State of Platform Engineering Report

VOLUME 3

Platform Engineering

SPONSORED BY

humanitec Gitpod StackGen Google Cloud

State of Platform Engineering Vol 3

About the sponsors

humanitec

Humanitec is the leader in the platform engineering space. Our core products, the Humanitec Platform Orchestrator, and Humanitec Portal are used by mid and large-size engineering organizations, from 100+ developer scale-ups all the way to Fortune 100s.

Humanitec also initially developed the OSS workload specification Score, which is a CNCF Sandbox project now. It lets developers describe their workloads and dependencies as code in a platform-agnostic way.

The Humanitec Platform Orchestrator, the graph-based backend of your Internal Developer Platform, dynamically generates app and infra configurations as a Resource Graph with every new deployment, driving standardization across the entire software delivery lifecycle.

This means no more ticket ops or waiting times for developers, resulting in 4x higher deployment frequency and 30% faster time to market.

Gitpod

Gitpod is a platform for automated, standardized development environments that run both on your laptop and in the cloud. Gitpod is trusted by over 1.5 million developers including some of the largest financial institutions in the world. Visit gitpod.io and try it for free with your whole team.

StackGen

Don’t let infrastructure block your teams. The StackGen generative infrastructure platform deterministically generates secure cloud infrastructure from any input - existing cloud environments, IaC, or application code. StackGen adapts to your current workflows—no need to rewrite pipelines. Let StackGen analyze, visualize, secure, generate, and manage your existing environments effortlessly.

Google Cloud

Unburden your developers by shifting workloads and manual toil onto platforms powered by Google Cloud. With a comprehensive suite of managed services and golden paths, Google Cloud makes it easy to build, manage, and scale internal developer platforms.

There isn't a one-size-fits-all solution when it comes to platform engineering—Google Cloud's services, golden paths, and robust infrastructure act as building blocks to help you build a custom platform for your unique needs. By leveraging the power of Google Cloud, your teams can realize the full potential of automation, enhanced security, increased productivity, and accelerated time to market.

Learn more about how platform engineering on Google Cloud can help you increase developer productivity, improve reliability and security, and go to market faster.

Introduction

Since our last report in November 2023, platform engineering has continued to expand, both as a community and as a professional discipline, shaped by the evolving needs of modern organizations. This third volume dives into current trends, from balancing developer experience (DevEx) with scalable approaches to infrastructure orchestration to tackling the organizational challenges that accompany the growth of platform engineering.

One of the key areas we explore is the organizational hurdles around platform adoption, and aligning platform goals with broader business outcomes. With more teams prioritizing Internal Developer Platforms (IDPs), issues like securing executive buy-in, managing tech complexity, and fostering collaborative cultures have gained significance. In response, we are launching an ecosystem of courses, certifications and trainings, helping organizations and individuals build expertise and align with best practices. Additionally, platform engineering salaries are another topic, with data showing a steady increase in compensation as demand for these specialized skills rises across industries.

Executive summary

Community growth and PlatformCon

With over 35,000 registrations and 100,000 views, PlatformCon 2024 showcased the expanding community and strong interest in platform engineering best practices.

Shifting focus on infrastructure

While DevEx remains a priority, the need for scalable, infrastructure-focused solutions has led organizations toward a more balanced approach, supporting both development and operations teams.

Key trends

The rise of platform orchestration and reference architectures demonstrates how teams are standardizing IDP design, using blueprints to streamline platform structure and function.

Platform engineering maturity

Adoption continues to grow, though only a subset of organizations report high maturity with cross-functional collaboration and measurable outcomes, showing varied progress across the industry.

Real-life insights

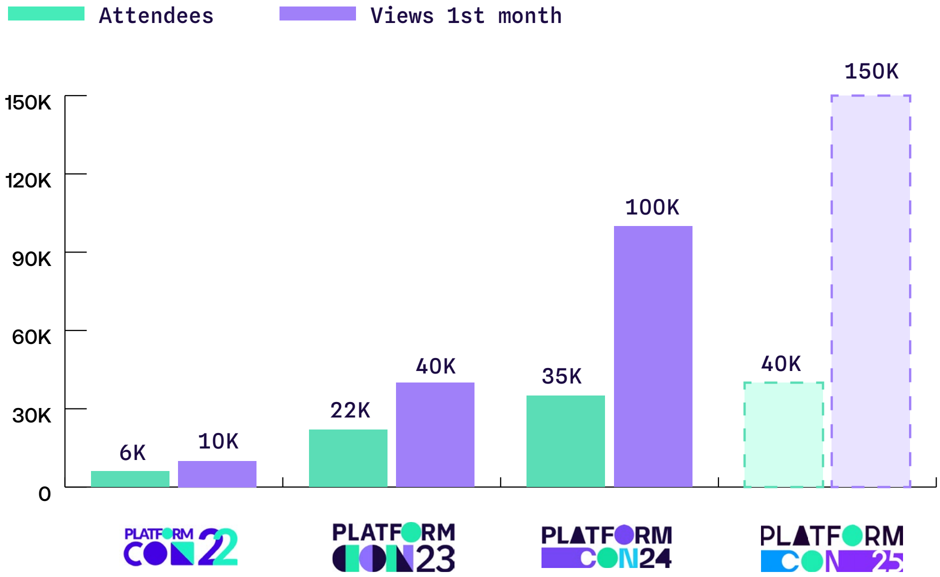
A survey of platform engineers provides data on their main focus areas, job titles, and salary trends. Notably, platform engineering roles continue to command higher salaries than comparable DevOps positions, reflecting the growing demand for these specialized skills.

What happened since Vol 2?

A lot has happened in the last year since the publishing of Volume 2. And there is nothing more representative of the key trends in the platform engineering community and the broader industry than the largest platform engineering event in the world, PlatformCon.

Let's dive in.

COMMUNITY GROWTH AND PLATFORMCON



PlatformCon24 was an absolute celebration, by the community for the community. With 35k+ registrations and 100k+ views on the official [Platform Engineering Youtube]() channel (just in the first 4 weeks), PC24 surpassed any expectation. This growth really speaks to the continuous compounding of the platform engineering space year over year and how much its key value propositions keep resonating with so many in the industry.

But the numbers weren't the only impressive thing at PlatformCon this year. The community experimented with lots of new successful formats: from a day dedicated entirely to hands-on workshops, to an 8+ hour live stream of panels hosted by media partners like DevOps.com, TNS and DZone, to the first ever Live Day, with 300+ people meeting in London for a full day of platform engineering deep dives and networking.

2024 NUMBERS

35K+

Registrations

42K+

Views during PlatformCon

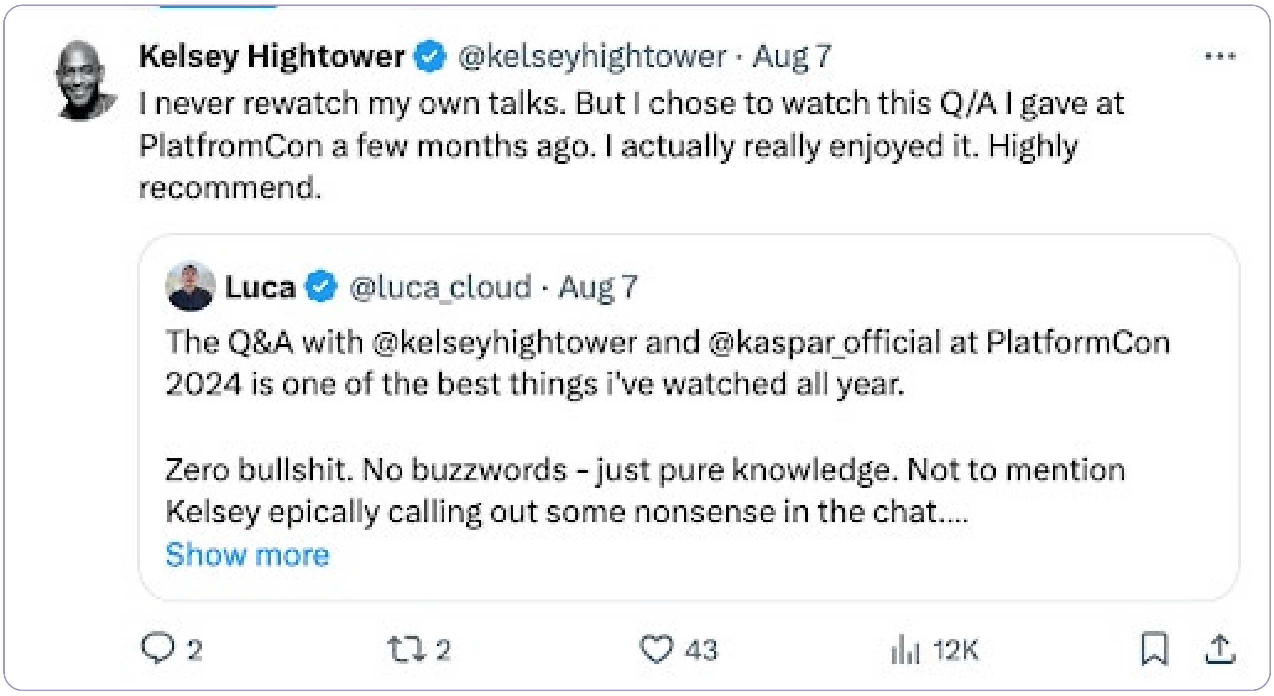
38.5

Attendee NPS Score

1.2M+

Impressions across social media

The quality of the content was truly on a whole new level compared to the previous editions. The [fireside chat between Kaspar von Grünberg and Kelsey Hightower]() opened the dances with a bang, with (un)controversial quotes like "silos are fantastic" that made for a [heated Q&A session]().



Kelsey Hightower @kelseyhightower · Aug 7

I never rewatch my own talks. But I chose to watch this Q/A I gave at PlatfromCon a few months ago. I actually really enjoyed it. Highly recommend.

Luca @luca_cloud · Aug 7

The Q&A with @kelseyhightower and @kaspar_official at PlatformCon 2024 is one of the best things i've watched all year.

Zero bullshit. No buzzwords - just pure knowledge. Not to mention Kelsey epically calling out some nonsense in the chat....

A deluge of 140+ talks followed throughout the rest of the week, with some fascinating key themes emerging and standing out.

Platform as a Product is still king

First popularized by Pais and Skelton in their foundational Team Topologies book, Platform as a Product remains an indispensable feature of platform engineering. It is no surprise that it was a crucial thread throughout PlatformCon 2024, with talks like Platform as a Product: Stories from the Field by Cansu Kavili Örnek and Anna Ciula or Helping a platform team switch to a product based approach by Stéphane Di Cesare, Platform Engineer at DKB (one of Germany largest banks).

Platform Engineering Maturity Model

In her talk, Maturing your platform engineering initiative, Nicki Watt showcased the value of the CNCF platform maturity model and how it can be a useful guiding tool for organizations starting their platform engineering journey.

Where did the Portals go?

At PlatformCon 2023 a third of all talk submissions were about Developer Portals. Despite contrasting opinions on them, with both positive and negative talks on the subject, portals were definitely the "thing" last year. Contrast that to this year where we have witnessed a 90% drop in portal-related talk submissions, with very few talks overall focusing solely on this topic.

A possible answer can be found precisely in these talks. In their discussion "Is platform engineering only about internal developer portals? Think again", Atulpriya Sharma and Vishal Biyani explain how portals should be seen as a useful interface of Internal Developer Platforms, but the majority of the impact of a platform initiative will come from tooling beyond portals. Derik Evangelista emphasized how "Your Backstage needs a Platform", highlighting the risks of betting too much of your platform engineering initiative on portals alone.

New to the party: Infrastructure platform engineering

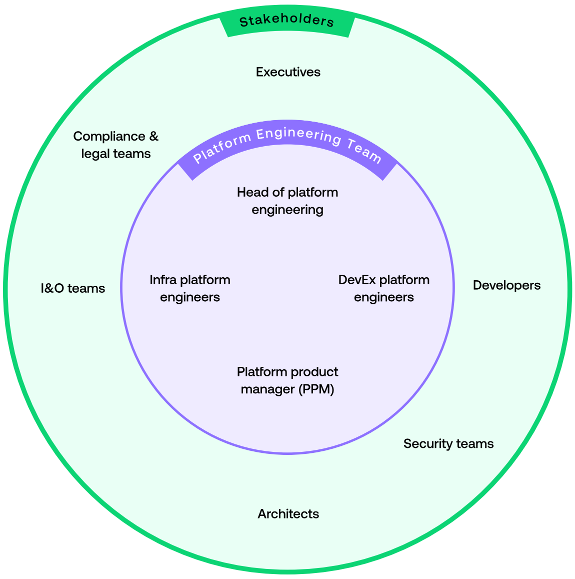
At PlatformCon, Kaspar von Grünberg discussed how an overemphasis on DevEx can lead teams to overlook the broader purpose of platform engineering as a "multiplayer game" that should serve Infrastructure & Operations (I&O) teams as well. Infrastructure platform engineering represents a holistic approach to platform engineering that ensures that all teams can benefit and work efficiently together. The growing interest in the new subdiscipline of infrastructure platform engineering reflects the critical relationship between platform and I&O teams and underscores that I&O buy-in is essential for a platform initiative's success.

WHAT IS A PLATFORM TEAM?

Group of people building, running and continuously improving an Internal Developer Platform

Managing key platform stakeholders throughout the entire organization

Treating the Platform as a Product and developers as customers



Internal Developer Platforms (IDPs) that are overly optimized for DevEx, without tangible benefits for I&O, are likely to fall short. Infrastructure Platform Engineering aims to close this gap by balancing developer and I&O perspectives. In a recent article, we explored the ideal composition of platform teams breaking down how successful initiatives integrate DevEx with Infrastructure Platform Engineering functions to support both developers and operations effectively.

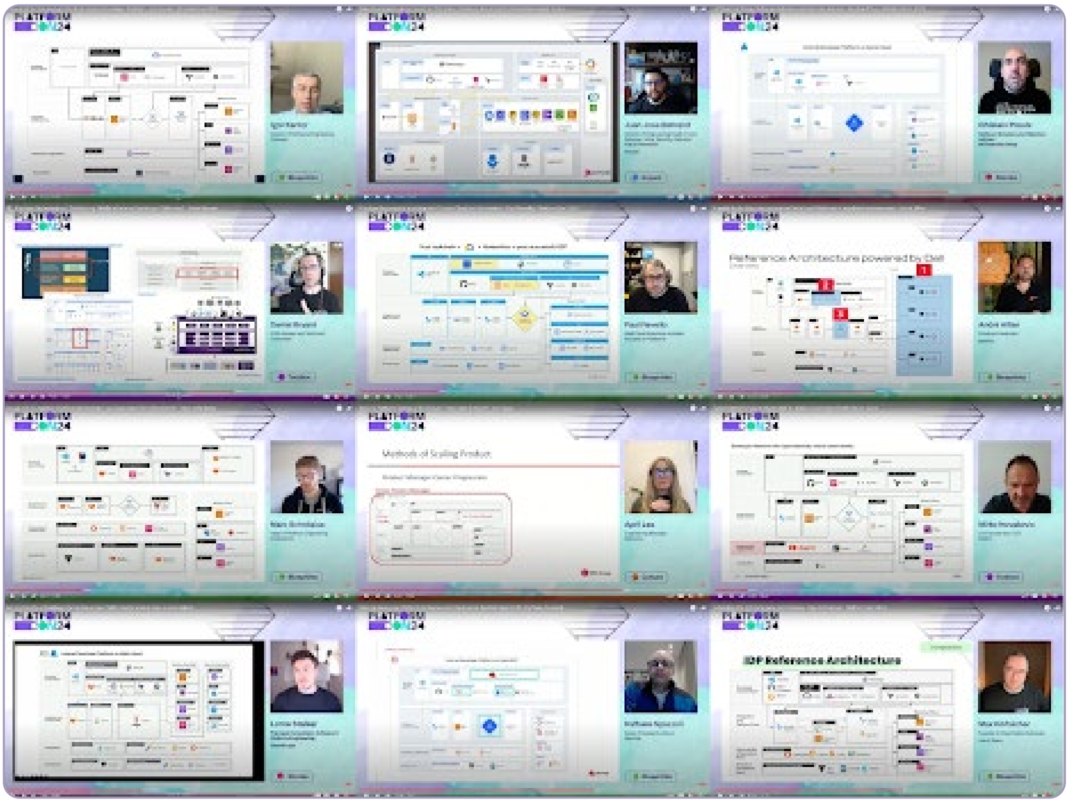
Platform Orchestrators: the missing link

With conversation around Platform Orchestrators growing almost ten-fold since 2023, Platform Orchestrators asserted their position as a key cornerstone of platform engineering. Specifically, as the missing component complementing portals and building the bridge to infrastructure platform engineering. From talks like "Platform Orchestrators: The Missing Middle of Internal Developer Platforms?", showcasing orchestrators as the core configuration engine of their Internal Developer Platform, to deep dives like "Platform Orchestrator: The platform engineering game changer" by Luca Galante on the benefits of graph-based backends, Platform Orchestrators established themselves as the backend of enterprise IDPs. This is an important trend we'll dissect in detail further down below.

