weaveworks

eBOOK

GITOPS FOR ABSOLUTE BEGINNERS

AUTHOR: TWAIN TAYLOR

weaveworks

Who this guide is for

You've likely heard of GitOps at a KubeCon, or read about it online, or heard about it in a conversation with your colleague. You've been curious about the idea, but have not fully understood it yet. You're ready to take the first steps to GitOps adoption, but don't know where to start. If this sounds like you, this guide is the best starting point for your GitOps journey.

What you'll get in this guide

- A quick way to assess your organization's current situation and readiness for GitOps adoption
- An overview of the key benefits of GitOps
- The four key principles of GitOps as defined by the Open GitOps project
- Specific directions on how to get started with GitOps no matter if your organization is at a beginner or advanced level with DevOps

weaveworks

Signs you're ready for GitOps

Like every software methodology before it, GitOps requires organizations to be ready for its adoption. Whether you come from a large enterprise setup, or are part of a scrappy startup team, there are ways to tell if your organization is ready for GitOps. Here are signs that you're ready to adopt GitOps:

1

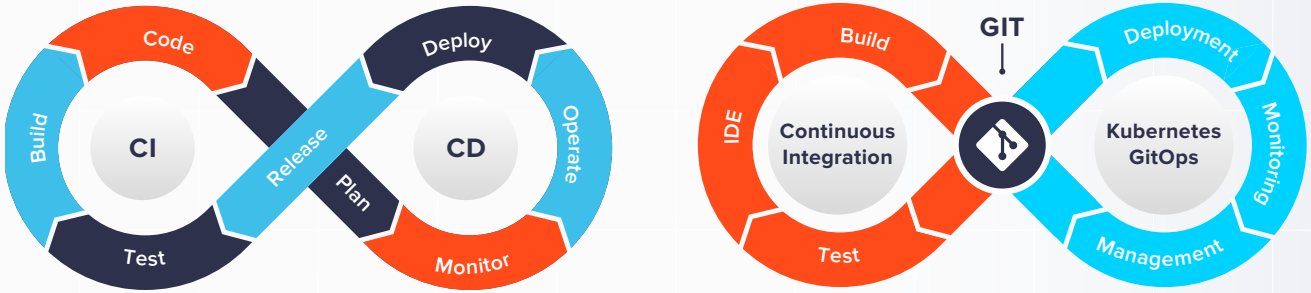
You have a CI/CD pipeline

If you come from the world of DevOps, you have a fairly functional CI/CD pipeline in place. This is a great starting point for GitOps. The problem with CI/CD is that it isn't as automated as it promises to be. This is because of the numerous manual tasks required at each step. GitOps extends CI/CD with end-to-end automation.

2

You've adopted the cloud

You run infrastructure and applications in one of the cloud vendors like AWS, or Azure, and may even be considering multicloud setups. If you've used a lift and shift model to migrate your systems from on-premise to cloud, you may still be building software using the traditional waterfall approach. If you feel you're not making the most of the cloud, GitOps is a great way to change this situation.



CLICK TO NEXT PAGE

3

You run Kubernetes clusters in production

You have been riding the containerization wave, and are an early adopter of Kubernetes. However, it's not all smooth sailing. Your teams struggle with the complexity of Kubernetes. You find yourself struggling to keep up with the latest fad in cloud-native technology. GitOps can bring a lot of sanity and clarity to your Kubernetes management.

4

Your team's developer productivity can be improved

Developer experience (DX) needs to be elegant and effortless for developers to be productive. This includes making it easy for developers to access the resources they need on-demand, without the assistance of Ops. This, in turn, requires that Ops does not provision these resources manually, but instead has them all templatized and automated. GitOps excels at enabling this.

