WHITE PAPER–MAY 2021

Why You Should Treat Your Platform as a Product

By Zac Bergquist and Joe Fitzgerald

Table of contents

Why you should treat your platform as a product

Your platform is not an off-the-shelf piece of software; it's an evolving set of reusable services, integrated with your existing systems to create valuable outcomes for your "business. The platform capabilities change in response to the needs of its users—your app developers—among whom it's a recognizable internal brand.

In other words, you should treat your platform as a product.

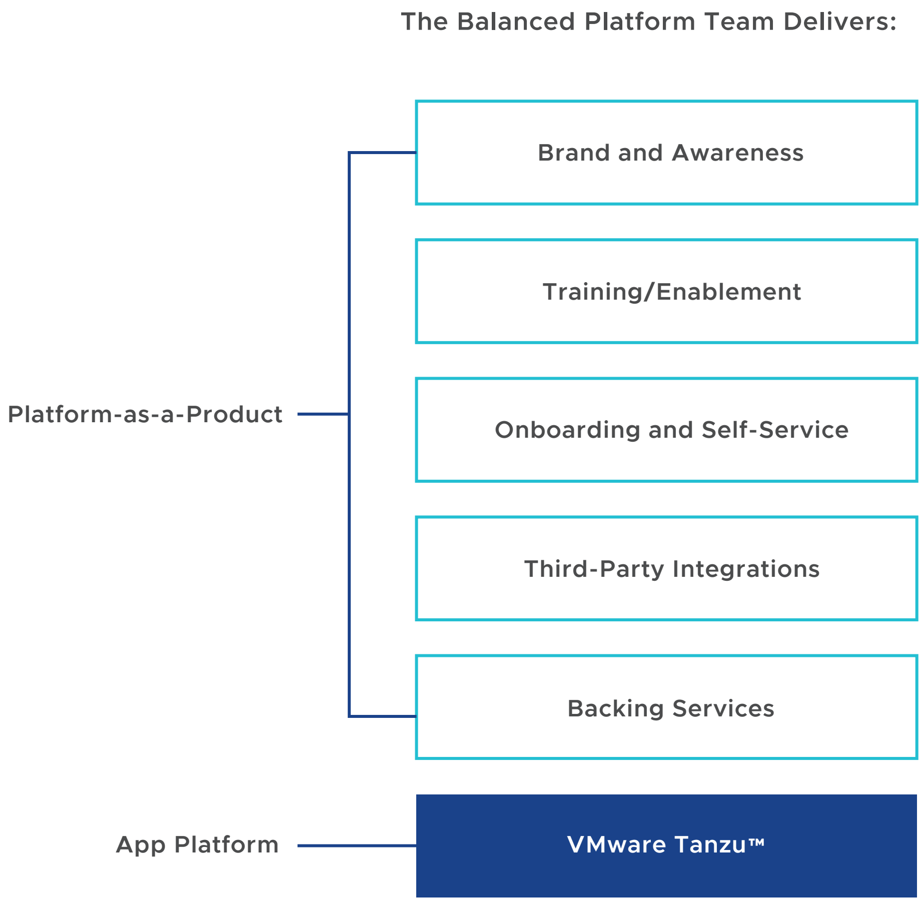

FIGURE 1: Building a platform product.

The VMware Tanzu Labs platform-as-product approach

The VMware Tanzu Labs platform-as-product approach combines product management, user-centered design (UCD), agile, extreme programming (XP), and site reliability engineering (SRE) practices. A dedicated, *balanced* platform team uses these practices to both build and run the platform product.

Here's what the four different areas in the Tanzu Labs approach look like in practice:

- **Lean product management:** Determines what the platform team will build.
- **UCD:** Helps to inform both the what and the why using feedback to de-risk decisions and enable continuous improvement.
- **Agile software development with XP:** Provides a clear methodology for how the team should build the platform.
- **SRE**: Defines how the team runs the platform product, including how to balance the conflicting goals of feature delivery, security, and availability.

This approach allows an app development team to have an idea in the morning that can run in production by the afternoon. Treating your platform as a product maximizes the value of the platform and minimizes delivery time, risk, and waste, while retaining the ability to detect and remediate security issues early in the development lifecycle. Done well, it will change the way your organization builds and runs software.

Product management

Lean startup practices make it possible for your platform team to continuously respond to the changing needs of its customers—your app developers—by maximizing customer value and minimizing waste.

