THE ART OF PLATFORM ENGINEERING

A RUSSIAN DOLL EFFECT

+ explore Self-Service and Internal Developer Portals inside

cycloid

TABLE OF CONTENTS

INTRODUCTION

Demystifying Platform Engineering, SSP, and IDP

Ever found yourself scratching your head over which tools are the perfect fit to kick-start your platform engineering journey? You're not alone. It's easy to get lost in the jumble of acronyms like IDP and SSP, especially when they seem to promise similar things but deliver quite differently. You might expect an Internal Developer Portal (IDP) to save your devs from repetitive tasks and boring work, while what you are actually searching for is a Self-Service Portal (SSP) to simply streamline your automation.

This ebook is here to clear up the confusion once and for all. It will guide you through the maze of platform engineering tools, from the SSP that gets your automation humming, to the IDP that streamlines team collaboration, from smooth app building and documentation to easy deployment environments. And let's not forget the bigger picture – Platform Engineering solutions that go beyond the code, helping you design, implement, and maintain a robust ecosystem for software development and infrastructure management.

The Russian Doll Effect

But we're not just here to regurgitate industry jargon. We're shaking things up by challenging the status quo. According to Gartner, the core of any platform engineering strategy is an IDP that **provides a curated set of tools, capabilities, and processes for easy consumption by the development team**. But let's face it, the need to consume IT infrastructure, particularly in large organizations, goes beyond developers.

This ebook is reclaiming the narrative that Platform Engineering focuses on empowering everyone across your organization, from coders to stakeholders, and tackles the challenges of modern software development and infrastructure management together.

In fact, the relationship between SSP, IDP, and Platform Engineering resembles a Russian doll – each nestled within the other, revealing layers of complexity. It's like a puzzle where every small detail matters, even if it seems insignificant at first glance.

Software Development & Infrastructure Management

Let's also consider that while platform engineering has core principles that apply universally, how it's put into practice can vary greatly depending on factors like the industry, goals, and services of the organization. Think of it as a tailored suit – what works for a software company might not be the best fit for other types of organizations. This customization, along with the scale and whether the focus is more on product development or internal services, can shape whether an IDP, an SSP, or Platform Engineering is the top priority.

CHAPTER 1

A self-service portal at the heart of Platform Engineering

A functional self-service portal is the beating heart of your digital transformation journey and an integral part of Platform Engineering. Not to toot our own horn, but here at Cycloid we are somewhat experts at what makes a great SSP, since after all, it is our bread and butter.

3 most important things that will make SSP a success in your organization:

Governance – create order amidst chaos

Start with understanding the whos and whats of your process and create a clear governance framework. Who is the project owner? A good self-service portal should have defined roles and accesses for every user.

Service Catalog – build a service buffet

Once you've got your roles down, choose which services you want to integrate into your service catalog and self-service portal which is maintained by the platform team. We would also advise a full GitOps approach for this to enhance efficiency.

Automation – relieve your devs' plates

A great SSP lets anyone interact with tools and the cloud without needing to be a technical expert. Roles and accesses are built-in and there's minimal room for human error because your automation takes care of the manual tasks.

A note on DevX

While the Gartner definition of Platform Engineering focuses heavily on the developer experience, that's just one piece of the puzzle. Your platform isn't just for developers; it's a tool used by a variety of teams, including Solution Architects, Ops Teams, DevOps, FinOps, Security Teams, or Network Teams. To truly succeed, it's vital to take into account their satisfaction and efficiency. After all, it's the people who make your business tick.

Developers vs End-users experience

Developers by trade	Developers by default
Developers, software engineers	Solution Architects, Ops Teams, DevOps, Security also Researchers in Universities, Webmasters, or anyone else who is required to code as part of their job but it is not their core function
Working with deployment tools, automation, infrastructure	Working with internal tools, dashboards, or admin interfaces
DevX: boosting developer productivity by reducing friction in tasks such as coding, debugging, testing, and deployment.	**UX:** boosting team productivity by providing intuitive interfaces, robust documentation, and efficient workflows.

Here’s a real-life use case:

A market research company finds themselves in need of infrastructure to support their work. While coding may not be their main gig, it's often necessary. A self-service portal becomes their go-to solution, offering the necessary tools for both the developers writing code and interns conducting market research. As for developers, the usefulness of an IDP can vary depending on the complexity of the code and the level of collaboration needed. It's all about finding the right tools to streamline the process and get the job done efficiently.

