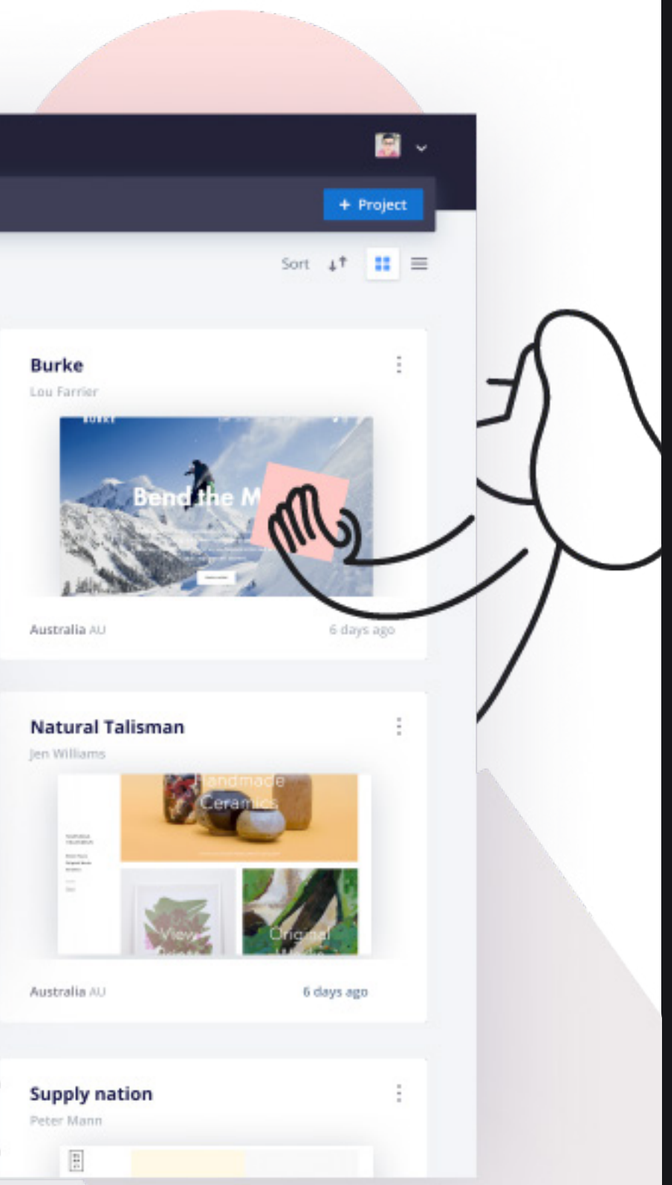


Solving the challenge of website fleet management

An alternative to Kubernetes

platform.sh

The challenges of website fleet management

Organizations face a major challenge in ensuring their online services, applications, and websites are deployed and managed effectively and economically. In recent years, with interest in infrastructure technologies such as Kubernetes and Docker surging, container orchestration technologies have emerged as a core technology. While many organizations have adopted some level of containerization within development, adoption for production remains relatively low, with organizations citing as barriers a lack of expertise and increasing security challenges.

Today, many organizations choose to rely on Platform-as-a-Service (PaaS) offerings that provide a complete platform for development and deployment, including container orchestration. They also provide the complete infrastructure needed for production, including security infrastructure, load balancing, and high availability—all fully supported by the PaaS provider.

PaaS offerings vary significantly. This paper will cover some of the features and technologies that Paas provider Platform.sh incorporates specifically. Currently, Platform.sh has more than 5,000 business customers (2,000-plus enterprise); 65,000 developers rely on the Platform.sh PaaS, with a large proportion doing so for production.

This paper will explain the benefits and technologies of containers, container orchestration, and PaaS, alongside what each technology does and doesn't provide. We'll cover some of the functionality necessary for organizations to leverage containers in production.

These users range in size from small and midsize enterprises up to globally distributed large enterprises and come from a range of sectors, including finance, retail, life sciences, and higher education. A significant proportion of our larger enterprise customers have migrated from Kubernetes management systems. Customers have been successful in implementing 12 Factor App methodologies, which have emerged as a dominant standard for web apps or Software-as-a-Service.

