White Paper: **January 2024**

A Practical Approach to Getting Started with Platform Engineering

vmware by Broadcom

Table of contents

Why platform engineering

In an ever-changing digital landscape, a platform offers a consistent and scalable foundation for technology, services and infrastructure that engineers can develop applications and products on. It enables development teams to concentrate on creating business-differentiating value and not be distracted by managing infrastructure and navigating complex governance policies. Platform engineering merges multiple concepts, including DevOps and platform as a product (PaaP), providing an efficient and effective system for software development and operations.

This white paper presents practical steps for getting started with platform engineering, informed by the real-world experiences of Tanzu Labs by Broadcom. Tanzu Labs, previously known as Pivotal Labs, is a forerunner and leader in PaaP, which simplifies application deployment, scaling and management. In the realm of platform engineering, Tanzu Labs methodologies have become a cornerstone for organizations around the world.

Building a successful platform

Beginning the platform journey while building the desired capabilities and getting them to a production-ready state can be overwhelming. We recommend using agile methodologies with quick feedback loops, combined with data-driven decisions, to prioritize, develop and improve the platform features, treating the platform like a product.

The initial step in building a new developer platform is to collect data from your existing platform and its users. The data collected helps you prioritize which features to develop for your new platform and which existing features to enhance. The DevOps Research and Assessment (DORA) program recommends four basic metrics to determine the performance of your platform: the amount of time it takes to get code to production, how often features are released to production, the percentage of deployments that cause a production failure, and how long it takes to restore service after a failure.

Tanzu Labs has broadened the DORA metrics to assess a platform's health and maturity as a whole and help guide decision-making for provisioning new platform capabilities.[1] In our experience, the ability to measure the current state and evaluate the needs of the platform's users provides the information necessary to make informed decisions about platform vision and strategy. These health markers evaluate the many facets of a platform team and the culture that they are operating in, which are often highly correlated with achieving healthy DORA metrics. Understanding the bigger picture and addressing underlying challenges are essential in building a successful platform.

1. See The 10 Platform Journey Health Markers: A Roadmap to Continuous Improvement.

Focusing on platform health

The Tanzu Labs platform teams have identified these areas of focus and key metrics for understanding the health, resilience and effectiveness of a platform.

- **Platform as a product** – The platform is treated like a product that not only includes the platform software but also provides access to any services and integrations that make it a viable environment for applications to run, as described in Why You Should Treat Platform as a Product. As such, the platform's capabilities can change in response to the needs of the users.
- **Balanced team** – The platform team consists of a product manager and at least two platform engineers with a combination of infrastructure and software engineering skills. The balanced team focuses on delivering user experience (UX) by taking advantage of user-centric design (UCD).
- **Path to production** – Golden paths are created to provide consistent pipelines to get application workloads into production quickly and securely. Developers have self-service access to the correct infrastructure abstractions and services and can take full advantage of the platform via modern and optimized tools and processes.
- **Monitoring and metrics** – System health is made observable. By understanding and measuring the desired service behavior, such as response latency, error rate and peak utilization of resources, discrepancies can be corrected.
- **Capacity planning** – Future demand is forecasted, and platform capabilities that applications depend on have enough capacity to satisfy load demands under normal conditions and the ability to scale as needed to meet service-level objectives (SLOs).
- **Platform updates** – Platform capabilities are upgradable while preserving desired service levels. Activities can include canary deployments, rolling upgrades, tested quick fail-back procedures, and governing service changes with error budgets, as defined by site reliability engineering best practices. Platform lifecycle management is automated to drive operational efficiency and cost-effective infrastructure consumption. The platform is kept consistently secure, compliant and patched.
- **Emergency response** – Performance degradation is observable and incidents are responded to effectively to stay within error budgets.
- **Self-service** – Self-service options are provided, such as for documentation, tutorials, APIs and SDKs, as well as easy access to platform features through portals or dashboards, to foster innovation and enable developers to work autonomously and at their own pace.
- **Performance optimization** – Service component performance, efficiency and resource utilization are characterized and tracked to identify and address regressions and drive improvements.
- **Business continuity** – Plans for business continuity are frequently tested and demonstrated to meet service level, recovery time and recovery point objectives.

