State of Platform Engineering Report

VOLUME 1

humanitec

State of Platform Engineering Report

Volume 1

About this report

Platform engineering is one of the hottest trends in DevOps and infrastructure. Gartner recently added it to their Hype Cycle for Software Engineering. Some DevOps leaders consider it the natural evolution of the "you build it, you run it" paradigm that kickstarted DevOps back in 2006.

The community growth speaks for itself. In the two years since its inception, the platform engineering community has seen the creation of nineteen Meetup groups pop up around the globe, with individual groups like Platform Engineers Austin gaining over 2k members. The Platform Engineering Slack has over 5k engaged contributors. In Summer 2022, the first ever platform engineering conference, PlatformCon, attracted over 6k attendees.

Despite its growing popularity, people in and outside of the community still have questions about this new discipline: What does platform engineering actually mean? What are the required skills to become a platform engineer? How much does a platform engineer make? What does the tooling landscape look like?

This paper will provide guidance on how to think about these topics and key resources from different corners of the community. By the last page, you'll be able to explain what platform engineering is and why your organization might want to pay attention to it.

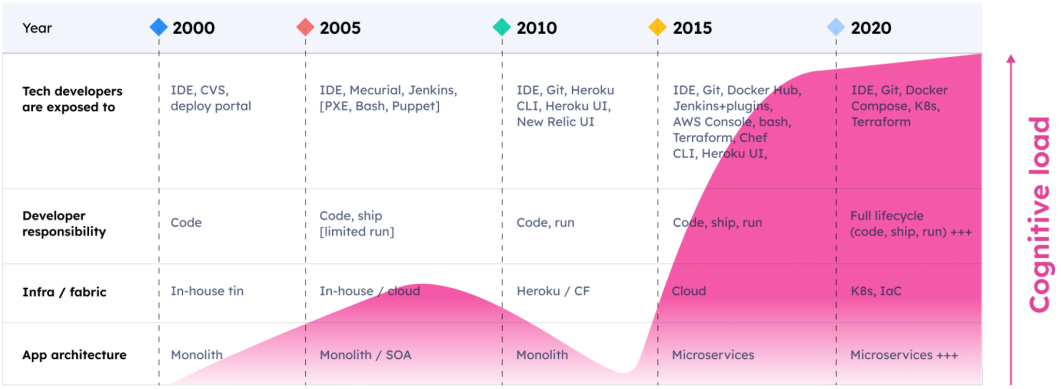
Let's dive in.

Where does platform engineering come from?

Just five years ago, platform engineering was not a thing people talked about. What was and has been on everyone's mind for the past decade instead is DevOps. Ever since Werner Vogels famously yelled “you build it, you run it” at the AWS launch in 2006, DevOps has become the de facto gold standard most engineering organizations have tried to build towards.

This drove a massive shift left movement, with developers being responsible for more and more of their applications’ lifecycle and delivery workflows. At the same time, more complex microservice architectures and technologies like Kubernetes, GitOps, and Infrastructure as Code (IaC) also became the industry standard.

These trends resulted in a modern cloud native setup that is infinitely more complex than it was just a few years ago. Even a simple task, like changing an environment variable before redeploying an application, requires developers to have an end-to-end understanding of their enterprise toolchain. This increased cognitive load on engineers, creating inefficiencies like shadow operations.



Inspired by Daniel Bryant at PlatformCon 2022

While many teams hopped on the DevOps hype train and tried to get developers to not only develop and ship features, but also own their infrastructure and deployment pipelines, most top performing companies took a different route.

It became increasingly clear that engineering organizations that invested into building an Internal Developer Platforms (IDPs) showed better performance on all DORA metrics than the vast majority of orgs that got stuck somewhere along the way of this DevOps transformation.

