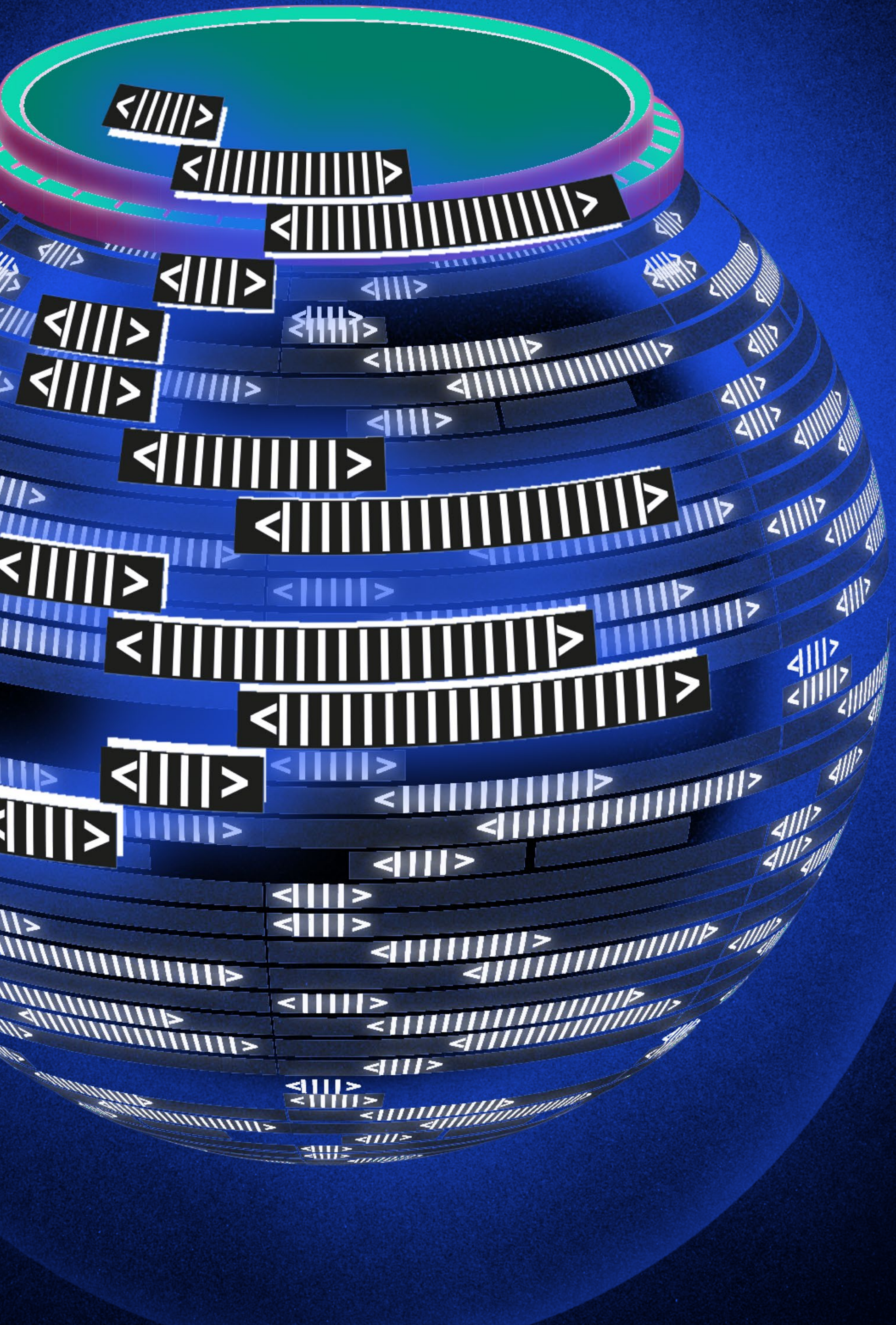
State of Platform Engineering Report

VOLUME 2



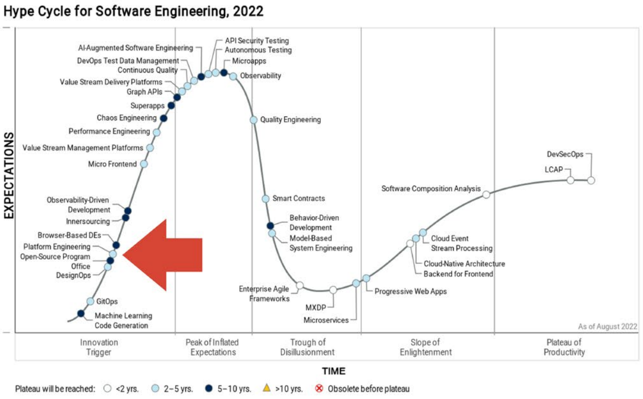
humanitec

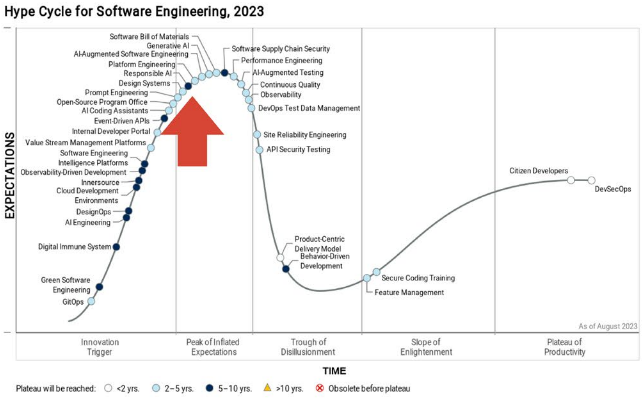
Table of contents

What's happened since vol 1?

Since the State of Platform Engineering Report volume 1, platform engineering has seen phenomenal growth as both a discipline and a community. Just one year after the first PlatformCon 2022 which hosted 6k+ attendees, PlatformCon 2023 attracted a staggering 22k+ attendees. Backed by 27 sponsors, the event received 514 talk submissions and hosted 169 talks across five new talk tracks. This year's virtual event was bigger and better in every way. But that's not all. The Platform Engineering YouTube channel now boasts nearly 16k subscribers and the Slack channel is a constant buzz of conversation fuellled by 16.3k members.

Growth isn't the only major change that's happening. In August, platform engineering entered the "Peak of Inflated Expectations" on the new Gartner Software Engineering Hype Cycle. This is the second of five phases that make up the hype cycle, and is used to describe innovations that show an increase in product usage — but where there's still more hype than proof that the innovation can deliver what organizations need to deliver value.