Focusing on user needs (UX)

How do you make a platform that app developers use and appreciate so much that they evangelize it to other members of their organization?

The answer is to focus on the user experience. We know the challenges that can impact an app developer's ability to successfully leverage the platform and want to share our insight. We want to see organizations benefit from their investment in building a platform.

One way to achieve that is to grow your platform team's skills with UCD practices. Using these practices will ensure that the team builds a usable platform that meets user needs through consistent user research, usability testing, and subsequent iteration.

Agile development with XP

Your platform team focuses on the incremental, iterative, and flexible delivery of platform software. The team uses XP practices to build high-quality, well-tested platform code that evolves in line with your app developers' requirements. XP anticipates changes in customer requirements, avoids building features until they are needed, uses pair programming and automated testing practices, and requires continuous communication between all stakeholders.

Site reliability engineering

SRE treats operations as an engineering problem by using software to manage and maintain systems. Agile software development practices are applied to add new features to your platform and automate away toil and waste. In SRE, toil is a manual, repetitive, and automatable work tied to running a production service.

SRE balances the traditionally conflicting goals of velocity and reliability. Your platform team and your application team will determine an appropriate level of reliability for the platform and each application, while maximizing feature velocity. SRE regards one hundred percent reliability as the wrong target for the platform or for applications running on a platform. Instead, the platform team and app development team together define an error budget, which quantifies how reliable the product needs to be over a particular time period, as experienced by the end user. The app development team is responsible for planning how much change they can safely introduce—and on what schedule—to stay within their error budget.

Prerequisites for adopting a product mindset

Tanzu Labs' experience with hundreds of clients has shown that adopting a product mindset is most successful when the adoption is recognized as transformational and is supported by more than the platform team.

Part of the transformation is a shared desire to change for the better and challenge the status quo.

In this new way of working, everyone must work together in a blameless culture that accepts failures as learning opportunities.

There must be a level of patience and trust in the process and people.

The push to build a platform

Your organization should have a strong motivation to build a platform product. In some cases, your IT organization knows that it wastes large amounts of effort building platform-like capabilities and wants to reduce costs. In other cases, a business unit in which multiple app development teams have already been forced to solve their own platform challenges wants to increase velocity.

Reliable local infrastructure

Don't introduce changes in your infrastructure-as-a-service (IaaS) layer at the same time as embarking on the platform journey. You need a stable IaaS capability during the platform creation period.

Establish a production platform capability that's located close to—or ideally in the same physical location as—the existing applications and data where the platform needs to function. Otherwise, latency issues may be blamed on the platform. If you must have distance between the applications and data, establish a baseline with monitoring that you use to disambiguate changes to the platform or application.

Shared goals

Many stakeholders are involved in building a platform product, including app development teams, business application owners, change management, networking, production operations, release management, security, virtualization, and their end users. Defining shared goals ensures that stakeholders work together rather than against each other. Shared goals also change the conversation from, "Here's why this won't work" to "How can we make this work?"

Tanzu Labs uses an *inception* exercise to align teams on the importance of the platform and how it fits into the bigger picture for your organization. Define quarterly objectives and key results (OKRs) for the platform (e.g., deploying the first app to production during the launch phase). Variable compensation schemes, rewards, and recognition can also be linked to OKRs.

Platform champion

A platform champion has the motivation and the political capital to protect the platform team as they embark on the platform journey. The champion is often at—or close to—the CxO level and has a track record of internal entrepreneurship or organizational transformation. This person understands and can articulate the value of a platform product and evangelizes its use and growth within your company.

Dedicated platform team

Your platform team builds and runs your platform product. The team must have the authority to change the production process as well as the platform itself. A platform team consists of at least three full-time employees, satisfying the following functions:

- Product management focuses on building the thesis for the platform product, testing that thesis with the target customer (internal application developers), and iterating on the features provided to achieve the desired customer and business outcomes.
- Engineering works through a feature backlog to implement new platform features on a weekly basis, and manages the delivery of those features. This involves balancing the rate of change introduced with the desired reliability objectives.

How to build your platform product

Your platform team will start small and gradually extend the platform's capabilities. We see three successive phases during which your platform increases in maturity, taking you from creating a platform team to running thousands of apps in production on the new platform:

- Launch the platform capability.
- Extend the platform product.
- Use the platform at scale.

Launch the platform capability

The main objective in this phase is to launch a minimum viable product (MVP) version of the platform and deploy a single production app on that MVP. This MVP of the platform is intentionally just enough to get the first app to production, not an attempt to bring parity with existing platforms.