The ABCs of containers

Containers are built around core Linux kernel (LXC Containers and cgroups) and OS features that allow complete native isolation of an application's view of the operating environment, including process trees, networking, user IDs, and mounted file systems. In other words, the Linux kernel itself provides a way of securely virtualizing an application without the need to spin-up a virtual machine.

Containers offer developers a solution to those in-application/"it works on my machine" issues. If an application is deployed within a container, its view of the world is always the same. The segregation/sandboxing means another application can't overwrite the memory being used, and an application runs the same independent of the underlying hardware and infrastructure used, perfect for the cloud. Containers can be useful for individual developers and managed/scripted on a small scale by numerous mechanisms. Proprietary container technologies, such as Docker or rkt, have become popular to help associate applications with containers and services.

Using Kubernetes to orchestrate containers

Kubernetes (often referred to as K8s) is one of the most popular frameworks for container orchestration.

Container orchestration involves many things:

+ Managing the storage resources a container can use (where the storage physically is; whether that storage is AWS, Azure, or on-premises)
+ Deploying containers
+ Monitoring applications

An application and its services run in a container. Kubernetes can be used to build a platform that then allows containers to be operated, deployed, moved, and scaled to maintain the desired state of the application and end service. The application itself runs on a distributed system of cloud and physical servers, using orchestration to ensure the resources are available and used optimally for the whole system, balancing and adjusting according to the needs of the applications.

Theoretically, this means with the right triggers and monitoring a web application can respond to the demands on it. If demand surges, additional copies of a container can be spun up in seconds in geographies nearer to the demand. This, however, relies on building the infrastructure and tools within Kubernetes or leveraging and integrating the right third-party tools and functionality.

How to build a platform with Kubernetes

To leverage container orchestration in production you need to build a platform, typically with functionality to ensure the platform provides:

+ High availability
+ Security updates
+ Environment cloning for developers to test their changes
+ Backups
+ Automated generation of staging clusters
+ Web application firewalls
+ Storage allocation and purchasing
+ Content delivery network
+ Monitoring and feedback

Kubernetes itself is not a platform; it's a framework within which you can build one. The types of products and services that are added to Kubernetes give an idea of what's needed: storage/host management (AWS, Ceph, Terraform); container templating/service registration (Docker, Rancher, Helm, JFrog); logging/metrics (Grafana, fluentd); code building and publishing (Jenkins, GitHub); load balancing (Netscaler/Citrix ADC).

Technology layers needed to manage fleets

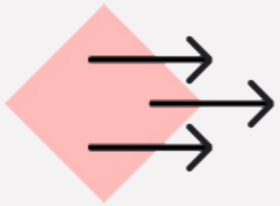
- Provisioning
- Routing + Edge Security
- Continuous Integration
- Application Code
- Application Data
- Application Patching
- App Runtimes
- Data Services
- Continuous Deployment
- Container Orchestration
- Infrastructure

To build a platform around Kubernetes, an organization needs not only to evaluate, license, and support numerous tools and technologies, but also to take it upon themselves to maintain, license, and support those components and their interactions.

Beyond maintaining and patching an infrastructure around Kubernetes, a production deployment needs significant development, tooling, and processes to integrate technologies that manage containers and their contents. Containers make it easier for developers to build applications faster. However, much of that software can contain vulnerabilities when developers end up relying on outdated components that haven't been updated/patched or are unsupported. Later we'll cover how these challenges can be overcome by leveraging declarative architecture features within PaaS.

Business outcomes with Platform.sh

Standardize across different tech stacks



Accelerate time to market

Meet business demands (say yes more)

Focus on your core business

PaaS lets you focus on your apps

Platform.sh supports development stacks that include Go, Java, Node.js, PHP, Python, Ruby, Drupal, eZ, Laravel, Magento, Symfony, Typo3, WordPress, and others. We power fleets consisting of thousands of websites and applications for brands and organizations such as:

A Platform-as-a-Service provider takes on the overhead of building and supporting the platform and all the components. Developers and architects are then free to focus on developing their applications.