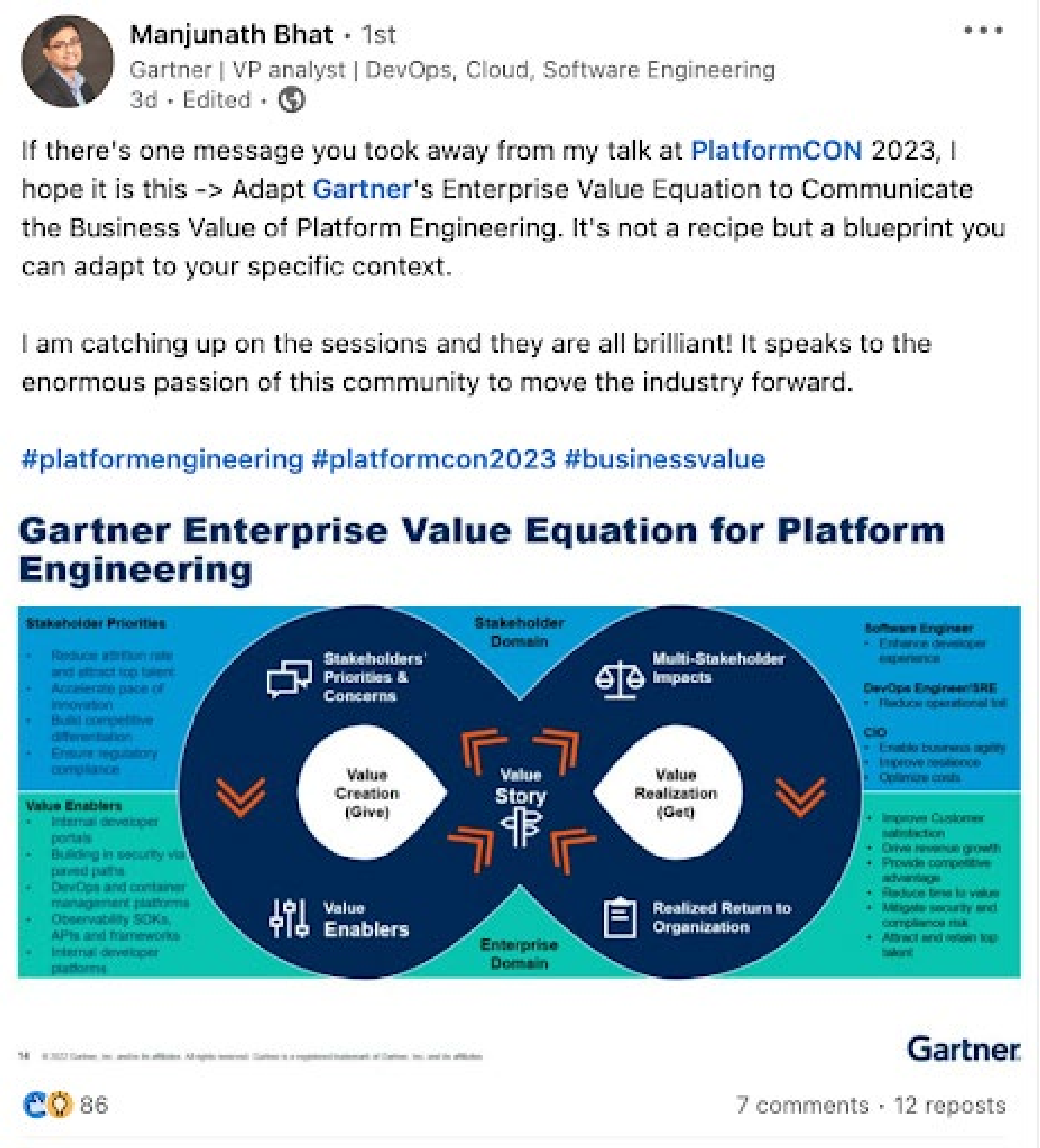

Hype Cycle for Software Engineering, 2022


Hype Cycle for Software Engineering, 2023

Recently Thoughtworks Technology Radar Volume 29 recognized platform orchestration as a technique and a new generation of tools that surpass the traditional platform-as-a-service (PaaS) model. According to Thoughtworks, tools like Humanitec Platform Orchestrator are capable of enforcing organizational standards while granting developers self-service access to variations through configuration. Thoughtworks recommends "platform teams assess these tools as an alternative to pulling together your own unique collection of scripts, native tools and infrastructure as code.

We're also seeing more organizations recognizing the importance of treating their platform as a product, to meet the unique needs of their developers. There's been an increase in platform engineering community-driven webinars, in-person workshops, and events to facilitate and expand conversations. And we're thrilled to see more businesses speed up innovation and impact key business metrics such as time to market, by successfully rolling out an enterprise-grade Internal Developer Platform (IDP).

When it comes to advancing their platform engineering journey, many organizations gained a wealth of knowledge from industry-leading platform practitioners at PlatformCon 2023. For example, Gartner's Manjunath Bhat noted that many organizations find it hard to explain the value of platform engineering in his talk "How to communicate the business value of platform engineering." According to Bhat, the key to communicating the business value of platform engineering lies in knowing how it extends beyond DevOps. Also, priorities vary from stakeholder to stakeholder, so platform engineers need to know how to speak their language.



Manjunath Bhat • 1st
Gartner | VP analyst | DevOps, Cloud, Software Engineering
3d • Edited •

If there's one message you took away from my talk at PlatformCON 2023, I hope it is this -> Adapt Gartner's Enterprise Value Equation to Communicate the Business Value of Platform Engineering. It's not a recipe but a blueprint you can adapt to your specific context.

I am catching up on the sessions and they are all brilliant! It speaks to the enormous passion of this community to move the industry forward.

#platformengineering #platformcon2023 #businessvalue

Gartner Enterprise Value Equation for Platform Engineering

Stakeholder Priorities
Reduce attrition rate and attract top talent
Accelerate pace of innovation
Build competitive differentiation
Ensure regulatory compliance
Value Enablers
Internal developer portals
Building in security via paved paths
DevOps and container management platforms
Observability SDKs, APIs and frameworks
Internal developer platforms
Stakeholders' Priorities & Concerns
Value Creation (Give)
Value Enablers
Stakeholder Domain
Value Story
Enterprise Domain
Multi-Stakeholder Impacts
Value Realization (Get)
Realized Return to Organization
Software Engineer
Enhance developer experience
DevOps Engineer/SRE
Reduce operational toil
CIO
Enable business agility
Improve resilience
Optimize costs
Improve Customer satisfaction
Drive revenue growth
Provide competitive advantage
Reduce time to value
Mitigate security and compliance risk
Attract and retain top talent

14 © 2022 Gartner, Inc. and/or its affiliates. All rights reserved. Gartner is a registered trademark of Gartner, Inc. and its affiliates.

Gartner

86

7 comments • 12 reposts

During PlatformCon 2023 we were also introduced to proven platform blueprints, reference architectures, and key design considerations when building an IDP. In “Platform as Code: Simplifying developer platform design with reference architectures” McKinsey’s Stephan Schneider and Mike Gatto introduced their new reference architecture for IDPs on Amazon Web Services (AWS) based on learnings from hundreds of real-life platform setups (more on this later). This inspired Humanitec to create their own IDP reference architectures for AWS, Azure, and GCP-based setups, for which the implementation codes have now been open-sourced (AWS and GCP setups only).

More platform engineering benefits were revealed by Electrolux SRE Product Owner Kristina Kondrashevich and SRE Manager Gang Luo in their talk “Why we skipped SRE and switched to platform engineering”. The pair shared a multitude of developer pain points their team faced such as needing new clusters, applying Terraform changes, and provisioning service infrastructure. They discussed how they built their IDP with integration to their Cloud and various toolchains, and how embedding SRE principles helped Electrolux reduce complexity, cut delivery time, and strengthen security.

So there you have it. The ongoing community growth speaks for itself. But despite its increasing popularity, there are many questions still looming, like what exactly is platform engineering? Where’s the best place to start on your platform journey? How do you avoid costly mistakes, and what’s the big deal with the platform as a product approach?

This paper offers guidance on these topics and key resources from across the community. By the last page, you’ll be able to explain what modern platform engineering looks like, why you need a reference architecture in your life, and why you need to be thinking more about platform engineering and AI. Let’s dig in.

What is platform engineering?

Platform engineering is the discipline of building Internal Developer Platforms (IDPs) and is one of the most important software engineering trends right now. Actually, the truth is it's much more than just a trend. It's the future of software delivery. By designing and building effective enterprise-grade IDPs, platform engineering promises to solve some of the biggest problems facing software engineering organizations in the cloud-native era. Ie, the high complexity of tech and tools which impacts developer productivity and feeds ticket ops.

> **"Platform engineering is the discipline of designing and building toolchains and workflows that enable self-service capabilities for software engineering organizations in the cloud-native era. Platform engineers provide an integrated product most often referred to as an "Internal Developer Platform" covering the operational necessities of the entire lifecycle of an application."**

Luca Galante Product at Humanitec

When done well, IDPs radically simplify software delivery. They drive standardization by design and enable developer self-service throughout the entire life cycle of an application. This helps boost developer productivity and removes Ops bottlenecks, by enabling developers to easily access the resources they need to work more efficiently without waiting for Ops support. And when developers can focus on what they do best (writing code), organizations can slash time to market (TTM) by 30%, achieve 4x higher deployment frequency, and accelerate lead time by 30%. This in turn helps drive revenue growth and empowers businesses to respond faster to a competitive landscape.

> **"An Internal Developer Platform (IDP) is built by a platform team to build golden paths and enable developer self-service. An IDP consists of many different techs and tools, glued together in a way that lowers cognitive load on developers without abstracting away context and underlying technologies."**

internaldeveloperplatform.org

So now we know what platform engineering is, let's take a look at where it all started — and what the future holds.

A few years ago some organizations had started patching complex tech and tools together into something that developers could use independently of Ops. But as a discipline, platform engineering did not yet exist.

In the 2010s tech giants like Google, Facebook, Airbnb, and Netflix began building platforms to drive developer and Ops efficiency. Then in 2018 Evan Bottcher defined a digital platform (what we now call an Internal Developer Platform) as "a foundation of self-service APIs, tools, services, knowledge, and support which are arranged as a compelling internal product." This is also when the concept that platforms are and should be treated as a product started out, as a foundational pillar of platform engineering.

Team Topologies was then released by Matthew Skelton and Manuel Pais in 2019. It was the first book to talk about the platform team and its relationship with the rest of the engineering organization. The book identified DevOps antipatterns, where certain team structures made ownership between developers and Ops unclear — and provided a clearer separation of concerns.

