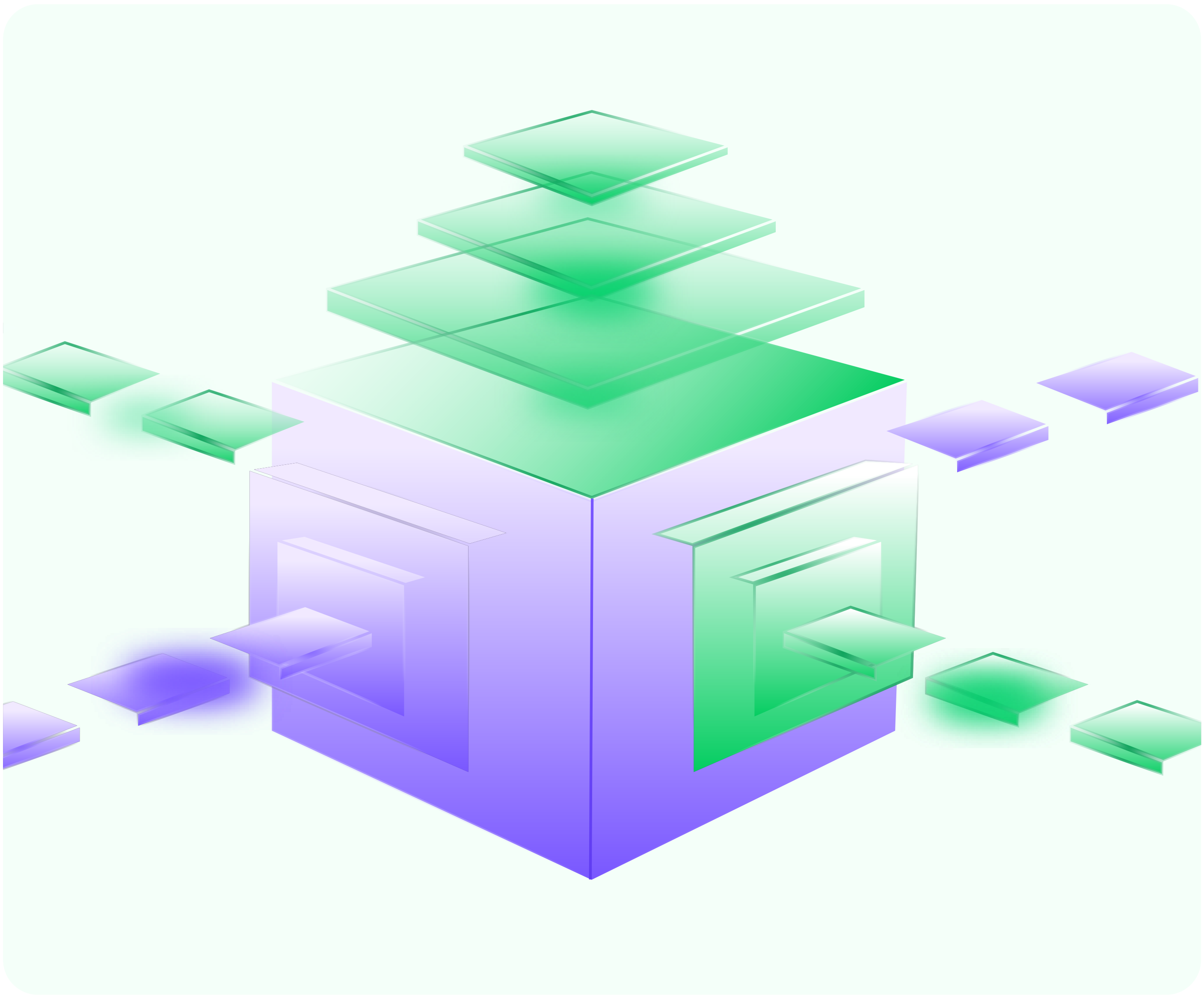
Platform as a Product: the key to platform engineering success

Platform Engineering

This short whitepaper is aimed at defining one of the foundational concepts in platform engineering, Platform as a Product. We'll go over what it means and why it's so important to our emergent discipline. We'll also discuss the main challenges in adopting a Platform as a Product approach and look at best practices distilled from working with 100s of platform engineering teams.



What is Platform as a Product and why does it matter

Let's start with a definition. Platform as a Product refers to building the end result of a platform engineering initiative, an Internal Developer Platform (IDP), as a product. This means there's a dedicated product team (the platform team) that follows established product management principles and, crucially, treats developers as their internal customers. This platform team would be responsible for building tight feedback loops across the different application development teams to ensure the IDP is solving their challenges and their major pain points.

