

THE GITOPS GUIDE TO BUILDING & MANAGING INTERNAL PLATFORMS



Executive summary

Internal platforms are a great way to enable an outstanding developer experience. GitOps is the best way to enable this platform approach. These two sentences are a concise summary of the key ideas in this paper.

In the last few years, the role of the developer has changed drastically. The “benevolent dictator for life” of any given codebase is being replaced with a team of developers who have to collaborate on a codebase to move it forward. Instead of a single developer who owns the codebase making all the decisions, a group of developers works together to make a product.

In this situation, developer experience is of utmost importance as developers collaborate around code. The way to foster an outstanding developer experience is to leverage the internal platform approach. An internal platform helps to get resources to developers on-demand and in a secure way.

“FAANG and startup companies are increasingly building DX teams to work with platform teams to improve overall organisational productivity and development velocity.” James Governor,

— [Source Redmonk.com](#)

Adopting the platform approach alone is not enough. Organizations need to think strategically about how they would go about building and maintaining the platform. Further, how they would build a new development culture around the platform that makes software delivery seamless.

In this paper, we look at how an internal platform serves to deliver an outstanding developer experience. Further, we look at how GitOps is the way to build internal platforms at scale. With these key strategic initiatives in place, software delivery teams can release software continuously, and with full confidence in the reliability of the system.





Self-service developer experience (DX)

In the past, the only way a developer could get access to servers, data, or software was to ask a system administrator. In the modern era, however, developers require on-demand access to a variety of tools and resources that enable them to be more productive and deliver applications more quickly. The tools and services that developers need to build and ship applications are what make up the developer experience (DX). Crafting a seamless DX is in the best interest of any organization as the popular book by Stephen King says developers are ‘The New King Makers.’

A lot of companies are still struggling to provide a developer experience that will improve developer productivity. But it’s not just about providing better tools. It’s about providing a better way to access these tools and resources that eliminate obstacles. Developers should not have to file a list of support tickets and wait forever to get what they need.

It’s a good idea to have a consistent process in place, but it’s even better to have a consistent experience for your developers. Especially if your organization has numerous different teams, each with its own set of products, and priorities.

“ A great developer experience gets out of the way, leaving the developer in a flow state. DX allows developers to be more effective, by making the good thing the easy thing – in areas such as testing, security and observability this is increasingly important. Good DX allows for shift left.”

— James Governor, [Source Redmonk.com](https://source.redmonk.com)

2021 State of DevOps report

The 2021 State of DevOps report released by Puppet and DORA (DevOps Research & Assessment) has surveyed many organizations and organized them by low, mid, and high DevOps maturity. The report highlighted a DevOps maturity model that shows 80% of organizations being ‘stuck in the middle’, yet to reach high levels of DevOps maturity.

The report points out that there are two types of teams that are prevalent in highly mature DevOps organizations - Stream-aligned teams, and Platform teams.

Use of internal platforms and level of DevOps evolution

Source: Puppet



Stream-aligned teams are essentially application development teams that build features and release code into production. Platform teams, on the other hand, exist to support application development teams and make resources available to them whenever needed.

Application development teams

Application development teams deliver applications and products to end-users. These teams are aligned according to the organization's priorities. A common approach is for organizations to organize their development teams by-product, although in some cases it can be by department. Application development teams care about moving faster and deploying new features and updates with little or no friction.

The internal platform approach

While there are many ways to deliver resources and tools to application developers, the favorite approach of high-performing DevOps teams is to deploy an internal platform that developers use to create and access the resources they need on their own.

The platform team has a critical mission: to provide the resources and capabilities that enable application development teams to work effectively. A platform team delivers self-service APIs to developers. This team builds and maintains the platform that is consumed by development teams. The platform they build is a collection of self-service APIs, tools, services, knowledge, and support.

Development teams have a high degree of autonomy, but they also rely on the platform team to deliver the resources they need to get their work done. In this way, application development teams are internal customers of the platform team. In order to properly fulfill its role as an internal service provider, the platform team must take a strategic approach to resource management.

“The challenge is not in the technology itself, or the tools... The challenge is mainly building the structure inside the teams. We’re building many centers of excellences across all of our business units and all of our teams. To build a structure across 10,000 developers plus is a major challenge.”

— **Amr Abdelhalem, Fidelity Investments**

Why a platform approach?