Blueprints, blueprints, blueprints

Last year at PlatformCon 2023, we saw the first examples of reference architectures for enterprise-grade IDPs going mainstream, with "Platform as Code: Simplifying developer platform design with reference architectures" by Stephan Schneider and Mike Gatto. It's been truly amazing to see how much of an essential blueprint these have become in the last 12 months. Over 20% of speakers this year used a reference architecture as the guiding standard to talk through the key components and planes of their platforms. This is once again proof of the growing maturity of the platform engineering space, which is quickly consolidating around a few key standards and conventions.

REFERENCE ARCHITECTURES AT PLATFORMCON

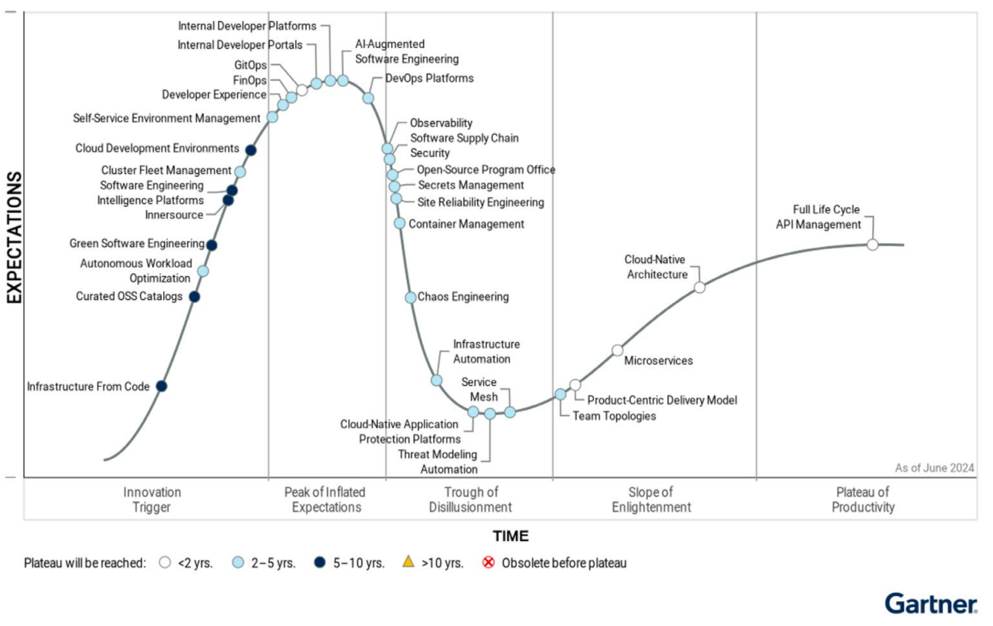


In general, PlatformCon clearly showed the rapid progress of the community in answering crucial questions on how to design IDPs (reference architectures), how to get started building them (backends, see below), and how to think about adoption (Minimum Viable Platforms, more on that later). Platform engineers are hungry for standards to share best practices with one another, and proven frameworks to accelerate their platform engineering journey.

The hype cycle tsunami

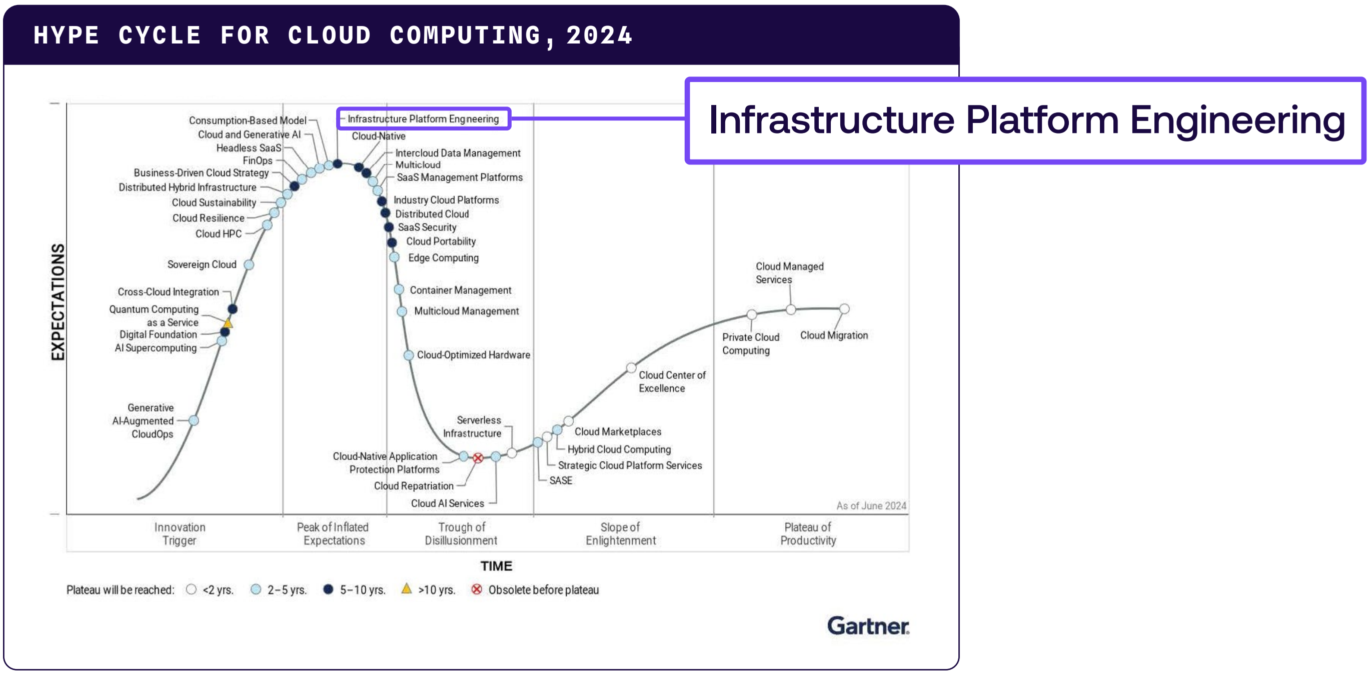
The community and PlatformCon are not the only useful data points to track the growth of platform engineering in the last year. Analysts reports come in handy too. In particular, platform engineering exploded on the Gartner front throughout 2024. While it had already been identified as an up-and-coming trend in two Hype Cycles in 2022 and 2023, this year it's now present across 10 different Hype Cycles and it even has its own Hype Cycle for Platform Engineering:

HYPE CYCLE FOR PLATFORM ENGINEERING, 2024



It is also interesting to see that infrastructure platform engineering, the key new subdiscipline, showed up on 8 Hype Cycles, e.g. for Cloud Computing, SRE, Infrastructure, and Operations or I&O Automation.

HYPE CYCLE FOR CLOUD COMPUTING, 2024



Platform engineering initiatives

As the discipline of platform engineering continues to grow, the platform engineering community has grown to match. The Platform Engineering community launched three major initiatives this year to help nourish the industry's desire for learning, content, and collaboration. These initiatives, a certification track, trainings, and the community ambassador program, are designed to help prevent the same challenges that plagued other engineering hype trains like DevOps and GitOps, while at the same time offering the community more opportunities to work together and grow.

Certification

The first-ever official platform engineering certification track. The track begins with a fundamentals course covering the core concepts of platform engineering like Platform as a Product, platform maturity, golden paths, its relationship to DevOps, Infrastructure Platform Engineering, starting and selling your platform engineering initiative, and much more. It then continues onto an advanced course on how to design, build, and sell your first Minimum Viable Platform (MVP), and the process of internal marketing and selling to secure buy-in and budget to mature your MVP to a fully-fledged Internal Developer Platform. The track then closes with a certification exam to ensure a proper and consistent understanding of these concepts.

Training

As the community and the industry have matured, the race to ensure your organization is effectively following these best practices and standards has advanced as well. Fortunately, the community ecosystem of training and education has grown to match it. By connecting platform engineering experts with organizations eager to better understand and master this growing discipline, the community training services have successfully helped organizations of all sizes, geography, and platform maturity build, launch (or save) platform initiatives effectively while supporting the spread of best practices.

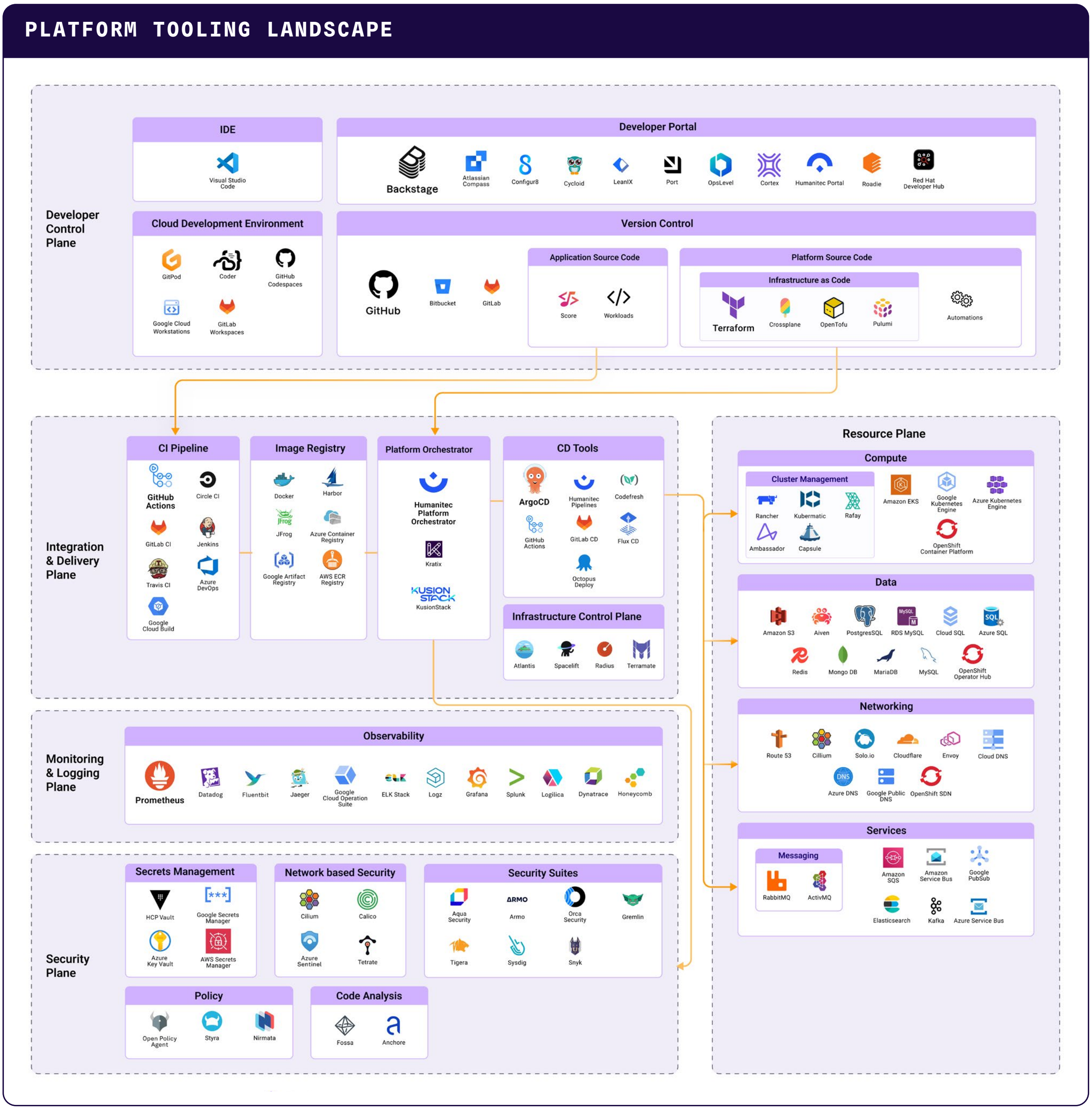
Ambassador program

As a collective effort to grow and nurture the platform engineering space, the Community launched its Platform Engineering Community Ambassador program. It works as an effective way for practitioners and thought leaders in the space to come together and discuss ideas, best practices, and lessons learned and to get support in sharing that knowledge with the wider community. The Ambassador program is a fantastic way to bring together experts and help them gain recognition, new experiences, and to contribute to the community.

Platform architectural patterns

As mentioned above, reference architectures have become one of the foundational standards of the community and the broader platform engineering space. Most platform teams at this point are using them as templates to design, understand, and discuss how the different components of their IDPs interact with one another.

The Platform Engineering Tooling Landscape reflects the same pattern with five IDP planes (Developer Control, Integration and Delivery, Resources, Monitoring and Logging, and Security). It's become an important reference in the industry to map out tooling requirements and integration points for IDPs.



On the same community page, you can find templates to draw your own reference architecture and share it with the community, check them out.

THE 5 PLANES

01 Developer Control Plane

This is the primary configuration layer and interaction point for the platform users. Components include Workload specifications such as Score and a portal for developers to interact with.

02 Integration and Delivery Plane

This plane is about building and storing the image, creating app and infra configs, and deploying the final state. It usually contains a CI pipeline, an image registry, a Platform Orchestrator, and the CD system.

03 Resource Plane

This is where the actual infrastructure exists including clusters, databases, storage or DNS services.

04 Monitoring and Logging Plane

This provides real-time metrics and logs for apps and infrastructure.

05 Security Plane

This manages secrets and identity to protect sensitive information, e.g., storing, managing, and security retrieving API keys and credentials/secrets.

These reference architectures and respective planes are useful for understanding how the different tools fit together. They have been instrumental in helping platform teams ensure they have the same baseline when sharing IDP designs and best practices. However, they don’t tell you much about where you should start when building your platform.

Why not the frontend?

It’s important to address right away one of the clear antipatterns that has emerged in the past couple of years in the platform engineering space. Many platform teams (especially those rolling out IDPs for the first time) who received a mandate and budget for their platform initiative tend to make the mistake of optimizing for short-term gains. They need a win under their belt to prove to management that the platform deserves additional funding, so they look for the quickest way of making an impression on their executives.

That's where they fall for the portal trap. As discussed above, portals are a very important tool in the platform engineer toolbelt. However, it's extremely important to keep in mind the reference architectures from the previous section and remember that portals are just interfaces on top of your Internal Developer Platform. They can't replace the platform itself, they are simply a frontend for developers to access the underlying platform's capabilities.

> **Internal developer portals serve as the interface through which developers can discover and access internal developer platform capabilities.**
>
> Source: A Software Engineering Leader's Guide to Improving Developer Experience by Manjunath Bhat, Research VP, Software Engineering Practice at Gartner. (Full report behind paywall)

Starting to build your platform from the frontend is like starting to build a house by the windows or front door (great analogy from Aaron Erickson, who built the IDP at Salesforce). It's just something you shouldn't do. Especially in the last year, it has become widely acknowledged in the space that building a platform is really not that different from building a regular application. And if there's one thing we all learned in the last decade(s) building app architectures, you should always start from the backend.