By providing a user-friendly interface to define and deploy cloud resources, you can abstract away the time and effort required to set up complex infrastructure configurations. This is the main difference between a self-service portal and an IDP:

- **Removing the requirement for users to understand how the infrastructure is being deployed.**
- **Allowing non-technical users to provision infrastructure and applications using a self-service portal**

In the next part, we’ll dive further into the specifics of internal developer portals.

CHAPTER 2

Internal Developer Portal as the bridge between teams

An internal developer portal (IDP) is a layer on top of your self-service portal that offers a suite of tools to streamline the entire software development lifecycle. Unlike a self-service portal that focuses on functionality and automation, an IDP is all about developers' collaboration and simplified access to automation for everyone involved.

An IDP is typically designed with the expanded user circle in mind and includes both developers by trade and developers by default. This means that the portal features better, more intuitive UX to improve DevX and overall team experience alike. By providing a centralized and cohesive environment, an IDP aims to enhance developer productivity, foster collaboration among team members, and ultimately accelerate the delivery of high-quality software products.

Take a look at the following add-on features that distinguish a successful internal developer portal.

Access to automation for non-dev team members

Non-developers that we outlined in the last chapter, such as project managers, QA testers, and even business stakeholders, all benefit from having access to automation tools through an IDP. These tools empower non-developers to keep track of projects, identify bottlenecks, and gain insights into the development process without needing deep technical expertise.

Simplified UX for better DevX

As we said before, DevX is not simply about pizza parties or even extremely efficient workflows – it's about creating an environment where everyone in your organization, from software engineers to management, can do their best work. An IDP expands the definition of DevX to include non-technical members of the team, which means the portal has to accommodate their non-technical needs. That's why UX – everything from pretty icons to intuitive functionality – will be the puzzle piece that makes your platform complete. It's not just user-friendly – it's user-oriented! Here are a few UX-specific things an IDP should have:

- **Address the users' needs – satisfy both devs and devs "by default."**
- **Provide efficient functionality – can it do the thing for a minimal number of clicks?**
- **Safeguard important infrastructure – don't underestimate human error and put security guardrails in place**

Team harmony through improved collaboration

For a successful IDP, collaboration isn't just a buzzword – it's baked into every corner of the portal.

Think of shared repositories where developers can stash their code for safekeeping, version control to keep everything running smoothly, and code reviews to make sure everyone's on the same page. And let's not forget about those integrations with all your favorite development tools, making it easier than ever to work seamlessly across platforms.

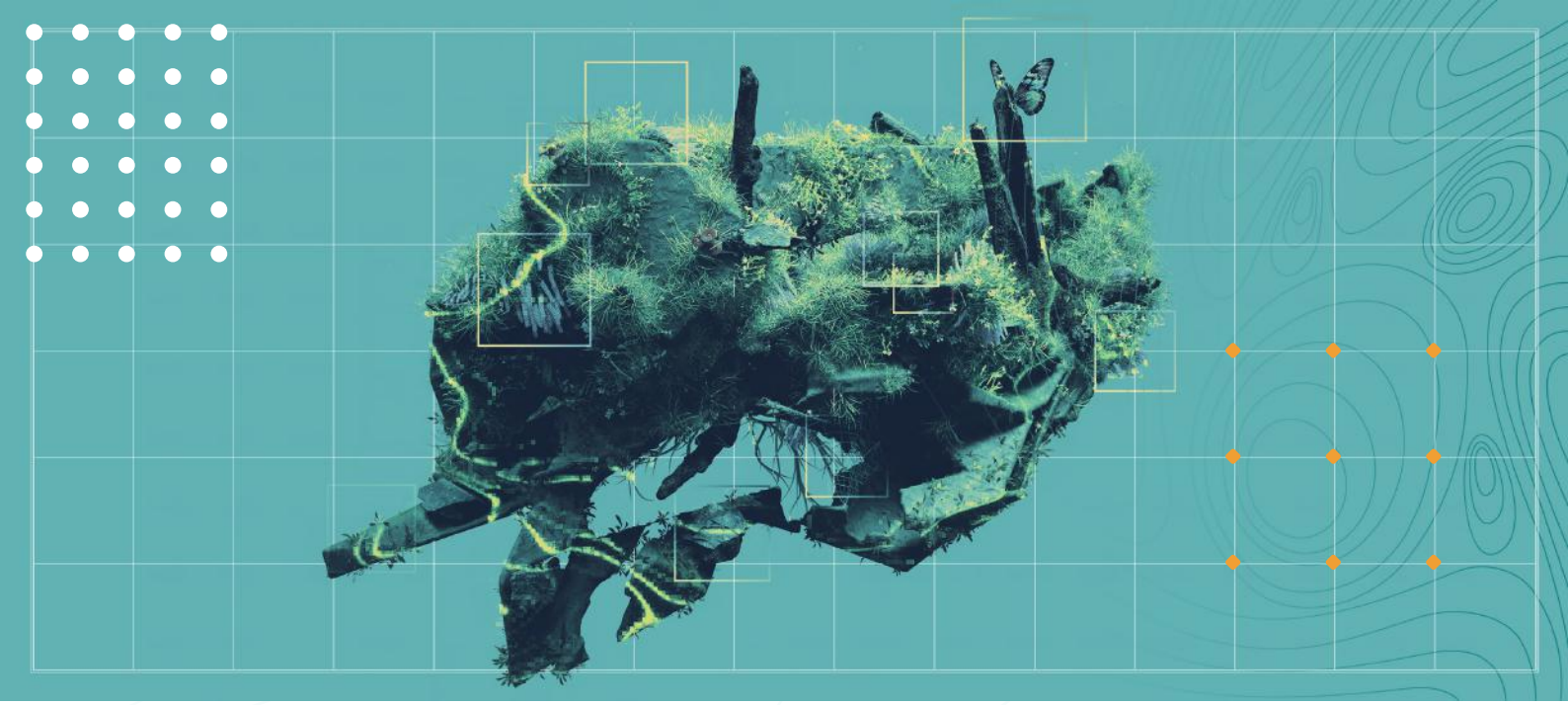
But here's the best part: all of these features aren't just there to make developers' lives easier (although that's definitely a bonus!). They're designed to streamline the entire software development lifecycle, from ideation to deployment, and foster open communication and collaboration within your development teams.