Orange

Platform.sh delivers a web application fleet management solution for more than 6,000 Orange Business Services SMB customers. Orange Cloud for Business gives customers the power to build and run their websites with the CMS they choose, whether it be Drupal, WordPress, Joomla, or PrestaShop.

The British Council

In just three months, Platform.sh helped The British Council migrate its in-house system to Platform.sh. The system now supports 1,000 staff across 120 multilingual sites with 115 million users in 110 countries.

The University of Missouri

Working with Platform.sh, the University of Missouri consolidated hundreds of websites and 13 different content management systems.

Platform.sh helps organizations to scale and secure their entire website fleets

Regardless of the stacks our customers use, we focus on measurable business value features and practical use case scenarios. We enable customers deploying website fleets to:

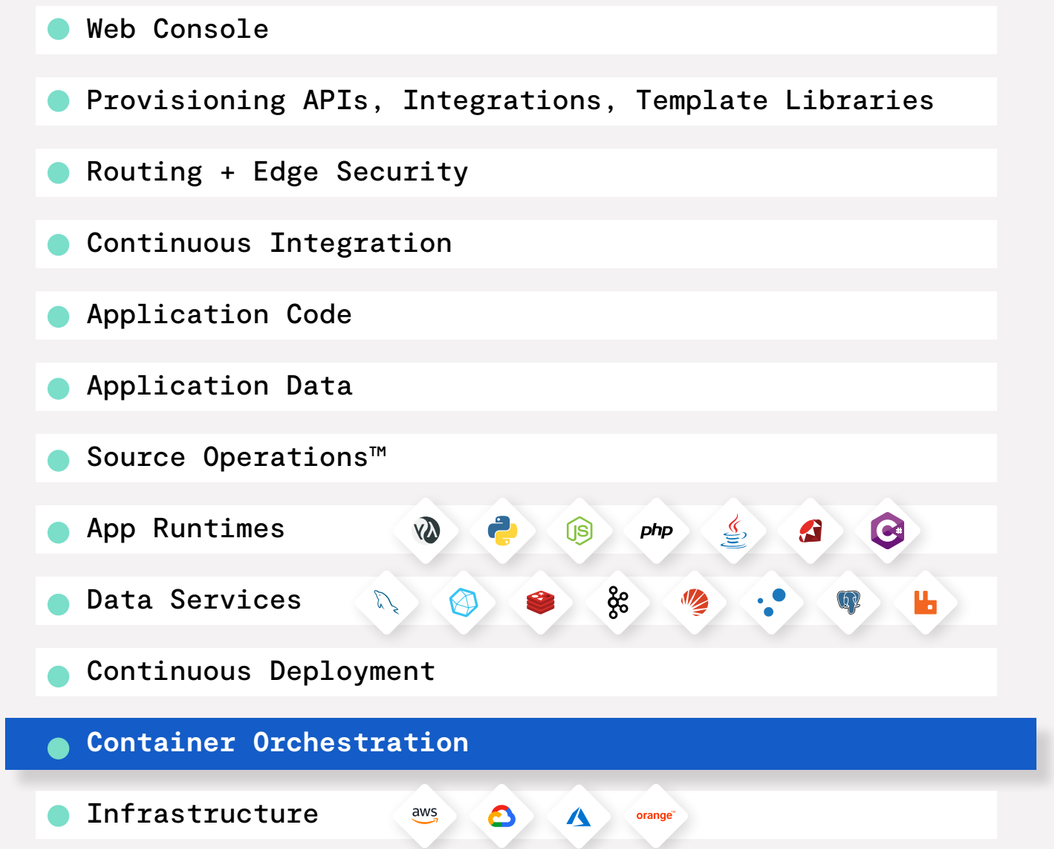
+ Patch all their sites centrally
+ Secure their data with fully managed database services
+ Ensure governance over technology and processes
+ Provide high availability SLAs (up to 99.99%)
+ Rely on 24x7 global support

Focusing on the specific use cases and productivity features that a platform needs to provide, developers can often help an organization expose the security and reliability issues of in-house Kubernetes management.