That's where the logic modules and the foundation of your house platform need to be designed and configured. You can then add any frontend you like on top of that. But starting from the portal and trying to shoehorn business logic into your platform frontend not only violates the single responsibility principle (from e.g. microservice architectures) and exposes sensitive business logic and potential vulnerabilities, but importantly it also creates a load of technical debt your platform team will have to face sooner or later. If you do get the extra budget from your management for your platform engineering initiative, you'll now have to misuse a tool that's designed to be an interface as a backend logic orchestrator, which it was simply not built for.

Whether you think of it as frontend vs backend or 3 tier architecture (useful visualization from Daniel Bryant's PlatformCon 2024 talk below), what matters is starting from the foundations and defining your infrastructure orchestration and application configuration golden paths first, and only then making your way up the stack to the different interfaces developers can use.

3 TIER PLATFORM ARCHITECTURE

Layer	Why and How?	Who?	What?	Example tech
Application Choreography	"Code, ship, run" Developer Control Plane	App developers, full stack engineers, DevOps, SRES	UI (Portals), CLI, Declarative config *Software dev lifecycle*	Backstage, Heroku CLI & Netflix Newt, Score & KubeVela (OAM)
Platform Orchestration	"Design, enable, optimize" Platform Orchestrator	Platform engineers, Engineering enablement, DevEx engineers, SRES	Platform API *Platform lifecycle*	Kratix Promise, Humanitec Resource Definition, Argo/Flux CRDs
Infrastructure Orchestration/ Composition	"Plan, build, maintain" Infrastructure Control Plane	Platform engineers, DevOps, Operators, Sysadmins, Infrastructure engineers	IaC, Bash scripts, CRDs *Infrastructure lifecycle*	Terraform, Crossplane Managed Resource, Ansible

Daniel Bryant: Platform Engineering: Orchestrating Applications, Platforms, and Infrastructure

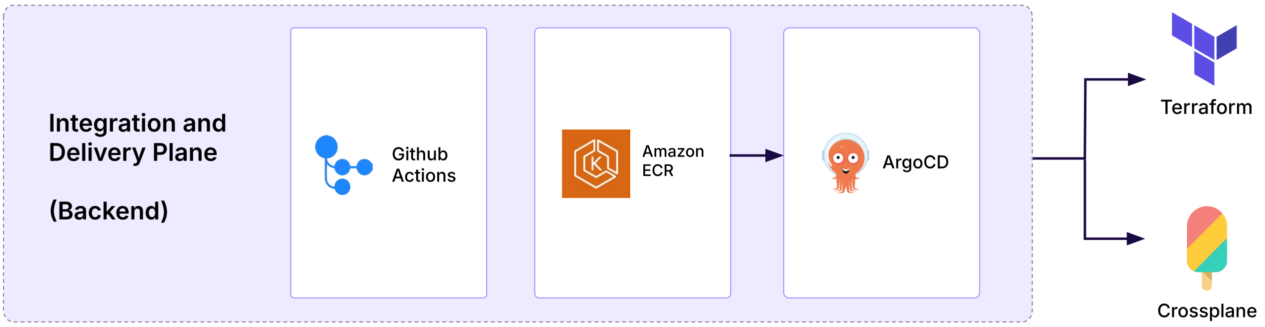
Platform backend designs

Alright, backend first it is. But which design? Two main contenders emerge as common patterns in the industry: pipeline-based and graph-based backends.

Pipeline-based backends

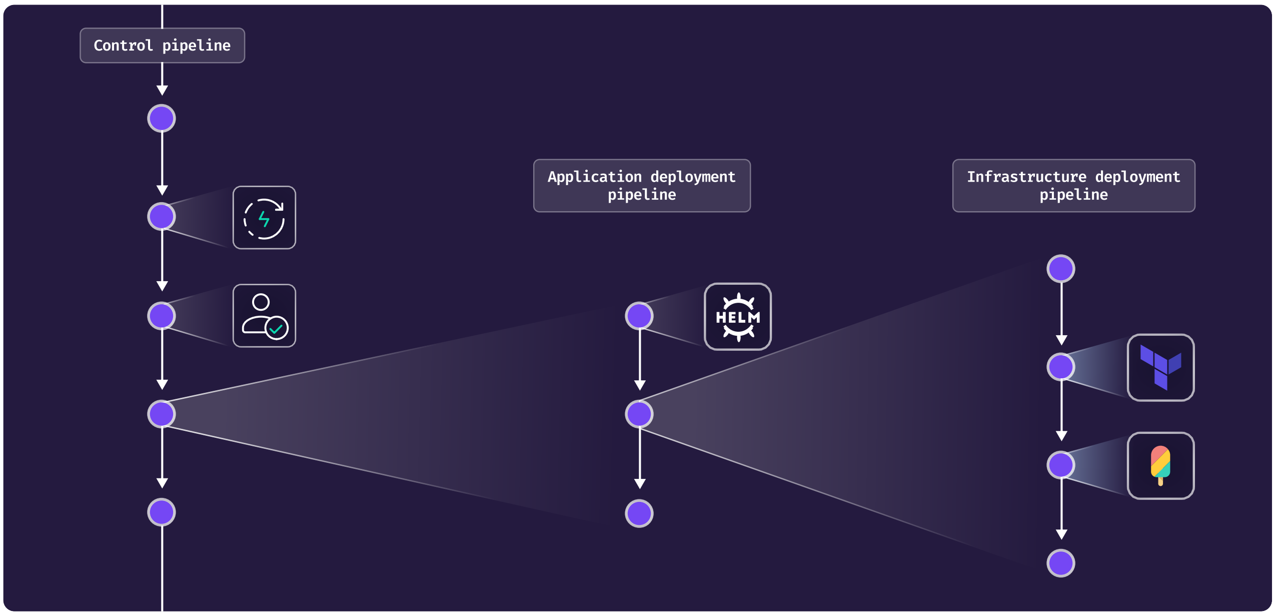
As the name suggests, pipeline-based designs leverage your typical CI/CD pipeline to handle developers' infrastructure requests and app configuration. The advantage of course is that most teams are very familiar with this type of setup. The drawback for this kind of use case however is that these are start-stop systems that are not designed to have any type of advanced logic built into them.

PIPELINE-BASED BACKENDS (CI/CD + IAC)



Business and provisioning logic that goes beyond simple environment progression won't really scale. Or rather, it can scale, but linearly with the complexity of the overall setup. This means that as you want your pipeline-based platform backend to handle more complex use cases, you'll quickly have to resort to nesting pipelines into one another. And while this is not as bad as trying to build your business logic into your frontend, you are once again misusing a system that wasn't designed to handle complex logic that spans tens or hundreds of dev teams and thousands of resources.

NESTED PIPELINES

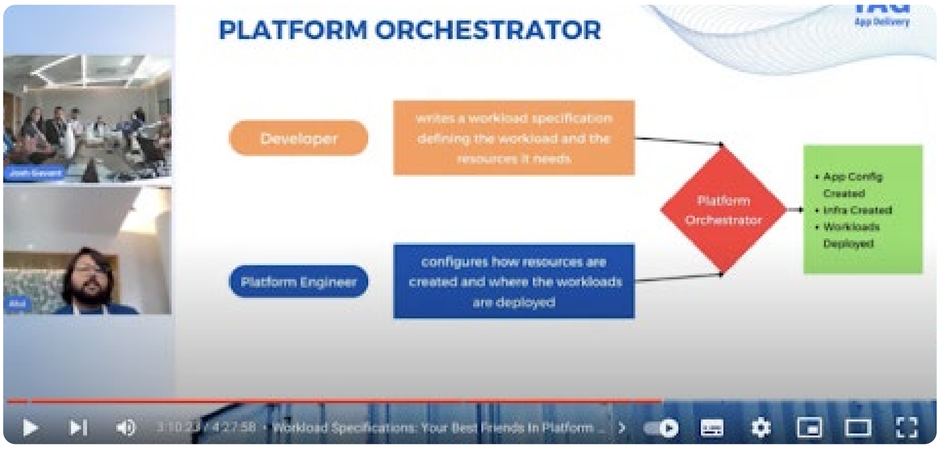

Control pipeline
Application deployment pipeline
Infrastructure deployment pipeline
HELM

The issue here is that you are back at square one, with a setup that's overwhelming for application developers and very hard to maintain and scale for your Infrastructure and Operations colleagues. That is why top-performing platform teams go for graph-based backends.

Graph-based backends

Graph-based backends, also known as Platform Orchestrators, are designed specifically to handle complex business logic and any degree of infrastructure provisioning and orchestration. One of their key advantages is that they allow developers to specify what their workload needs (e.g. through a workload specification like the CNCF project Score, which was recently mentioned in the ThoughtWorks Technology Radar) in an abstract format, without having to worry about any configuration files or implementation details.

GRAPH-BASED BACKENDS → PLATFORM ORCHESTRATORS



PLATFORM ORCHESTRATOR

Developer — writes a workload specification defining the workload and the resources it needs.

Platform Engineer — configures how resources are created and where the workloads are deployed

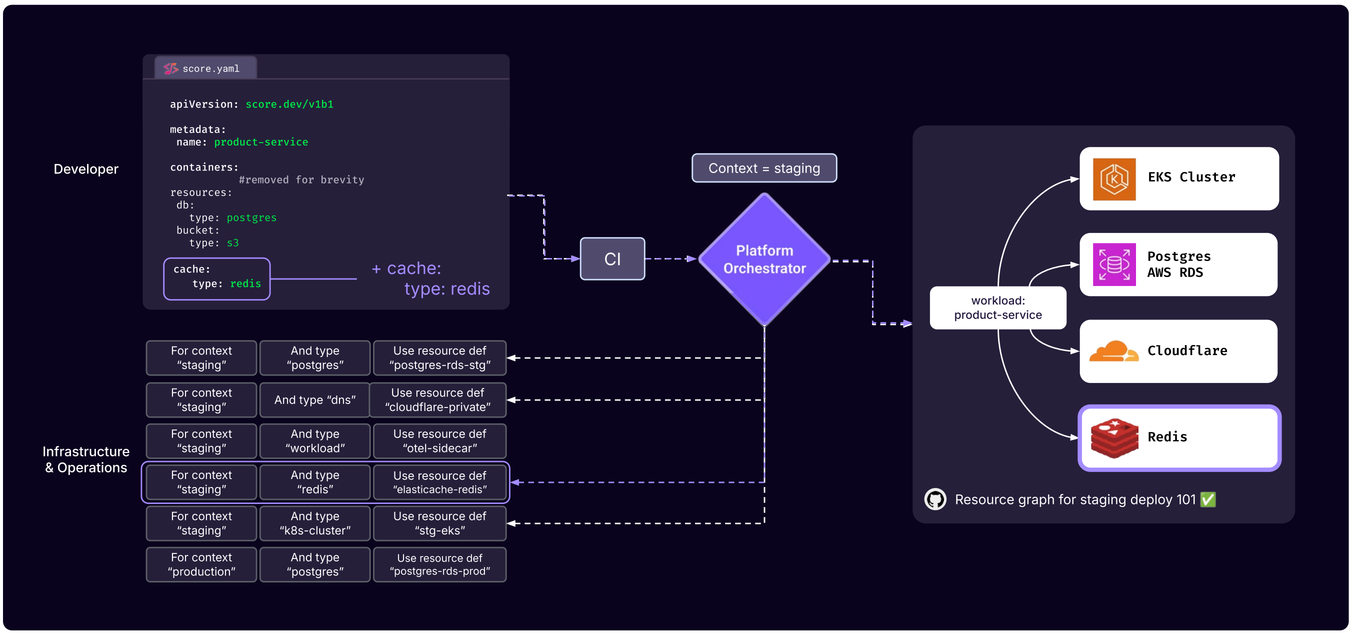
Platform Orchestrator

- App Config Created
- Infra Created
- Workloads Deployed

TAG
App Delivery
3:10:23 / 4:27:58 • Workload Specifications: Your Best Friends In Platform ...

Source: CNCF TAG App-Delivery @ KubeCon NA 2023

The Platform Orchestrator then matches this request to the rules and baseline configuration templates defined by the platform team and creates a graph-based representation (hence the name) of the workload or service and all its dependencies.



score.yaml

```yaml
apiVersion: score.dev/v1b1

metadata:
 name: product-service

containers:
           #removed for brevity
resources:
 db:
   type: postgres
 bucket:
   type: s3
cache:
   type: redis
```

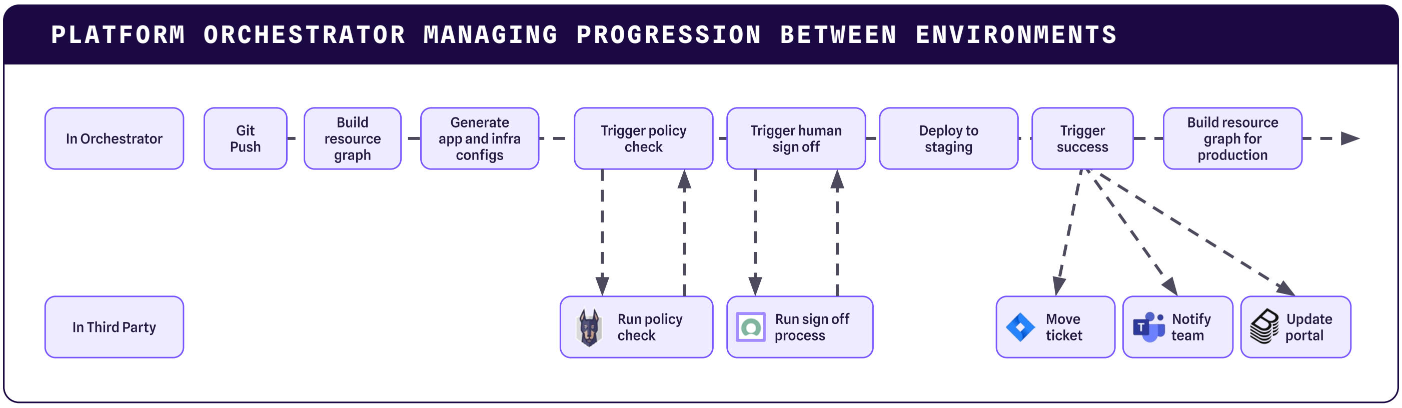
For context "staging"	And type "postgres"	Use resource def "postgres-rds-stg"
For context "staging"	And type "dns"	Use resource def "cloudflare-private"
For context "staging"	And type "workload"	Use resource def "otel-sidecar"
For context "staging"	And type "redis"	Use resource def "elasticache-redis"
For context "staging"	And type "k8s-cluster"	Use resource def "stg-eks"
For context "production"	And type "postgres"	Use resource def "postgres-rds-prod"

Resource graph for staging deploy 101 ✅

The graph is then deployed directly by the orchestrator or handed over to existing deployment solutions (e.g. ArgoCD, FluxCD, Jenkins, etc.) in the form of infrastructure and application configuration files (e.g. Terraform modules, Helm charts, etc.).

This modular architecture lets Platform Orchestrators handle any degree of complexity in terms of deployment and provisioning workflows. These tools however can also manage environment progression, saving extra time on all the handover points between the different stakeholders (whether human or machine).