Choose your platform team

The platform team consists of a product manager and at least two platform engineers. The first members of the team will be responsible for staffing and training future team members so choose them wisely.

Product manager

IT organizations often do not have agile product management experience, so you may need to recruit an external-facing product manager from a business unit. We found that successful product managers fulfill several of the following personas:

- **Alchemist:** Takes disparate requirements and distills them down to a succinct product vision
- **Visionary:** Doesn't allow themselves to be constrained by legacy thinking and processes
- **Influencer:** Has strong relationships with business partners, application teams, and IT
- **Minimalist:** Understands the value of shipping early and often
- **Lean champion:** Relentlessly pursues the elimination of waste

Platform engineers

The platform team requires a combination of infrastructure and software engineering skills. Because managing change in a sustainable way requires that you treat operations as a software problem, your platform engineers must code. In many cases, training will be needed to help platform engineers recruited from software teams to deepen their infrastructure knowledge, and for platform operators to build their software engineering skills.

The launch platform team should include at least two of the following personas. As the team matures it should contain all three personas:

- **Infrastructure architect who codes:** This person is very experienced in IaaS primitives (compute, storage, network), usually has experience in production operations, and is able to automate away manual, repetitive activities.
- **Natural automator:** You'll often find this person doing continuous integration/continuous deployment work, automating your current release management processes, or doing system automation using tools such as Chef, Puppet, Salt, and Ansible.
- **Curious software engineer:** You'll find this person in an application product team that previously solved its own platform needs by automating infrastructure.

Pick a first production app

- Choose a single production application that all stakeholders agree will go into production on the new platform. The ideal launch app is not mission critical but has existing production users and is maintained by an app development team that can assist in moving the app to the platform product.
- Do not fall into the trap of choosing a mission-critical app to prove to detractors that "important" apps can immediately go into production on the new platform. Mission-critical apps can follow quickly once the platform team gains expertise by running the first real workloads in production.

Define a platform MVP with a brand

The platform product manager defines an MVP platform product with a self-service landing page and a product brand. An MVP is the simplest and most easily built version of the platform that can help validate or invalidate a product management hypothesis about user behavior or features.

Tanzu Labs customers who have successfully delivered platform products in large organizations define a recognizable, internal brand for the platform. A brand creates a sense of ownership and pride in the platform team and helps to build a customer base by raising awareness among app developers. Examples of a branded platform include *U.S. Air Force's All Domain Common Platform* and *Liberty Mutual's CloudForge*.

Define a new path to production

An effective platform allows application teams to release software faster, which stresses systems and processes that assume centralized control and throttle change in an attempt to achieve reliability. Do not shoehorn your platform into your existing path to production, or you'll make it easier for apps and features to pile up waiting to get to production on the platform.

Deploying the first production app on the platform will prompt important conversations among stakeholders about the processes in your current path to production and production operations (e.g., change management, compliance, release management, security, and testing). Customer *journey maps* and *value stream maps* will be created to guide these conversations.

The result should be a new and deliberately naïve path to production that emphasizes velocity and is only suitable for a small proportion of your app portfolio. As you extend the platform, the path to production is enhanced to meet the production needs of a larger proportion of your app portfolio, including things like artifact management, automated patching, and vulnerability scanning.

Create customer journey maps

A customer journey map is a compact visualization of an end-to-end customer experience. The platform product manager will generate journey maps of your app developers' current experience of going to production to understand where there are opportunities for improvement.

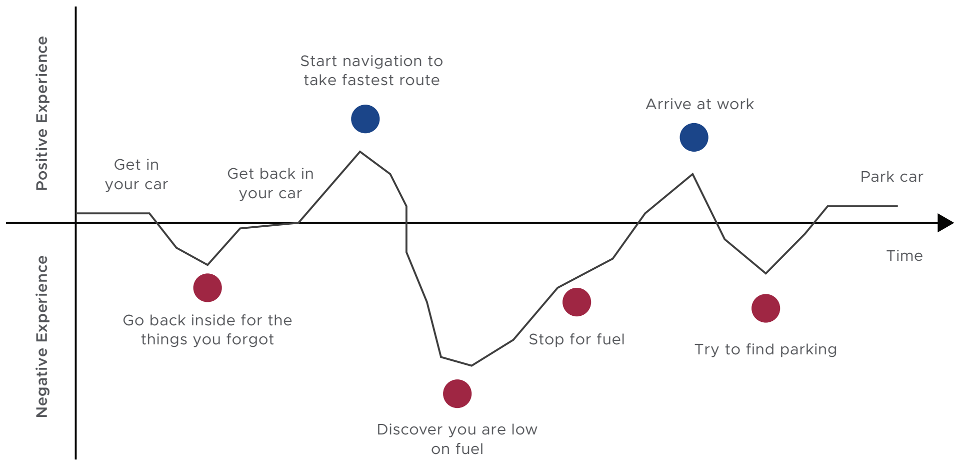

FIGURE 2: A drive-to-work journey map.