Sharing in the knowledge of documentation

Documentation guides developers through the jungle of code and features. For newcomers, documentation is their lifeline – it helps them bootstrap their projects and hit the ground running. It's also a valuable resource for seasoned developers, offering insights into advanced features, best practices, and tips and tricks to supercharge their workflow. Moreover, documentation serves as a central hub where developers can share their knowledge, contribute their insights, and collaborate on projects together.

Myth buster: should you use Backstage?

Backstage, the Spotify open-source platform has made big waves in the platform engineering space, making every company believe they can achieve developer bliss with just a bit of tinkering. And it's true, its plug-in system makes internal developer portals extremely customizable and collaborative. However, if you're new to this world, its DIY approach can make it more confusing than worth it. Backstage works best in tandem with other, off-the-shelf solutions to guide your digital transformation. That way, you get the best of both worlds – the accessibility of an established platform with the versatility of Backstage.



CHAPTER 3

The Art of Platform Engineering

We've finally arrived at the big doll, the big boss that everyone (in our humble opinion) gets wrong. That's why it typically takes nearly 3 years, and often even longer, to start seeing results from platform engineering.

An engineering platform is a comprehensive solution that transcends the boundaries of SSPs and IDPs. It's an overarching concept that spans a wide array of functionalities, including resource management, ecosystem collaboration, and plug-in integration.

Designed to cater to individuals of all skill levels and departments, it offers a user-friendly interface and customizable features to suit diverse needs and workflows. Whether you're a developer, a project manager, or a member of the operations team, the platform is tailored to empower you with the tools and capabilities necessary to thrive in today's fast-paced digital landscape. Let's take a look at what these tools are.

Project lifecycle management – observability

Platform engineering allows users with a stake in the game (managers, CFOs, or even Sustainability Officers) to zoom out of the day-to-day software development tasks and look at the bigger picture. Project lifecycle management and observability are crucial for gaining insights into the platform's performance and behavior, which involves integrating tools like event and project monitoring, cloud cost transparency, application performance, infrastructure health, and user interactions.

Efficient management entails monitoring key performance indicators (KPIs), setting up alerts for critical events, and leveraging logs and analysis tools for enhanced visibility into application behavior, error tracking, and issue resolution. Additionally, setting up tracing systems helps to follow requests as they move through different microservices and components. This helps your teams identify performance bottlenecks, understand latency issues, and optimize system behavior.

Ecosystem collaboration – plug-ins

Plug-ins allow you to empower users to tailor the platform to their specific needs. Any missing feature you can think of – simply integrate it into the platform via a plug-in and extend its capabilities!

Such flexibility improves scalability, so as your user base grows or changes, you can customize your platform accordingly. And if third-party devs want to join in the fun and contribute to the ecosystem growth, they can, as new functionalities are easily integrable.

Additionally, it speeds up development by recycling existing components, meaning less time spent reinventing the wheel!

Continuous Deployment and orchestration

Continuous Deployment involves automating the process of pushing code to the infrastructure. While some applications are straightforward and stateless, complex industry applications often have dependencies that require updates or notifications.