PLATFORM ORCHESTRATOR MANAGING PROGRESSION BETWEEN ENVIRONMENTS



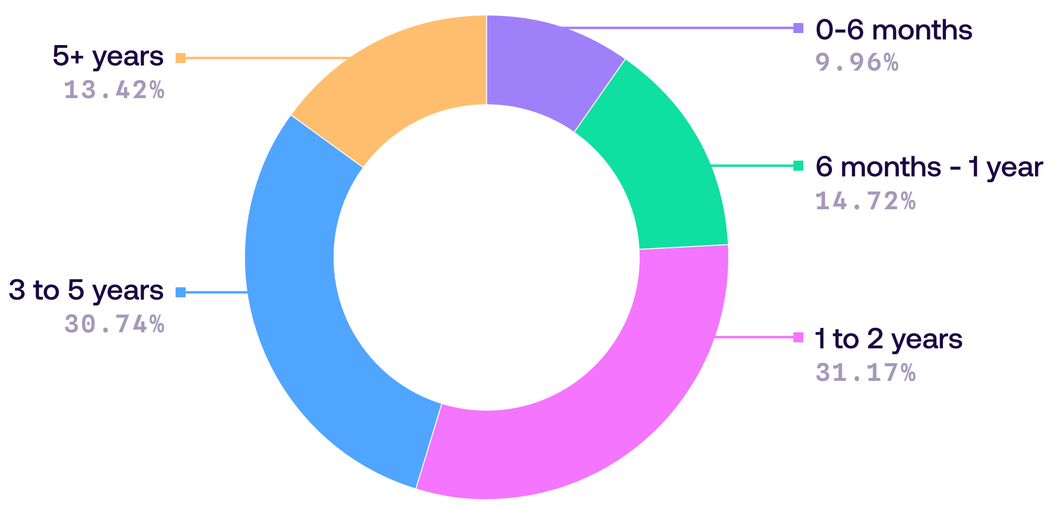
While graph-based backends check all the boxes of good architectural design (API-first supporting multiple interfaces, enterprise features like RBAC and SSO, automatic secrets injections and sign-offs, etc.), their main drawback is they require a new mindset. They are not your familiar CI/CD tooling, and teams can sometimes be scared of adopting this new approach to configuration. It should also be noted that they really only make sense at a certain scale, below 100 developers they are probably overkill.

Platform engineering survey results

New platform engineering initiatives are the majority

281 platform engineering professionals took part in the survey, those without a platform team (38) were filtered out.

HOW LONG HAS THE PLATFORM ENGINEERING TEAM BEEN IN PLACE AT YOUR ORGANIZATION?

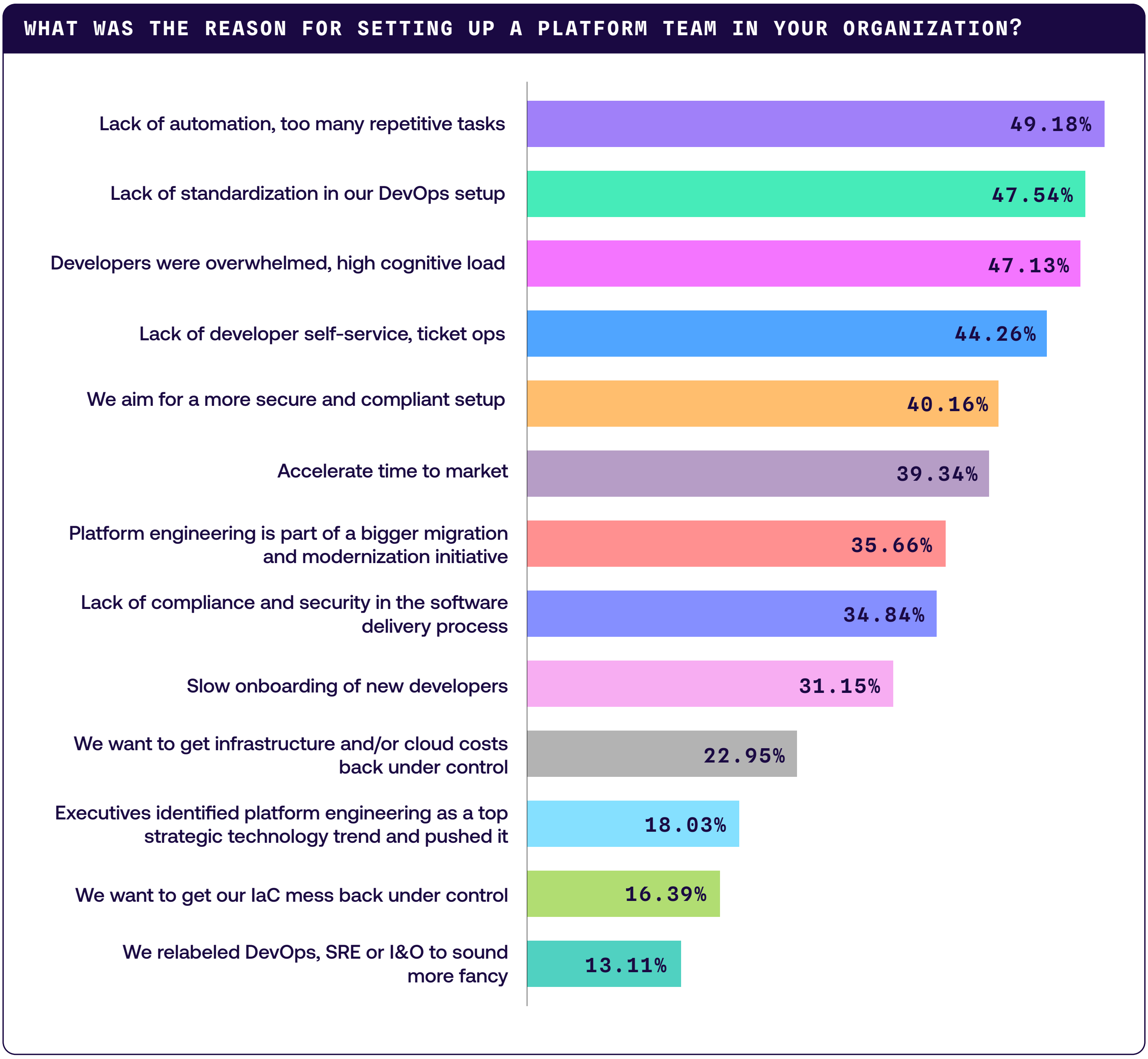


The sum of the percentages for platform engineering teams with a tenure of 0 months up to 2 years is approximately **55.84%**. This indicates that over half of the respondents' platform engineering teams are relatively new, having been established within the last two years. In comparison, only 13.42% have existed for 5 years plus.

55.84%

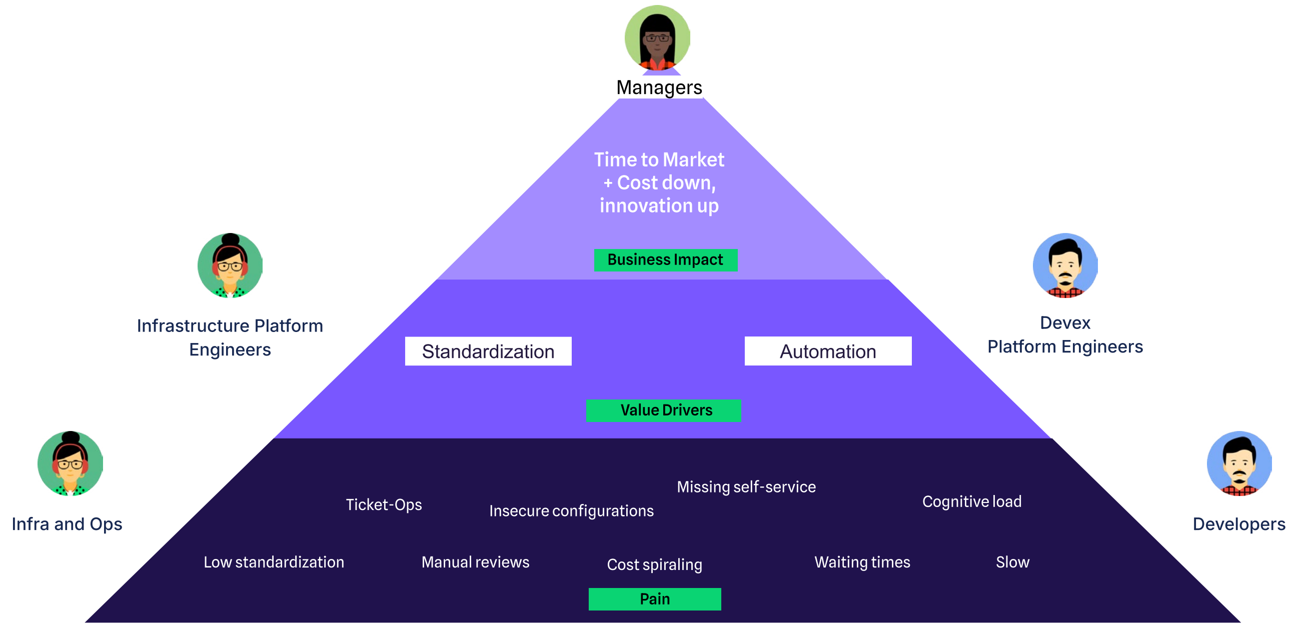
of platform engineering teams have a tenure of 0 months up to 2 years

Automation and standardization are the main drivers



The survey results clearly show that **automation** and **standardization** are the main reasons organizations are setting up platform engineering teams. The most frequently cited motivation was a **lack of automation** and an overreliance on repetitive tasks, with **49.18%** of respondents selecting this as a key driver.

WHY YOUR PLATFORM MATTERS



Close behind, **47.54%** of organizations mentioned the **lack of standardization in their DevOps setup**, which often leads to inefficiencies and fragmented workflows. While **47.13%** of respondents highlighted that their developers were overwhelmed, pointing to cognitive load as the primary challenge they planned to solve.

Other common reasons for establishing platform teams included the absence of **developer self-service** capabilities (44.26%), the need for a **more secure and compliant infrastructure (40.16%)**, and the desire to **accelerate time to market (39.34%)**.

These results heavily emphasize the dual role of platform engineering teams in automating infrastructure while also improving DevEx.

47.54%

mentioned the lack of standardization in their DevOps setup

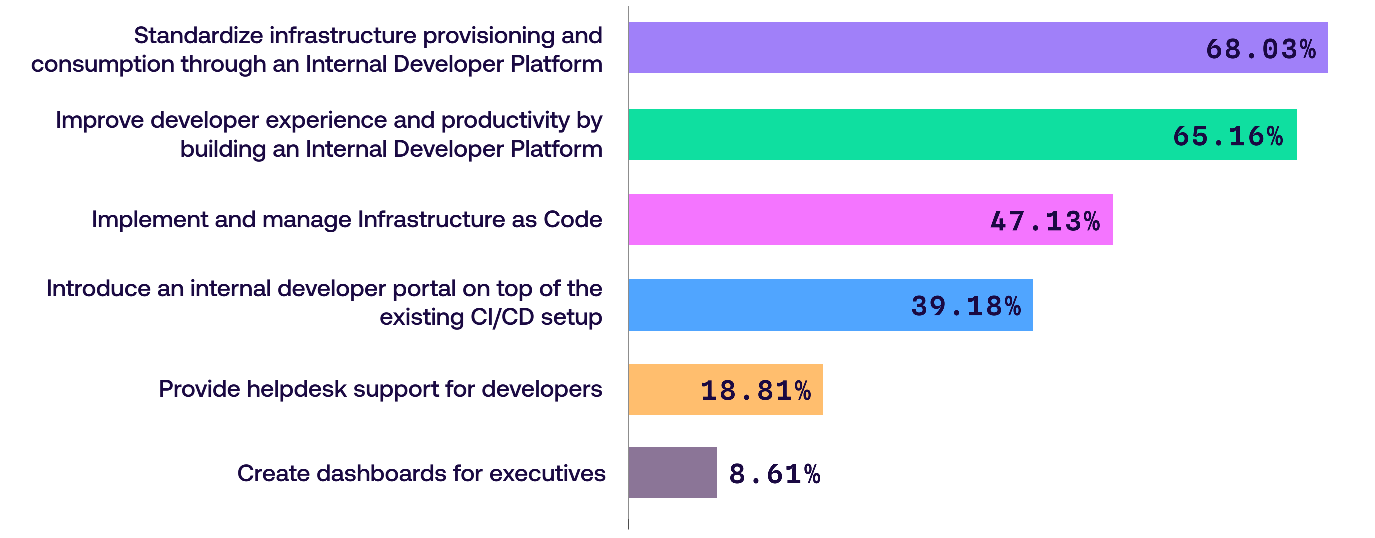
44.26%

included the absence of developer self-service capabilities

Platform engineering teams focus on both: Infra and DevEx

Platform engineering should not solely focus on enhancing developer experience (DevEx), but also ensure that **infrastructure and operations teams** are well-supported. In talks like "You're forgetting Infrastructure Platform Engineering", Kaspar von Grünberg stresses that over-optimizing for DevEx can harm the efficiency of infrastructure teams. He emphasizes the need for balancing the requirements of all teams involved in platform operations, insisting that platform engineering is a "multiplayer game" where all stakeholders need to be considered.

WHAT IS THE MAIN FOCUS OF THE PLATFORM ENGINEERING TEAM?



While earlier efforts overemphasized DevEx, recent trends show a shift towards **infrastructure platform engineering** to address the complexity of cloud environments by optimizing how infrastructure is provisioned, monitored, and maintained. According to the survey, **68.03%** of teams prioritize infrastructure standardization, while **65.16%** focus on enhancing DevEx through Internal Developer Platforms (IDPs).

This holistic view aligns with the survey results and further showcases that many platform teams see both **standardization of infrastructure** and **developer self-service** as part of their goals and remit.

68.03%

of teams prioritize infrastructure standardization

Platform Engineering Maturity

The CNCF Platform Engineering Maturity Model outlines stages for organizations implementing platform engineering practices. It categorizes maturity into four key stages: Provisional, Operational, Scalable, and Optimizing. Each stage is evaluated across dimensions like investment, adoption, interfaces, operations, and measurement, helping organizations assess their current platform engineering capabilities and develop a roadmap for improvement.

Our platform engineering survey asked question based on the platform engineering maturity model, these are the results.

PLATFORM ENGINEERING MATURITY

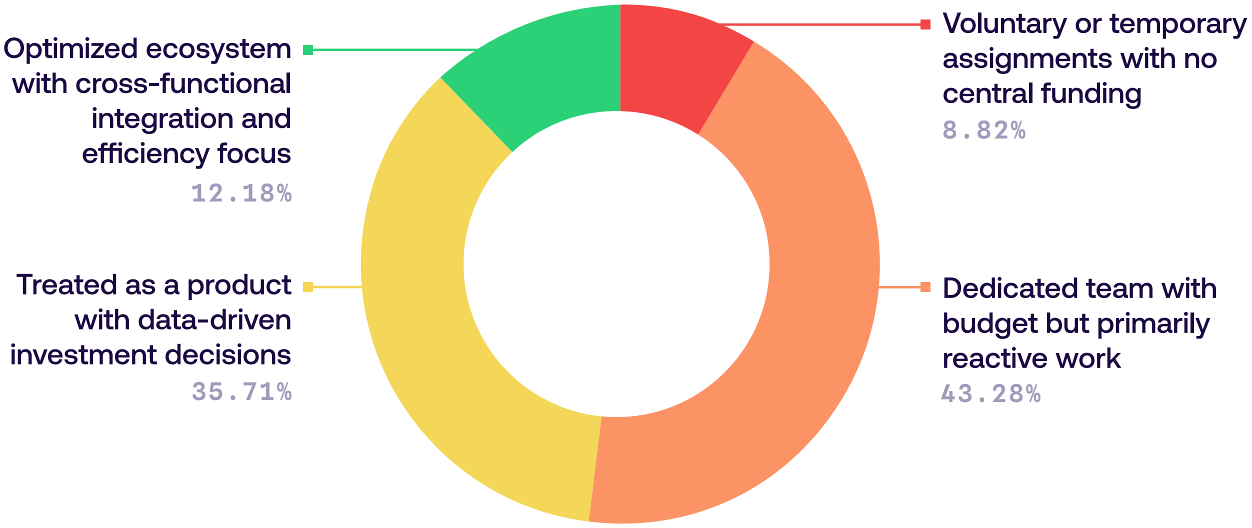
Aspect		Provisional	Operational	Scalable	Optimizing
Investment	*How are staff and funds allocated to platform capabilities?*	Voluntary or temporary	Dedicated team 43.3%	As product	Enabled ecosystem 12.2%
Adoption	*Why and how do users discover and use internal platforms and platform capabilities?*	Erratic	Extrinsic push 35.8%	Intrinsic pull	Participatory 17.3%
Interfaces	*How do users interact with and consume platform capabilities?*	Custom processes	Standard tooling 42.1%	Self-service solutions	Integrated services 9.1%
Operations	*How are platforms and their capabilities planned, prioritized, developed and maintained?*	By request	Centrally tracked	Centrally enabled 39.3%	Managed services 10.7%
Measurement	*What is the process for gathering and incorporating feedback and learning?*	Ad hoc 42.5%	Consistent collection	Insights	Quantitative and qualitative 10.4%

Source: Platform Engineering Maturity Model

The data suggests that the platform engineering industry is still in the process of maturing. The **majority** of organizations are in earlier stages of platform maturity, focusing on ad-hoc operations, extrinsic push for adoption, and limited measurement practices. On the other hand, **high performers** who have reached advanced stages of platform engineering, with optimized investment, intrinsic adoption, integrated interfaces, and comprehensive measurement represent a far smaller group. This indicates that while many companies are embracing platform engineering, only a few are fully leveraging its potential in a highly mature, and scalable way.