Already back in 2017, Thoughtworks Tech Radar outlined the difference platform engineering product teams were making:

> *The adoption of cloud and DevOps, while increasing the productivity of teams who can now move more quickly with reduced dependency on centralized operations teams and infrastructure, also has constrained teams who lack the skills to self-manage a full application and operations stack. Some organizations have tackled this challenge by creating platform engineering product teams. These teams operate an internal platform which enables delivery teams to self-service deploy and operate systems with reduced lead time and stack complexity. The emphasis here is on API-driven self-service and supporting tools, with delivery teams still responsible for supporting what they deploy onto the platform. Organizations that consider establishing such a platform team should be very cautious not to accidentally create a separate DevOps team, nor should they simply relabel their existing hosting and operations structure as a platform.*

Team Topologies co-author Manuel Pais noted how, in retrospect, it's surprising that platform engineering didn't get more attention sooner. There were good reasons for it to have done so. All of the top performing engineering organizations from the previous decade have one thing in common: they invested heavily in building their own Internal Developer Platforms to enable developer self-service, while finding the right level of abstraction to minimize cognitive load on engineers. Jason Warner, ex-CTO of GitHub, explained that the platform approach was the secret sauce behind GitHub's impressive infrastructure scale up. Courtney Kissler made a similar case for the transformation she led at Nike and Starbucks: "It simply wouldn't have been possible to operate at that scale without an IDP."

These stories used to be few and far between, but five years ago, they slowly started gaining more traction. In 2019, Manuel Pais and Matthew Skelton's "Team Topologies" brought the idea of platform teams building Internal Developer Platforms to a broader audience for the first time. From there, platform engineering exploded, with the first Platform Engineering Meetup groups popping up in 2021.

What is platform engineering?

Now that you know the history behind platform engineering, you'll better understand what it looks like in practice.

Luca Galante, core contributor of the platform engineering community and product leader at Humanitec, describes platform engineering as

> "*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."

But agreeing on a common definition is as tricky as it is important. In his PlatformCon keynote, Puppet's field CTO Nigel Kersten raised concerns that platform engineering might wind up like DevOps (a term that means both everything and nothing depending on who you ask) if the community doesn't come up with a descriptive model for it.

A common starting point for industry experts and thought leaders is the Team Topologies model.

Pais and Skelton differentiate between 4 different types of teams:

- Stream-aligned teams, aligned to a flow of work from a segment of the business domain and work on core business logic.
- Enabling teams that help stream-aligned teams overcome obstacles and detect missing capabilities.
- Complicated subsystem teams which form whenever significant mathematical/technical expertise is needed.

- Platform teams that provide a compelling internal platform to accelerate delivery by stream-aligned teams

This also makes clear what a platform engineering team is not: it's not another shadow Ops entity or another fake SRE team that handles tickets from devs who got stuck.

DevOps engineering tends to focus on certain teams and their individual challenges while SREs focus on production reliability, in their relationship to developers they are often said to act as gatekeepers. Platform engineering on the other hand is driven by a product mindset and sees developers as the customers they are serving by building their product, the Internal Developer Platform. To learn more, check out DevOps vs. SRE vs. platform engineering, some simply say, platform engineering is the next evolution of DevOps.

Some platform engineering principles & best practices

So now you want to get on the platform engineering bandwagon. But how do you do it well?

How can you make sure your platform team is aligned with the rest of your organization? What are design best practices? How do you know if your platform is creating value? Here, we'll look at the principles and insights shared by top platform teams and practitioners.

Clear mission and role

Your platform team *needs* a clear mission statement set in stone early on. The mission should fit into the overall goals of your organization and should focus on improving developer experience and productivity. "Building reliable workflows that enable developer self-service" is a strong example.

Proactively define the role of your platform team within your organization. The platform team should not be seen as another operations support that can provision DBs on demand. Rather, it should be seen as its own product team that builds products for its internal customers, the developers.

Treat your platform as a product

Once you have defined your objectives, you can use a product mindset to achieve them. As Gregor Hohpe emphasized at his PlatformCon keynote, you can try to "be smarter than everyone else and anticipate all of their needs or you evolve the platform based on user needs."

A product mindset requires conducting user research, soliciting user feedback, and getting internal buy-in. It helps platform teams focus on building features that provide real value to developers. The tight feedback loop ensures that the platform continues to be useful to its

engineers. Teams that treat their platform as a product don't get distracted playing with shiny new tech, they optimize for what developers actually need.

Focus on common problems

Once you have a good understanding of your developers' pain points, you can easily identify shared challenges across your organization. Your platform team should tackle those problems first by building a golden path that has a common solution built-in. You'll know if your golden path is working by gathering developers' feedback and looking at engineering KPIs.

Glue is valuable

Operations teams and (mistakenly) platform teams are often seen as a cost center within the organization because they don't ship any customer-facing features. They are simply the glue that holds the setup together. However, that glue is very valuable. Platform engineers need to embrace this function and advertise it internally as a key value-add they deliver.