The platform should abstract the complexity of the underlying infrastructure stack, while at the same time providing developers with the right amount of context they need to do their job. In order to strike this delicate balance it's essential for the platform team to listen closely to developers and gain an intimate understanding of the problems they are facing.

To better understand this, a helpful analogy is seeing the platform team as a startup that's developing a product (and a respective go-to-market motion). Imagine this hypothetical startup' product has a total addressable market (TAM) that is equal to all the application developers in the engineering organization. Similar to a startup, the platform team will launch a V1 of the platform (a Minimum Viable Platform, or MVP), then iterate on it continuously based on the feedback of its users (and customers) - the developers. The platform team is not only responsible for developing the IDP but also for marketing it internally, progressively gaining support from developers as well as other key stakeholders (e.g. architects, infrastructure and operations, security, executives, etc.). In larger enterprises, there might be multiple platform initiatives and respective teams vying to conquer the engineering org's TAM by marketing their own IDP solutions.

For engineers especially with Infrastructure and Operations or DevOps background this might require a significant mindset shift!

This customer-centric focus is an important change from the traditional approach of infrastructure and DevOps engineers (or SREs, Cloud Ops, etc.), who have historically been focused on adding and maintaining infrastructure, “teaching” developers how to use it for their needs best. This approach doesn’t scale and leads to developers being frustrated and overwhelmed by the increasing complexity of their enterprise toolchains, waiting on operations colleagues to provide them with the resources (e.g. DBs, environments) that they need to do their job. The flip side of that is ops teams becoming a bottleneck as they fail to deal with the growing backlog of TicketOps.

In contrast to this, platform engineers glue the tech and tools of an enterprise into opinionated, predefined golden paths that enable developer self-service and reduce cognitive load on the individual contributor. They build an Internal Developer Platform as a product, pre-packaging the underlying infrastructure in a self-serviceable layer that drives both automation and standardization across all workflows and teams. Unlike their predecessors, they don’t look at the platform as a “one and done” project that has a beginning and an end. Platform teams continuously iterate on the IDP features to ensure growing stakeholder buy-in and developer adoption.

Platform as a Product is essential to provide an Internal Developer Platform that lets developers move faster without breaking things (i.e. staying compliant and following security best practices). This is what ultimately drives down time to market for the entire organization.

Key challenges

Hopefully it's now clear why a product mindset is essential for a successful platform engineering initiative. However, there are important challenges to implementing a proper Platform as a Product approach. Let's take a look at the main ones.

Shortage of product management talent

There are not enough good product managers (PMs) who have the experience to build such complex products catering to the needs of so many different stakeholders. At the same time, most companies hesitate to put their best technical product managers on platform topics. Although leveraging your best PM talent for the IDP buildout and rollout might actually be the highest ROI option, many organizations prefer to allocate them to customer-facing products, while the platform is treated as a cost center.

WHO CAN HELP?

Upskilling with courses like the first-ever official Platform Engineering community course, 'Platform Engineering Fundamentals,' which focuses on key concepts such as Platform as a Product and helps you evangelize them to both your platform team and the broader engineering organization.

Politics

Product management is a tricky balancing act and it only gets trickier when it comes to platform product management. While customer-facing products benefit from the most direct feedback there is (sales), internal products such as IDPs are affected by more complex dynamics that go beyond simple supply and demand and often include a higher degree of internal politics. Who decides how to build the platform, developers or the CIO? It's not always as clear as it should be.

Listening too closely

We mentioned the importance of tight feedback loops between platform teams and developers. That often means relying on one of the pillars of good product management: user research. An easy trap to fall into, however, is interviewing developers and then going on to build exactly what they said they wanted, just to have the frustrating experience of no platform adoption right after. How's that possible? Didn't I give them exactly what they asked for?

Doing user research is not about building what developers ask you. It's about identifying and understanding their pain points and coming up with solutions on a higher abstraction level. Just remind yourself of the famous Henry Ford line: "If I'd asked customers what they wanted, they would have told me, 'A faster horse!'" Source: Walter Isaacson, HBR

And many more

For the sake of this whitepaper's brevity, we won't go into detail on all of the many challenges (we'd rather focus on some of the solutions and best practices, see next section👇), but here are some other worthy mentions including:

- The fact that platform engineering means innovation and change, and most people (and organizations) are naturally averse to change.
- Platform engineering initiatives touch so many different parts of your organization and the workflows of virtually every stakeholder. Platform Product Managers (PPMs) thus need to be able to navigate this complexity and face the daunting task of needing to bring everyone on this journey.
- Ending up with a huge feature backlog for your platform and a foggy vision for your product roadmap (a classic of product management).
- Measuring the wrong metrics e.g. lagging vs leading indicators, therefore focusing on optimizing the wrong things.