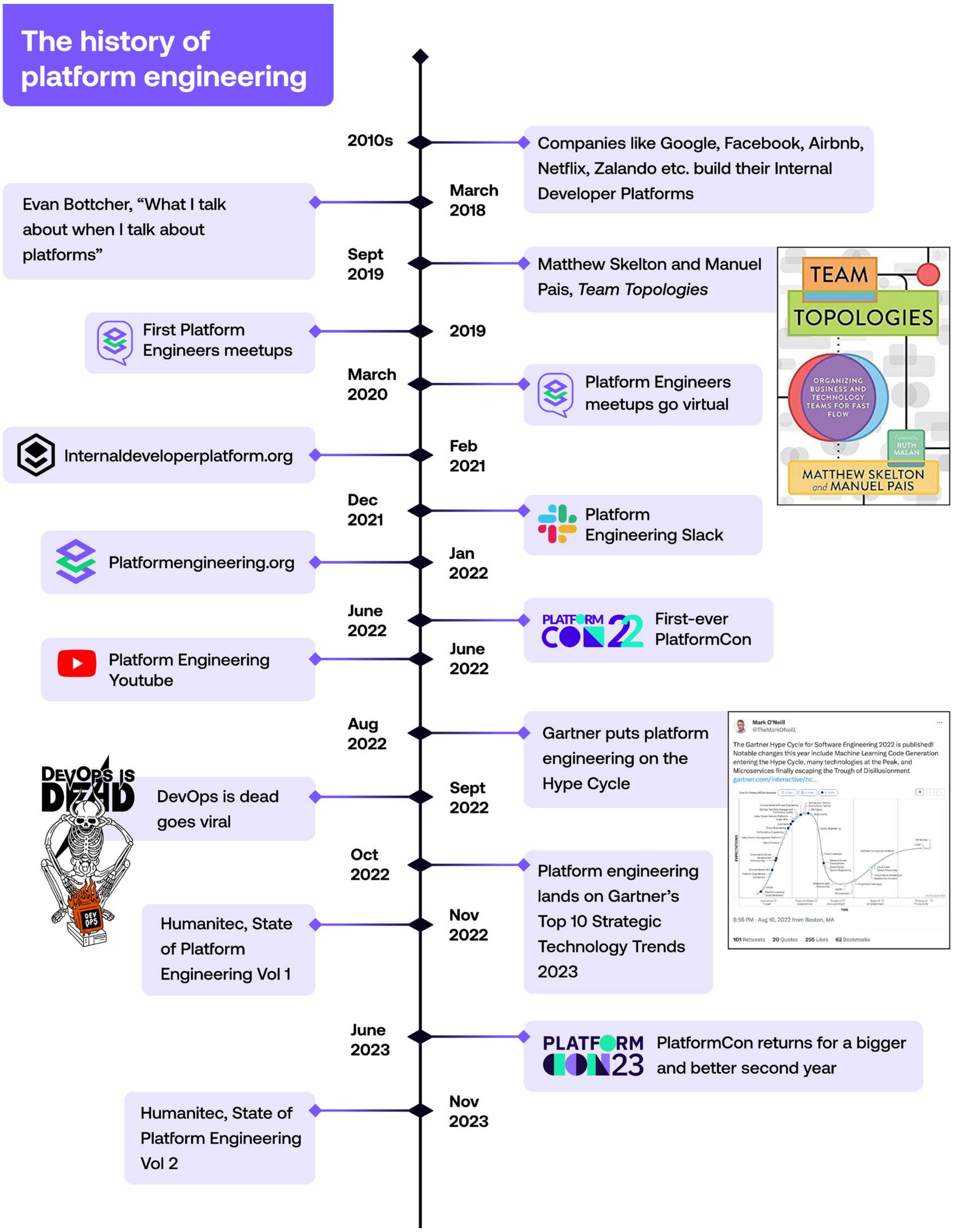
Things changed in 2019 when the first platform engineering meetup happened in Berlin, formed of a small bunch of engineers fed up with their organization's DevOps situation. Think spiraling complexity of cloud-native technologies combined with the expectation that developers should be the ones taking it all on. The result was burnt-out developers trying to ship code at light speed with ridiculously high cognitive load. Meanwhile, Ops were being flooded with tickets from overwhelmed developers. The thinking was to create a space to share learnings and get advice from seasoned practitioners. The small community rapidly grew in size and the term platform engineering was defined, a discipline that exists to build Internal Developer Platforms.

In 2021 the Platform Engineering Slack channel was launched, followed by platformengineering.org a year later — born as a hub for conversations, insights, and information shared by community members. Soon after came the idea of PlatformCon, the first virtual conference of its kind created by and for platform engineers. amongst the 78 speakers were then-Puppet field CTO Nigel Kersten, OpenCredo's Nicki Watt, Team Topologies co-author Manuel Pais, and Cloud Strategy author Gregor Hohpe.

The momentum continued when in August 2022, platform engineering made Gartner's Hype Cycle. A month later, DevOps is dead, embrace platform engineering went viral for its controversial headline. Author Aeris Stewart remarked when developers don't agree on the extent to which they should do Ops tasks, it forces "everyone to do DevOps in a one-size-fits-all way has disastrous consequences." The News Stack also recognized that "Platform engineering uses a product approach to enable the right amount of developer self-service and level of abstraction"

Following this, platform engineering was again named one of Gartner's Top Strategic Technology Trends for 2024. They shared that platforms used by organizations like Nike provided the ability to respond faster to change, slash time to market, and innovate rapidly. And in November 2022 Humanitec published the first-ever State of Platform Engineering Report which covered the evolution of platform engineering and gave a consolidated view of the platform tooling landscape. It also compared platform engineering and DevOps salaries, and detailed community growth opportunities.

The history of platform engineering



2010s — Companies like Google, Facebook, Airbnb, Netflix, Zalando etc. build their Internal Developer Platforms

March 2018 — Evan Bottcher, “What I talk about when I talk about platforms”

Sept 2019 — Matthew Skelton and Manuel Pais, *Team Topologies*

TEAM
TOPOLOGIES
ORGANIZING BUSINESS AND TECHNOLOGY TEAMS FOR FAST FLOW
Foreword by RUTH MALAN
MATTHEW SKELTON and MANUEL PAIS

2019 — First Platform Engineers meetups

March 2020 — Platform Engineers meetups go virtual

Feb 2021 — Internaldeveloperplatform.org

Dec 2021 — Platform Engineering Slack

Jan 2022 — Platformengineering.org

June 2022 — First-ever PlatformCon

June 2022 — Platform Engineering Youtube

Aug 2022 — Gartner puts platform engineering on the Hype Cycle

Mark O'Neill
@TheMarkONeill

The Gartner Hype Cycle for Software Engineering 2022 is published! Notable changes this year include Machine Learning Code Generation entering the Hype Cycle, many technologies at the Peak, and Microservices finally escaping the Trough of Disillusionment gartner.com/interactive/hc...

8:56 PM · Aug 10, 2022 from Boston, MA

101 Retweets 20 Quotes 255 Likes 62 Bookmarks

Sept 2022 — DevOps is dead goes viral

Oct 2022 — Platform engineering lands on Gartner’s Top 10 Strategic Technology Trends 2023

Nov 2022 — Humanitec, State of Platform Engineering Vol 1

June 2023 — PlatformCon returns for a bigger and better second year

Nov 2023 — Humanitec, State of Platform Engineering Vol 2

Looking ahead, we expect to see platform engineering’s popularity continue to skyrocket. The impact on developer productivity and employee satisfaction is huge, because of it’s potential to reduce developer cognitive load and reduce the overall burden on Ops. According to Puppet research, 94% of organizations agree the adoption of platform engineering is helping them realize the benefits of DevOps. 68% say they are experiencing increased development velocity, and 53% of firms that have been practicing platform engineering for more than three years report improved development speed.

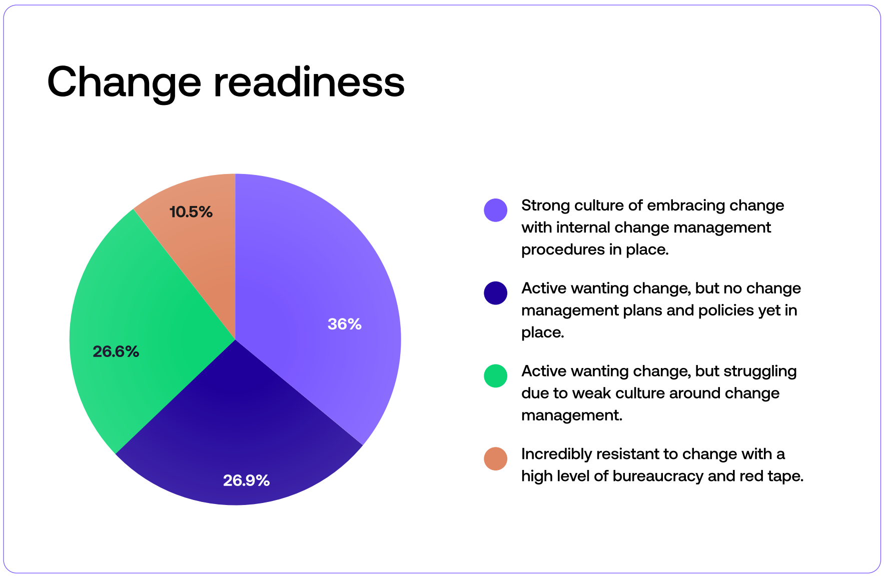
Platform engineering maturity survey reveals lack of best practice

In April 2023 Syntasso drafted the first version of the Platform Maturity Model, which they donated to the Cloud Computing Foundation (CNCF) for further discussion. The model was designed to help organizations assess their platform engineering continuous improvement capability. One key point was that engineering organizations are able to mature faster when they align with established best practices — such as embracing a platform as a product mindset — and when they foster the right organizational culture.