Once you pave golden paths for developers and draft the blueprints to drive standardization by design, the main value your platform team creates is through connecting the different parts of your toolchain together. This improves the developer experience (DevEx) and self-service capabilities for the organization.

For a deeper dive into internal marketing for platform engineering, check out Galo Navarro's Salesman Tricks to get executive buy-in and developer adoption for your platform. Here's a quick summary:

- **Executives** → Sell them how the platform can level up key projects in a measurable way. Sell value creation, not cost reduction.
- **DevOps and Sysadmins** → Make clear the platform is an opportunity, not a threat.
- **Developers** → Make clear you build the platform for them so they can win the race.

Don't reinvent the wheel

Successful platform teams prevent the need for other teams to find creative ways to solve the same problems, hence reinventing the wheel over and over again. So it's important that platform engineers don't fall into the same trap. It doesn't matter if your platform team's homegrown delivery setup is better today. If it's a problem worth solving, OSS and commercial vendors will catch up eventually.

Spend your time wisely. Building another CI solution or metrics dashboard when there are other businesses dedicating all of their resources to the same thing is probably not the best use of your time.

You create the most value when you tailor off-the-shelf solutions to the specific requirements of your organization. Commercial competitors are more likely to optimize for generic needs, but your platform team can customize the solution to your teams.

Platform tooling landscape

Let's get more pragmatic here. Principles are a much needed starting point, but where do you go next? What tools should you actually consider for building your platform? The unexciting answer is, of course, it depends. It depends on your organization goals, on your developers' pain points and on some external factors like regulatory requirements.

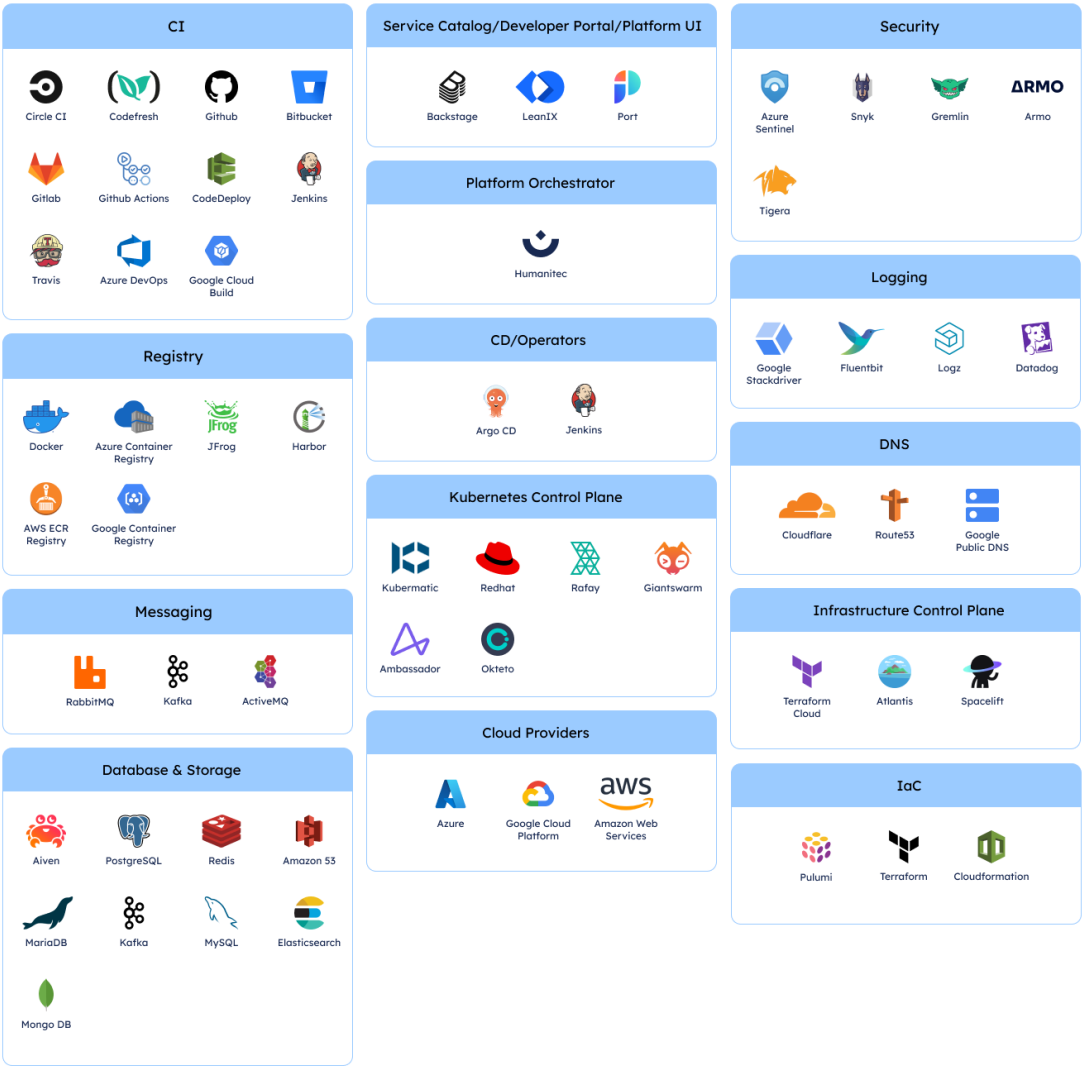
So while it's impossible to give a one-size-fits-all answer here, there are a few heuristics you can follow to look for tooling for your IDP.

