How DevOps and GitOps differ and how they overlap

Tony Le

May 2022

Abstract

DevOps is an increasingly popular approach to software development, and GitOps has been touted as its possible next step. This essay breaks down what DevOps and GitOps are, their benefits and challenges, and discusses their differences and similarities. The findings of the essay is that they share a lot of similarities, notably in that they embrance continuous integration and continuous deployment. Some of the challenges faced by DevOps when it comes to implementation is that it is heavily reliant on tools, from which there are too many to choose from. The use of many new tools make it costly and complicated to adopt, which GitOps has less of an issue with as it is tied to Git and Kubernetes, the former of which a lot of developer should be familiar with. The conclusion of this essay is that even though they share similarities and GitOps improves on certain aspects, GitOps is less of an evolution of DevOps, but closer to a subset of it.

Contents

1 Introduction

DevOps is a buzzword in software development that is increasingly popular nowadays. Several of its aspects come from the Agile methodology, and it is widely considered to be an evolution of it. GitOps, on the other hand, is a relatively newer concept, and it has been touted as possibly the next step of DevOps, by the pioneers of the former [Weaveworks Blog, 2022] on their blog. The main aim of this essay is to dig into DevOps and GitOps and what they are, how they are defined, and how they relate to each other. To draw any conclusions regarding whether it can really be considered the next evolution of DevOps or not, this essay will study what constitutes DevOps and GitOps, which challenges they typically face, and the two will be compared regarding their differences as well as overlaps. Hopefully readers of this essay will gain a better understanding of what DevOps and GitOps are, as well as how they are linked with each other.

2 Background

2.1 DevOps

DevOps is buzzword, and there is no single concrete description of what it really means [Khan et al., 2022]. One way of describing the idea of DevOps is as a culture or a set of practices that combine software development and IT operations, making use of automated development, deployment and infrastructure monitoring. It is an organizational shift in which, instead of having siloed teams that work with development and operations, cross-functional teams work with both development and operations for continuous operational feature deliveries. [Ebert et al., 2016].

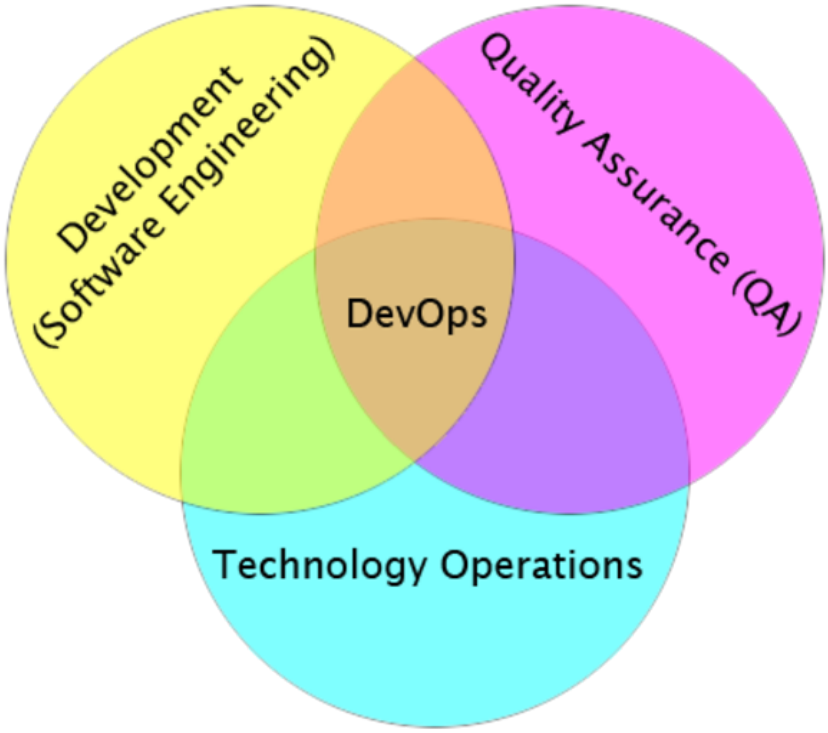

Figure 1: Image showing DevOps as an intersection. Image from Wikimedia Commons.

DevOps is an approach that helps reduce problems arising from miscommunication between teams and additionally allows for continuous delivery of value [Ebert et al., 2016]. Automated continuous integration and continuous delivery (CI/CD), which make up the DevOps pipeline, are critical components of DevOps. This means that building, testing and deployment of software is automated, and that allows DevOps teams to receive feedback and ship software faster, also more reliably.