5

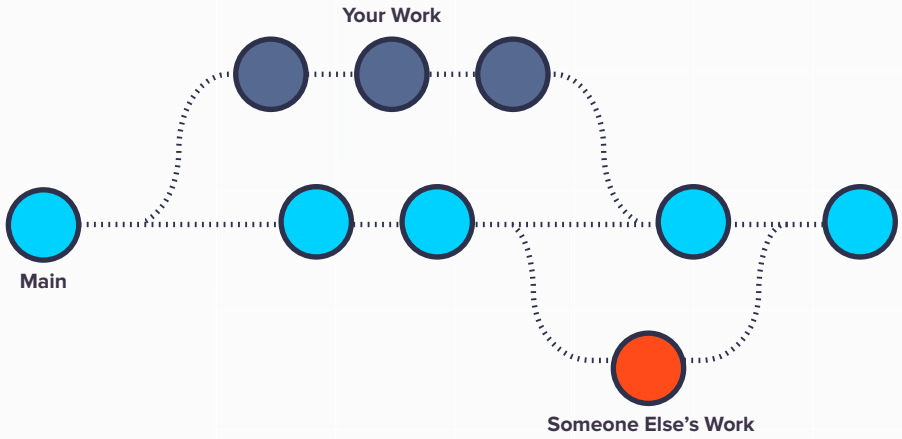
Your business needs faster deployment velocity

The 2020 DORA report talks about how 'developer velocity equates to business success.' In order to have more frequent deployments, you need to automate every step of the pipeline from start to end. This is easier said than done. GitOps shines here allowing for advanced deployment automation.

6

You're big on Git

Your Dev teams collaborate using Git platforms like GitHub and BitBucket. The branching structure, pull/merge requests, and in-built versioning are things you enjoy about Git. However, you've felt there's more to Git than this. You'd like to get the most out of these Git tools and need the right processes to complement them. That is just what GitOps brings to the table.



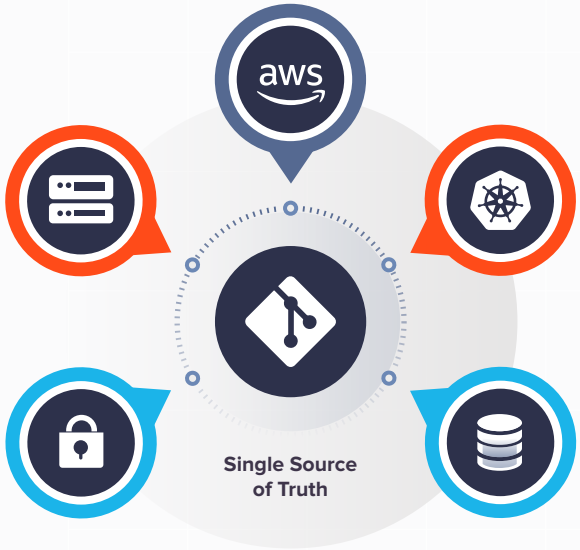
The more you can relate to these signs, the more ready your organization is to adopt GitOps. But before that, why should you care about GitOps? There are a bunch of reasons.

Benefits of GitOps

GitOps has many strengths and benefits that appeal to different organizations. In the list below, you're bound to find something that resonates with you.

1 **Consistent, standardized workflows**
With GitOps, Git becomes the single source of truth. This avoids duplication of work, compartmentalizing of knowledge, and communication gaps. Software delivery processees become seamless as silos are broken across Dev and Ops.

2 **Everything auditable, automatically**
GitOps leverages Git's built-in versioning capabilities to track every change to every repository. These changes greatly facilitate auditing and compliance.



3 **Predictable operations with end-to-end pipeline automation**
GitOps prescribes completely automated operations. This means right after a pull request is merged by a developer, the change is automatically applied into production (unless it's a bad change, of course. More on this later). This automation reduces human error and makes operations more predictable.

CLICK TO NEXT PAGE

4 **Greater developer productivity**
A natural offshoot of automation is greater deployment velocity. It's no longer just the tech innovators like Google and Amazon who can deploy on-demand, GitOps brings this ability to any organization.

5 **Avoid drift in production**
GitOps uses agents that watch production clusters for any deviation or drift from the desired state described in Git. This drift is automatically corrected by reverting to the original state.