The platform approach is being adopted by every kind of organization, from every industry, across every part of the world. Mae Large, Architecture Manager at State Farm says that taking a platform approach is about enabling every product team to deploy changes to platforms in a simple, compliant, and repeatable manner.

“Our goal is to give every product team the ability to deploy their changes to our modern strategic platforms in a simple, compliant and repeatable manner.”

— **Mae Large, State Farm**

The job of the platform team in this case is to enforce compliance while giving more power to developers.

There are numerous reasons for adopting a platform approach to software delivery. Here are some of the key reasons:

- 1 Manage deployments from on-premise to cloud to edge
- 2 Scale infrastructure to meet demands
- 3 Enable built-in compliance
- 4 Enforce zero-trust security

Let’s look at each of these reasons in detail.



Benefits of the platform approach

1. Manage deployments from on-premise to cloud to edge

As the infrastructure stack becomes more complex, what organizations need is a consistent way to build and deliver applications from on-prem to edge. Deutsche Telekom is a great example of an organization that needs to support legacy applications and extend them with cloud capabilities and modern practices like GitOps.

The platform approach at Deutsche Telekom enables thousands of application owners and vendors to securely deploy workloads and other services on-premise, across multiple sites, clouds, and at the edge with the standard GitOps operating model.

“ Our application teams have more or less all the basics in their hands to start deploying and running their application in our very specific environment. We are running everything on-prem. We run workloads also traditionally in many locations. We have a handful of what we call core locations, 20 to 25, then there are near edge locations and there are more than 10,000 edge locations.”

— [Vuk Gojnic, Deutsche Telekom](#)

With this tremendous infrastructure complexity, Deutsche Telekom relies heavily on the consistency that the platform approach brings.

2. Scale infrastructure to meet demands

Platforms are exceptionally good at making components repeatable and reusable. This trait is especially useful when an application needs to be scaled at cloud burst speed.

Being in the retail space, MediaMarktSaturn has many days of unusually high traffic spikes. For example, to prepare for a Black Friday sale, the platform team had to start ordering additional servers two to three months in advance, and always over provision. Very short internal deadlines for big product launches, sometimes 30 minutes or less, had the teams struggling to scale up to support a 10 fold increase in web traffic. Some of their applications that usually experience around 300 OPS (operations per second) can shoot up to 3000 OPS in a matter of minutes.

By leveraging the platform approach, MediaMarktSaturn was able to adopt a cloud-first approach. This gave them the power of cloud scalability as they could easily configure new resources in minutes to meet the surge in demand.

During the pilot stage, the evaluating product team discovered that they were able to operate their entire stack with only 3 developers. This instilled confidence in the team to move away from traditional architecture, and to rather build consistent end-to-end workflows that simultaneously increase consistency and introduce standardization. Today, MediaMarktSaturn is confident they can meet any spike in traffic without a hassle.

3. Enforce zero-trust security

Security comes first for any organization, particularly an organization like the Department of Defense (DoD). With the world's most powerful weapons arsenal in its control, there is no room for error with security. The DoD has been transforming itself in recent years from a traditional organization with bureaucracy and legacy systems to one that is founded on open source software and modern cloud-native practices. Along the way, they have kept DevSecOps as their top priority. To make this transformation possible, they leveraged the platform approach to systems building.

Security is baked into the platform at DoD so that developers only need to apply tags to certain resources to have capabilities, like authentication, enabled. Application developers are able to consume these security resources without ever reaching out to the cybersecurity people outside of their team. This bakes in security from the get-go and allows a security-first mindset even when developers are working on an MVP.

“ Everything we do is based on Zero Trust architecture - both for ingress, egress, and east-west traffic. We use the Istio service mesh, and that's the foundation of everything in terms of security.”

— **Nicolas Chaillan, former Chief Software Officer of the U.S. Air Force**


4. Sidecars enable better security

The DoD leverages Istio as its service mesh, and one aspect of Istio is its reliance on sidecars for secure network communication. Sidecars, which are language agnostic, act as service proxies and allow for all traffic (ingress and egress) to flow through them before reaching or leaving a container. This greatly improves security as bad actors cannot easily hide their actions from the sidecar.

The first benefit of a sidecar model is, because it is injected alongside and not inside a container, you can independently update the sidecar or the container. The second benefit is that sidecars are injected automatically, no matter the workload. This means that even if your software team does not know about sidecars, they are still going to utilize the sidecar and its benefits. This is what baked-in security looks like in practice.

