resources to minimize leakage or risk.
DORA metrics	Key DevOps performance measures: deployment frequency, lead time, change failure rate, and mean time to restore.
Feature store	Central repository for creating, managing, and serving ML features for training and inference.
GenAI (Generative AI)	AI that generates new content such as text, images, or code, typically using large language models (LLMs).
GenAIOps	Operational practices for generative AI systems across data, models, and runtime governance.
GPU orchestration	Scheduling and managing GPU resources in clusters for AI model training and inference.
Hallucination (rate)	Confident but incorrect outputs from AI systems, often measured as a quality metric.
Inference endpoint	API or service that hosts and serves AI/ML models for real-time or batch predictions.
LLMs (Large language models)	Advanced AI models trained on massive text datasets, capable of understanding and generating human language.
LLMOps	Practices for operating large language models, including versioning, routing, evaluation, and governance.
MCP (Model context protocol)	Standard for connecting tools/data to LLMs with structured, contextualized input.
MLOps (Machine learning operations)	Application of DevOps principles to ML lifecycle management—training, deployment, monitoring.



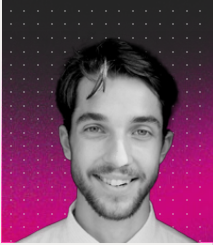
<b>Model drift</b>	Decline in model accuracy over time due to shifts in input data or environment.
<b>Model gateway</b>	Policy and routing layer for managing multiple AI/ML model backends or providers.
<b>Model registry</b>	Central catalog that tracks ML models, their versions, lineage, and promotion status.
<b>Platform as a product</b>	Treating internal platforms as products with defined customers, success metrics, and roadmaps.
<b>Platforms for AI</b>	Platforms that provide infrastructure and tooling for AI/ML workloads (e.g., GPU clusters, MLOps, feature stores).
<b>Policy as code (PaC)</b>	Defining policies in machine-readable form for automated enforcement in CI/CD and runtime.
<b>Prompt engineering</b>	Designing and structuring inputs to guide AI systems toward desired outputs.
<b>RAG (Retrieval-augmented generation)</b>	Framework connecting LLMs to external knowledge bases for accurate, context-rich answers.
<b>Shadow AI</b>	Unauthorized or uncoordinated AI tool usage within organizations, bypassing governance frameworks.
<b>Temperature</b>	A generation parameter that controls randomness; higher values yield more diverse but less predictable outputs.
<b>Ultra Ethernet</b>	Next-gen Ethernet standard designed for high-performance AI and HPC workloads.
<b>Vector database</b>	Specialized database optimized for vector embeddings and similarity search, often used in RAG.





# Authors

This whitepaper was written by Sam Barlien and Luca Galante, with contributions from Ajay Chankramath, Rickey Zachary, and Sylvain Kalache.



## *Sam Barlien*

Sam Barlien is the Head of Ecosystem for the Platform Engineering Community. He is a tech nerd, and has been involved in tech communities for more than 10 years. He helps manage Platform Weekly, co-hosts PlatformCon, and drives the community Ambassador program, blog and Youtube channel. He speaks to 100s of platform engineers a year and translates their experience into articles, webinars and reports for the wider community.



## *Luca Galante*

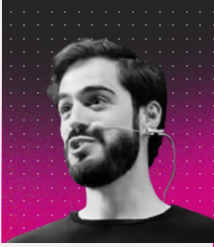
Luca Galante is the Core Contributor to the Platform Engineering community, the world's largest platform engineering community with over 200,000 members. He routinely speaks to dozens of engineering teams every month, and summarizes his learnings and takeaways from hundreds of setups into crisp, insightful content for everyone in the industry, from beginner-Ops to cloud experts. He is the host of PlatformCon, the world's largest platform engineering event, and writes to over 100,000 engineers every Friday in his newsletter, Platform Weekly.



## *Ajay Chankramath*

Ajay Chankramath has over 30 years of technology leadership experience. He co-authored Effective Platform Engineering (Manning) and is a frequent speaker, panelist, and writer on platform engineering and developer productivity. With a strong background in computer science and advanced degrees in business and technology management, Chankramath focuses on driving digital transformation through domain-driven platform engineering and enhancing developer experience at scale.





## *Sylvain Kalache*

Sylvain Kalache is a technologist and strategic communicator with two decades of experience at the intersection of engineering, public relations, and developer advocacy. He leads AI Labs at Rootly, developing AI-driven open-source reliability tools and prototypes, sponsored by Anthropic, Google DeepMind, and Google Cloud.

Previously, he co-founded Holberton School – training highly skilled software engineers in 25+ countries – forging partnerships with leaders such as LinkedIn CEO Jeff Weiner and Grammy-winning artist NE-YO. Graduates have gone on to work at top-tier companies including Apple, Nvidia, and Meta.

Earlier, as a Senior SRE at LinkedIn, he co-designed a patented self-healing infrastructure. He regularly writes for tech publications, including TechCrunch, The New Stack, and VentureBeat.



## *Rickey Zachary*

Rickey Zachary specializes in designing and implementing platform engineering solutions that solve enterprise-scale challenges. With over two decades of experience, he helps organizations establish internal developer platforms that increase productivity, reduce cognitive load, and accelerate software delivery. Rickey guides global clients in creating self-service capabilities, implementing platform governance models, and measuring platform effectiveness through DORA and SPACE metrics.

His industry recognition includes speaking engagements at major technology conferences like KubeCon, Google Next, and AWS Re:Invent, where he shares insights on platform team structures and operating models. His research in platform standardization, developer portals, and golden paths enables organizations to transform their technical foundations while maintaining the balance between developer autonomy and enterprise governance.



# References

Kaspar von Grünberg. [Why platform engineering will eat the world](#). Platform Engineering Blog.

Luca Galante. [AI and platform engineering](#). Platform Engineering Blog.

Patrick Debois. [Why AI needs a platform team](#). PlatformCon 2025

Manjunath Bhat. [How platform teams can help scale generative AI application delivery](#). PlatformCon 2025

Ajay Chankramath. [GenAI in platform engineering: 5 keys to adoption & impact](#). PlatformCon 2025.

Ralf Huuck. [Platform engineering and AI: How to ensure the ROI](#). PlatformCon 2025.

Boyan Dimitrov. [10 years of platform engineering at SIXT: Lessons in scaling and innovation](#). PlatformCon 2025.

Introducing SONIC: [A reference architecture for AI developer experiences in platform engineering](#). PlatformCon 2025.

Kevin Cochrane. [Building AI-Native infrastructure with platform engineering](#). Platform Engineering Blog.

Google Cloud. [What is LLMOps \(large language model operations\)?](#)

Luca Galante. [How to build your platform engineering team](#). Platform Engineering Blog.

Digital Adoption. [What is Meta-Prompting? Examples & applications](#).

Agile Analytics. [How to connect DevEx metrics with business success | Step-by-Step guide](#).

Jellyfish.co. [15 DevEx metrics for engineering leaders to consider: Because 14 wasn't enough](#).

Mia-Platform. [Team Topologies to structure a platform team](#).

Conflux. [Team Topologies in action: Effective structures for machine learning teams](#).

