WHO ARE YOU, PLATFORM ENGINEERING

Abstract: Platform Engineering is a leading technological trend embraced by companies worldwide, as they develop their Internal Development Platforms (IDP). This methodology has gained popularity amid the ongoing digital transformation of businesses. Building IDP platforms offers competitive advantages to businesses in the era of digital transformation by streamlining development processes, enhancing efficiency, and enabling innovation.

Key words: cloud, platform engineering, internal development platform, devops, sre

Authors: Nguyen Thi My Tu

The problem defined: DevOps is Dead with the Rise of Platform Engineering

There is a lot of discussion going on, where some people are asking whether platform engineering replaces DevOps. Many people say it goes hand in hand and is rather an addition to DevOps, but it’s a bit more complicated. So let's clearly define, what platform engineering exactly is.

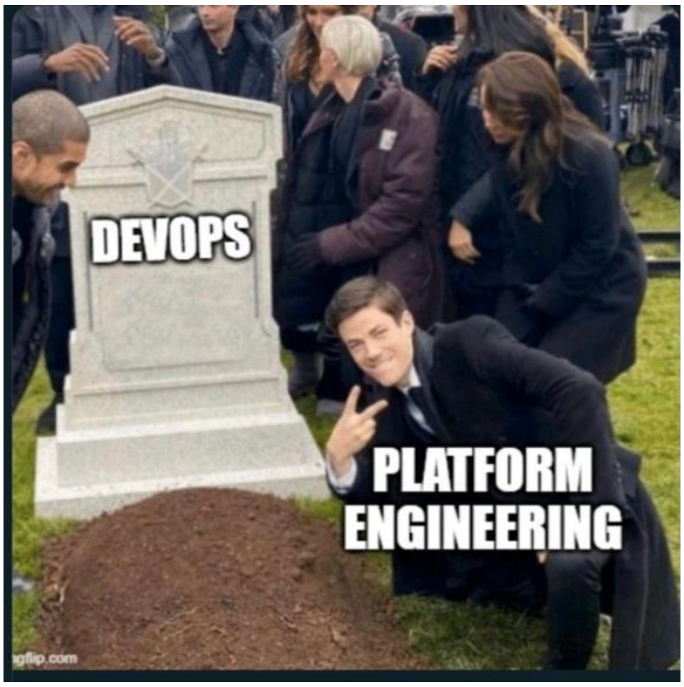

Figure 1 – Platform Engineering meme

1 DEFINITION

Platform Engineering is one of the major technological trends of 2024. It is a methodology for designing and integrating tools and workflows, creating abstractions that help reduce the cognitive load on developers in the era of cloud technologies. In other words, it is a way to organize your specific corporate universe to make life easier for development teams by simplifying access to resources in the broadest sense. In return, businesses gain acceleration in product development cycles.

The collection of such tools and processes is called an Internal Developer Platform (IDP). An IDP is created by a team using the "platform as a product" approach. This means that the platform team must treat developers as their internal customers. This is logical because the main goal of creating and implementing an IDP is to relieve developers of non-core tasks. Additional productivity doesn't come from nowhere: to add another ticket to a programmer's workday, you have to remove some other ticket from it.

Why has this need arisen? What problem couldn't be solved by other methods and means that necessitated the invention of Platform Engineering?

2 TRADITIONAL TEAMS ARE FALLING BEHIND

The specialization of workers to increase productivity was devised even before the First Industrial Revolution. The traditional structure of an IT department consists of several separate teams of developers and system administrators. Developers are responsible for programming the application. When everything is ready, they hand over the packaged application to the administrators. Administrators are responsible for deploying and running the application. Thus, we always have dedicated operational groups with the appropriate expertise.

Everything was fine as long as speed was not a critical factor for business. The process was inflexible and slow. Developers waited for feedback from administrators when they needed some infrastructure changes or infrastructure resources. On the other hand, administrators waited for developers to fix something in the application, which affected deployment.

3 DEVOPS TO HELP YOU BUT …

With the advent of Agile, the DevOps paradigm emerged, addressing the most critical issues of traditional teams. Most importantly, it resolved communication problems between application development, deployment, and operations. This became a unified team, resulting in a much more flexible and faster way of working.