An internal platform is built not bought

A platform is an intricate system and it cannot be bought ready made from any vendor. It's something that needs to be custom-built and tailored to the specific needs of each organization. No two organizations will have the same kind of platform. However, there is one common trait that all platforms have. They need to be built using modular 'building blocks' that work across any type of infrastructure, and are flexible enough to meet all the varied needs of the business.



One option that has proved to be an elegant and effective solution to platform building is Git. Git is widely used in software delivery, and its success is the reason why platforms like GitHub and GitLab are household names in every software delivery team today. Git has led to the rise of a modern approach to software delivery called GitOps.

GitOps - The way to build internal platforms

GitOps views Git as the single source of truth. The entire system needs to be declared in Git and any changes are initiated using a simple pull request.

GitOps allows you to create and manage your entire software stack through Git and treat your servers as your source code. Every change and configuration is done through a pull request and everyone has access to the application, which can be rolled back if needed. Just like with your code, you should trust the platform you are building on. With GitOps, you can make sure that each configuration change is verified by your peers and automatically tested before it is deployed.

Your platform should not be managed by one single person. It's a bad idea to have one person who deploys code without anyone else knowing about it. GitOps allows you to have a team of people who are responsible for each change being made on the platform. Beyond this, it helps to build a platform that is modular, flexible, and importantly, secure.

GitOps for internal platform building

GitOps is a way to manage platforms from Git repositories. It requires that every part of the platform be defined in Git. With Git as the single source of truth, and all platform configuration stored in Git, platform operations can be automated with ease. Whenever configuration drift occurs, it is immediately spotted and corrected.

GitOps is great for building internal platforms because it has a set of strong defaults for versioning using Git repositories. The GitOps framework can greatly simplify and speed up the process of building and maintaining internal platforms. It is much easier to manage internal applications when using a strategy like GitOps because you can share and replicate resources, tools, and workflows across various teams from a single platform.

“ GitOps is game changing for the industry. It is a replicable, automated, immutable construct where your change management, everything happens in Git... Everything is founded on that infrastructure as GitOps mindset where your design stays in Git, and everything is in Git. Your production or staging environment will pull from Git continuously to implement whatever change you want to make, to ensure you have no drift and no change between production and your desired state. So we ensure that thousands of clusters who end up running in DoD don't drift between each other.”

— **Nicolas Chaillan, former Chief Software Officer of the U.S. Air Force**

With GitOps, everything must be declarative, which means deployments are reliable and that any action can be replayed or rolled back at any time.

“ Most of our teams operate their software and systems and are managing their infrastructure including Kubernetes clusters on their own. [The] GitOps approach was the enabler so that even small teams can operate in the cloud on their own. ”

— [Florian Heubeck, Principal Engineer, MediaMarkt Saturn](#)

Benefits of GitOps for platform building

Here are the top benefits of using GitOps to build and manage your internal platform:

- 1 Make risk-aware decisions
- 2 Detect system and configuration drift
- 3 Enable built-in compliance
- 4 Implement progressive delivery

Let's dive into each of them.

1. Make risk-aware decisions

Risk awareness in deployments is about being aware of what lines of code have changed, and having the ability to rollback specific changes if necessary. This kind of precise control over deployments and change management is a hallmark of the platform approach to software delivery.

Thanks to GitOps, MediaMarktSaturn now experiences almost no downtime. Before GitOps, it was the norm for them to experience outages during a deployment. Now, instead of outages, they have automatic rollbacks. In fact, they had an incident where a corrupt configuration that made it to production did not even raise an alert because the release was so quickly rolled back.

2. Detect system and configuration drift

Drift of any sort plagues operations teams. Especially as the stack becomes more complex, drift leads to incompatibility between different parts of the stack. GitOps counters this challenge by embracing a declarative approach to systems building. This ensures that the system always stays faithful to the original declared state as in Git.



“ The key is to monitor your staging and your production environment, and look at anything that’s not in Git. Anything that’s drifting or not present in Git is effectively either malicious or adrift. And that should never happen.”

— **Nicolas Chaillan, former Chief Software Officer of the U.S. Air Force**

GitOps effectively reduces your attack surface by not allowing people to make manual changes to production - changes can only be applied through a reviewed change in Git. In the cluster, in the infrastructure, in your identity service, you do everything in Git. Drift detection doesn’t just make infrastructure management easier and more predictable, it greatly tightens security controls.