Inspired by the Platform Maturity Model, the platform engineering community wanted to understand how much organizations follow platform engineering best practices. A short survey was created by and for the community, and received responses from 296 individuals revealing that most organizations still struggle to use best platform engineering practices.

Change readiness

One such best practice is change readiness. When it comes to fostering a culture that embraces change, the survey found the majority (64%) of respondents have no change management processes in place. Over a third (36%) report having strong internal change management processes in place, and 26.9% say they actively want change — however are lacking in any policies or plans. 26.6% also actively want change but are running into difficulties because of a weak change management mindset. Only 10.5% struggle due to a high level of bureaucracy and red tape and their organization being extremely resistant to change.

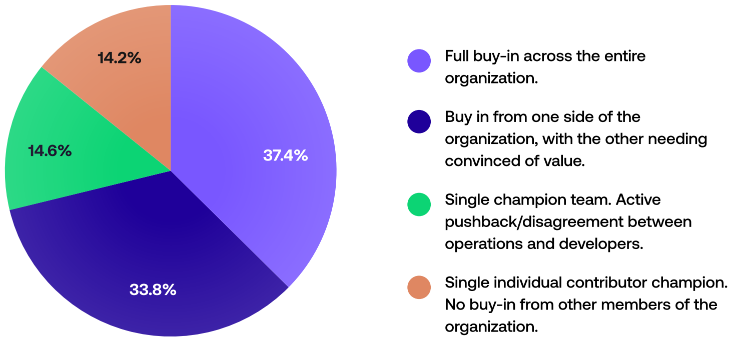

Change readiness

Organizational buy-in

When trying to create a culture that embraces change, developers are not the only key stakeholders. To deliver a successful platform you also need sustained buy-in from other areas of the organization, such as senior management and C-suite.

According to the survey, 37.4% of respondents are able to secure full buy-in across the organization. 33.8% agree one side of the organization remains a work in progress. 14.6% rely on a single champion team, while 14.2% say they rely on a single individual contributor champion. And with two-thirds (62.6%) of respondents lacking full buy-in for their platform, it's clear that platform advocacy remains a challenge for the majority of organizations.

How much support does your platform have from the oganization?



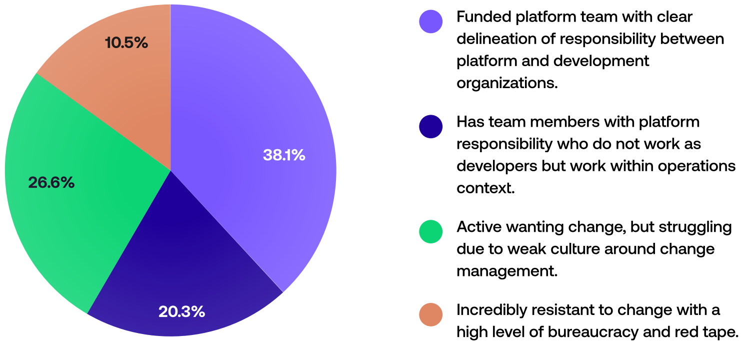
Platform team structure

The next important takeaway is that few organizations have a platform team aligned with best practices. When thinking about the platform team structure, organizations should avoid anti-patterns — such as having no clear separation of responsibilities between teams.

"That's an anti-pattern I've seen very often, where the platform becomes a bucket for everything, and just becomes a huge mess with lack of ownership, lack of focus and a lot of waste, where teams are overloaded and working on a lot of stuff that's not really a priority," explained Team Topologies co-author Manuel Pais (on The New Stack).

According to the survey, only 38.1% of respondents have a funded platform team with a clear delineation of responsibility between themselves, and development organizations. 26.3% have team members with platform responsibility working in an Ops context. 26.6% have some platform knowledge and responsibility divided up between several team members in full-time development roles. And 15% report little to no platform knowledge with people in full-time development positions also doing Ops tasks.

What best describes the structure of your platform team?

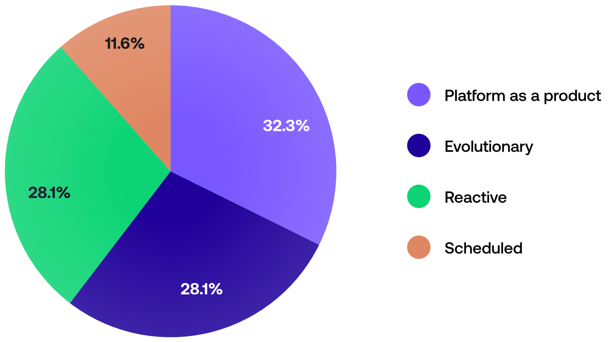


Platform product management

The survey explored three aspects of the platform as a product approach: identifying features, strategy for driving platform adoption, and organizational buy-in.

When it comes to identifying new platform features, just under a third (32.3%) of respondents follow a platform as a product approach. 28.1% take an evolutionary approach, collaborating with users to define the scope of work before optimizing for wider use. 26.1% of respondents take a reactive approach, and only 11.6% identify new features by following an infrequently updated set list of requirements.

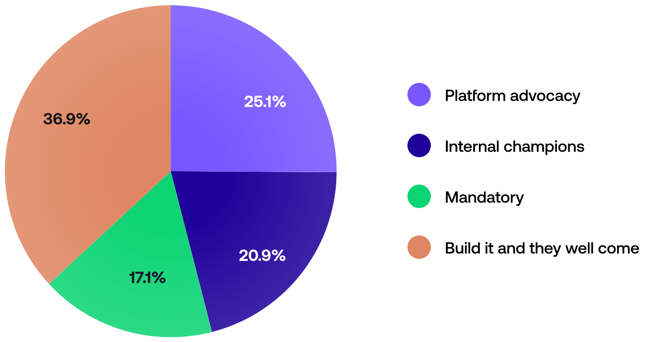
How are new features identified and priotize for your platform?



Platform adoption strategy

No matter how innovative and user-friendly an Internal Developer Platform (IDP) may be, much of its success hinges on its adoption rate. The attitude of “build it and they will come” is not a viable strategy for driving platform adoption. Just over a quarter (25.1%) of respondents recognize this, and report establishing platform advocacy to help drive adoption. 20.9% say they rely on internal champions, 17.1% mandated platform usage, while the majority (36.9%) still take a “build it and they will come” approach.

What is the organization’s strategy for driving platform adoption?



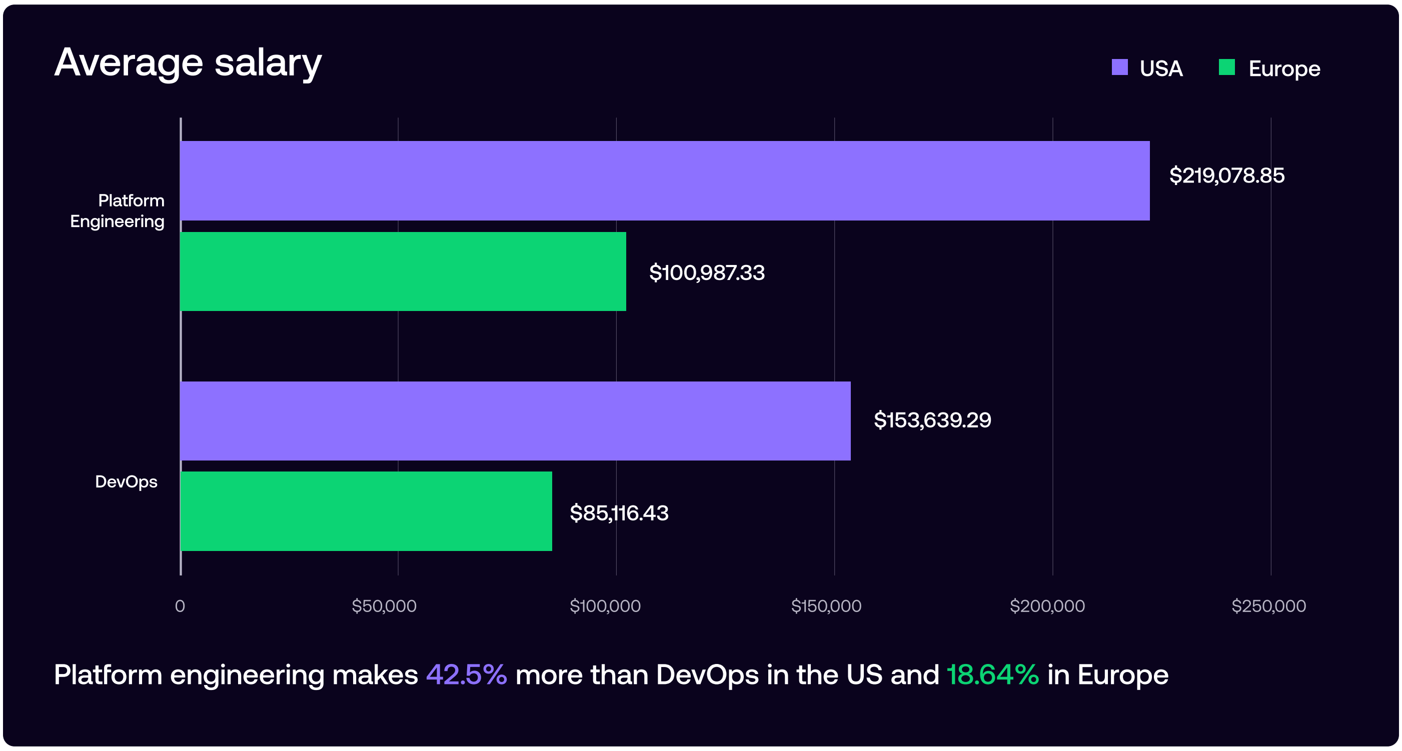
The platform engineer's world

As platform engineering grows it's important we understand what the role of a platform engineer looks like. How much do they earn? What does their working life look like? And who actually is a platform engineer?