Now there is a single team where either the developers themselves or a dedicated DevOps engineer manage Docker containers, configure CI/CD platforms and create pipelines, write Kubernetes manifests, set up logging and monitoring, and maintain all this infrastructure as tools evolve and new service releases come out. And all this is in addition to the actual application development.

The DevOps paradigm enhances flexibility, speed, and efficiency, but it also adds a colossal cognitive load to the team.

Let's say there are 10 such teams developing various applications. They all face the same problems and needs. Each team does things differently, using their own stack. As a result, we end up with 10 DevOps teams creating and operating 10 different platforms for delivering and running their applications. This is extremely inefficient, wasteful, and creates a high burden on individual engineers. Consequently, the teams have less time to implement business requirements. Moreover, each team needs to add experienced engineers, leading to increased costs for human resources. And scaling all of this becomes increasingly difficult.

4 SELF-SERVICE CHECKOUT

How does Platform Engineering address the problems outlined above?

In the era before supermarkets, you had to ask the shopkeeper across the counter if they had sour cream, then find out what fat content you wanted, and specify how much you needed and in what container to pour it. A great quest for developing communication skills, but it created queues, often led to misunderstandings, and significantly reduced the store's throughput. It would be great to standardize all products and eliminate the human factor from this scheme! But wait a minute...

Every application in any project requires a version control system, CI/CD, deployment in some runtime environment, such as Kubernetes, which, in turn, requires basic infrastructure. The application's infrastructure needs monitoring, logging, proper security, and so on. In general, there are many areas that need to be configured for the application to function correctly. And each of these areas has many different tools and technologies that the platform team standardizes across the entire organization.

It sounds like we've come back to traditional interaction, where developers request infrastructure resources from another team. Therefore, to provide access to infrastructure, the platform team takes the necessary tools, standardizes them, and configures them properly, creating a layer of abstraction with convenient interfaces. This can be a WebUI, API, or CLI application.

Now development teams can independently service any services and tools they need. Thus, the platform team creates an Internal Developer Platform, serving as an internal developer platform.