However, there are a lot of things that you might believe are Internal Developer Platforms that are not. Platform as a service (PaaS) offerings like Heroku, end-to-end DevOps solutions like Gitlab, Azure DevOps, or open source tooling packages like Argo (ArgoCD, Argo Workflows, etc.) are not Internal Developer Platforms. They all cover certain parts of the software delivery flow, but they are not a realistic answer to any real life brownfield enterprise setup.

If you are a team of ten developers trying to get to market quickly, a PaaS like Heroku is more than enough to manage your infrastructure and configuration workflows. But that won't be sufficient for any engineering organization with more than fifty developers.

Developer portals or service catalogs like Backstage are sometimes also confused for an Internal Developer Platform. The same holds true for Kubernetes control planes, Environment as a service providers, infrastructure orchestration or DevOps Robotic Process Automation (RPA) tools.

An Internal Developer Platform, instead, is *the sum of all tech and tools that a platform engineering team binds together to pave a golden path or paths. Developers leverage this path to self-serve with low cognitive load*, Kaspar von Grünberg, CEO at Humanitec. But what are the tech and tools platform teams work with?



This overview attempts to visualize all relevant tooling categories you can use to build an IDP that also follows the Platform as a Product paradigm. While CI, registry, messaging, database & storage, security, logging, DNS, IaC and cloud providers should be pretty self-explanatory, let's have a closer look at the remaining parts:

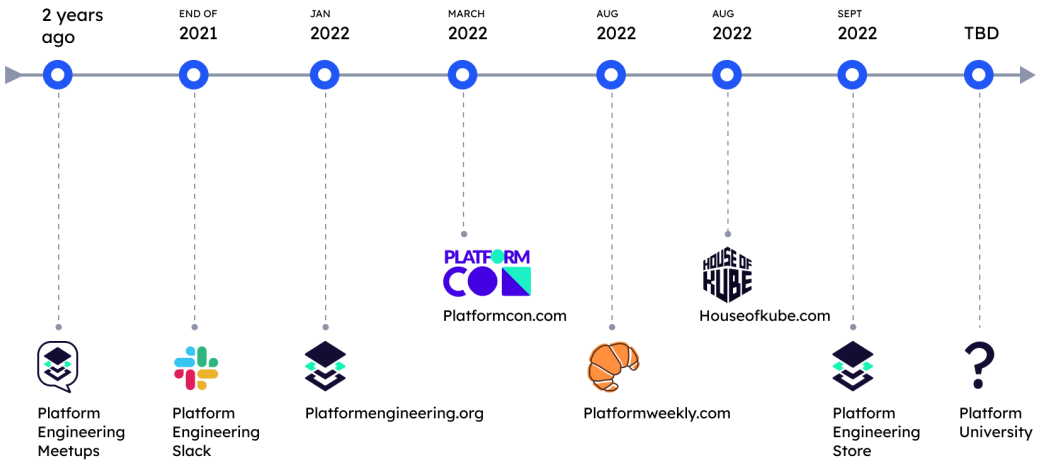
- Service catalogs, developer portals or platform UIs, e.g. Backstage: tools from this category are not an IDP, but they can play a very useful role in your IDP setup. As Gartner puts it:
  "*Internal developer portals* serve as the interface through which developers can discover and access *internal developer platform* capabilities."