If your teams spend too much time evaluating, discussing, and implementing Kubernetes architectural components (such as Pods, Labels, Replica Sets, and Config Maps) or debating whether and how to combine Rancher with Helm and RabbitMQ, then a PaaS could suit your organization.

Guides to demystifying Kubernetes or containers offer insight into the fundamental soundness of those technologies. But designing, documenting, and maintaining a working system is a very different matter. Managed Kubernetes services offer a wide range of individual components, but it's up to the individual organization to figure out how to achieve high availability backups and how to integrate load-balancing or security gateways.

The benefits of deploying with Platform.sh



This overview gives an idea of what Platform.sh provides to support a deployment. Please note that container orchestration is one small piece of the provision.

Key points:

+ Platform.sh leverages standard native containers and provides orchestration.
+ A vast amount of functionality is provided beyond container orchestration.
+ Key tools and applications are integrated into the platform, with the licensing and patching responsibilities undertaken by the platform.
+ Platform.sh provides a choice of infrastructure, sourced and priced at bulk (e.g., AWS, Azure).

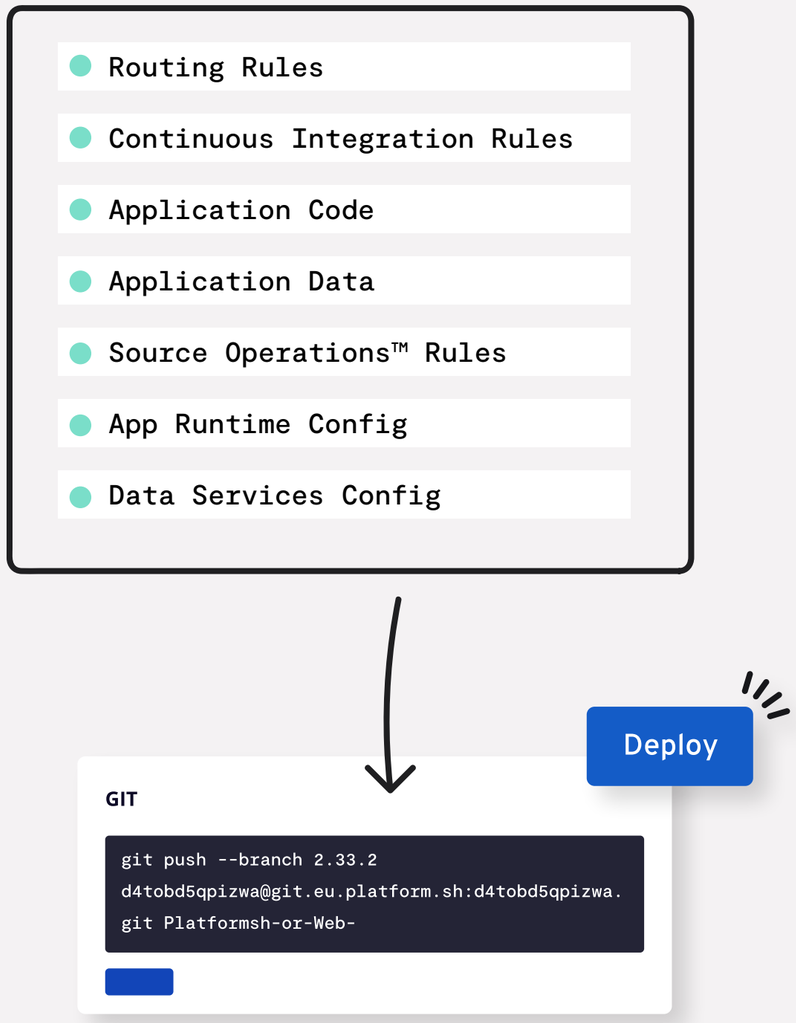
Achieving platform integrity

Containers make it easier for developers to build applications faster, but that software can contain vulnerabilities when developers end up relying on outdated components. Docker—a popular container templating technology—is based on images, and there are tens of thousands of container images available. An image is the result of a script; the definition of the contents of an image is usually opaque.

While Docker makes it easy to package up something yourself to run, unless you're building all your containers from source, they become difficult to maintain. Furthermore, controlling the use of third-party images is problematic. How can you trust they don't contain vulnerable libraries and executables or are being maintained or supported properly? How do you limit and document your staff's reliance on third-party organizations?