The product manager will also talk to teams currently running apps in production and generate journey maps of their experience of responding to incidents and performing root cause analysis. These maps will continue to help identify improvement opportunities for how teams run apps in production while increasing the velocity of change.

Compare the path to production to the requirements of the first production app. Since the app is not mission critical, it may not need to adhere to requirements for compliance or security demanded by other apps. Does it require all steps in the process? What can you change or leave out? This exercise helps define the essential steps in the new, naïve path to production. This set of essential steps will grow as you onboard new applications with additional requirements.

Define value stream maps

A *value stream map* is a visualization of the sequence of activities an organization undertakes to deliver on a customer request. In the case of the path to production, that request is getting a feature or app into production. The time between a particular trigger and the value delivery prompted by that trigger is the lead time. Shortening the lead time by removing steps and reducing waste in the activities used to deliver value is an objective of value stream mapping. Although customer journey maps are qualitative, value stream maps are quantitative.

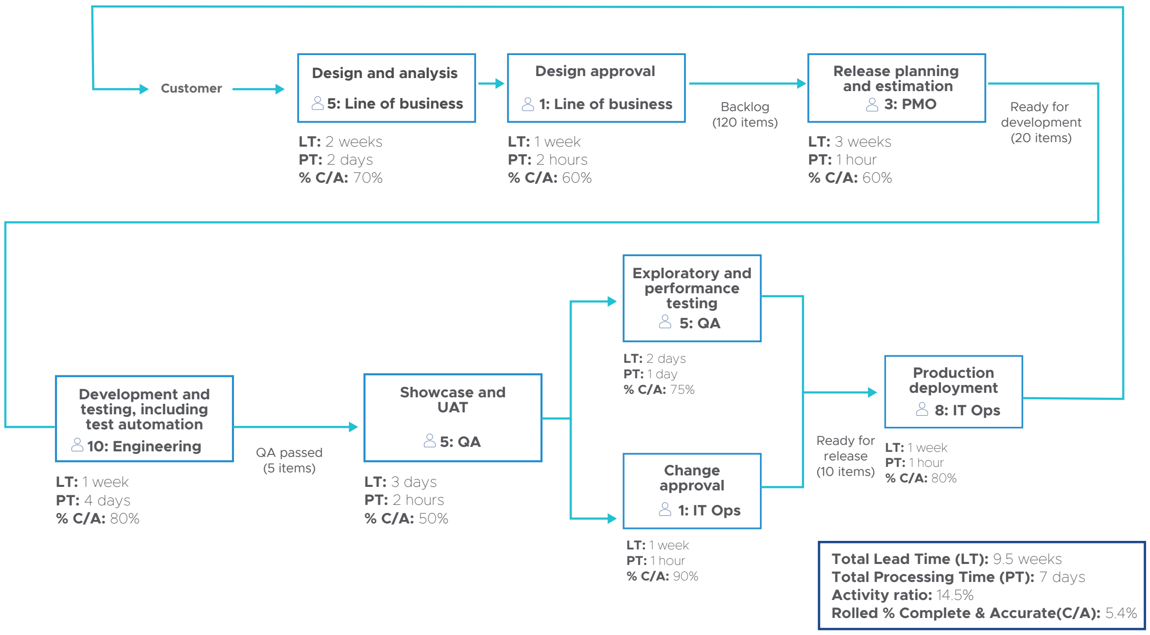

FIGURE 3: Software design and delivery value stream example.

The platform team will continuously evaluate the path to production and perform value stream mapping to find opportunities to remove waste and increase the rate of automation.

Deploy to production

Launch the platform MVP and deploy the first app to production using the new, naïve path to production.

Extend the platform product

In this phase, you'll extend the features of the platform so that a larger proportion of your app portfolio can go to production while meeting security, compliance, auditing, financial reporting, and other enterprise requirements.