6 **Greatly enhanced DX**
Developers need not wait for Ops to provision resources. They can simply pick from ready-made templates, and spin up cloud resources as and when needed. This enables them to move faster from code to production.

7 **Deployment guardrails**
GitOps allows for fast and easy rollbacks if a new deployment breaks production or is otherwise suboptimal. But how can teams set deployment guardrails that stop errors even before they reach production? Using a policy engine such as **Magalix**, configuration that could be a security weakness, reduce application resilience or not conform to coding standards can be detected when a pull request is submitted. Magalix also includes an admission controller that ultimately blocks configuration with policy violations from being applied.

Surprising fact: Speed and Stability support each other!

In our Whitepaper **"How GitOps boosts business performance: The Facts"** we examine how GitOps positively influences DORA metrics.

Learn more

8 **Improve reliability with progressive delivery**
GitOps, with the help of Flagger, enables progressive delivery approaches like canary releasing. These are complex to implement outside of GitOps. On the other hand, GitOps makes complex delivery strategies easy to implement as every minute detail is described and controlled in Git.

If you're excited about the possibilities with GitOps, it's time to dive into what GitOps actually is.

weaveworks

The Principles of GitOps

In mid-2021, the GitOps movement took a big step towards standardization as the CNCF (Cloud Native Computing Foundation) commissioned the **OpenGitOps project**. This project was created to bring vendors and customers together around the idea of GitOps.

The team's first task was to define four foundational principles of GitOps, which are as follows:

1 **Declarative**

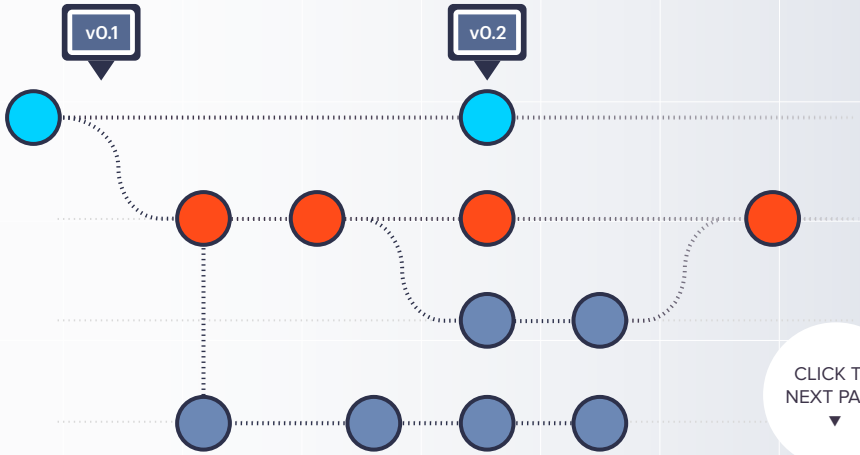
"A system managed by GitOps must have its desired state expressed declaratively."

'State' is the key word here. Every system (and every part of a system) in GitOps has a desired state. This desired state is the end goal of all software operations conducted within the system. GitOps gives more importance to this desired state than the means of attaining the state. Accordingly, systems are described in Git, and those descriptions are implemented by deployment tools using API calls, scripts, and more. A fundamental goal of GitOps is to describe everything in Git.

2 **Versioned & Immutable**

"Desired state is stored in a way that enforces immutability, versioning and retains a complete version history."

Git is the preferred store of the desired state mentioned above. State is 'immutable' in that any change to a new state requires a complete replacement of the previous state, not a change. Every time this happens, the action is recorded automatically (versioning). With the number of users, repositories, and changes taking place, it is essential to have a clear audit chain for the sake of system security as well as regulatory compliance.