Investment

INVESTMENT: HOW ARE STAFF & FUNDS ALLOCATED TO PLATFORM CAPABILITIES?



8.82% of organizations still rely on **voluntary or temporary assignments**, indicating a low level of maturity where investment is not yet a priority.

Most organizations (43.28%) allocate staff and funds to platform capabilities in a **dedicated, but reactive manner**. Fewer organizations (35.71%) treat their platforms as products, with **data-driven investment decisions**. And only a small percentage (12.18%) have reached the highest maturity: an **optimized ecosystem**, integrating cross-functional teams, and focusing on efficiency.

ONLY

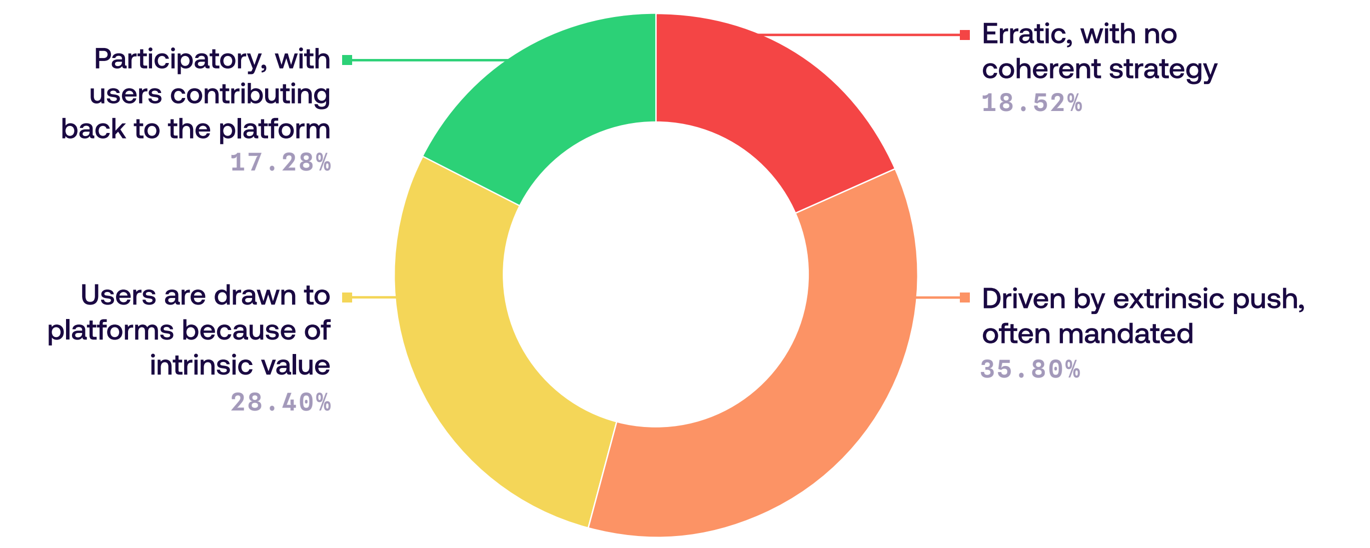
12.18%

of organizations have reached the highest maturity: an optimized ecosystem, integrating cross-functional teams, and focusing on efficiency.

Adoption

Most organizations (35.80%) try to reach platform adoption through an **extrinsic push**, where usage is often mandated. This reflects a low maturity level, where platforms may not yet offer compelling intrinsic value to users.

ADOPTION: WHY AND HOW DO USERS DISCOVER AND USE INTERNAL PLATFORMS AND PLATFORM CAPABILITIES?



A smaller but notable percentage (28.40%) sees **intrinsic pull**, where users are drawn to platforms because they genuinely find them valuable. On the positive side, **erratic** adoption where no clear strategy is reported is at only 18.52%, with a similar result to the most mature organizations (17.28%) having **participatory** adoption, where users actively contribute back to the platform.

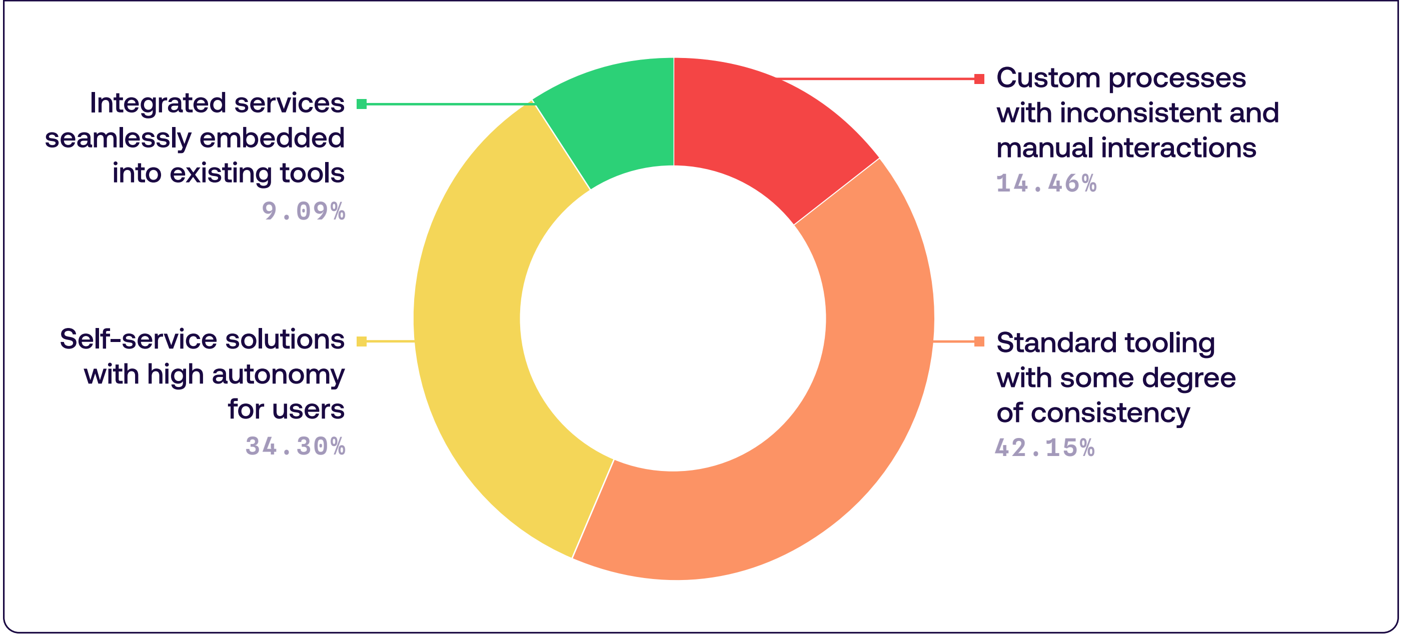
28.40%

see intrinsic pull, where users are drawn to platforms because they genuinely find them valuable

Interfaces

The majority (42.15%) of organizations provide **standard tooling** with some degree of consistency for users to interact with platform capabilities. **Self-service solutions** that allow users high autonomy are utilized by 34.30% of respondents.

INTERFACES: HOW DO USERS INTERACT WITH AND CONSUME PLATFORM CAPABILITIES?



A smaller group (14.46%) still relies on custom processes with inconsistent and manual interactions, and only 9.09% of organizations have achieved integrated services that seamlessly embed platform capabilities into existing tools, reflecting the most mature interface interaction.

This continues the reflection that most platform initiatives are still very immature.

ONLY

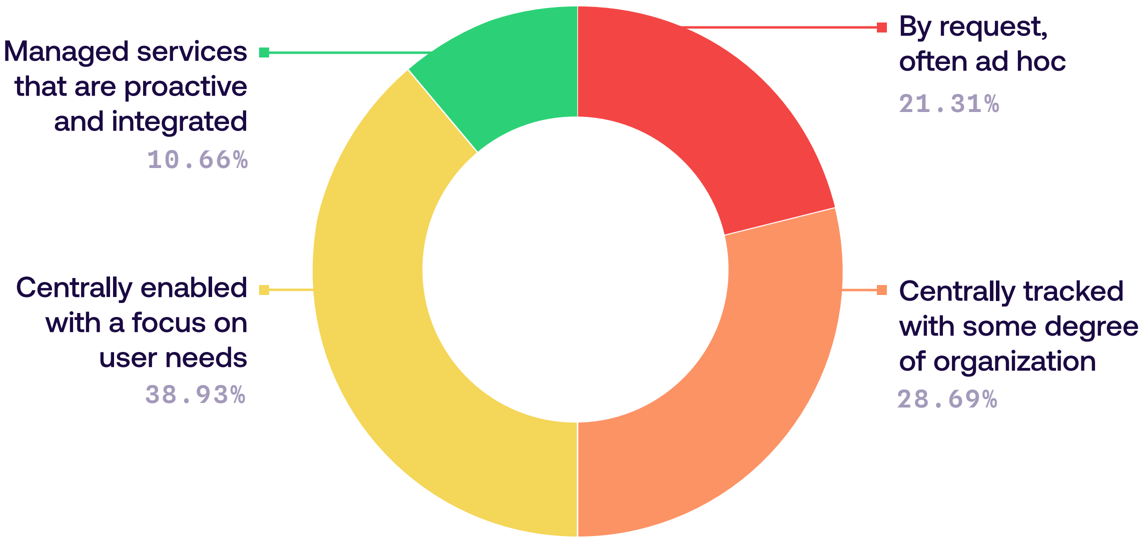
9.09%

of organizations have achieved integrated services that seamlessly embed platform capabilities into existing tools

Operations

39.26% of organizations have **centrally enabled** operations that prioritize user needs. A notable portion **(28.93%)** operates with **centrally tracked** systems, providing some organization but not fully optimized.

OPERATIONS: HOW ARE PLATFORMS AND THEIR CAPABILITIES PLANNED, PRIORITIZED, DEVELOPED, AND MAINTAINED?



Ad hoc operations, which are more reactive and request-driven, represent **21.49%** of organizations. While at the most mature level, **10.74%** of organizations have **managed services** that are proactive and fully integrated, which would indicate advanced maturity in how platform capabilities are maintained and developed.

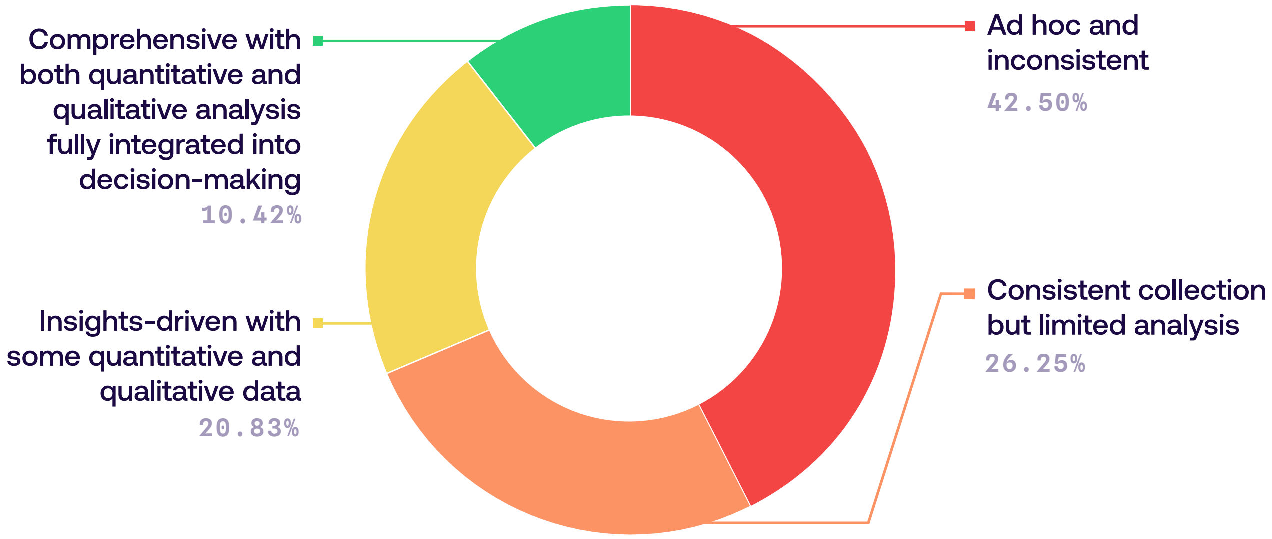
10.74%

of organizations at the most mature level have managed services that are proactive and fully integrated.

Measurement

When we discuss measurement, the majority of organizations (42.50%) have **ad hoc or inconsistent feedback mechanisms**, with only a small percentage adopting fully integrated, data-driven processes (10.42%).

MEASUREMENT: WHAT IS THE PROCESS FOR GATHERING AND INCORPORATING FEEDBACK AND LEARNING?



This conflicts with the **Operations** results from above, where a larger portion (39.26%) of organizations reportedly have centrally enabled operations that focus on user needs, implying a more mature process. The gap suggests that while many organizations may be prioritizing user needs operationally, they are not effectively measuring or analyzing feedback to improve platform capabilities in a structured way.

The gap is an indication that many organizations have a much better image of themselves than the reality.

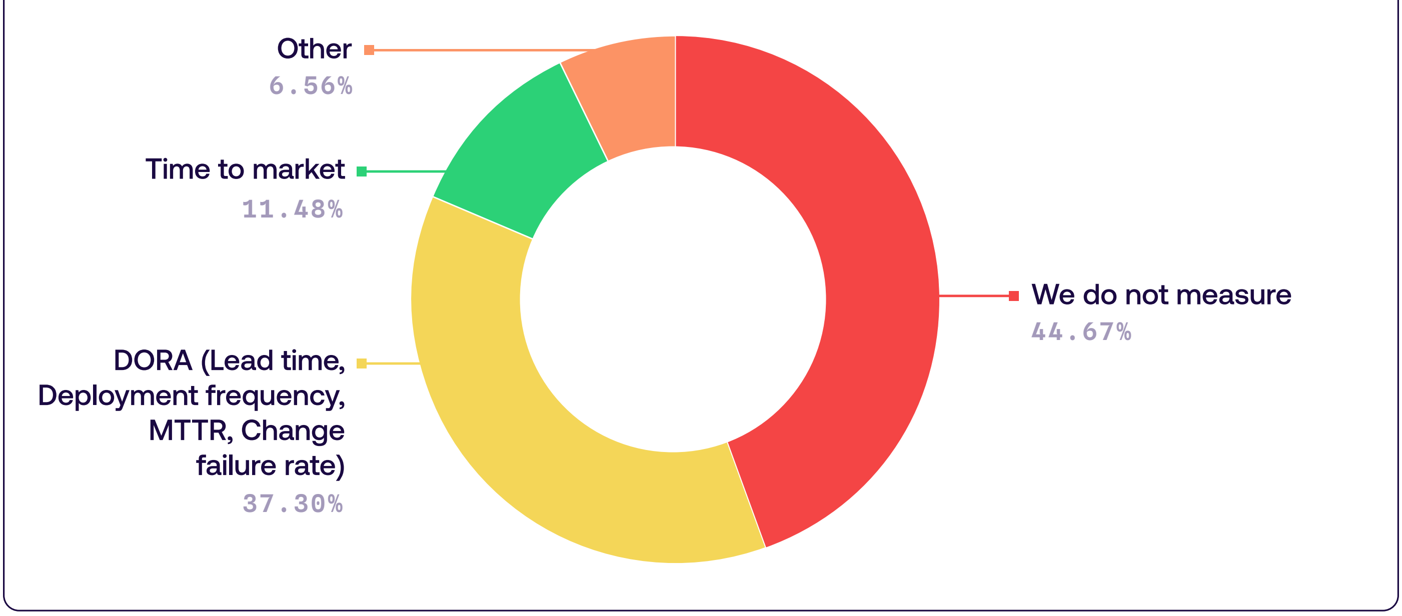
42.50%

which represents majority of organizations have ad hoc or inconsistent feedback mechanisms

Measurement is neglected and inconsistent

The survey results show that **44.67%** of organizations do not measure any metrics, indicating a substantial gap in success tracking for platform engineering initiatives. Meanwhile, **37.30%** focus on **DORA metrics** (deployment efficiency and reliability indicators), highlighting some attention to operational performance.