A Kubernetes implementation will usually require the development of a reproducible build chain. Without this, once you deploy your container, there's nothing to prevent it from changing; it's just a program and can write to disk and even update its own code. So you need to ensure you have a repeatable build process with deterministic results, coupled with an immutable read-only infrastructure, to make sure once a container gets into production, nothing can ever change it while it's there.

You already know how to use Platform.sh



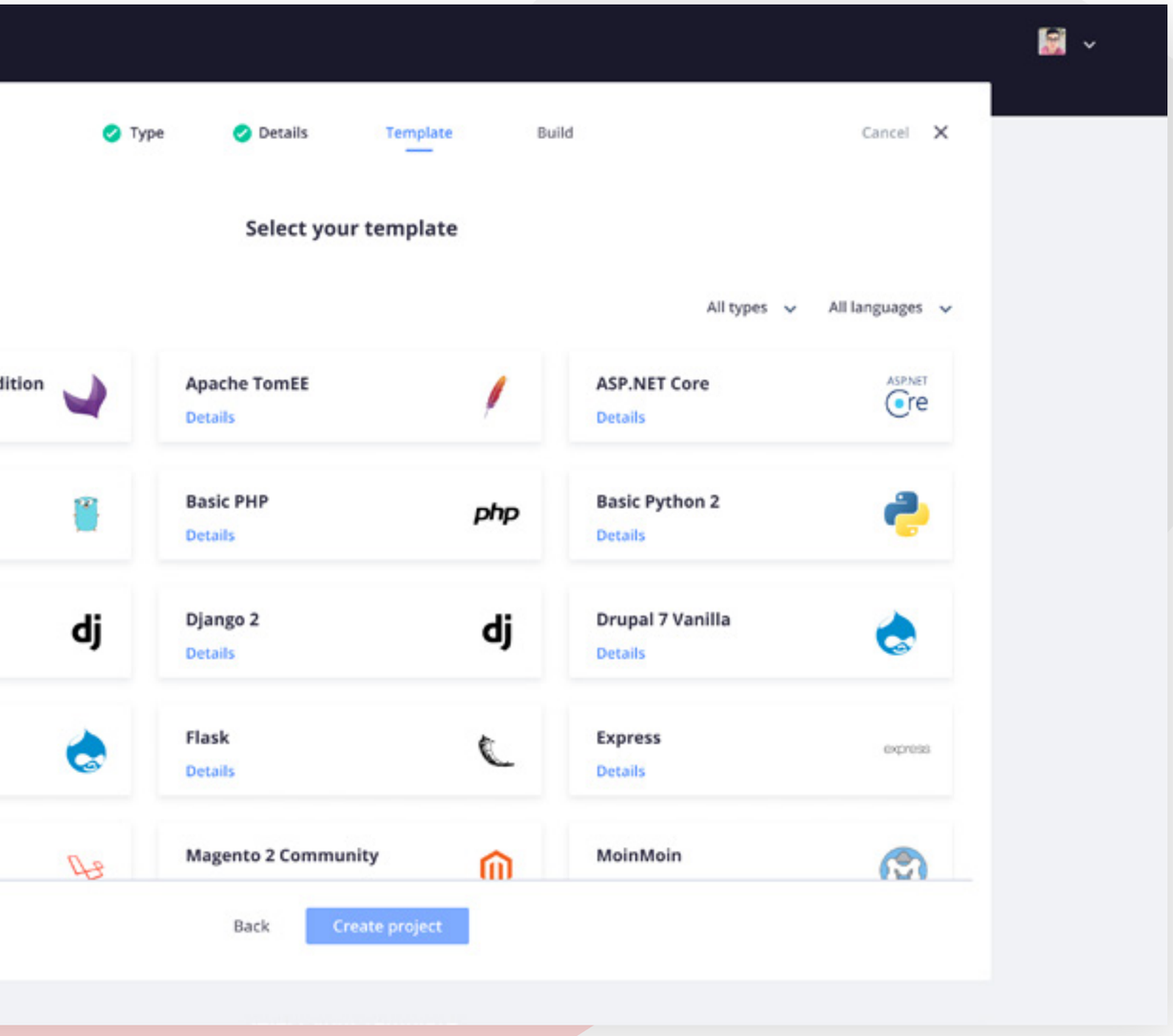
Platform.sh has solved these challenges by leveraging Declarative Infrastructure built around industry standard code control technologies. This means that complete websites and services can be managed in Git. This also means that developers already know how to use Platform.sh. It works how they do because they know Git and GitHub. Platform.sh also gives devs a nearly instant clone of their production apps for every change (data, services, CI/CD).