Evaluating platform maturity

Tanzu Labs evaluates team health and defines areas of improvement to bring maturity to the organization as well as the platform. The level of maturity—from chaotic to continuous improvement—guides the journey and which areas to prioritize for improvement as the platform evolves. Measuring platform maturity over time helps teams identify where to focus efforts within each platform health area.

Level 1: Chaotic – Metrics are not established. Procedures are unknown or not documented. Roles and responsibilities are not enumerated. Assigned results depend on the capabilities and circumstance of the individuals in the group and vary widely.

Level 2: Defined – Roles and responsibilities are defined and documented. Tasks are assigned to individuals and tracked to completion.

Level 3: Managed – Metrics are enumerated, and target values are established and documented. Procedures are documented.

Level 4: Measured – Metrics are measured and compared to target values, and tasks are created to correct deviations. Conformance to procedures is measured. External data on performance, such as customer satisfaction surveys, are collected.

Level 5: Continuous improvement – Target metric values are reviewed regularly with the customer base and modified as needed. Procedures are reviewed regularly and updated when improvements are found internally or externally.

The platform engineering team can assess the maturity of the developer platform by measuring its capabilities over time within all the focus areas and then adjusting plans and the future implementations of tools.

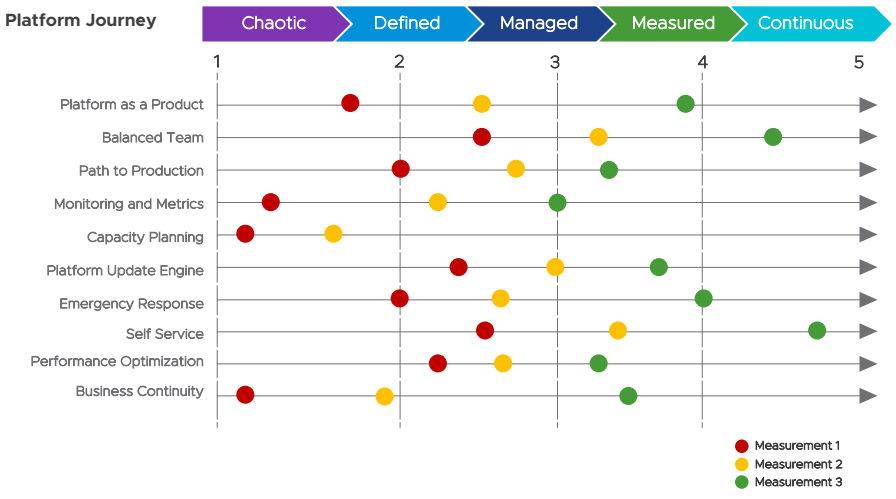

Figure 1: Evaluating team health and defining areas of focus using a maturity matrix helps assess capabilities and target areas of improvement on the platform journey.

Developing a balanced team

Having a dedicated and balanced team of individuals with a diverse range of skills, knowledge and expertise in various business, developer and technology contexts influences the success of a platform. It is also crucial that the platform team is empowered with the necessary resources and authority to execute the platform's vision.

Balanced Platform Team	
1	Platform Champion "The Protector"
1	Product Manager "The Visionary"
2+	Platform Engineers "The Architects"

Figure 2: A balanced team includes these roles.

Platform champion

The platform champion, who is often at or close to the C level, has the will and the political capital to protect the team as they embark on the platform journey. The platform champion creates an environment where change can happen and empowers the team to make it happen. This individual has a track record of internal entrepreneurship or organizational transformation with the ability to change a conversation from "here's why this can't work" to "how can we make this work?"