- Platform Orchestrator: this is a new category that enables dynamic configuration management. A Platform Orchestrator is the centerpiece of every dynamic IDP.
- Kubernetes control planes: these are abstraction layers on top of Kubernetes that reduce the complexity developers are exposed to. Be aware that everything beyond Kubernetes is not covered.
- Infrastructure control planes: these are abstraction layers on top of the IaC setup to reduce the complexity developers are exposed to. Be aware that everything beyond IaC is not covered.

All of these tools can come together in different ways to form the backbone of your Internal Developer Platform. Spend some time to determine which tools are the right fit for your organization.

Community growth and how to navigate it

The tooling landscape is not the only proof that platform engineering is blowing up. The Platform Engineering community's growth is a strong signal that this is a trend every organization should take seriously.



As of September 2022:

- **The Platform Engineering Slack grew to over 5k active members after only nine months online.** With community-built channels for product managers, Kubernetes enthusiasts, and job seekers, Slack is the heart of the global community. Platform

practitioners share their war stories, share advice, and discuss best practices with folks around the world.

- **There are over 6k active platform engineers in 19 different Meetup groups.** Members in cities like Austin, London, New York City, and Tel Aviv have hosted over 50 meetups to date. You can watch the recordings of those talks on platformengineering.org and the Platform Engineering YouTube channel.
- PlatformCon, the first-ever virtual conference for platform engineers, saw over 6k attendees and 78 community-submitted talks. The conference was supported by 15 leading brands in the space including Google, Hashicorp, Puppet and Humanitec.
- Platform Weekly is a community-driven email newsletter that delivers bite-sized pieces of the best of the platform engineering and cloud native worlds, straight to folks' inboxes.
- The new store section of platformengineering.org is getting rolled out in September and much more is to come (hint: it might be called Platform University).

The community is growing faster than any of its founding members ever imagined. As it grows, it is consolidating more insight, resources, and support for platform engineers worldwide.

Taking the platform engineering path

Is platform engineering worth it?

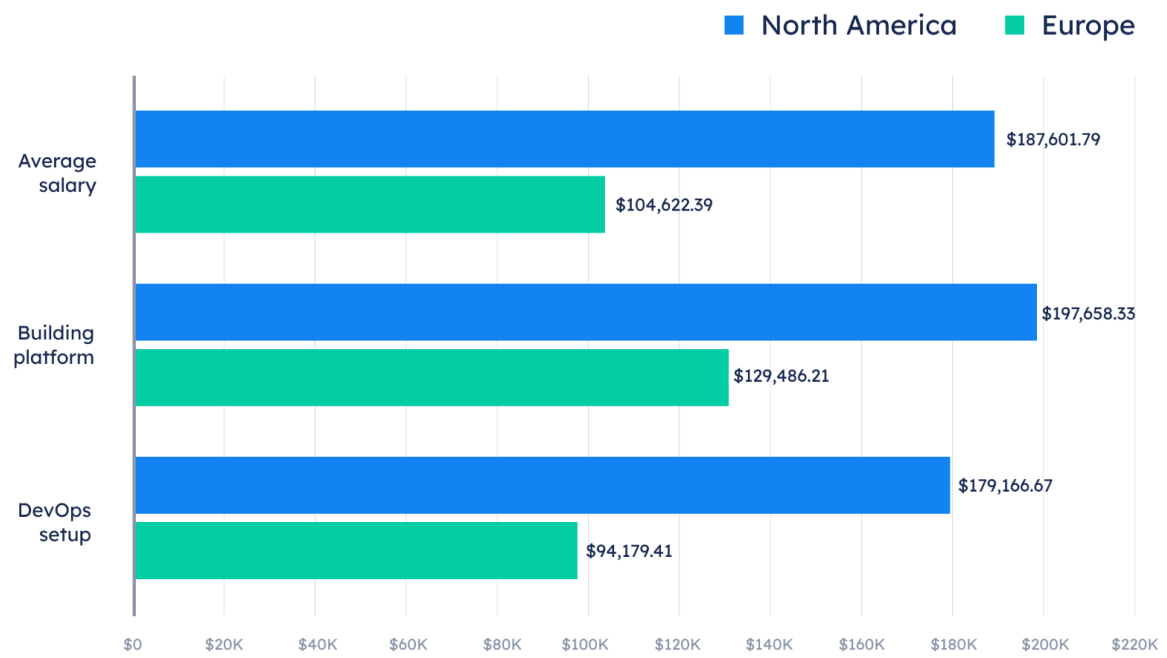
This is a question that frequently comes up from people who are considering taking the leap into a platform engineering career. If platform engineering is a trend, is it worth betting a career on? Does it pay off?