Keep things simple with Source Operations™

Simplifying site maintenance is important for single sites. It's even more critical if you're running thousands of sites, as some of our clients do. We have built the Source Operations technology to allow developers to define CRUD operations, automating code, content, and dependency updates. The main use case is for the source code to update itself, but it can also be used to update application dependencies such as a JSON stack.

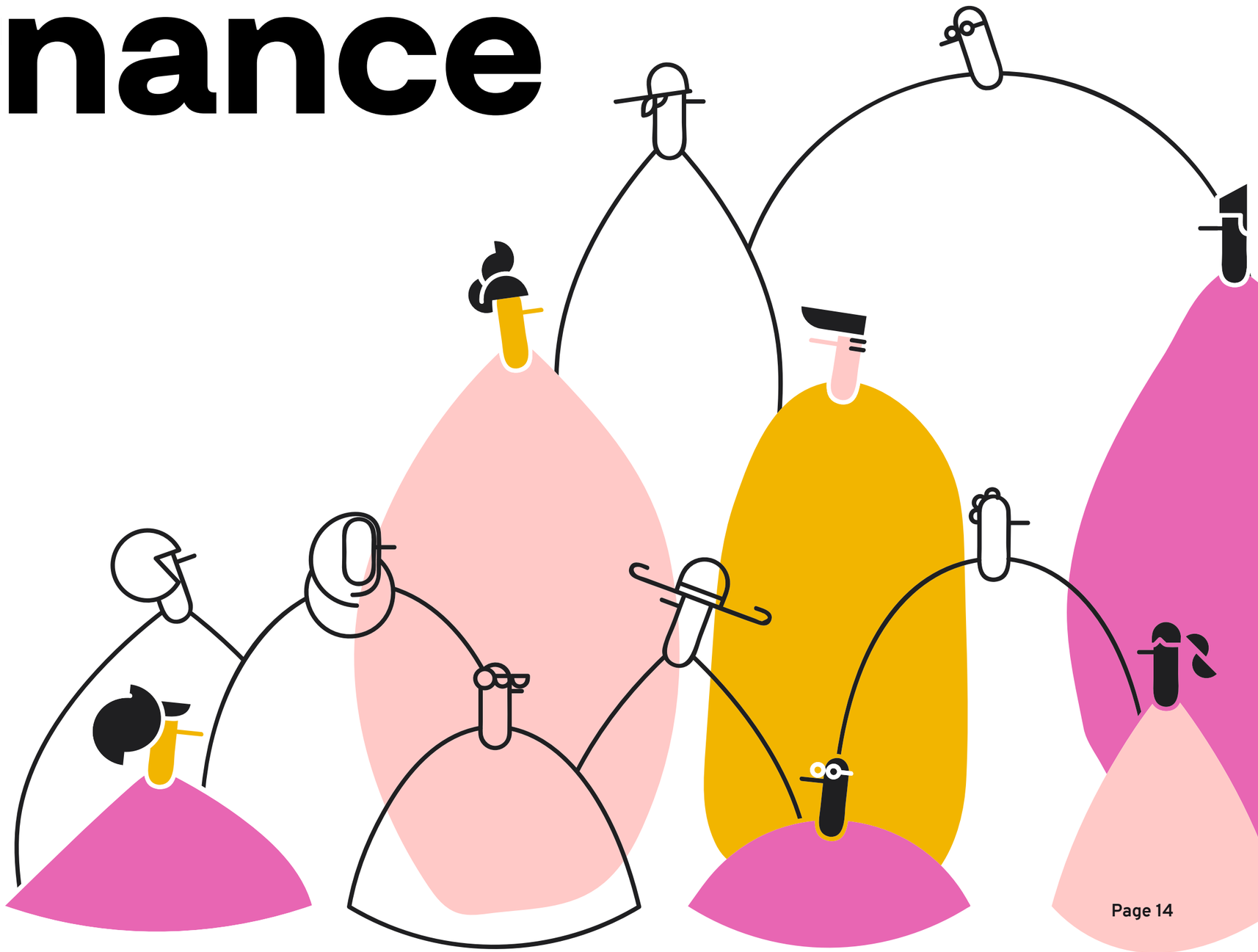
A common use case for many organizations is to have a single code base that's deployed many dozens or hundreds of times to different instances. The code itself needs to be centrally managed, but each instance is "owned" by some specific branch or department or perhaps is geographically localized. Frequently, this may be a Drupal distribution, such as OpenY, Opigno, or Open Social, or a WordPress template. But it could be anything. How, then, can the central organization manage dozens or hundreds of code bases? Source Operations enables any satellite project to be brought up to date with an upstream master repository controlled by process.