3. Enable built-in compliance

For organizations in highly regulated industries like financial services and healthcare, compliance is a high priority. They have specific needs to have data stored only on-premises and ensure a complete audit trail of all actions performed within the system. With these high expectations, ad hoc management will not hold up. Instead, only the platform approach can deliver the peace of mind required.

Fidelity Investments, a leading global financial services firm, faced the challenge that every application they create must meet a unique mix of regulatory, security, and governance requirements. The team at Fidelity wanted the scalability and reliability that comes with adopting Kubernetes, and also sought to leverage the ecosystem of available cloud-native open source projects in order to remain innovative and improve their time to market. However, they needed to implement cloud-native technologies within their highly regulated environment using their existing control and audit guidelines.

Fidelity built a GitOps-powered platform that allowed them to maintain regulatory compliance, yet didn’t impact the speed of innovation across their teams. Automating platform configuration and managing it with GitOps provided a way for the SREs to easily perform upgrades on these configurations and to introduce controls through pull requests. The result is a standard cluster development platform that spans environments with security guardrails and a common set of cloud-native patterns, like secrets management, in place that any development team can deploy to and from.

4. Implement progressive delivery

Progressive delivery is an advanced release pattern that allows you to release a new feature to a small percentage of users, let them test it out, and if it works well enough, make it available to everyone. It is a great way to enhance user experience and make the most of available resources. Progressive delivery includes approaches such as canary releasing and blue-green deployments. Flagger is an open source GitOps tool that is purpose-built to manage all aspects of progressive delivery.

MediaMarktSaturn uses Flagger to automate canary releases in a GitOps pipeline. Whenever the GitOps operator, in their case, Flux, detects a new version, Flagger spins up a new canary release and shifts traffic gradually over from an existing service to the new version while Flagger's load generator simulates customer traffic. Every 5 minutes, 5% of the traffic is moved from the primary to the canary version. If the release goes as planned, the canary will end up with 100% of the traffic and the old version will be retired.

The biggest difference MediaMarktSaturn experienced after Flagger is increased confidence in their deployment results. Stable rollouts at high deployment speed empower the team to now deploy during high-traffic times without issues. Even when the inevitable failure happens, they can recover quickly. The team can now spin up an identical full-stack production system within five minutes and recover from disaster.

“Right now, our systems are nearly 100% reliable.”

— Florian Heubeck, Principal Engineer, MediaMarktSaturn

The path to GitOps adoption

Take a phased approach

The time it takes for your organization to adopt a GitOps approach greatly depends on where you are in your DevOps journey. If you've already been leveraging Git for collaboration, have a CI/CD pipeline in place, and use open source tooling across your pipeline, you may see quicker adoption time. However, for many organizations, it can take more time.

State Farm, for example, had 3 phases when adopting GitOps.

- 1 Jan 2020 - General availability of GitOps (Public Cloud and Kubernetes on-premises)
- 2 Dec 2020 - GitOps for on-premises Cloud Foundry
- 3 Jan 2021 - GitOps for Kubernetes on-premises

This phased-out approach ensured adequate time for the various teams to get on board at their own pace. It also provided the platform team time to adapt to issues as they arose, ensuring a smoother transition.

Describe everything as code

Even for organizations that have been using Git for a while, the concept of 'everything as code' can be a far stretch. Yet it is essential for GitOps and to enable a platform model.

“ You always have to factor in scale. Describing things as code, specifically embracing the declarative nature of a lot of these modern platforms is such a key and powerful enabler.”

— Mae Large, State Farm

To build declarative systems, every part of the system first needs to be defined before it can be declared. This is why it is important to take the time to transpose the entire system from various disparate pieces to Git.

▼ Leverage open source tools like Flux & Flagger as a POC

GitOps is powered by open source tools that are backed by large developer communities. Flux and Flagger are two key open source tools that make up the GitOps toolchain.

“ Weaveworks and the CNCF delivered a transformative open source tool with Flux; managing Kubernetes deployments at scale is now easy and fast.”

— Nicolas Chaillan, DoD

Along with Flux and Flagger, leveraging open source tools like Prometheus, Istio, Helm, and Elasticsearch can bring much momentum to the transition from traditional to the platform approach.