To get more insight, we ran a small survey within the Platform Engineering community. We focused on North America (the United States and Canada) and Europe. In the survey, we asked a wide range of questions to get an idea of the current state of platform engineering: what folks are working on, the tools they use most often, their salary, etc.

Here's what we found.

Annual salary

Let's start with the good stuff. How much could you make as a platform engineer?



In North America, platform engineers make

9.4%

more than DevOps engineers

In Europe, platform engineers make

19.4%

more than DevOps engineers

Unsurprisingly, the average salaries in North America, across the board, are higher than those in Europe. This is consistent with most other reports in the industry, such as the Puppet DevOps Salary Report.

What is surprising is the gap between respondents building a platform and those who just look after the DevOps setup. In both regions, platform engineers earn more on average than their

DevOps engineers peers. The gap is almost 9,4% in North America and double that – 19,4% – in Europe.

Why is this the case? Companies are realizing that building a platform adds more value to their bottom line. Employees with the right experience (and product mindset) are hard to find, harder than your typical DevOps engineer.

Roles and titles

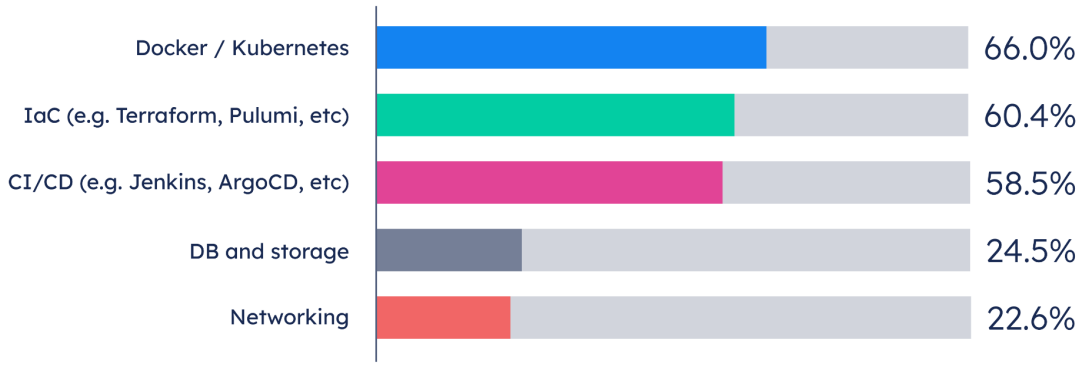
Another interesting finding. While 46,5% of the respondents claimed to work on building a platform, job titles like Platform Engineer or Head of Platform were still comparably rare within this group (22,64%).

While 46,5% of the respondents reported that they work on building a platform, only 22,64% reported having a job title like Platform Engineer or Head of Platform. Instead, most respondents had job titles like Senior Software Engineer, Principle Engineer, Senior DevOps Engineer, SRE, IT Architect, etc.

This disparity illustrates that we are still transitioning towards understanding what platform engineering is and defining platform roles accordingly.

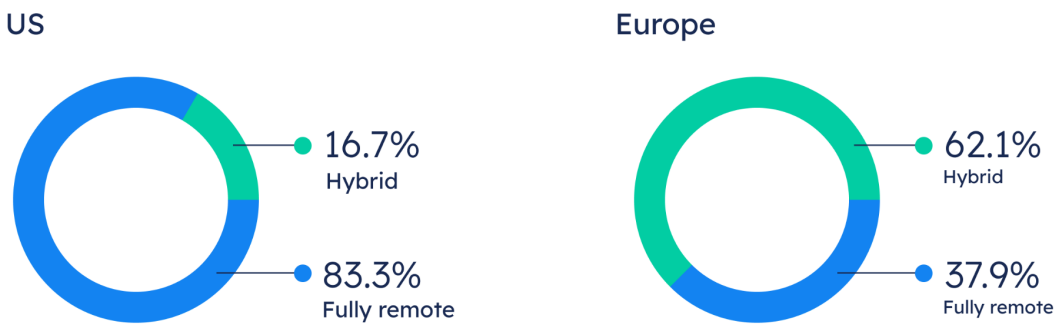
Technologies

To the question of what their main focus areas are, respondents who are building a platform answered as follows (multiple selection possible):