Iterate on features and reliability

The platform product manager will gather information from application and IT teams to determine what platform features are required to run additional apps from your portfolio in production on the new platform. Since the platform should create value for your business, the exact features will vary, but here are a few typical extensions:

- Additional *service* brokers to permit self-service access to application dependencies
- Additional *buildpacks* to allow new applications to run on the platform or to reduce the time required to modify an app to run on the platform
- Modifications to the onboarding portal to add more self-service features
- Additional backing services such as MySQL, RabbitMQ, or Spring Cloud Services
- Solutions for cross-cutting concerns like security, logging, metrics, and load balancing
- Integrations with other enterprise systems
- Multiple variants of the platform that have different service-level objectives (SLOs) and characteristics
- Additional deployments of the platform in new regions and/or on new infrastructure

Iterate on the path to production

The platform team will iterate on the path to production to meet nonfunctional and compliance objectives for security, auditing, financial reporting, or other enterprise requirements, and make it possible to deploy a larger proportion of your app portfolio on the platform.

The platform team will use the *5 Whys* or the *infinite hows* technique to determine the process objectives in your original path to production to achieve the same outcomes with less waste and more automation. The exercise should result in journey maps with improved sentiment (i.e., the emotional state of your app developers at each stage of the journey) and updated value stream maps with less waste.

Use the platform at scale

During this phase, you'll scale the platform and the platform team to serve thousands of applications while meeting all mandatory security, compliance, auditing, financial reporting, or other enterprise requirements.

Increase self-service

Self-service helps the platform team to scale sublinearly with the workloads running on the platform. At scale, app development teams are able to self-service their way to production, while individual developers are able to use the platform to learn and experiment with preproduction.

Self-service changes the communication pattern between the platform team and the app developers they serve. Instead of waiting for developers to send a request to IT, the platform team tracks self-service activities. The platform team is also proactive about reaching out to new and existing platform tenants whenever they see opportunities to educate, plan for the tenant's needs, or identify missing features in the platform's capability.

Scale the platform team

Two platform engineers are only sufficient while you launch the platform and have a small number of platform tenants. If your platform engineers spend more than 50% of their time on toil, or if you want to provide enhanced support for the platform such as 24/7 on-call support, then it's time to add more engineers.

The platform team will scale sublinearly with platform workloads and generally max out at 9–10 people. A team of this size is capable of building and running a platform that can service more than 3,000 apps.

Tanzu Labs recommends creating a sustainable work environment where each engineer works a maximum of 40 hours a week and spends most of their day working from a prioritized backlog. Each engineer should only be on call for a maximum of one week per month, ideally with a primary and secondary engineer on call during each rotation. Providing 24/7 support coverage in that environment requires 6–8 engineers per site.

Tanzu Labs has found that spreading those 6–8 platform engineers across multiple time zones in an effort to reduce "out of hours" work is the wrong approach. Maximizing time zone overlap between engineers is more important to ensure that context about platform changes is shared. Since platform incidents are almost always the result of planned changes, you can effectively schedule when incidents are likely to occur.

Add a developer enablement team

Adopting a new set of platform capabilities and transforming applications to run on the platform can be challenging for app development teams. A dedicated developer enablement team helps app developers improve their craft and establishes bidirectional communication between the platform team and lines of business. A developer enablement team can help transform your existing app portfolio and launch new products faster by advising other teams how to meet business requirements using the features of the platform product.

How to know if you're succeeding?

Tanzu Labs customers often ask how they can tell if they're building an effective platform product. The following signals show that you're on the right path:

- Features are delivered to end users faster.
- Developers ask for more platform capabilities.
- Deploying to production requires no special reviews, checks, balances, or ceremony. *Simply clicking "accept"* triggers automated deployment, audit trails, and approvals.
- Applications and their dependencies can be patched immediately in response to new common vulnerabilities and exposures (CVEs).
- Application outages are quickly traced to a root cause without a fire drill.
- Events that previously generated downtime are autohealed without paging anybody.
- Alerting "noise" is reduced, alerts become more actionable, and humans are only paged when there's a need for immediate action.
- The platform team reaches a level of platform mastery where they provide new insights to their peers or to Tanzu Labs.

Tanzu Labs does not have all the answers. We learn as much from our customers as we teach them. The successful platform journeys of Tanzu Labs customers, such as *Comcast* and *The Home Depot*, have informed much of the advice discussed in this white paper. If you have questions or insights about building a platform product you'd like to share, please contact us at *tanzu.vmware.com/contact*.