To find out, the platform engineering community rolled out the Platform Engineering 2023 survey, which targeted primarily the United States and Europe to gather the largest amount of data possible.

Platform engineers earn more than DevOps roles

The latest Platform Engineering 2023 survey shows that platform engineers in the US make on average 42.5% ($65,439) more money than DevOps engineers. In Europe the salary gap is smaller but still obvious, with platform engineers earning on average 18.64% ($15,871) more than their DevOps counterparts. This is a huge difference and could be a reflection of how platform engineers require a wider or more specialized skillset than DevOps professionals.

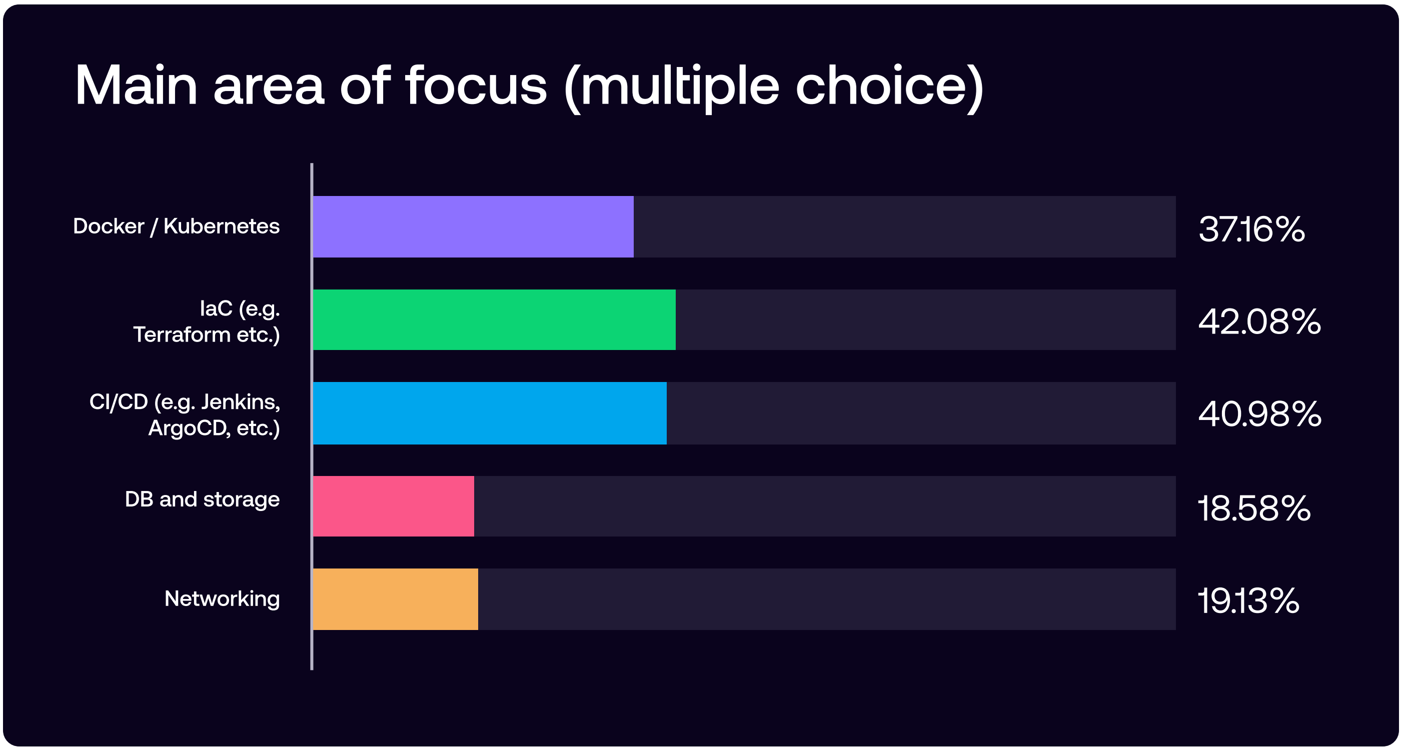


Platform engineering makes 42.5% more than DevOps in the US and 18.64% in Europe

Note: Data aggregated is based on description of what respondents "work on". Platform engineering was an aggregate of platform engineering and developer experience. DevOps was aggregated from infrastructure, DevOps setup, and Ops.

What are platform engineers' main areas of focus?

The majority of survey respondents working in platform engineering report IaC such as Terraform (42.08%) as their main area of focus. This was closely followed by respondents who focus on CI/CD such as Jenkins and ArgoCD (40.98%), and then by professionals who work mainly with Docker / Kubernetes (37.16%). In the minority were practitioners who focus mainly on networking (19.13%).

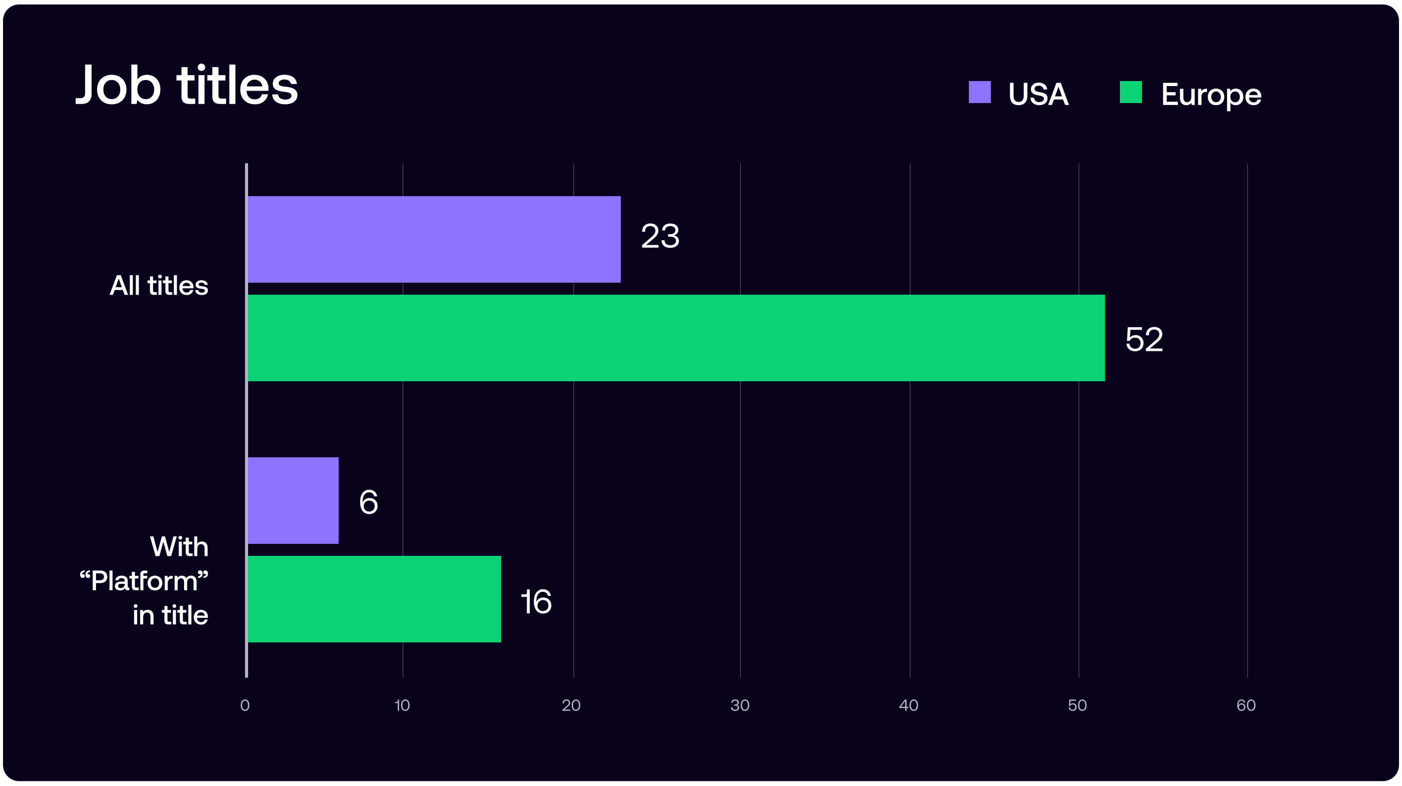


Common platform engineering job titles

Platform engineering is still a relatively new discipline, and so those working in this space do not yet have extremely defined roles. That said if we look at DevOps and SRE (both have been around for much longer) there's still much ambiguity as to what related titles actually mean. Going forward we hope to do better with platform engineering.