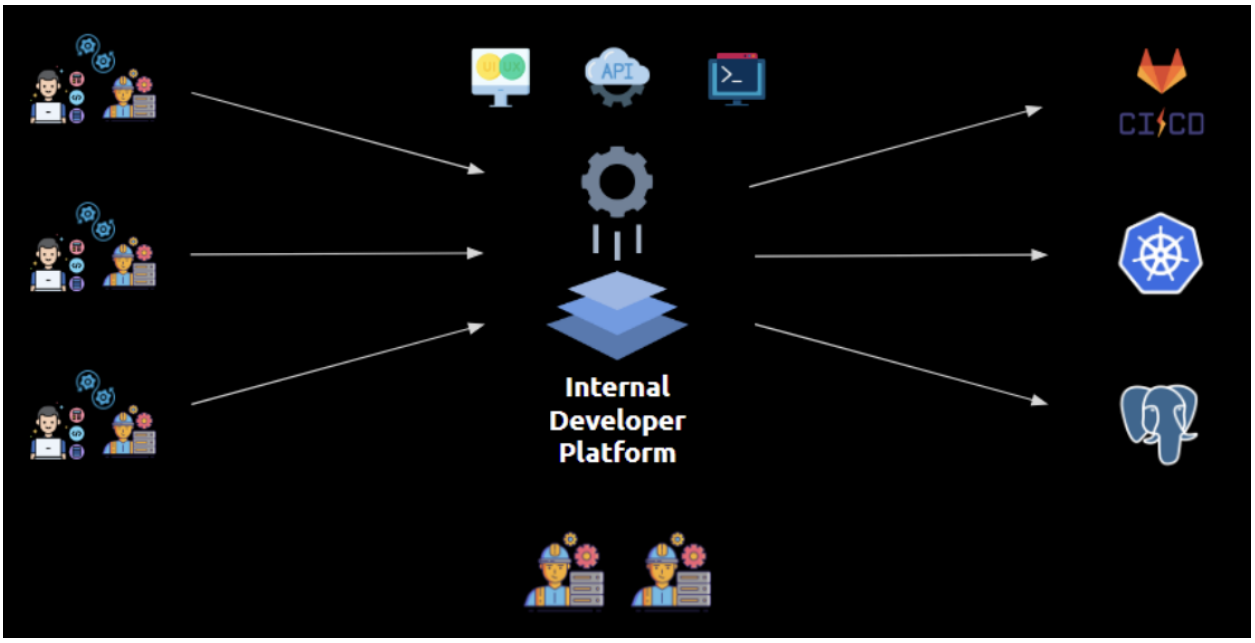

Figure 2 – How Internal Developer Platform looks like

5 PROS AND CONS, AS YOU LIKE IT

Let's summarize. Unlike traditional or DevOps teams, the structure of a Platform team has obvious advantages for both sides.

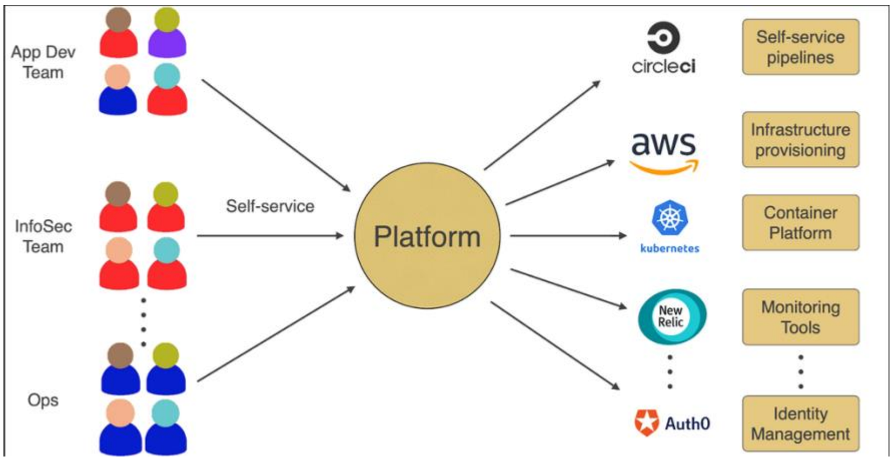

Figure 3 – Platform Engineering Team

Pros for the Platform Team:

– No TicketOps: There's no need to be distracted by developers' and testers' problems and requests, allowing focus on strategic initiatives.

– Standardized Infrastructure: Easier to maintain due to standardization across the organization.

Pros for Developers:

– Self-Service: Developers can easily deploy, reconfigure, or even destroy new environments without delving into the complexities of Kubernetes.

– No Local Environment: There's no need to set up the entire local infrastructure for development.

Cons for the Platform Team:

– Initial Setup Effort: Establishing the standardized infrastructure and abstraction layers requires significant initial effort and planning.

– Maintenance Burden: The platform team needs to continuously maintain and update the platform, ensuring it meets the evolving needs of the development teams and integrates new technologies.

Cons for Developers:

– Learning Curve: Developers need to learn how to use the self-service tools and interfaces provided by the platform team, which might be different from traditional tools they are used to.

– Dependency on Platform Team: If the platform team encounters issues or delays, it can impact the developers' ability to work efficiently.

6 FEATURES OF IMPLEMENTING AN IDP

Currently, almost all large companies are either exploring or already using IDP. In Russia, this trend is actively followed by major companies with in-house development centers and extensive expertise in IT infrastructure automation. In such cases, the resources and time spent on building and supporting IDP are fully justified.

In the enterprise segment, IDP platforms are often built on cloud orchestrators (such as HP CSA, VMware vRA, OpenStack, etc.), which provide an IT service builder and a wide range of plugins for quickly creating self-service portals with service catalogs. Typically, integrators with the necessary expertise assist companies in the enterprise segment along this path.

At the same time, many companies develop development platforms from scratch without using ready-made solutions. These are typically companies with a high level of development culture that have a clear understanding of what they want from the platform and are willing to invest significant resources in its creation and maintenance. This allows them to integrate their own custom infrastructure services (Observability, IaM, etc.), CI/CD tools, and other solutions into IDP.

In other words, IDP in such companies is tailored to the internal development business processes.