WHICH METRICS DO YOU MEASURE TO PROVE SUCCESS?

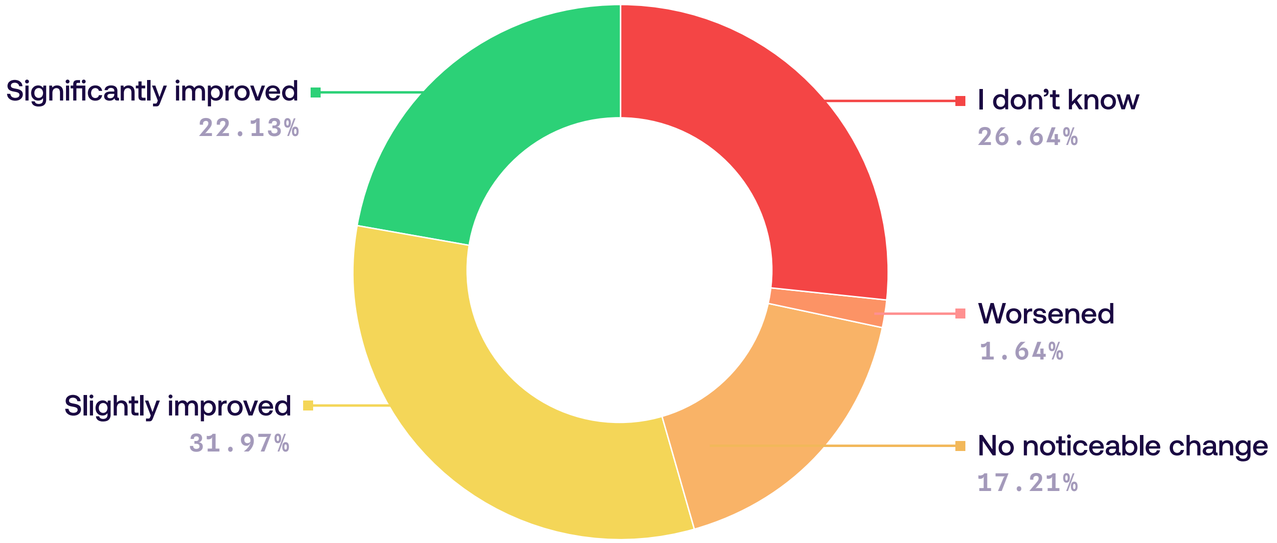


A smaller group (**11.48%**) measures **time to market**, reflecting a limited focus on product delivery speed. Overall, these results suggest that while some organizations prioritize essential metrics, a significant portion lacks standardized performance measurement.

When asked how metrics have improved since organizations introduced platform engineering, notably only 22.13% of organizations report significant improvements, while 31.97% see only slight gains. Concerningly, 17.21% report no noticeable change, and 26.64% are uncertain of any impact, indicating a substantial portion lacks clear evidence of progress. A small fraction (1.64%) even noted worsening metrics. These findings suggest that while some organizations experience benefits, many are still in the early stages, struggling to measure the tangible effects of platform engineering on key performance metrics.

This data reflects the continual challenge teams face in both prioritizing measurement, but also in effectively implementing and ensuring it.

TO WHAT EXTENT HAVE METRICS IMPROVED SINCE YOUR ORGANIZATION INTRODUCED PLATFORM ENGINEERING?

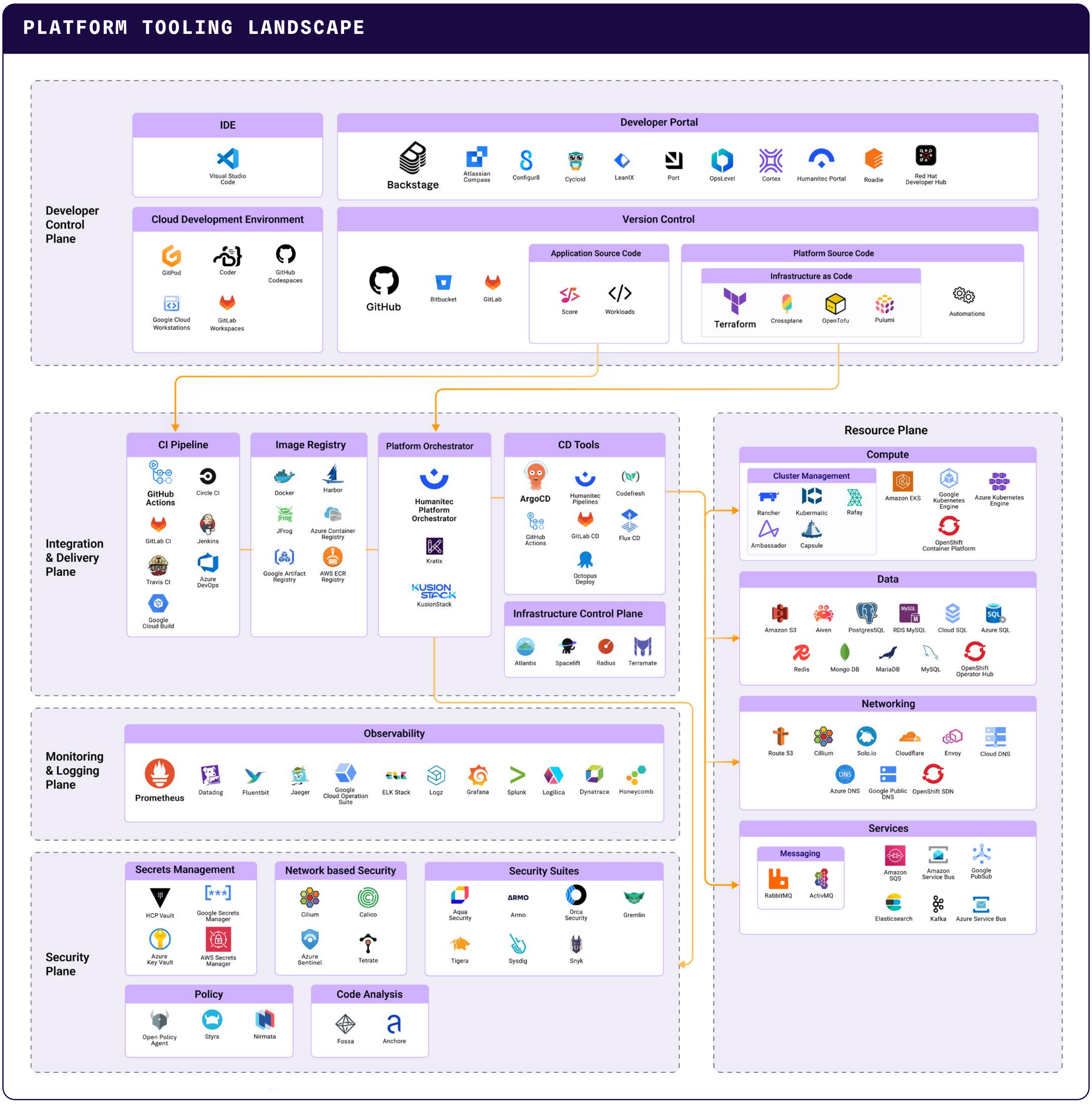


The data reveals an **18.03% discrepancy** between organizations reporting they do not measure any success metrics (44.67%) and those who do not know to what extent metrics changed since they introduced platform engineering (26.64%). This suggests that at least 18.03% of organizations who reported improvements, no changes, or even worsened metrics, may be assessing changes based on anecdotal evidence, informal observations, or ad hoc measurements rather than on formal tracking systems. This gap highlights a potential reliance on subjective feedback where there is a lack of structured metrics.

This emphasizes the clear need for more consistent measurement practices to accurately assess the impact of platform engineering initiatives.

Platform tooling landscape

We published a new version of the platform tooling landscape which now contains a more granular security plane but also a new category: Cloud Development Environments (CDEs), which have gained significant traction and visibility in recent months.



New category: Cloud Development Environments (CDEs)

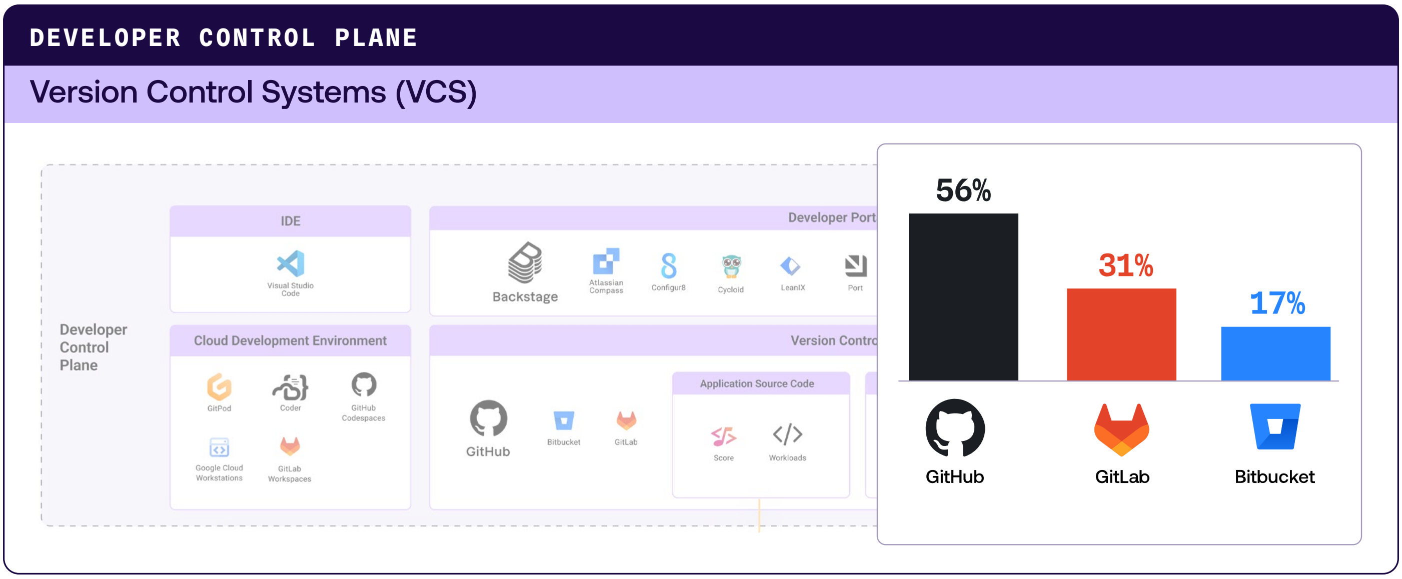
Unlike local dev environments, Cloud Development Environments (CDEs) are setup in an automated and standardized way. They provide developers with pre-installed tools, dependencies, and resources required to code, and provide platform teams a central place to manage the setup, troubleshooting, updating, and security of development environments.

According to Gartner, CDEs enhance productivity, improve security, and support compliance, making them a key driver of efficient software development; this is why CDEs have been mentioned in the Platform Engineering Hype Cycle as well as the subject of a Market Guide.

Most used tools

This year, we surveyed platform engineering teams to learn which tools they use to build their Internal Developer Platforms. We are pleased to share data from selected categories where there were enough responses to identify patterns, though the sample size is still too small to be fully representative. Respondents could select multiple tools within a category, reflecting the reality that many companies use more than one tool due to acquisitions or tool sprawl, even though this is often not ideal.

Version Control Systems



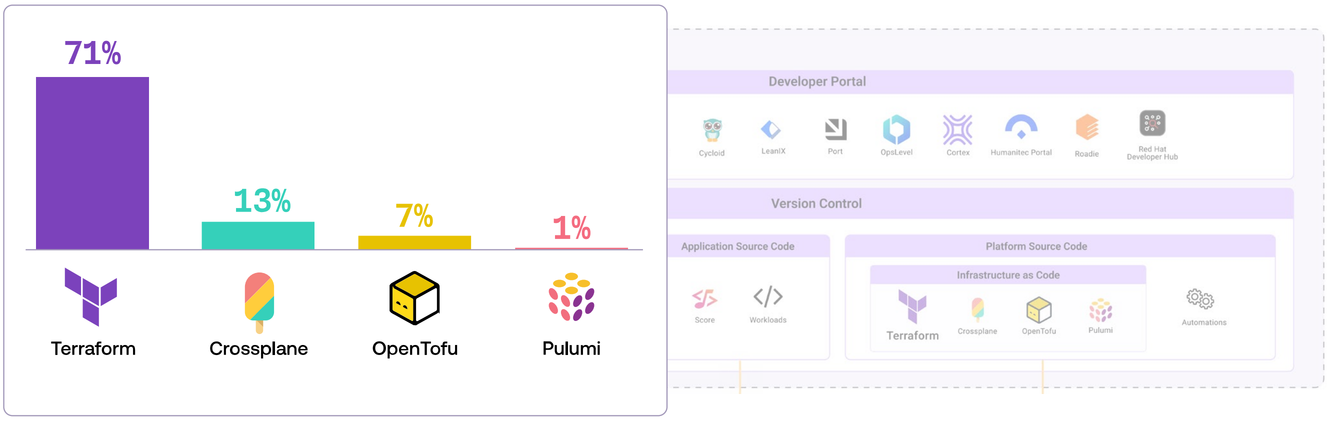
The survey shows GitHub as the most widely used version control system, with 56% of respondents relying on it for their development workflows. GitLab follows with 31%, while Bitbucket (Atlassian) is used by 17%.

Infrastructure as Code

Infrastructure as Code (IaC) serves as the source code of Internal Developer Platforms (IDPs), allowing platform teams to automate the provisioning and management of infrastructure. If implemented well and paired with a platform orchestration tool, IaC ensures consistency and reduces manual errors, making deployments more reliable.

DEVELOPER CONTROL PLANE

Infrastructure as Code

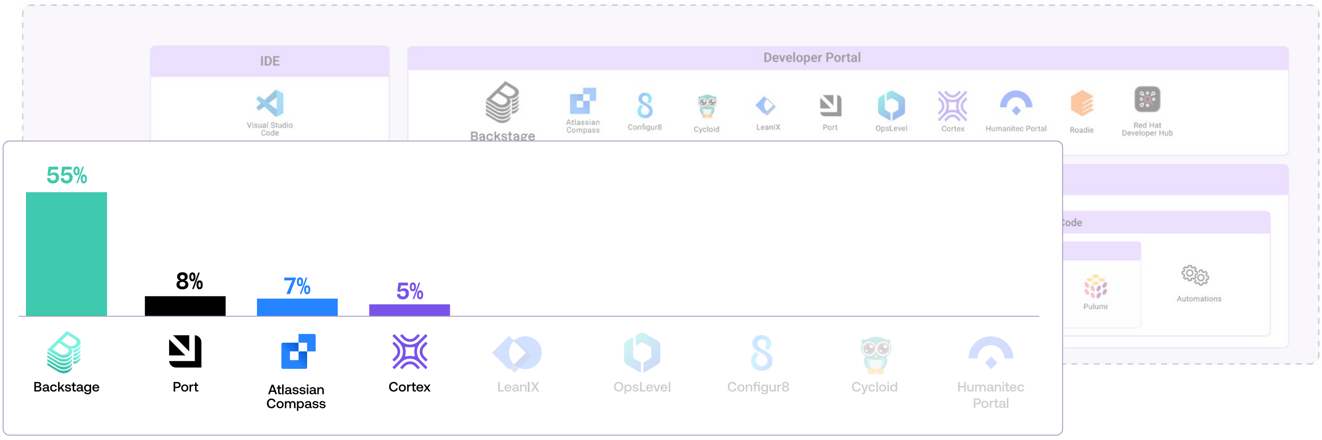


Terraform leads the IaC category, used by **71%** of respondents for its modular, cloud-agnostic features that streamline infrastructure management. Crossplane follows with **13%**, while OpenTofu, adopted by 7%, stands out as the open-source alternative to Terraform, providing similar functionality without vendor lock-in. Finally, Pulumi accounts for only 1% of those surveyed.