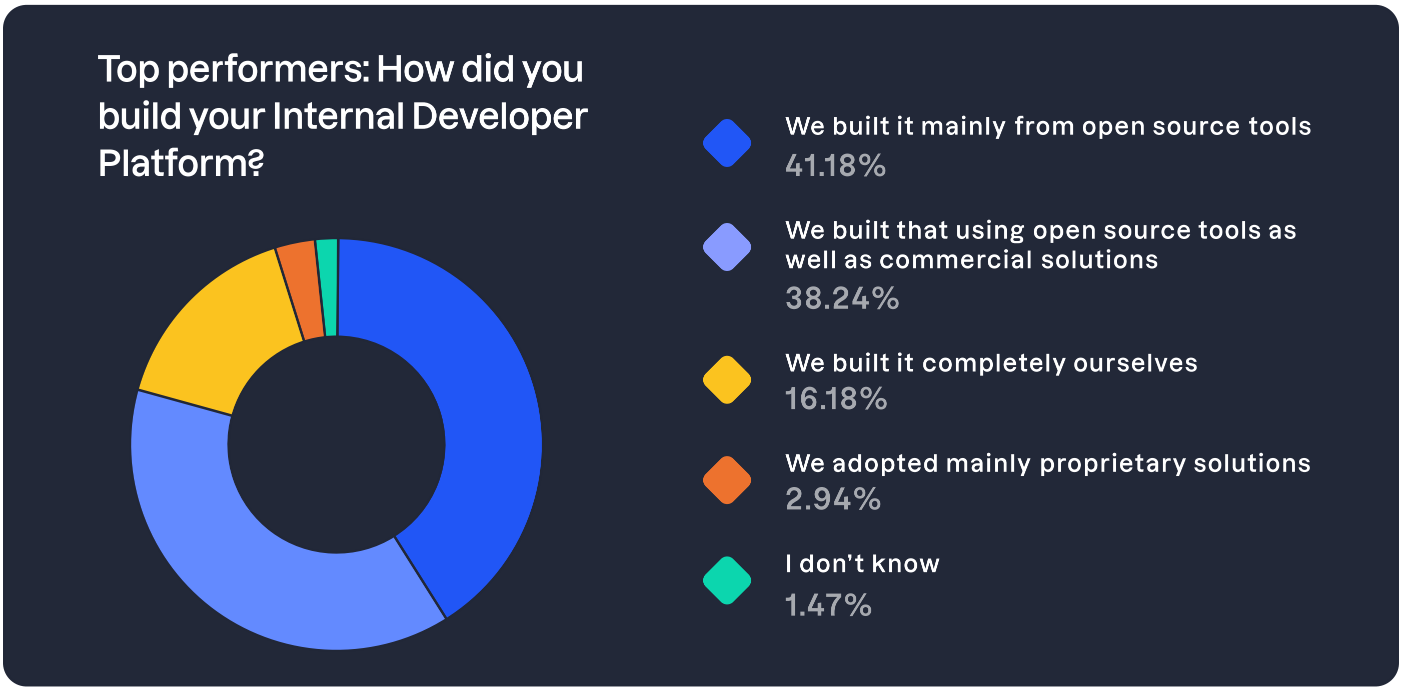
According to the survey, only 26.09% of respondents in the US and 30.77% in Europe have 'platform' in their title. Even less than that, 1% have 'product' and only 3.3% have a 'DevOps' title. The majority were a mix of general 'engineering' and 'cloud' job titles. So although only one-quarter of respondents have a 'platform' title, they were the majority compared to DevOps, SRE, cloud, and general engineering titles.

From these results, we can see that platform engineering job titles are still very inconsistent and undefined.



Reference architectures: bringing you closer to your enterprise-grade IDP

Treating your platform as a product means you build your own IDP, but it doesn't mean having to start from scratch. There are a multitude of tools available for you to use and tailor to fit your organization's needs. In fact according to [Humanitec's 2023 DevOps Benchmarking Study](), top engineering organizations use a combination of open-source and vendor tools to create their platform.


Top performers: How did you build your Internal Developer Platform?

But how exactly do you go about glueing all these available tools together in a meaningful way?

During their PlatformCon 2023 talk "[Platform as Code: Simplifying developer platform design with reference architectures](), McKinsey's Stephan Schneider and Mike Gatto unveiled their new IDP reference architecture for Amazon Web Services (AWS) platforms, using learnings from hundreds of real-life platform setups. McKinsey's Marco Marulli did the same for Google Cloud Platform (GCP)-based setups in his talk "[Streamlining developer platform design with reference architectures]()". The idea behind these reference architectures is to enable organizations to quickly stand up a minimum viable product (MVP) of the platform and in doing so, elevate developer experience (DevEx) and productivity.

From this, several reference architectures have been created for AWS, Azure, and GCP setups. They explain how IDPs consist of five architectural planes each with unique functionalities, and how tools can be clustered into each plane:

- **Developer Control Plane**
  This level is made up of the primary "interfaces" (code based with Score, UI, CLI, or API) that developers should be able to choose from when using the platform.

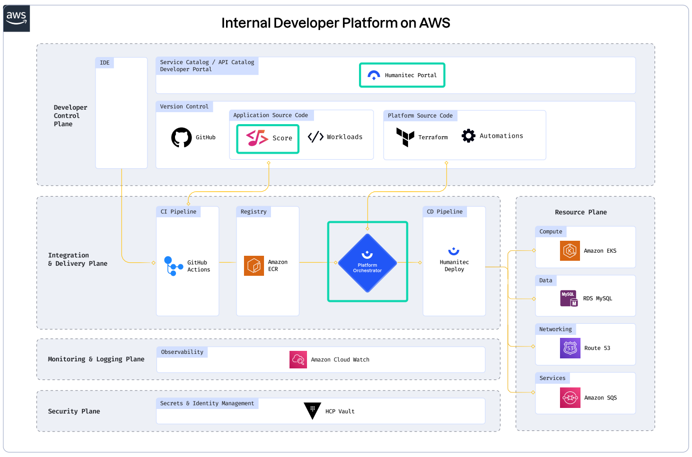
- **Integration and Delivery Plane**
  This level contains the tools that build, store, configure, and deploy requests that come from the Developer Control Plane.

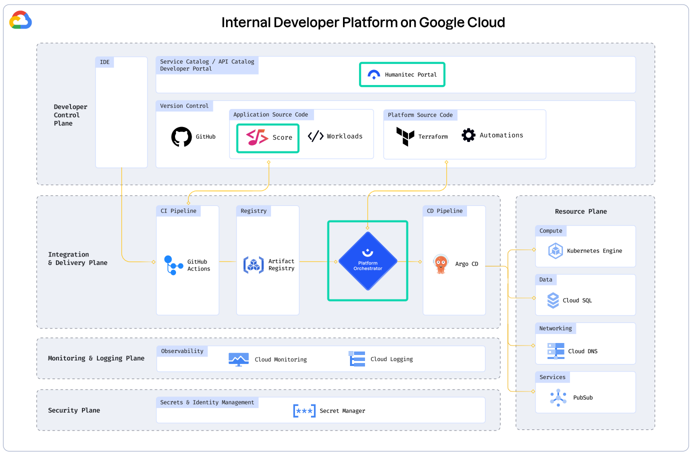
- **Resource Plane**
  Here is where all resource components necessary to run the app exist. The resources can be configured as code using tools like Terraform.

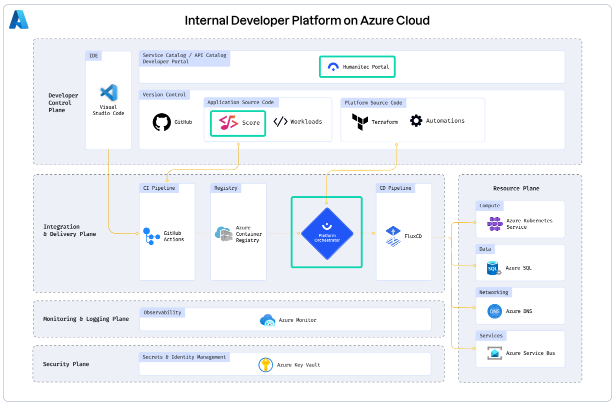
- **Monitoring and Logging Plane**
  This level provides app and infrastructure logs and real-time metrics that developers can use for observability, monitoring, and making data-driven decisions.