7 WHY PLATFORM ENGINEERING IS TRENDING?

The Platform Engineering approach has become a trend because the challenges it addresses have become widespread, and the benefits of its implementation are evident.

Positive outcomes of using Platform Engineering include:

1. Alignment of Technological Stack: Enabling different teams to use the same services and tools available in the IDP enhances cross-team efficiency, minimizes "shadow IT," and reduces competency silos.

2. Minimization of Technology Sprawl: Building a self-service portal helps establish a unified set of tools used across the company, reducing unnecessary duplication of solutions and simplifying stack support, administration, and licensing costs.
3. Knowledge Sharing: Developing comprehensive documentation accompanying the IDP guides users through common scenarios, ensuring expertise continuity and minimizing the bus factor.
4. Enhanced Security: The IDP platform serves as a single access point to tools, enabling the integration of security solutions for access control verification and approval procedures, utilizing approved security resources and IT services.
5. Standardization and Improved Collaboration: Platform Engineering involves using specific toolsets and frameworks, facilitating application development and establishing clear component standards for monitoring, logging, tracing, and integration, thereby simplifying development and increasing efficiency.
6. Accelerated Onboarding: Implementing an IDP platform with a unified stack and common tool operation rules reduces the onboarding time for new users or teams by integrating with development tools, providing single sign-on mechanisms, and ensuring pre-approved access rights.
7. Fast and Non-Bureaucratic Resource Acquisition: Platform Engineering enables easy access to computing resources through pre-agreed catalogs of IT services, reducing resource acquisition time from weeks to minutes.

All these aspects relieve development teams from solving typical tasks, allowing them to focus on delivering value to the business.

8 IMPLEMENTATION TIMELINES AND EFFECTIVENESS EVALUATION

Implementing an IDP platform is an ongoing journey without a final destination. Even after restructuring internal processes, building self-service portals, and moving away from technology sprawl, it's crucial for companies to continue developing the platform, updating the available stack, enhancing user experience, and more. Tools, market needs, and business processes evolve, so the IDP as a product also requires regular updates.

Typically, companies spend several years adapting their business processes to utilize an IDP platform. Timelines heavily depend on the size and competencies of the team involved in building the IDP platform, as well as the chosen approach to implementation — whether it's based on an existing solution or developed from scratch.

It's challenging to directly assess the effectiveness of investments in such projects. Therefore, when determining the rationality of investments, several criteria are considered conceptually.

– Ratio of Developers to DevOps Engineers For instance, if before implementing the IDP platform, 10 development teams required 10 DevOps specialists, then after the solution's implementation and scaling development by three times, the number of DevOps engineers would increase only slightly — for example, to 15 people (without the platform, the growth would be almost proportional).

– Release Frequency: Due to reduced coordination, checks, and secondary tasks, as well as improved interaction between development teams, the time-to-market is also reduced. This allows for an increase in the number of releases without compromising quality and significantly increasing the workload on developers.

– Number of Permitted Errors: Standardizing tools and technologies, as well as the ability to obtain necessary resources without complex manipulations, allows for a significant reduction in the number of errors during deployment and deployment.

– Effectiveness Evaluation: Formalizing approaches and tools enables setting and tracking metrics (both technical and business), which, in turn, help evaluate the effectiveness of changes or quickly identify bottlenecks."

9 EXAMPLES OF SUCCESSFUL USE CASE

The primary stakeholders driving the implementation of an IDP platform typically include:

– The technical department, aiming to increase development productivity and reduce the number of errors.

– Development teams seeking to address systematic issues related to DevOps processes and decrease reliance on DevOps specialists.

– The business, aiming to reduce time-to-market for development and enhance control over internal processes.

However, companies often undertake IDP platform implementations only after achieving a certain level of maturity in internal processes and team scale. There are numerous successful implementation cases, even in the Russian market. Here is a example: **Tinkoff**: A major bank serving millions of clients and providing dozens of IT services. Tinkoff is heavily focused on digital, boasting a large IT workforce and a substantial pool of IT tasks. To unify self-service APIs, tools, services, developments, and competencies, Tinkoff created an internal platform that effectively addresses the needs of thousands of internal developers.