Portals

DEVELOPER CONTROL PLANE

Portals

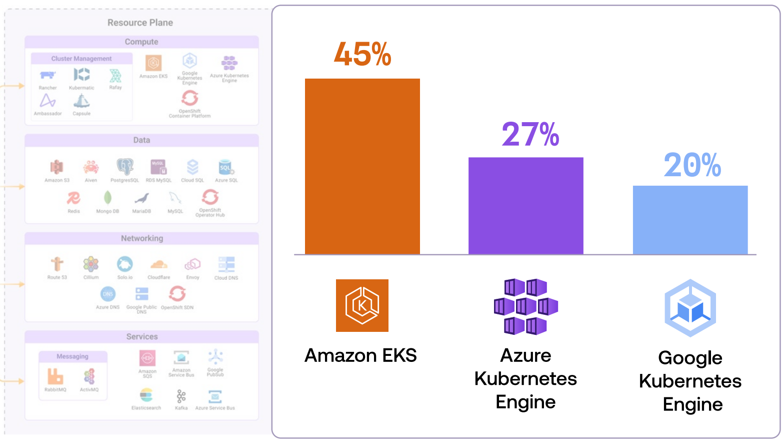


Portals remain one of the most prominent categories in platform engineering, serving as centralized hubs for features like service catalogs, dashboards, and scaffolding. Backstage, developed by Spotify, leads the space with 55% of respondents. Port follows with 8%, and recently raised $35 million in a Series B round, highlighting growing interest and investment in enhancing developer productivity. Cortex, adopted by 5%, has also gained significant traction, raising $60 million in a Series C round.

Managed K8s

RESOURCE PLANE

Cloud Provider - Managed K8s



The adoption of managed Kubernetes services aligns closely with global cloud infrastructure market shares. Amazon EKS leads the way with 45% usage, reflecting its dominance within Amazon Web Services (AWS), which holds a 31% cloud market share.

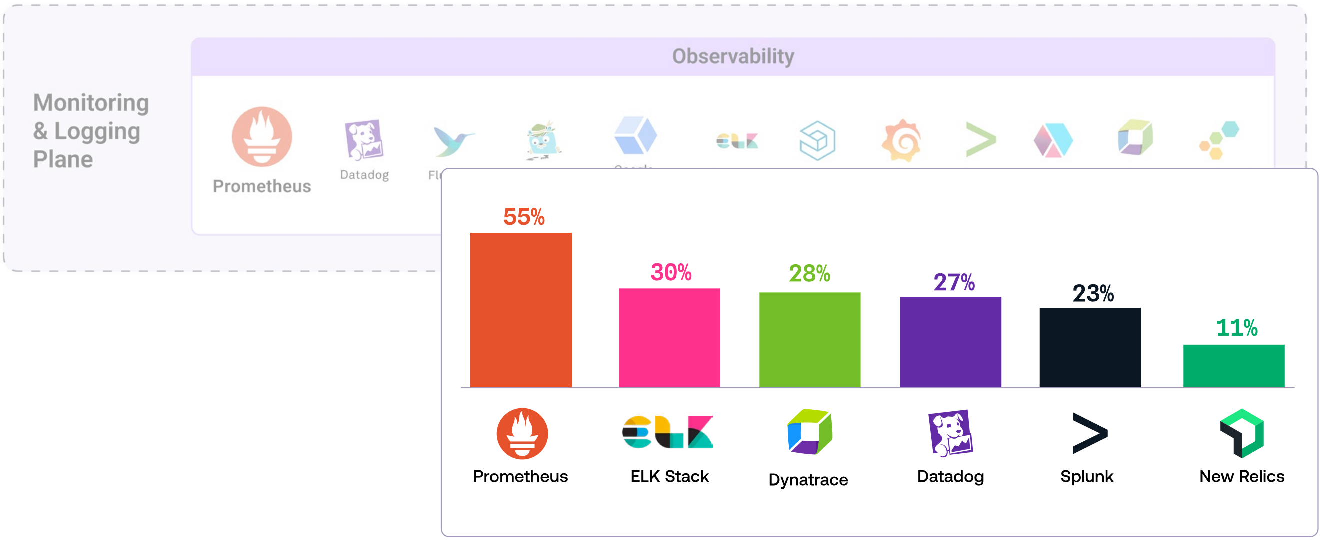
MARKET SHARE CLOUD INFRASTRUCTURE

	Amazon Web Services (AWS)	Microsoft Azure	Google Cloud Platform (GCP)
Org Size Tendencies	Startups, SMEs, large enterprises	Mid-sized, large enterprise	Startups, SMEs, tech-focused
Org Age Tendencies	Newer and older companies	Older, established enterprises	Younger, tech-savvy companies
Market share **Source: Statista**	31%	21%	10%
Key industries	Financial services, media, government	Government, healthcare, manufacturing	Retail, e-commerce, media, gaming, and data-centric sectors

Observability

MONITORING AND LOGGING PLANE

Observability

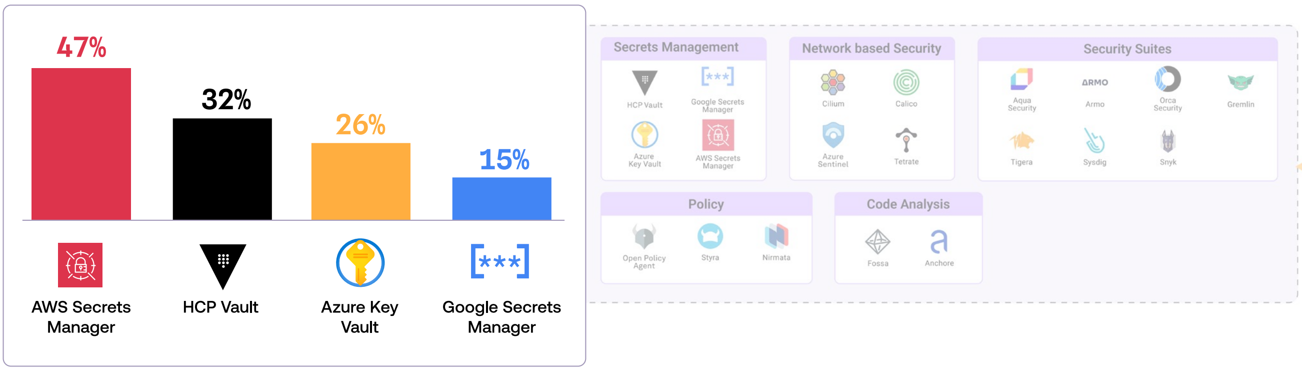


The survey data on observability tools shows a clear preference for Prometheus, with 54.59% of respondents using it, reflecting its popularity for real-time, customizable metrics monitoring, especially in Kubernetes environments. Dynatrace (28.38%) and Datadog (27.51%) follow closely, while ELK Stack (29.69%) remains a go-to for log management, and Splunk (23.14%) and New Relic round out the group at 11.35%. This diverse toolset highlights the balance between open-source flexibility and the advanced capabilities of commercial platforms.

Secrets management

SECURITY PLANE

Secrets management

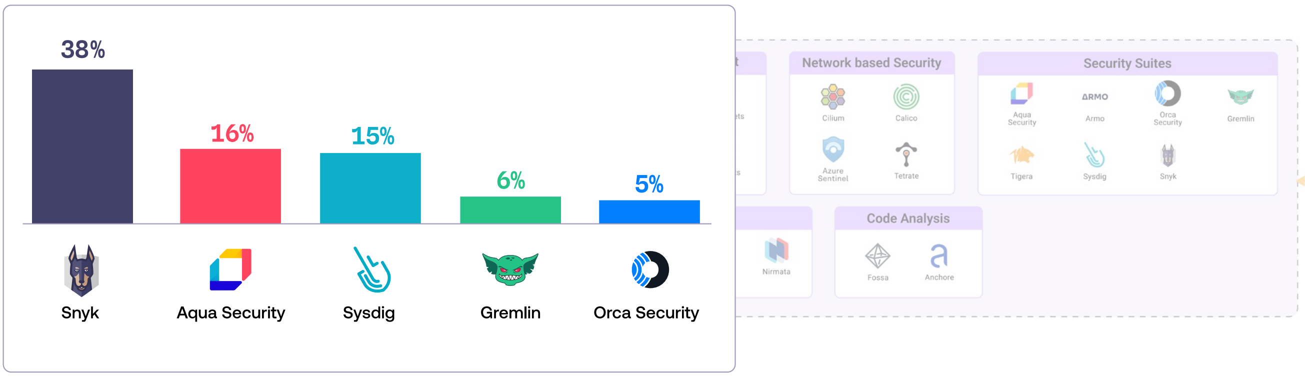


In the realm of secrets management within platform engineering, AWS Secrets Manager leads with 47% adoption, reflecting its deep integration within the AWS ecosystem. HashiCorp Vault (HCP Vault) follows at 32%, known for its flexibility and robust security features across multiple cloud providers. Azure Key Vault is used by 26%, aligning with Azure's broader services for enterprise clients, while Google Secrets Manager holds a 15% share, primarily favored by teams within Google Cloud environments. This distribution suggests a preference for secrets management tools that closely integrate with each organization's primary cloud platform, as the usage share of these tools closely matches that of the wider industry market share.

Security suites

SECURITY PLANE

Security suites



The survey indicates a range of security tools used within platform engineering, reflecting diverse security needs. Snyk leads at 38%, highlighting its role in embedding security checks directly into development pipelines. Aqua Security and Sysdig follow, at 16% and 15% respectively.. Smaller shares for Gremlin (6%) and Orca Security (5%) point to more specialized use cases, such as resilience testing and agentless cloud security. These numbers suggest a varied approach, where teams choose tools based on their specific requirements for cloud-native environments and integrated security.

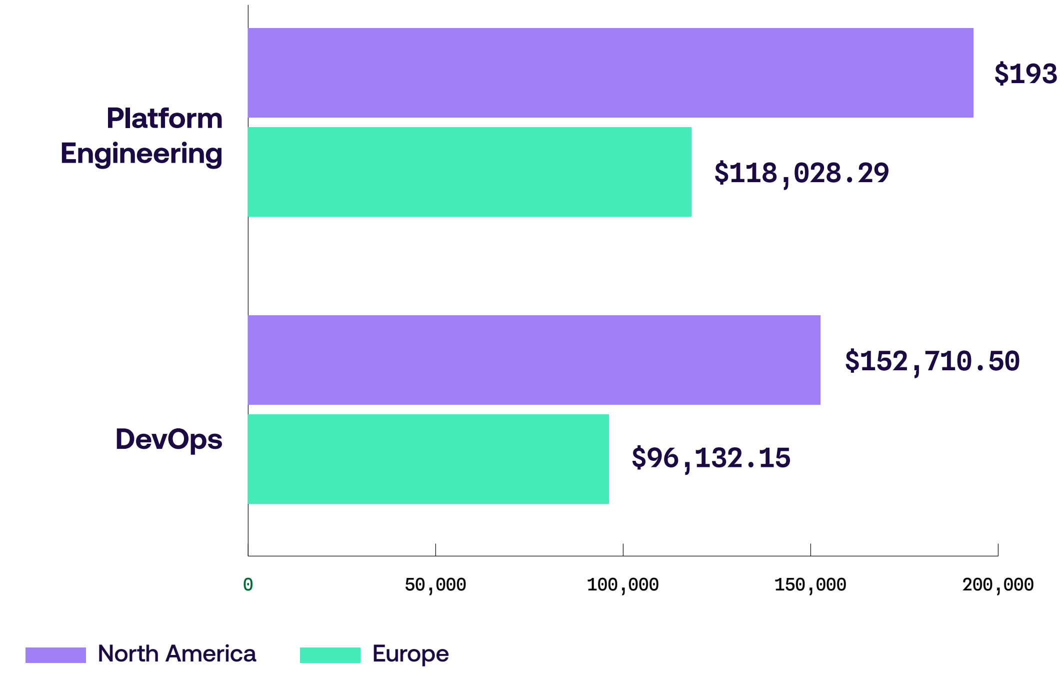
Platform engineering in real life

In addition to our organizational platform engineering survey, we conducted a parallel survey with 492 DevOps and Platform Engineering respondents. This survey aimed to gain insights into their daily responsibilities, job titles, and salary ranges. It also explored what areas these professionals primarily focus on, whether it be platform engineering, DevOps setups, or infrastructure and operations. This dual approach allowed us to better understand both the structural aspects of platform engineering at the organizational level and the individual experiences and focus areas of those actively working within the field.

Salaries

The salary data from our survey shows a clear difference between Platform Engineers and DevOps professionals, especially across regions.

AVERAGE SALARY



In North America, those working mainly in platform engineering are pulling in an average of $193,412, while DevOps earn around $152,710 - about 26.6% more for platform engineers. It reflects how highly specialized and crucial these roles have become.

In North America, platform engineers earn about

26.6%

more than DevOps roles

Over in Europe, the trend is similar but with lower overall numbers. Platform engineers earn $118,028 on average, compared to $96,132 for DevOps roles, which is roughly a 22.78% difference.

These figures highlight how companies are willing to pay a premium for platform engineering skills, recognizing the value of expertise that helps build scalable, efficient platforms.

In Europe, platform engineers earn roughly

22.78%

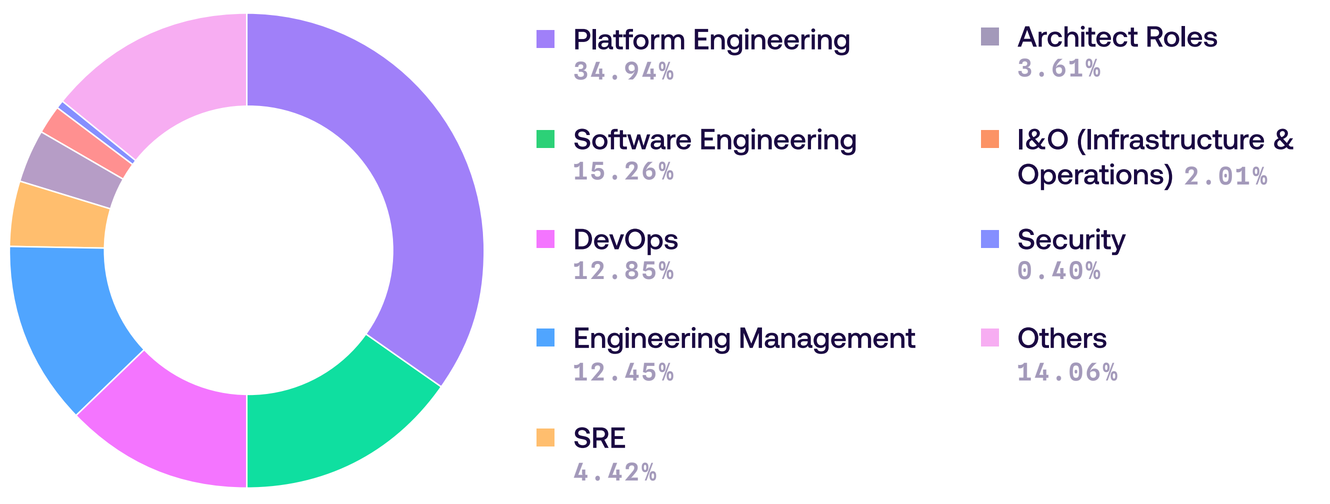
more than DevOps roles

It also reflects the growing gap between US and European tech salaries. And simultaneously a closing gap year over year between platform engineering and DevOps, as last year the survey showed how platform engineers in the US made on average 42.5% ($65,439) more money than DevOps engineers.