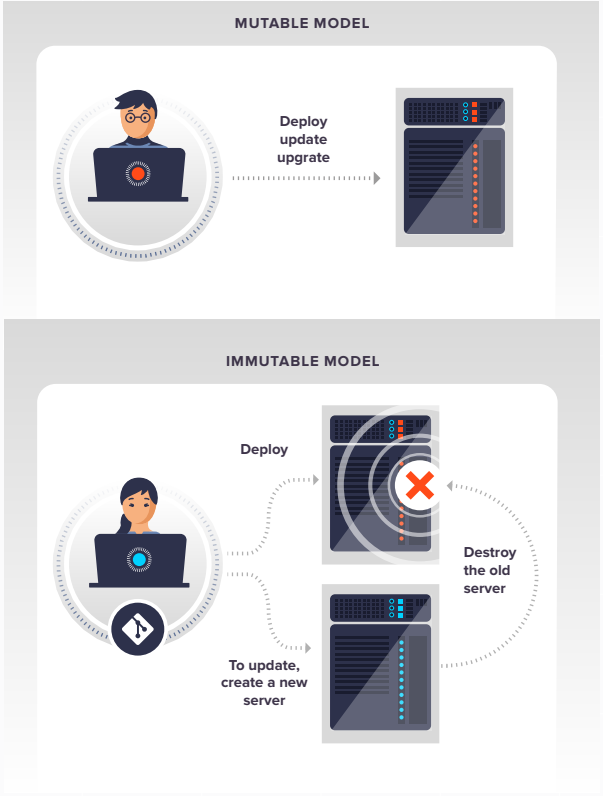


CLICK TO NEXT PAGE

4 Pulled automatically

"Software agents automatically pull the desired state declarations from the source."

Traditionally, developers push changes to a production system. With GitOps, it's the reverse. A software agent acts as a bridge between a Git repository and production environment. This agent watches for any changes to the Git repository and automatically 'pulls' every new change from the repository and 'merges' it into the production environment. This automation is a crucial difference between GitOps and its predecessor CI/CD.



5 Continuously reconciled

"Software agents continuously observe the actual system state and attempt to apply the desired state."

Systems are continuously changing in production. This means that as soon as changes are deployed, they tend to drift. GitOps prevents drift from occurring by automatically reconciling the live system according to its desired state in Git. This is a powerful feature of GitOps and has many ripple benefits in terms of system security, performance, and more.

For a more detailed and formal description of these four GitOps principles, please refer to the **Glossary** created by the OpenGitOps project.

weaveworks

A typical GitOps pipeline

Though software delivery pipelines vary by organization, there are common traits that most GitOps pipelines share. It starts with the developer merging a pull request from their development environment. From here, a GitOps agent like Flux notices the change and ensures that it is compatible with the production system. If a change fails, it is not allowed to be deployed. If it passes, it is automatically applied to the target production environment.

This is a basic GitOps flow, but there can be numerous additional steps beyond these. For example, the Ops team can set the flow to require a manual review of certain types of changes before they can be deployed. Or, after Flux approves of a change, the progressive delivery tool Flagger could take over and initiate a complex canary release sequence. The possibilities are endless, but a simple GitOps flow starts with Git, is carried forward by an agent like Flux, and ends in a production environment (usually a Kubernetes cluster).

Push versus Pull pipelines for CICD

Read more on

How to adopt GitOps even if you don't

The annual **State of DevOps repor**t released by Puppet and Google Cloud groups organizations into three DevOps maturity levels - low, mid, and high.

We'll use these same categories to outline what's required for your organization to adopt GitOps from where you're at today.

	Low	Mid	High
Deployment frequency	Monthly or less often	Between daily and weekly	On Demand (whenever we want)
Lead time for changes	Between a week and 6 months	Less than a week	Less than a hour
MTTR	Less than a week	Less than a day	Less than a hour
Changes failure rate	Less than 15%	Less than 15%	Less than 15%

CLICK TO NEXT PAGE

1

Low DevOps maturity

For organizations that are new to the cloud, or DevOps, the first step to GitOps adoption is to start using Git for developer collaboration. It is key to standardize this across both Dev and Ops teams. This will require Ops teams to familiarize themselves with how Git works. From here, the next step is to define your systems as much as possible in Git repositories.