To address these and other challenges, we have collected 10 best practices, distilled from looking at how top-performing engineering organizations and their respective platform teams drive successful platform engineering initiatives.

10 PROVEN BEST PRACTICES

01 Build vs. buy vs blend

The market reached a maturity level where it doesn't make sense anymore to build everything from scratch. You should make a business case and do your ROI calculations as early as possible. According to the 2023 study, the best-performing platform engineering teams blend OSS with commercial vendor offerings but do not build everything from scratch. Check the platform tooling landscape for inspiration.

02 Apply well established architecture patterns

With a three-tier architecture (presentation, application, data): start building from the backend; do not simply put a developer portal as a presentation layer on top of your existing setup and build additional logic into it. Build the house first, then the front door.

03 Everything as code

Consider code as the single source of truth which helps maintain transparency, increase reliability, and simplify maintenance. An IDP that's code-first at its core allows for disaster recovery, versioning, and structured product development principles. This does not exclude further interface offerings such as a UI (portal), CLI, or API.

04 Build golden paths over cages

Do not try to please everyone. To adapt to different situations, meet diverse needs, and benefit from evolving technologies, staying open-minded is key. But your platform design should not try to cover every technology on earth or convince every developer in a user base. Do not assume that you will be able to please 100% of developers. Instead, consider achieving 80% a great win.

05 Take an 80/20 attitude to platforming

Do not try to please everyone. To adapt to different situations, meet diverse needs, and benefit from evolving technologies, staying open-minded is key. But your platform design should not try to cover every technology on earth or convince every developer in a user base. Do not assume that you will be able to please 100% of developers. Instead, consider achieving 80% a great win.

06 Leave platform interface choice to the developer

To ensure adoption, give developers the freedom to use the interfaces they're most comfortable with and that best meet their needs. Provide the option to use an OSS workload specification like Score, a portal (GUI), CLI, or API.

07 Security from scratch

To get buy in, implement security best practices from the get-go. If the V1 of your platform doesn't fulfill security and compliance requirements and if there is no proof that the platform will even support ensuring security and compliance by design, security teams will veto and your platform initiative is dead before it could even properly start.

08 Measure from the beginning

Measure success with hard numbers to support informed decisions and generate stakeholder buy-in. Choose metrics wisely, considering both leading (e.g., automation and complexity scores) and lagging indicators (e.g., DORA metrics). Track leading indicators in non-production environments early on. Remember to include NPS scores for developer satisfaction, as well as stability metrics, SLOs, and SLAs.

09 Gain stakeholder buy-in

Make sure all stakeholders have a seat (besides the developers - your customers). From security to compliance and legal teams, from architects to I&O teams, and important for the funding of your platform engineering initiative: executives. Make sure you build a platform team where important stakeholders are represented by heralds and the team goals are aligned with those of your stakeholders.

10 Think about adoption from the first day

If the platform is not used, it is dead. This is about internal marketing/evangelism. Identify the right first team to onboard and make them advocates of your platform. They are essential for platform success and developer adoption.

How to get started

You might be sold on the importance of Platform as a Product and eager to become a best practice platform engineering team, but the question lingers: how do you even get started building your Internal Developer Platform as a product?

Here again, we can draw from well-established best practices in product management:

- Leverage existing blueprints, use for example our reference architectures to design your target platform.
- However, don't spend months designing and planning for a full-blown enterprise-grade IDP in all its parts and with a full rollout.
- Instead, start small and optimize for quick iteration. Start with a Minimum Viable Platform and follow a structured approach to gain stakeholder buy-in and adoption.
- Make a strong business case and walk your management through it. We are happy to help.

We are available to help you design and implement your MVP, and we can support you with making a strong business case. If you want to take the next step in your platform engineering, , contact us.

PlatCo GmbH

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

PlatCo Inc

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

PlatCo Ltd

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

E-mail: info@platco-group.com
Website: https://platco-group.com

CEO: Kaspar von Grünberg
Registered at Amtsgericht Charlottenburg, Berlin: HRB 262650
VAT-ID according to §27a UStG: DE367439464
Responsible for the content of humanitec.com ref. § 55 II RStV: Kaspar von Grünberg