Product manager

The platform product manager defines the strategy, manages the feature backlog, makes data-driven decisions regarding assumptions and features, and communicates the product roadmap to customers and stakeholders. The product manager is a visionary who typically represents one of these archetypes, depending on the desired outcome for your platform and your current business and organizational needs:

- The **dreamer** who asks "why?" and is not constrained by legacy thinking and processes
- The **alchemist** who can take many disparate requirements and distill them into a succinct product vision
- The **influencer** who has strong relationships with business partners, application teams and IT
- The **minimalist** who understands the value of shipping early and often
- The **lean champion** who relentlessly pursues process change

Platform engineers

The platform team also requires a combination of infrastructure and software engineering skills. Because managing change sustainably requires treating operations as if it is a software problem, platform engineers must be able to code. Identify at least two people with complementary domain expertise across business, developer and technology contexts. Your platform engineers need to encompass these traits:

- The **infrastructure architect who codes** and has experience in production operations and automating repetitive tasks
- The **natural automator** for continuous integration and deployment (CI/CD) work, automating the current release management process, or doing system automation
- The **curious software engineer** who is agile in solving platform needs by going down the stack to automate infrastructure

Practical first steps with platform engineering

Embarking on a platform engineering initiative can seem daunting at first. However, there are logical places to start that set you up for success.

View your developers as customers

To adopt the discipline of platform engineering and build a successful platform, one of the key shifts is to have a customer mindset regarding developers' needs. Developers want a platform that is user friendly, flexible and customizable. It is crucial to map the journey from a business idea to a production-ready application feature to understand the key requirements and expectations of the developers who will use the platform.

Developers generally want to reduce friction in their work by having access to tools, APIs and documentation as well as robust testing and debugging functionality. Providing a platform that offers seamless integration with other services and systems enables them to leverage existing technologies, easily scale their applications, and add new tools and technologies as needed. Regular updates, bug fixes and reliable technical support are also crucial to ensure a smooth development experience.

Start by understanding how both your external and internal customers leverage your platforms and services to achieve their goals.

- Collaboratively map out customer journeys in a path-to-production workshop to encompass all key stakeholders—developers, infrastructure, security, lines of business and so on. Visually capture the process from initial idea through product launch and operation. While describing the process, identify pain points and blockers and their impact on the process.
- Gather feedback through surveys, focus groups and user observation. Capture frustrations and suggested improvements.
- Analyze platform telemetry and production monitoring data. Look for patterns of issues and outliers, especially as they relate to existing or past inefficiencies and reliability issues.

Establishing stronger empathy for a platform's users provides the necessary context to inform platform priorities. Which capabilities really move the needle for them? Where are the biggest opportunities to improve? These insights ensure that the platform's capabilities evolve in response to usage and demand rather than hypothetical needs.

You want to avoid platform engineering teams becoming disconnected from their customers' needs and forging a "build it and they will come" approach to their platform. This approach often results in work that isn't valuable to the customer and is never used.

Many of the platform teams that Tanzu Labs has worked with have groups of developers that meet at their company, either formally or informally. One way to develop stronger empathy and make connections with developers is by asking to be included in these channels to build a relationship. Platform team members can listen to the challenges that their users are facing, volunteer to present a view of the platform, and solicit feedback from stakeholders during these events.

Adopt a product mindset

A key element of platform engineering is shifting from a project-based orientation and instead focusing on the product-market fit of your platform. Traditionally, technical delivery is organized around projects with a defined start and end date. In contrast, platform engineering treats the platform as an ongoing product that requires continuous investment and improvement to meet the evolving needs of your developers.

A product team takes full ownership of the entire lifecycle, including defining the roadmap based on customer input, driving design and development, and supporting and enhancing the product long term. In addition, full ownership of the platform lifecycle fosters responsibility and commitment among the team members. When individuals know that their opinions and ideas are valued and that they have the authority to make decisions, they are more invested in the success of the effort. This increased level of commitment motivates team members to produce their best work and ensure that the product meets or exceeds expectations.

Empowering the team can result in greater continuity, providing a smoother workflow and minimizing knowledge gaps. Continuity translates to less time spent on educating new members and prevents common mistakes that can arise when transitioning between different contributors. It also enables faster iteration. With the authority to make decisions and implement changes, the team can quickly adapt and respond to customer feedback and market demands. This agile approach to development promotes faster problem-solving, improved efficiency, and the ability to pivot quickly in response to changing circumstances.