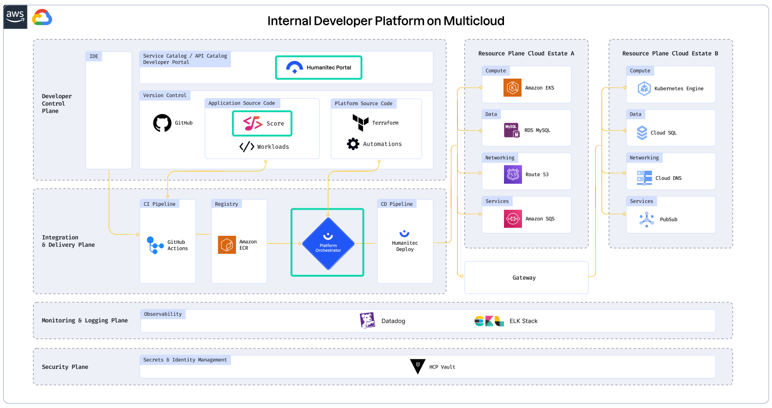
- **Security Plane**
  This plane manages secrets and identity to protect sensitive information, like storing, managing, and retrieving API keys and passwords.

Below are examples of what the different architectural planes would look like based on an AWS, Azure, and GCP platform setup:


Internal Developer Platform on AWS


Internal Developer Platform on Google Cloud


Internal Developer Platform on Azure Cloud


Internal Developer Platform on Multicloud

Most recently the implementation codes for the reference architectures based on AWS and a GCP setup have been open-sourced. In addition, new learning paths are now available to help organizations master their IDP:

- Open source reference architecture for AWS
- Open source reference architecture for GCP
- IDP learning path

Organizations who leverage this codebase will be able to accelerate and streamline the creation of their IDP, empowering teams to build an MVP with greater ease than ever before.

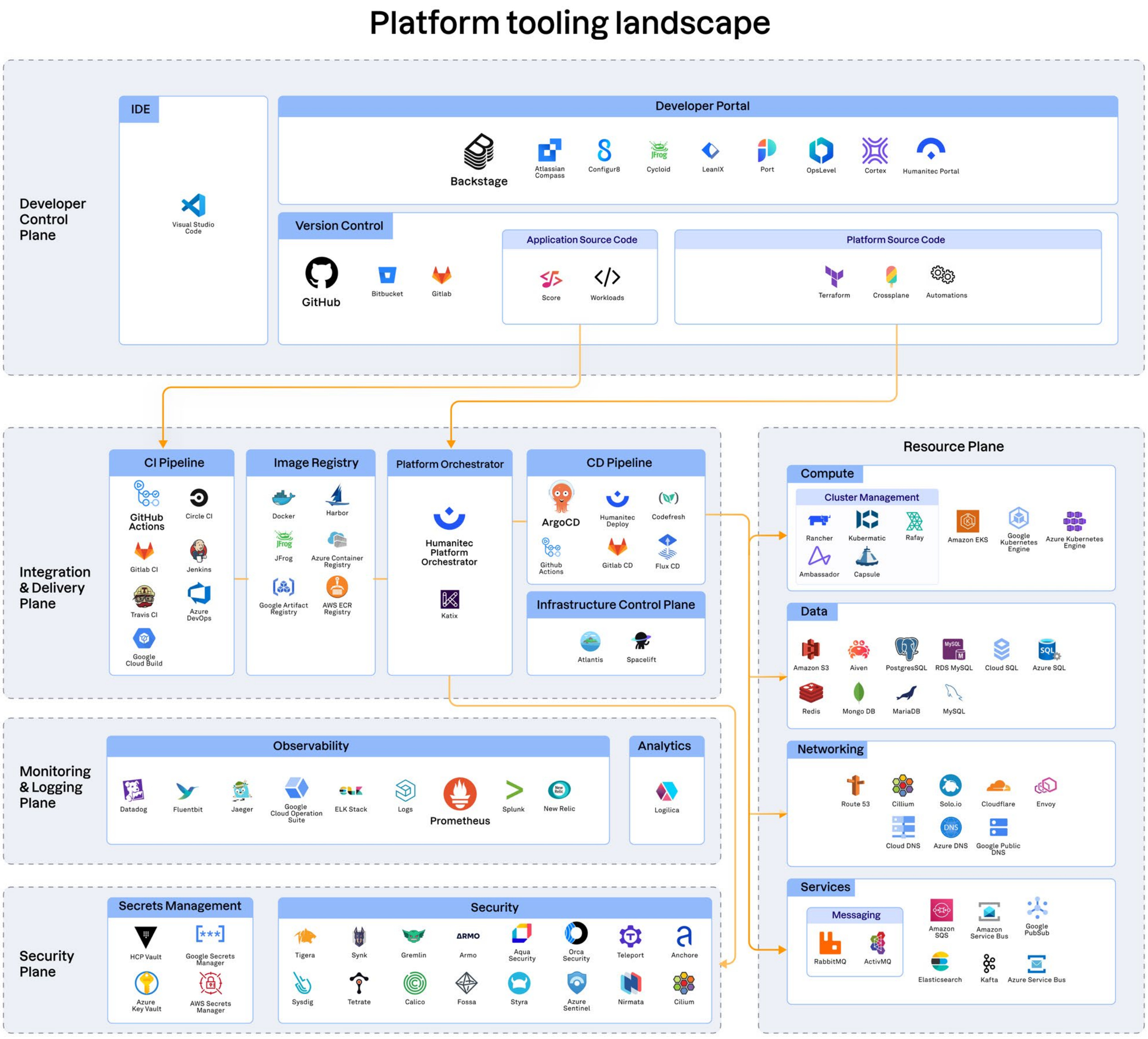
No matter your setup, using a reference architecture is essential for any organization beginning its platform engineering journey. Not only can it be used to teach platform teams proven IDP design principles; it paints a clear picture of how architectural components fit together — and how to design great interaction patterns for both engineers and developers.

The latest platform tooling landscape

Now you know what architectural planes make up a platform and how they work, where do you go next? What are the tech and tools platform teams work with, and that you should you consider for building your platform?

> “An Internal Developer Platform (IDP) is the sum of all the tech and tools that a platform engineering team binds together to pave golden paths for developers.”
>
>
>
> **Kaspar von Grünberg** CEO, Humanitec

The below overview attempts to show all the relevant tooling categories you can use to build an IDP, that also follows the Platform as a Product paradigm:



Platform tooling landscape

The impact of AI on platform engineering

So what does the future hold for platform engineering? We couldn't answer this question without considering how AI is changing the future of platform engineering. Specifically, how large language models (LLMs) are helping platform teams build more compelling IDPs. We're already seeing lots of use cases for LLMs, and the potential for organizations to further automate repetitive tasks in a standardized way. With this in mind, now is the time to start thinking about what AI means to you. How useful would LLMs be to you and your role? While the technology is not quite there yet, is there a chance that in the future, an LLM could take over your job completely? Or will there always be a need for the human touch?