Job titles

Among those primarily working in platform engineering, only 34.94% hold job titles explicitly related to platform engineering, indicating that many still operate under broader roles. For instance, 12.85% have DevOps titles, and 15.26% are identified as Software Engineers. Other roles include Engineering Management (12.45%), SRE (4.42%), and Architect positions (3.61%). Smaller shares are seen in Infrastructure & Operations (2.01%) and Security (0.40%). This distribution suggests that platform engineering responsibilities are often spread across various roles and functions.

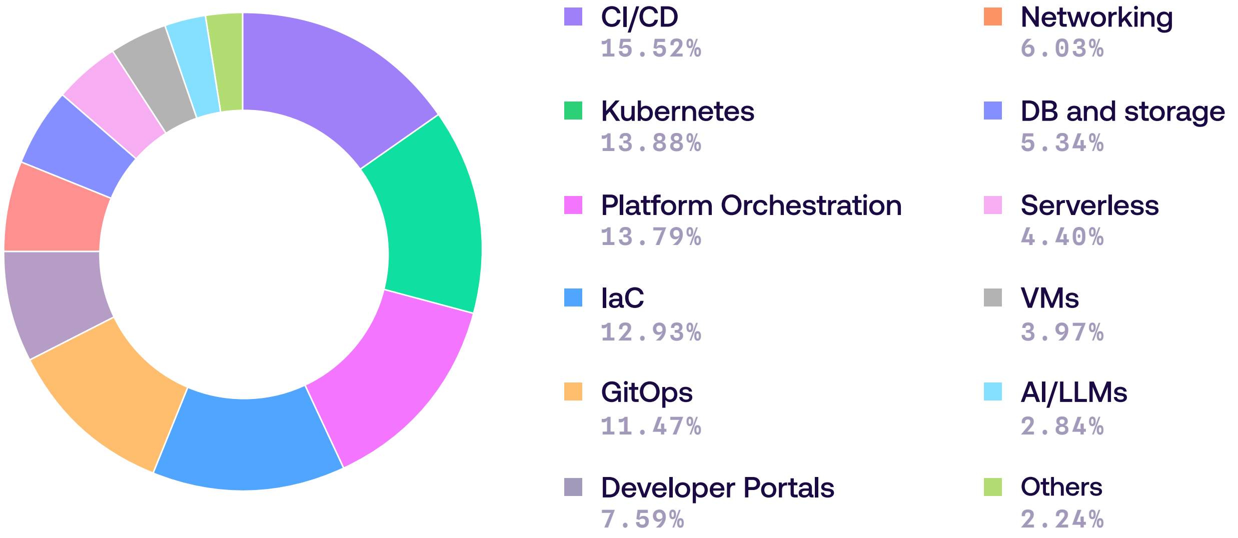
JOB TITLES



Main focus of platform engineers' work

For those primarily working in platform engineering, the main areas of focus are diverse, reflecting the range of responsibilities in the field. CI/CD tops the list at 15.52%, followed closely by Kubernetes (13.88%) and Platform Orchestration (13.79%). Infrastructure as Code (IaC) is also significant at 12.93%, alongside GitOps (11.47%). Other areas like Developer Portals, Networking, and Serverless see lower engagement, but each represents crucial elements of platform engineering. Emerging fields like AI/LLMs also make a small yet growing appearance at 2.84%.

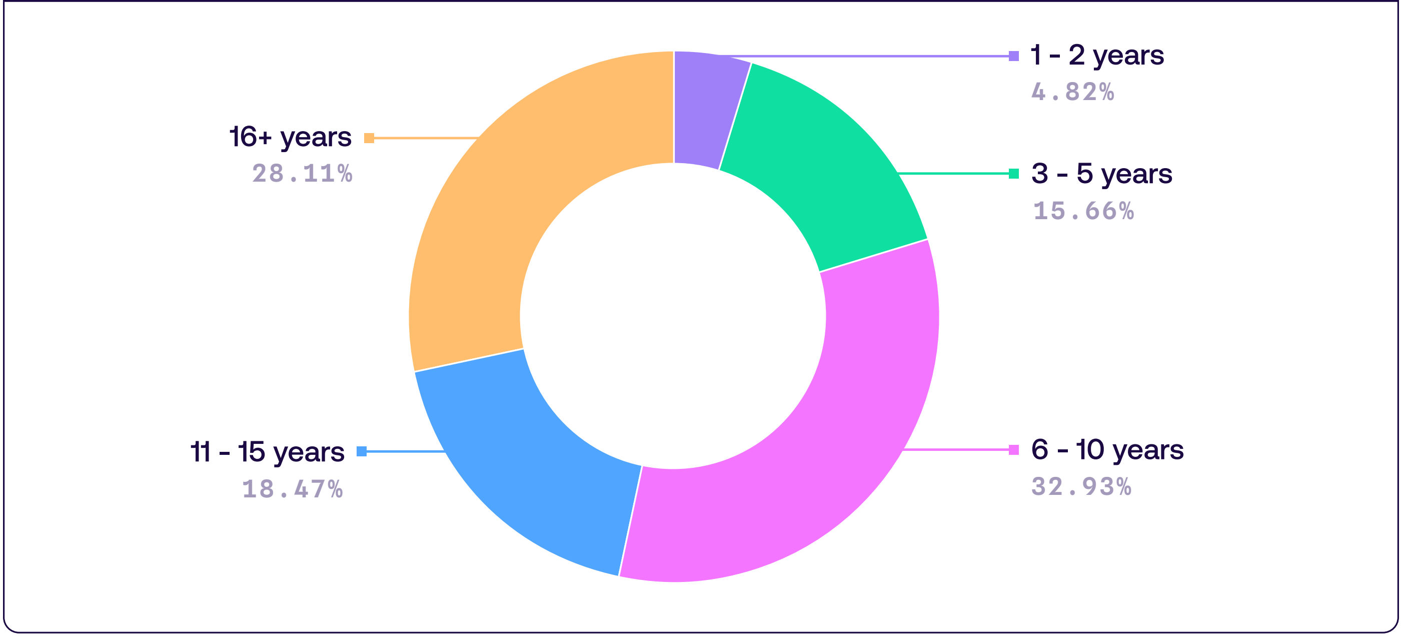
WHAT ARE YOUR MAIN AREAS OF FOCUS?



Experience levels among platform engineers

For those primarily working in platform engineering, the main areas of focus are diverse, reflecting the range of responsibilities in the field. CI/CD tops the list at 15.52%, followed closely by Kubernetes (13.88%) and Platform Orchestration (13.79%). Infrastructure as Code (IaC) is also significant at 12.93%, alongside GitOps (11.47%). Other areas like Developer Portals, Networking, and Serverless see lower engagement, but each represents crucial elements of platform engineering. Emerging fields like AI/LLMs also make a small yet growing appearance at 2.84%.

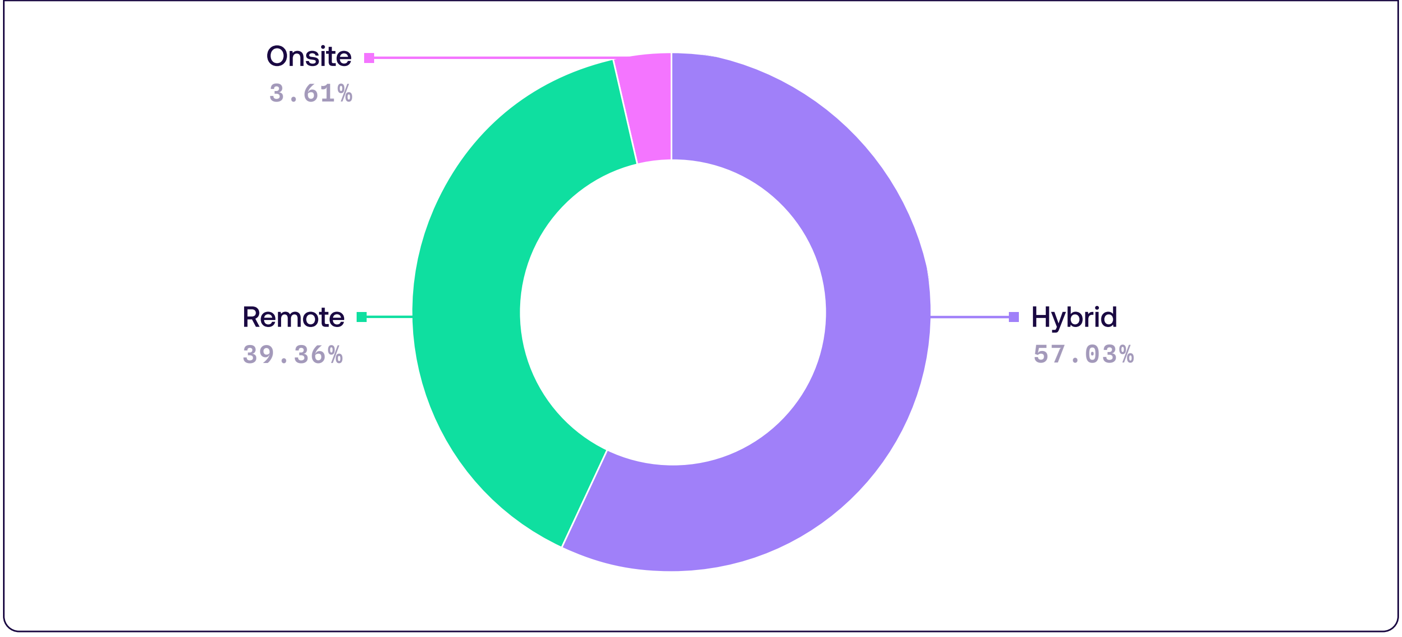
HOW MANY YEARS OF EXPERIENCE DO YOU HAVE IN YOUR FIELD?



Work setup

The survey shows that 57.03% of platform engineers work in a hybrid setup, reflecting a preference for flexibility. 39.36% operate fully remote, allowing them to work from anywhere, while only 3.61% are on site, suggesting that despite media attention around back to the office mandates, in-office roles are less common in this field.

WORK SETUP

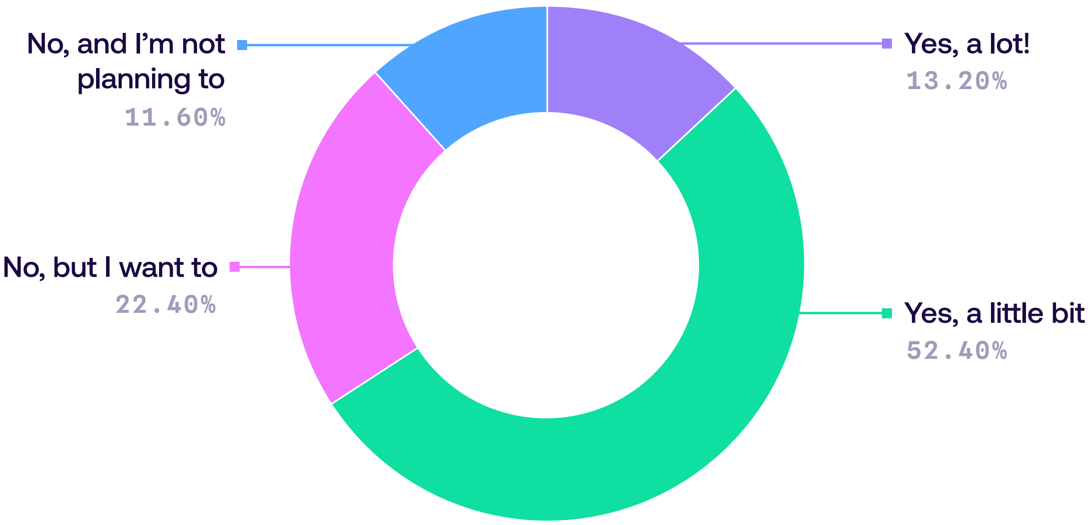


AI usage among platform engineers

The survey reveals that AI is gradually being integrated into platform engineering. 52.40% of respondents use AI a little, applying it in specific tasks or automations. 13.20% report using AI extensively, indicating deeper integration across their workflows. Meanwhile, 22.40% do not currently use AI but express a strong interest in adopting it. This shows a growing trend toward leveraging AI to enhance platform engineering processes, with potential for wider adoption in the future.

This reflects the slow but steady growth in the usage of AI amongst platform engineers, as last year's results highlighted that only 4.81% of respondents answered that they used AI “a lot”. And only 48.08% answered that they used AI at all.

ARE YOU USING AI IN YOUR WORK?



Outlook: What will the future bring?

Platform engineering is set to keep growing, but we're clearly moving beyond the hype phase. In uncertain times, teams must demonstrate the value of their platform engineering initiatives more quickly to maintain sponsorship. This means advancing platform engineering maturity and being able to measure and show success in concrete, data-driven ways.

Too often, organizations dive into platform building without a solid framework or sufficient product management expertise, making it difficult to demonstrate value beyond anecdotal evidence.

That's why we've invested heavily in our community offerings (from the community, for the community), launching new courses and trainings to help platform teams tackle these challenges in a structured, scalable way. Topics include starting with a Minimum Viable Platform (MVP), building a business case and calculating platform ROI, generating stakeholder buy-in, preparing for production readiness, rolling out effectively, ensuring adoption, and scaling with confidence.

We're also expanding our support network by launching an Ambassador Program to enable practitioners to share their insights and learnings with others in the field.

If you haven't joined yet, be part of our community on Slack, and we'll see you at our events and next PlatformCon!

Resources

Manjunath Bhat, Thomas Murphy (Gartner): Market Guide for Cloud Development Environments (behind paywall)

Manjunath Bhat (Gartner): A Software Engineering Leader's Guide to Improving Developer Experience (behind paywall)

Daniel Bryant: Platform Engineering: Orchestrating Applications, Platforms, and Infrastructure

Ajay Chankramath and Sridhar Kotagiri: Measuring the Value of Your Internal Developer Platform Investments. Enterprise Technology Leadership Journal, Spring 2024

CNCF TAG App Delivery: Platform Engineering Maturity Model

Aaron Erickson: What to build first: the house or the front door?

Luca Galante: What is a Minimum Viable Platform (MVP)?

Luca Galante: What is platform engineering?

Luca Galante: Why your Internal Developer Platform needs a backend

Kaspar von Grünberg: The role of infrastructure teams in the platform engineering era

Matthew Skelton and Manuel Pais: Team Topologies: Organizing Business and Technology Teams for Fast Flow. IT Revolution, 2019

Selected talks from PlatformCon 2024

André Alfter: A guide for building secure Internal Developer Platforms in regulated environments

Daniel Bryant: Platform Orchestrators: The missing middle of Internal Developer Platforms?

Stéphane Di Cesare: Helping a platform team focused on engineering to switch to a product-based approach

Ajay Chankramath: Harmony in engineering platforms: A decision tree symphony for Orchestrator selection

Derik Evangelista: Your Backstage needs a platform

Luca Galante: Platform Orchestrator - the platform engineering game changer

Tim Hansen: Everything is code: embracing GitOps at Spotify

Kelsey Hightower and Kaspar von Grünberg: Real talk: platform engineering with Kelsey Hightower and Kaspar von Grünberg

Kaspar von Grünberg: You’re forgetting Infrastructure Platform Engineering – you shouldn’t

Gregor Hohpe: Platforms = (Architecture + DDD + Automation) x Wholesale

Manuel Pais: From fragile to antifragile internal platforms

Cansu Kavili Örnek and Anna Ciula: Platform as a product: stories from the field

Paul Revellio: Platform engineering in action with Google Cloud and Humanitec

Nicki Watt: Maturing your platform engineering initiative