This complexity can be difficult to manage manually.

Automating the process not only minimizes human error but also promotes consistency across development environments, increasing developers' confidence. Moreover, the incorporation of continuous monitoring and feedback mechanisms within CI/CD pipelines ensures that only thoroughly tested and reliable code progresses to production.

FinOps & GreenOps

Benjamin Brial, the founder of Cycloid, stated in an article for DevOps.com:

> ***By combining FinOps and GreenOps approaches and placing sustainability at the orchestration layer to empower users across the organization to consume less infrastructure, organizations can reduce software delivery/cloud costs and even their carbon emissions.***

Keeping an eye on where and how your cloud budget is being spent, as well as slashing your carbon emissions is an integral part of a platform engineering strategy. An engineering platform can integrate cloud cost management and carbon footprint management to ensure that your software development process is not just human resource-efficient, but also financially streamlined and environmentally conscious.

Drift detection – an ecosystem matter

When using Infrastructure-as-Code (IaC) tools to set up infrastructure, keeping track of Drift is crucial. Drift happens when there are differences between what you planned and what actually got set up, often because someone made changes manually. Fixing these differences might mean tweaking your IaC code or rolling back changes. Drift detection is an ongoing process that needs continuous monitoring to make sure everything stays on track. It's like keeping your garden tidy – you have to keep an eye on it regularly to prevent weeds from taking over.

When it comes to drift detection, different situations call for different approaches. For portals focused on technical teams (IDPs), drift detection might not be the main focus. But at the Platform Engineering layer, where there's more comprehensive infrastructure management through a Self-Service Portal, drift detection becomes really important to make sure everything stays in line.

Drift detection isn't just useful for developers, though. It's also important for security teams to catch any potential vulnerabilities, and for compliance and governance teams to make sure everything follows the rules. Ops teams use drift detection to fix issues before they become big problems, and business stakeholders care about it because it affects how reliable their apps and services are.

Resource management – keeping in control

Resource management offers a way of maintaining control across multiple cloud platforms and accounts and keeping resource usage in check per project and environment.

As companies increasingly adopt dedicated accounts per project across various cloud platforms, management becomes challenging.

Centralizing resources into a unified platform facilitates the detection of “shadow IT” (informally deployed IT solutions outside of official channels). This not only helps find resources that aren't being used to save money but also helps make cloud practices more eco-friendly by getting rid of resources you don't need (which is also great for the environment!).

Multi-tenancy – controlled flexibility

Multi-tenancy is a software architecture concept which means that one software system serves many users or groups at once, with each user having their own private space. These users or tenants can be individuals, groups, or even entire organizations, each with their own isolated set of data and configuration settings. Multi-tenancy allows for efficient resource utilization, as multiple users share the same infrastructure while maintaining data privacy and security.

But it’s not simply about sharing; rather, it's about fostering autonomy and promoting governance best practices. Multi-tenancy effectively dismantles isolated solutions that various teams might be utilizing, bringing the entire organization together. By embracing multi-tenancy, platforms can ensure streamlined operations, enhanced collaboration, and strengthened governance across the board.

CONCLUSION.

We hope you get the reference to the Russian doll now. A self-service portal is the heart of your platform, responsible for automation and governance. It's typically designed by Platform Engineers and used by developers.

The internal developer portal takes all the great bits an SSP offers and makes them better, focusing mostly on developer collaboration and DevX.

Platform Engineering is all about setting up a smooth system, no matter how big your crew gets.

Each of these services – SSP, IDP, and Platform Engineering work as a complex layer on top of each other, but there wouldn't be an engineering platform without a self-service portal.

Whether you're starting with a small team of developers or scaling up to involve the whole organization, remember to keep the focus on governance, repeatability, and reproducibility. When it comes to picking the right fit, think about your team's needs, your business goals, and what tools will make their jobs easier.

So go ahead, and make your choice with confidence. After all, with the right tools and the right team, there's no challenge you can't conquer!

About Cycloid

Cycloid is the sustainable platform engineering company with a mission to promote efficient infrastructure and software delivery alongside digital sobriety. Cycloid optimizes platform engineering, alleviates cognitive load on IT teams, and enhances Green IT and FinOps practices. Placing sustainability at the orchestration layer, the Cycloid engineering platform is a comprehensive solution for platform engineering teams and end users, delivering optimal UX with modular self-service portal access to project lifecycle, resource management, FinOps and GreenOps capabilities. With a zero lock-in, GitOps first approach, Cycloid encourages a culture of digital sobriety that flows through an organization, making DevOps and cloud delivery more efficient and cost-effective.