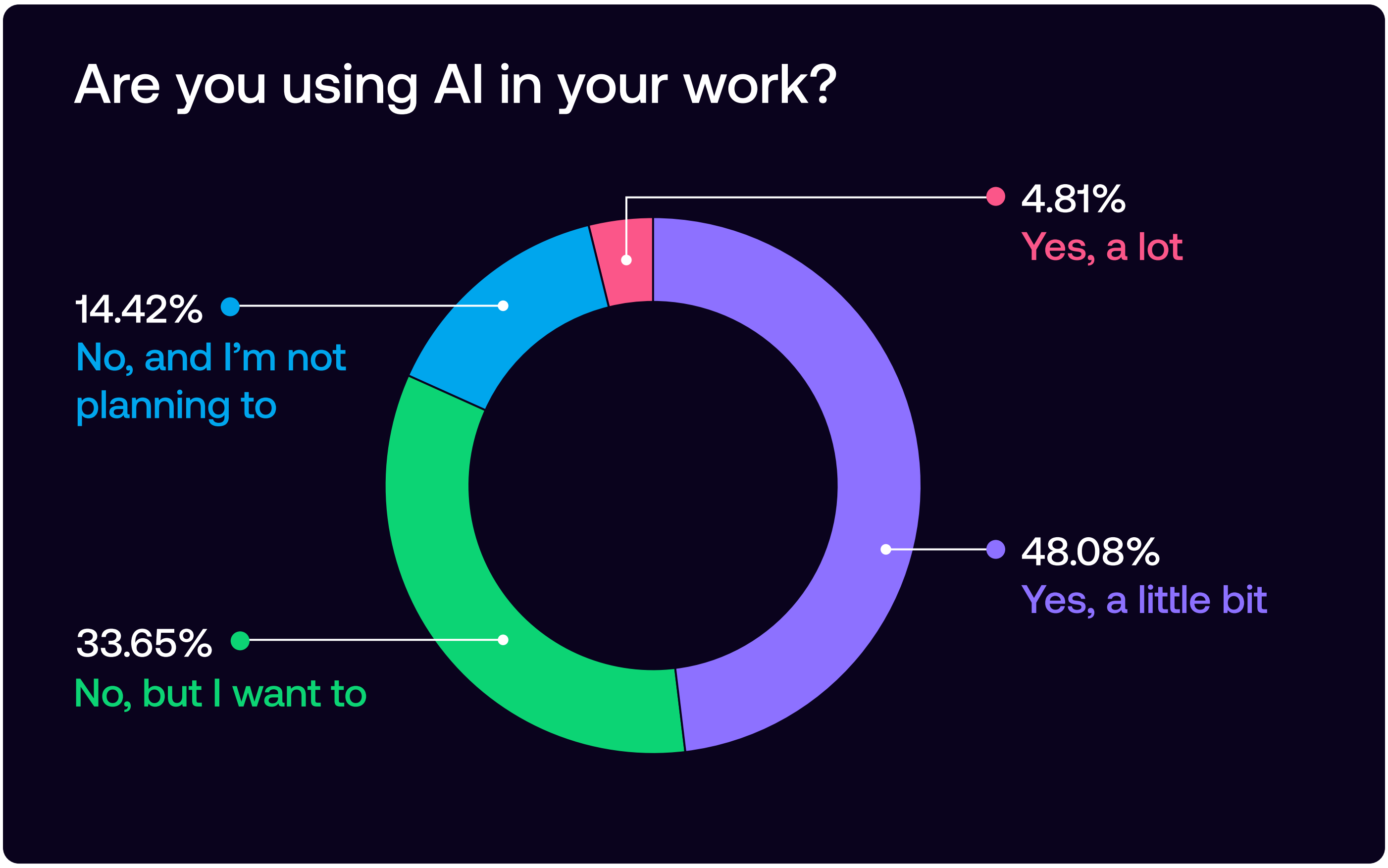
As a starting point check out the table below which shows some potential use cases for LLMs in platform engineering. Think about the ones that apply to you. Which would be easy to implement, and would be difficult?

🟩 Easy to implement 🟨 Medium 🟧 Hard 🟥 Very complex, will take expertise in LLMs

USE CASE PERSONA	Creating configurations	Usage of an IDP	Enablement / Documentation	Support
Developer	🟩 Create config files from natural language description. 🟨 Describe what config file does in plain English (reverse). 🟨 Fix errors in config files. 🟧 Co-pilot supporting real time creation of configs.	🟧 Query for deployment context and get a verbose response. 🟥 Commands and deployments through natural language interface.	🟩 Answer queries around products in general, features, application examples / sample code, command syntax, etc. with a 3rd party solution. 🟩 Answers queries around training content, training videos, tutorials, use case samples, etc. with a 3rd party solution.	🟧 Answer queries to general topics and get a verbose response. 🟧 Answer specific support queries around errors, issues, potential bugs, requests for samples code, use cases, also see Enablement/ Documentation. 🟧 Answers questions to problem sets around specific infrastructure configurations, security and compliance, etc.
Platform engineer	🟧 Co-pilot supporting real time creating of configs.	🟧 Query for deployment context or issues and get a verbose response. 🟧 Query for infra stats (change dates, list of resources, cost information, etc.). 🟥 Commands and deployments through natural language interface.	🟥 Answer queries around products in general, features, application examples / sample code, command syntax, etc. with a DIY solution. 🟥 Answer queries around training content, training videos, tutorials, use case samples, etc. with a DIY solution.	
Economic buyer / manager	n/a	🟥 Query for usage reports (usage statistics, list of resources and services, time saved through the IDP, cost information, ROI, etc.) as verbose response.	🟥 Verbose answers to value proposition, product, fit for pot. use cases, appl. by industry, ROI, etc.	🟥 Verbose answers to value proposition, product, fit for pot. use cases, application by industry, ROI, etc. 🟥 Guided questionnaire for benchmarking current stage of platform engineering journey.

Some of these use cases are low-hanging fruit that can more easily be deployed independently or with a 3rd party solution, and are reliable at least 95% of the time within weeks. In contrast, other use cases are harder to implement, demand specialist LLM expertise, and take longer to become reliable.

LLMs will undoubtedly change technology as we know it, along with how we use it. AI is already covering every quadrant of every Gartner hype cycle diagram out there, and its potential impact on the platform engineering space is being discussed in dozens of articles. But is it actively being used yet? The latest Platform Engineering 2023 survey confirmed that like in most verticals of software engineering (except maybe for copilot types of use), the AI hype has yet to be fully realized in the actual day-to-day work of platform engineers:



While it's still early days for these technologies, what's clear is that early adopters could quickly outpace their competitors when it comes to code output quality and quantity. And by getting ahead of the game and thinking about how you can leverage LLMs and AI, you'll be best placed to design a next-level, next-gen IDP that's future-ready and built for success.

A bright future for platform engineering

Platform engineering isn't just a trend. It's the future of software delivery. The discipline exists to design and deliver effective enterprise-grade IDPs, and promises to solve some of the biggest challenges faced by today's software engineering organizations. This includes the high complexity of the cloud-native landscape which hinders developer productivity and feeds ticket ops. By driving standardization by design, removing Ops bottlenecks, and enabling true developer self-service, IDPs radically transform software delivery. Through reports like this one, we continue to offer insight and guidance on key platform engineering topics.

It's an undeniably exciting time to be part of the platform engineering movement, and we can't wait to see where it goes next.

Next steps

To get started and build your enterprise-grade IDP fast, simply follow our new reference architectures with open-sourced implementation codes:

- Open source reference architecture for AWS
- Open source reference architecture for GCP

And head to our learning path for more guidance on how to master your IDP.

Resources

Manjunath Bhat: “A Software Engineering Leader’s Guide to Improving Developer Experience” (Gartner)

Lee Ditiangkin: “Why putting a pane of glass on a pile of sh*t doesn’t solve your problem”

Luca Galante: “What is platform engineering?” 2021

Luca Galante: “Internal Platform Teams: What Are They and Do You Need One?” 2021

Luca Galante: “DevOps vs. SRE vs. Platform Engineering? The gaps might be smaller than you think” 2023

Luca Galante: “Results are in: the 2023 platform engineering survey” 2023

Kaspar von Grünberg: “What Is an Internal Developer Platform?” 2021

Kaspar von Grünberg: “What is a Platform Orchestrator?” 2022

Kaspar von Grünberg: “What is Dynamic Configuration Management?” 2023

Aaron Erickson: “Why it is worth investing in Product Management for Internal Developer Platforms”

Paula Kennedy: “Whose cognitive load is it anyway?” 2022

Aeris Stewart: “AI is changing the future of platform engineering” 2023

Aeris Stewart: “Platform engineering maturity model: what we learned from our survey of ~300 orgs” 2023

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

Gartner” “Hype Cycle for Software Engineering” 2022

Gartner: “Hype Cycle for Software Engineering” 2023

Humanitec: “DevOps Benchmarking Study 2023”

Thoughtworks: Technology Radar, Vol. 16, March 2017

Thoughtworks: “Platform Orchestration” (Technology Radar, Vol 29) September 2023

Selected talks from PlatformCon 2023

Kaspar von Grünberg: “Build golden paths for day 50, not for day 1!”

Ralf Huuck: “How to measure ROI and make your platform effect visible”

Gregor Hohpe: “Build abstractions, not illusions”

Kristina Kondrashevich and Gang Luo: “Why we skipped SRE and switched to platform engineering”

Charity Majors: “The future of Ops is platform engineering”

Marco Marulli: “Streamlining developer platform design with reference architectures: Google Cloud use case”

Manuel Pais: “Beyond engineering: The future of platforms”

Stephan Schneider and Mike Gatto: “Platform as Code: Simplifying developer platform design with reference architectures”

Jon Skarpeteig: “Why treat your new Internal Developer Platform as a startup”

Susa Tünker: “How to eliminate config drift between environments”

Nicki Watt: “Why is it so hard to create a great Platform-as-a-Product?”

Humanitec GmbH

Wöhlertstraße 12-13, 10115 Berlin, Germany
Phone: +49 30 6293-8516

Humanitec Inc

228 East 45th Street, Suite 9E,
New York, NY 10017

Humanitec Ltd

3rd Floor, 1 Ashley Road
Altrincham, Cheshire WA14 2DT
United Kingdom

E-mail: info@humanitec.com
Website: https://www.humanitec.com

CEO: Kaspar von GrünbergRegistered at Amtsgericht Charlottenburg, Berlin: HRB 196818 B
VAT-ID according to §27a UStG: DE318212407
Responsible for the content of humanitec.com ref. § 55 II RStV: Kaspar von Grünberg