It is not surprising to see Kubernetes and Infrastructure as Code leading the pack. Platforms are normally built as a layer on top of clusters and infrastructure, streamlining both configuration management and infrastructure orchestration.

Working setup



Platform engineers are at the forefront of the remote work revolution! In North America especially, platform engineers are overwhelmingly remote. In Europe, there's a more balanced split between fully remote and hybrid options. Either way, 100% in-office is 100% dead.

What's next?

Now that you know what's up with platform engineering, what comes next?

If you want to connect with fellow platform builders, the Platform Engineering Slack channel is the best place to start.

If you want to learn more about best practices, check out the talks from PlatformCon or any of the community webinars.

You can also contribute to the community blog, host a meetup, or write a quick bite for the Platform Weekly newsletter.

We also have lots of exciting projects in the works: in-person events (like this post-Kubecon techno party and interactive workshops), platform blueprints, reference implementations, and more. The Platform Engineering community is just getting started.

Don't know where to start or have a question we didn't answer? Drop us a line at info@platformengineering.org.

Resources

Ditiangkin, Lee: Why putting a pane of glass on a pile of sh*t doesn't solve your problem, https://platformengineering.org/blog/why-putting-a-pane-of-glass-on-a-pile-of-shit-doesnt-solve-your-problem

Galante, Luca: Internal Platform Teams: What Are They and Do You Need One? https://humanitec.com/blog/internal-platform-teams-what-are-they-and-do-you-need-one

Galante, Luca: DevOps vs. SRE vs. Platform Engineering? The gaps might be smaller than you think, https://humanitec.com/blog/sre-vs-devops-vs-platform-engineering

Gartner, A Software Engineering Leader's Guide to Improving Developer Experience by Manjunath Bhat, Research VP, Software Engineering Practice at Gartner, https://www.gartner.com/document/4017457

Gartner, Platform Engineering, Hype Cycle for Software Engineering, 2022, https://www.gartner.com/interactive/hc/4017202

Grünberg, Kaspar von: What Is an Internal Developer Platform, https://humanitec.com/blog/what-is-an-internal-developer-platform

Grünberg, Kaspar von: What is a Platform Orchestrator? https://humanitec.com/blog/what-is-a-platform-orchestrator

Grünberg, Kaspar von: What is Dynamic Configuration Management? https://humanitec.com/blog/what-is-dynamic-configuration-management

Humanitec DevOps Setups: A Benchmarking Study 2021, https://humanitec.com/whitepapers/2021-devops-setups-benchmarking-report

Kennedy, Paula: Whose cognitive load is it anyway? https://platformengineering.org/blog/cognitive-load

Puppet State of DevOps Report 2020 https://puppet.com/resources/report/2020-state-of-devops-report

Puppet State of DevOps Report 2021, https://puppet.com/resources/report/2021-state-of-devops-report

Puppet DevOps Salary Report 2021 https://media.webteam.puppet.com/uploads/2022/03/Puppet-2021-DevOps-Salary-Report.pdf

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

Thoughtworks, Technology Radar, Vol. 16, March 2017

https://www.thoughtworks.com/content/dam/thoughtworks/documents/radar/2017/03/tr_technology_radar_vol_16_en.pdf

Selected talks from PlatformCon 2022:

Hohpe, Gregor: The Magic of Platforms
https://www.youtube.com/watch?v=WaL3ZbLgMuI

Kersten, Nigel: A Prescription for Platform Engineering?
https://www.youtube.com/watch?v=u4o1jcQIaYo

Navarro, Galo: Salesman tricks for the Platform Engineer
https://www.youtube.com/watch?v=ApEOiNC4GrA

Pais, Manuel: Platform as a Product
https://www.youtube.com/watch?v=b8YHCDMxqfg

Watt, Nicky: People, Process & Platform - A community focused approach
https://www.youtube.com/watch?v=JRlHGRklQxY