The empowered team works to incrementally build up the platform and drive a progressive evolution of platform capability to meet customer needs. New capabilities are added, outdated ones are removed and others are improved—similar to commercial software products.

Start with a minimum viable platform

When embarking on a new platform initiative, start with a "minimum viable platform" (MVP) that establishes just enough capability to demonstrate value and solicit feedback. When the MVP is in production, release small enhancements in a continuous delivery model rather than trying to launch an expansive platform all at once. This agile build-measure-learn approach reduces risk and ensures that the platform meets actual needs. Developers, who are the platform customers, provide input to help shape priorities and uncover hidden assumptions.

Many of the platform teams that Tanzu Labs works with have been successful with this model by identifying a dev team that will be a consumer of the new platform functionality and including them in the planning work. One key is framing the functionality as problem-solving outcomes for the development team customer and what can be accomplished instead of simply being a list of features. An example of such an outcome would be, "Developer's code is built, tested and deployed to the staging environment." Demonstrating the work in progress and having the dev team deploy earlier than expected—"uncomfortably early"—is an effective way for a platform team to understand if they are solving problems for the dev team, iterating faster, and moving more quickly to the goal of a mature platform.

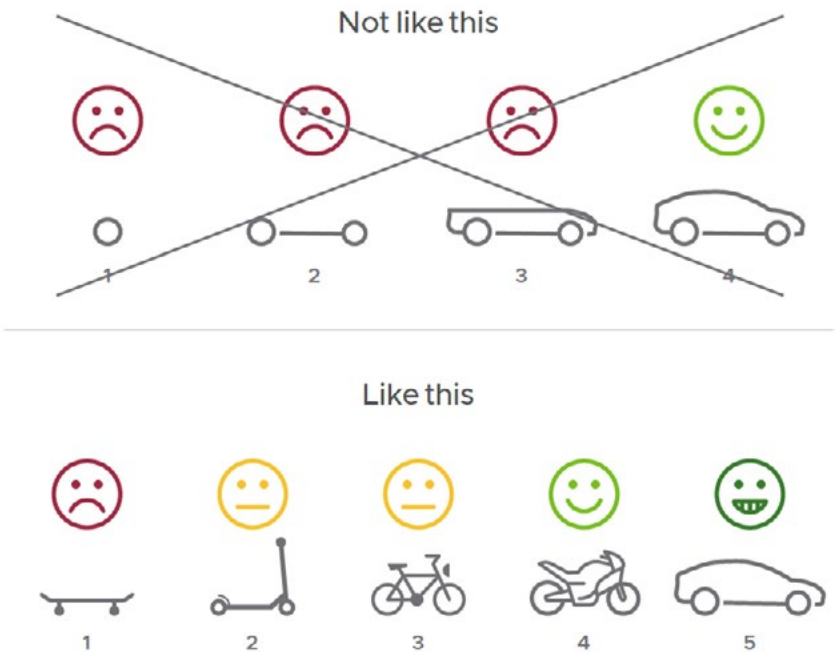

Figure 3: Iterative development of a platform allows for earlier, less feature-rich versions that can elicit development team feedback sooner instead of waiting for the platform to be done.

Define the platform's purpose and success metrics

Setting clear goals by understanding which problems the platform is fundamentally trying to solve aligns all efforts toward achieving the same objective and facilitates decision-making for the product team. Having objective measures of impact focused on outcomes rather than outputs helps steer the platform in the right direction. These metrics provide the feedback loop to know if investments are paying off.

For example, a platform's purpose could be to "democratize AI by providing self-service access to common models." When the product team understands the purpose, they can drive prioritization of capabilities and inform design choices.

It's also critical to define quantitative metrics and KPIs to measure the platform's success. These metrics tie back to the purpose and customer needs. The platform teams that Tanzu Labs has worked with have greater alignment from leadership when the team has a clear purpose and measurable KPIs. These teams are able to show the impact of their platform and justify its existence—and even make a compelling case for additional resources.