A change of culture is the hardest part for any organization. This is true with GitOps adoption as well. In this stage, training and documentation are essential to ensure the entire organization is on board with the GitOps adoption plan. Key stakeholders need to buy into the idea, and every team member needs to play their part in learning and practicing the core GitOps principles.

Once you're ready, you can get started with Flux, or even better, the open source **Weave GitOps Core**. This will give you the bare essentials required to get started with GitOps.

2

Mid DevOps maturity

This category consists of the largest group of organizations. They are usually stuck in the middle with routine practices and are unable to breakthrough to the high level.

If your organization falls in this category, you likely have an existing CI/CD pipeline, perhaps with Jenkins. This is a great starting point. It's most likely your systems are still not fully described in Git, and this will be a priority. From here, you should look to drive greater automation with the help of GitOps agents.

You should look to organize your teams into separate Platform and Application Development teams as recommended by the **2021 State of DevOps report**. The Platform team should aim to move from manual resource provisioning to automated templates and packages.

In terms of tooling, you could opt for Weave GitOps Core, or the more complete solution - **Weave GitOps Enterprise**. This has all you need to get started and up to speed with GitOps.

CLICK TO NEXT PAGE

3

High DevOps maturity

If your organization belongs to this category, it's likely you're already an advanced user of Kubernetes and its ecosystem of cloud-native tooling. You already believe in the Kubernetes approach of declarative ops. GitOps is the best way to realize this goal of declarative ops.

You could start by avoiding manual changes to Kubernetes clusters, and instead, deploy changes via pull requests only. Establish a working cadence of 'pull requests' and 'merges' between Dev and Ops teams. You likely already automate parts of your supply chain. The next step is to use GitOps to implement end-to-end automation. Once these tasks are checked off, you can turn your attention to progressive delivery. This will involve a tool like Flagger, and service mesh like Istio or Linkerd.

Weave GitOps Enterprise is the way to go as it includes Flux and Flagger, along with the other required components like Helm, and ready-made plugins for Prometheus and Istio. Weave GitOps Enterprise has all you need to achieve even higher levels of DevOps maturity.

Key takeaways

1. Each organization takes its own path with GitOps adoption. You'll need to assess where your organization stands in its readiness for GitOps. GitOps has numerous benefits that improve operational efficiency, deployment velocity, collaboration, and security and compliance. Knowing these benefits will help you better advocate the need for GitOps to your key stakeholders.

2. The Open GitOps project has laid out four fundamental principles of GitOps. Familiarize yourself with the key terms and ideas in these principles and begin applying them within your organization.

3. Compare your current software supply chain with the typical GitOps pipeline, and look for areas of improvement, and opportunities for change.

4. Whether you're new to DevOps, or an advanced Kubernetes user, there is a GitOps adoption path for you. Weave GitOps Core is great if you want to test the waters with GitOps, but if you'd like to cut short your adoption time frame, consider Weave GitOps Enterprise.

5. GitOps works not just for the application layer, but for the infrastructure layer as well. As you leverage Weave GitOps Enterprise, you'll find that you progress much faster to higher levels of GitOps maturity.

Try it yourself

Weave GitOps Core works with any Kubernetes cluster on your workstation or in the cloud:

- Kubernetes in Docker (kind)
- K3d / K3s
- miniK8s
- Minikube
- AWS EKS
- Azure AKS
- Google GKE

Use these steps to get started with Weave GitOps Core:

- **Bootstrap Flux** onto your Kubernetes cluster
- **Install Weave GitOps Core** via the Helm chart; with GitOps of course

Now you are ready to install your first application to Kubernetes, one pull request at a time.

- Add your Git repository
- Add a Kustomization from your Git repository
- Git add, commit and push

You can view the deployment state of your applications, source synchronisation status and the health of the Flux system components on the Weave GitOps dashboard. Alternatively from the command line with:

```
$ flux get all
```