Keep your projects consistent with GUI templates



Platform.sh provides GUI template-management functionality, ensuring teams share and reuse work and that workflows are consistent across projects, avoiding duplication or variations. Simple and complex applications or websites can be defined in code and added to centralized template libraries. Templates can also be used on demand via API or CLI, as required. Templates can define core project components such as an ecommerce site, a Drupal or WordPress site, or a Node.js microservice, for example.

Business governance



Platform.sh supplies savings and expertise

Providing PaaS to our thousands of customers is our sole business and allows us greater purchasing power and economies of scale than a customer can achieve as an individual organization. With more than five years of experience, Platform.sh focuses purely on PaaS delivery. That kind of specialized staffing is near impossible to achieve in-house or by recruitment without significant, often disproportionate training and investment over very long timeframes.

Platform.sh bulk purchasing contracts for compute and storage resources and the ability to achieve higher densities via consolidation typically result in at least a 50 percent saving for customers relative to the costs they can achieve from a hosting vendor directly. Deployment options are available on Amazon Web Services, Microsoft Azure, Google Cloud Platform, and Orange Business Services, without the overhead of having to manage any cloud infrastructure yourself.

With transparent per-user pricings and no unpredictable further support costs, Platform.sh can offer benefits for those businesses that want to budget and staff without the uncertainty that often comes with building and maintaining K8s infrastructure projects.