“ I wish GitOps practices were available a decade ago. GitOps increases the quality of everything we do. Right now, I can't imagine working differently anytime in the future.”

— Florian Heubeck, Principal Engineer, MMS

Weave GitOps for internal platform building

An internal platform is vital for modern cloud-native operations. The days of standing up infrastructure, and then maintaining it for years at a time, are long gone. The pace of business today means that infrastructure that is standing still is infrastructure that is getting outdated by the minute. As your organization looks to deliver an outstanding developer experience—and manage infrastructure from data center to edge—an internal platform is the key enabler.

Weaveworks, one of the key organizations behind the OpenGitOps project, recently announced the release of [Weave GitOps](#), a full-stack GitOps platform based on Weaveworks' highly successful open source projects Flux and Flagger. Weave comes with everything you need to build and manage internal platforms at scale. It implements concepts from GitOps to provide a powerful, yet simple approach to operating internal platforms.



Some of the noteworthy features of Weave GitOps are as follows

Reconciliation loop	It constantly monitors changes in Git repositories and enables you to automate their deployment by simply approving a pull request.
Simple profile bootstrap	Setup production-ready clusters in a fraction of the time it takes.
Drift detection	Constantly monitor for drift in the state production clusters and revert them back to the original declared state.
Cluster reuse	Rather than creating new resources from scratch, simply modify and reuse a previously created cluster.
Team workspaces:	Give each development team its own secure namespace with all the tools they need for development and monitoring.
Built-in security guardrails	With RBAC, SSO, and Git's version control, you can make core DevSecOps practices the norm without any added effort.
Enable progressive delivery	Leverage Flagger and a service mesh tool to execute complex progressive delivery patterns like canary releasing and blue-green deployments.initial workshop and architecture review

Weave GitOps Enterprise is the fastest way to take your organization to new levels of GitOps maturity. However, if your organization is at an early stage in this journey and you'd like to test the waters first, [Weave GitOps Core](#) is the underlying open source project that any organization can get started with. It includes many of the essential GitOps capabilities to simplify Git reconciliation, and cluster creation.

Key takeaways

Here are the key points to take away after reading this paper:

1

Developer experience matters: Development teams today require a self-service developer experience (DX) to enable them to deliver continuously.

2

Leverage an internal platform: Top performing DevOps organizations find the internal platform approach to be the most effective way to deliver a self-service DX. Here are the top benefits of a platform approach:

- ▶ Manage deployments from on-premise to cloud to edge
- ▶ Scale infrastructure to meet demands
- ▶ Enable built-in compliance
- ▶ Enforce zero-trust security

3

Use GitOps for platform building: Building and managing platforms can be challenging and requires an approach that is flexible, yet secure. GitOps fits the bill as the ideal solution for platform building. Here are the top benefits of using GitOps to build an internal platform:

- ▶ Make risk-aware decisions
- ▶ Detect system and configuration drift
- ▶ Enable built-in compliance
- ▶ Implement progressive delivery

4

GitOps adoption best practices: As you look to adopt GitOps in your organization, here's how to go about it:

- ▶ Take a phased approach
- ▶ Describe everything as code
- ▶ Leverage open source tools like Flux & Flagger

5

Weave GitOps: Leverage Weave GitOps Enterprise to greatly simplify and quicken your adoption of the internal platform approach. It is built on successful open source tools, and delivers GitOps best practices out of the box.





Further reading and sources:

1. **James Governor, Redmonk:** [What is Developer Experience? a roundup of links and goodness](#)
2. **Case Study:** [Deutsche Telekom uses Weave GitOps to delivery edge Kuberentes at scale](#)
3. **Podcast:** [Kubernetes at Deutsche Telekom - GitOps at the Edge](#)
4. **Case study:** [How Deutsche Telekom automates edge infrastructure using GitOps](#)
5. **Video:** [The world's largest telcos are now embracing GitOps. Deutsche Telekom explains why](#)
6. **Case Study:** [GitOps boosts collaboration, reliability and autonomy for MediaMarktSaturn](#)
7. **Case Study:** [MediaMarktSaturn finds confidence to implement progressive delivery with Flagger](#)
8. **Blog:** [DX at the Department of Defense: Platform One and GitOps](#)
9. **Case Study:** [How the Department of Defense \(DoD\) uses GitOps to bake in security](#)



www.weave.works