Start with a low-risk application

Choosing the right application to onboard onto the platform is a strategic decision. Ideally, the application should not be mission-critical but have existing users, some business value and a committed development team ready to move it to the platform. While the temptation might be to choose a high-stakes application to showcase the new platform's capabilities, it is prudent to start with a less-critical application to reduce risk and provide valuable learning experiences for the platform team.

The application can be any software or service that is used to perform a specific task or deliver a certain functionality within a larger system or infrastructure. It can encompass a wide range of software applications, tools or components that support various business processes or technical operations.

Most platform teams that Tanzu Labs has worked with have found success in beginning with an app that is used internally so that complexities like security and compliance are not as critical. For example, start with a tool that the team can use for internal retros. This "eat your own dog food" approach not only begins to exercise the features of the platform but can help develop empathy for the customer experience. It is also a great opportunity to begin introducing initial SLOs for the platform and develop observability around platform functionality, efficiency and reliability.

Map the path to production

Before placing more critical apps in production, it is crucial to have risk mitigation strategies for incidents, security, compliance and disaster recovery. Additionally, developers need to know how to consistently deploy application workloads onto the production platform. In a more user-friendly platform, an automated route to production involves deployment configurations that are

stored in code, known as infrastructure as code. The path to production provides developers and platform engineers with observability of what is happening as applications are deployed or updated on the platform. The most mature platform engineering teams leverage and refine secure software supply chains that developers can access via the platform.

These opinionated, well-defined, task-specific and supported paths for creating software are often referred to as "golden paths." A golden path enables an organization to build better software and deliver it to production faster with higher quality and greater control by helping engineers navigate complexity.

Tanzu Labs helps organizations achieve high-impact outcomes on their platform journey

Focus developers on code instead of infrastructure

37%
increase in developer productivity

Shorten the path to production

82%
increase in software to production

Strengthen your security posture

92%
less time to patch operating systems

Getting started on your platform journey with Tanzu Labs

Tanzu Labs helps organizations master the skills needed to build, run and manage an internal developer platform through hands-on pairing with technical experts. Our approach centers on using a product mindset and iterative approach to build the right features for your developers.

To get started on your own journey, begin to incorporate these concepts in your platform engineering practice.

- **Develop a roadmap** – Outline your platform engineering roadmap based on the insights gathered from customer journeys, feedback and analysis of platform telemetry and production monitoring data. Prioritize the features and capabilities that align with customer needs. Break down the roadmap into small, achievable milestones that can be executed in an agile manner.
- **Form a balanced team** – Assemble a platform engineering team consisting of a product manager and platform engineers with a combination of infrastructure and software engineering skills. Ensure that the team includes roles focused on design to prioritize user experience and usability in the development of the platform.
- **Implement monitoring and metrics** – Establish a system for monitoring and measuring the health and performance of your platform. Define and track metrics important to your developers, customers and end users, like those related to development frequency, lead time for changes, change failure rate, and time to restore service. Monitor system behavior, response latency, error rates, resource utilization and other performance indicators to identify areas for improvement and optimize the platform.
- **Enable self-service capabilities** – Implement self-service options that empower developers to work autonomously and at their own pace. Provide documentation, guides, tutorials, APIs, SDKs and easy access to platform features through portals or dashboards. Focus on enhancing the developer experience and providing tools and services that enable them to be more productive and innovative.

- **Continuously iterate and improve** – Adopt an iterative approach to platform development and improvement. Regularly release small enhancements and updates to the platform based on customer feedback and evolving needs. Gather feedback from developers and incorporate it into future iterations. Use an agile mindset to continuously learn, iterate and improve your platform to meet the changing demands of your customers.

Getting started with platform engineering requires a customer-centric approach, a shift to a product mindset, and agile, iterative development. Starting with a minimum viable platform and soliciting customer feedback allows for early validation and reduces risk. Additionally, choosing the right application to onboard onto the platform is crucial, balancing between low risk and the opportunity to show value by exercising platform capabilities. By following these practical steps, you can effectively embark on your platform engineering journey and build successful, innovative platforms that meet customer needs.

Item No: FY24-7514-BCM-PLATFORM-ENGINEERING-WP-USLET-20240118 1/24