Customer reported metrics

90%
Reduction in DevOps

15x
Faster user acceptance training

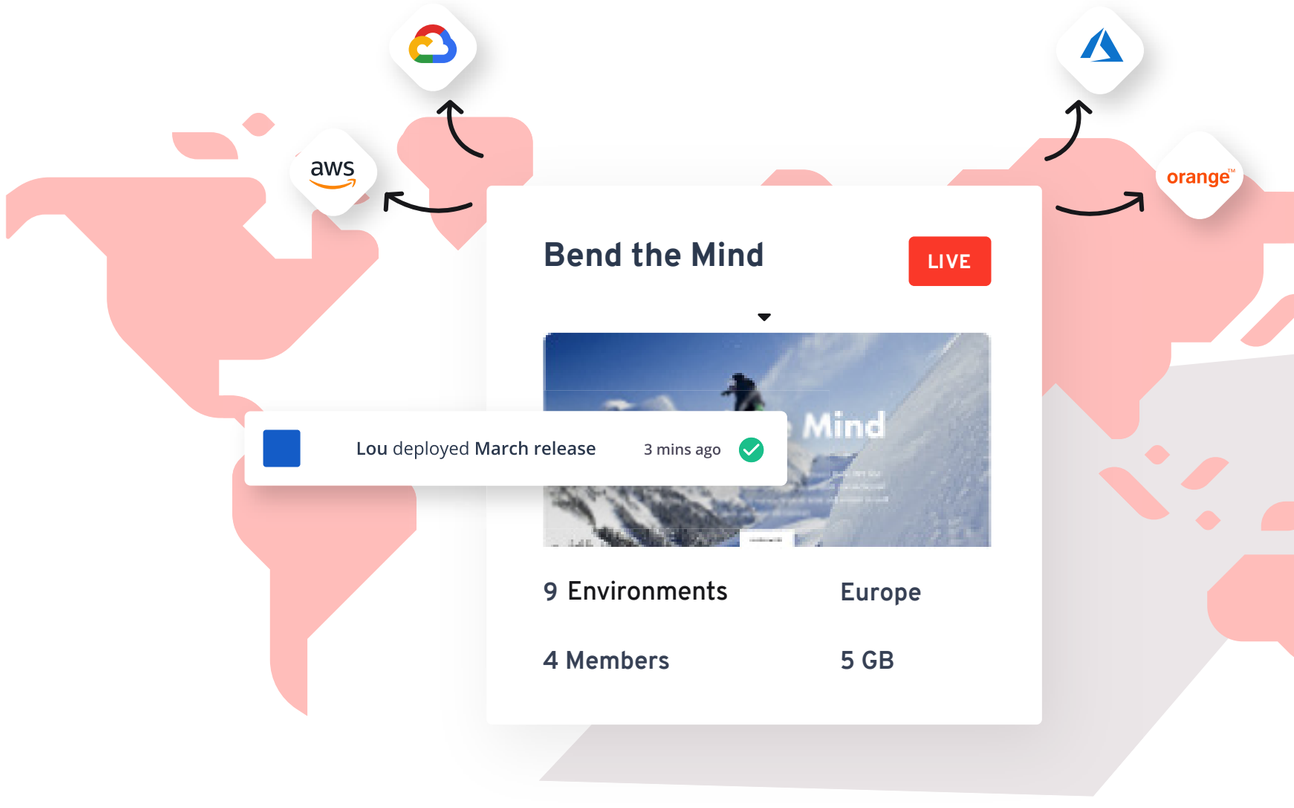
40%
Uplift in developer productivity

20x
Faster innovation cycles

Keep your data local by going worldwide

As one of the few PaaS vendors with the scale to support all major clouds, Platform.sh has the ability to take advantage of geographical locations (latency), optimal pricing, and resourcing. And Platform.sh enables customers to run exactly the same code/configuration across any of these clouds without any rework. In addition, dedicated enterprise clusters are deployed in 30 separate data centers. This enables Platform.sh customers to locate data to comply with their regulatory and data protection needs, as well as to minimize latency and optimize user experience by locating applications and websites near to developers and customers.

Regionally, Platform.sh currently hosts in Germany, USA, Ireland, Australia, Canada, and France (with Switzerland and UK planned to launch in 2020).



The support you need whenever you need it

Enterprise Platform.sh customers are entitled to full 24x7 support. Platform.sh not only builds and maintains your platform, but commits to resolving issues and providing help desk staffing to ensure your issues are resolved rapidly.

Since 2018, our 24x7 global customer support teams have grown more than 3x in size and span six continents. We've improved our response time on urgent enterprise tickets nearly 4x (while our contractual SLA is one hour, we often respond in ten minutes). Our customer satisfaction rating on all tickets averages greater than 92 percent. In addition to our Support teams, we have a Customer Care team that's dedicated to listening to your needs.

Don’t let your data get locked in

Platform.sh offers zero vendor lock-in. Platform.sh allows developers to describe—in the most succinct way possible—the requirements of their application, and the PaaS automatically deploys and manages those services. However, should you ever wish to migrate away from Platform.sh, all you’ll be losing is the automation of the management layer. Your code should run in precisely the same way on any widely available PaaS using PostgreSQL or a MySQL cluster as it would with any other hosting vendor. If you ever want to change your PaaS, you will be able to take your Git repositories and YAML file configurations and reuse them elsewhere.

Next steps

Don’t know if a bespoke Kubernetes solution meets your requirements? Platform.sh has detailed whitepapers available, including cost analyses between the Platform.sh PaaS and on-premises, build-your-own or managed Kubernetes options.

If you’d like to explore Platform.sh yourself, we offer a free trial to allow evaluation.

`https://platform.sh`

platform.sh