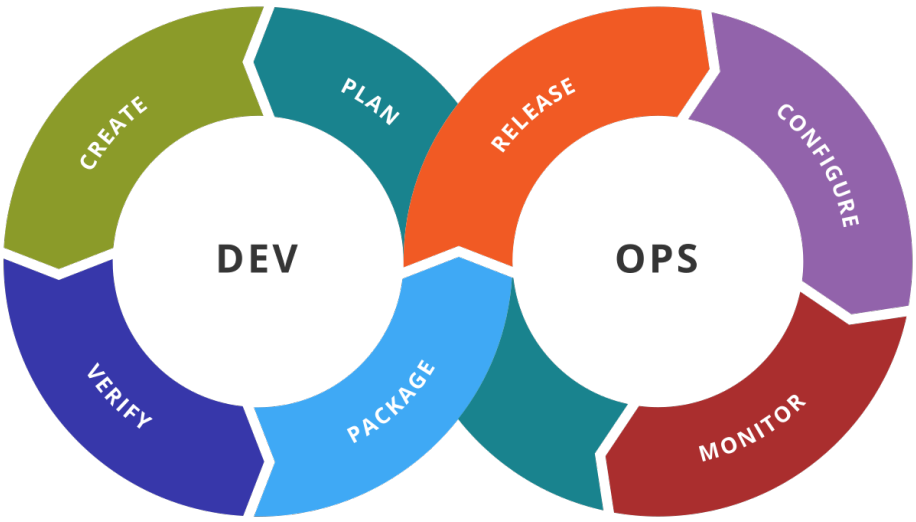

Figure 2: Visual representation of DevOps workflow. Image from Wikimedia Commons.

The following practices are commonly followed in DevOps practices, but they are not strictly required [Beetz and Harrer, 2021]:

a. *Continuous Integration (CI)*: Developers integrate their small pieces of their work frequently, often at least daily.

b. *Continuous Delivery (CD)*: Ensures that the result of the pipeline is frequently deployed.

c. *Communication*: Improve the communication within the team.

d. *Monitoring and Logging*: Use monitoring and logging technologies to collect metrics about the behavior of the software.

e. *Infrastructure as Code (IaC)*: Use IaC tools such as Puppet, Ansible or Terraform. IaC boils down to using configuration files as code, for generating the same infrastructure environments on every deployment.

f. *Microservices*: The scope of work of one team must be smaller and more focused, by using microservice architecture design approaches.

2.1.1 The benefits & challenges of DevOps

Some of the primary benefits of DevOps is that it helps reduce miscommunication [Ebert et al., 2016], and tightens the feedback loop between Dev and Ops [Gottesheim, 2015].

A study on 11 multionational software-intensive companies done by [Díaz et al., 2018] found that most of them confirm that DevOps brings many benefits, such as higher productivity, better feedback and lower IT cost.

One of the primary challenges of DevOps is that the implementation of its practices requires familiarity with both software development and IT operations. [Khan et al., 2022] found that there is insufficient technical knowledge and understanding of the key concepts, methods and tooling of implementing DevOps.

In practice, DevOps relies heavily on tools of various kinds, there exist specialized tools for each portion of a DevOps pipeline. [Zhu et al., 2016]. There are more than 100 DevOps tools available that are also continuously evolving. The use of many new tools makes it costly and complicated for organizations and team members, which has a negative effect on its adoption. [Leite et al., 2019].

2.2 GitOps

According to Weaveworks, GitOps, which was introduced in 2017, is an operating model that uses the Version Control System (VCS) Git as a single source of truth to facilitate Kubernetes cluster management and application delivery [Weaveworks Limited, 2022]. It takes DevOps best practices for software development, such as CI/CD and applies them to infrastructure automation [GitLab, 2022].

[Limoncelli, 2018] describes the GitOps workflow as the following:

1. Configurations are stored in Git.
2. Customers can change send requests as pull requests (PR).
3. A CI system runs automated tests to validate the request
4. A human manually approves the PR, which activates the mostly automated CI/CD functionality.
5. Once the PR is approved and all the tests are passed, the change is deployed by the CD system.

2.2.1 Kubernetes

Kubernetes is a orchestrating tool for containers, which provides them with abstracted functionality like internal networking and file storage, and necessities for orchestrating containers, and the infrastructure that they run on. [Tyson Matthew, 2022]. Kubernetes runs a workload by placing containers into Pods to run on Nodes, where a Node might be a physical or virtual machine [Kubernetes, 2022]. A set of nodes make up a cluster. The configurations of a Kubernetes cluster can be customized by submitting YAML documents to the API of a cluster.

A container simply explained is a pre-packaged environment with all dependencies an application needs to run. Similarities can be drawn to a virtual machine (VM) that is entirely self-contained, but while a VM virtualizes an entire machine including the hardware layer, containers only virtualize above

the operating system level. They typically are configured so that they always work the same, which simplifies testing and deployment.

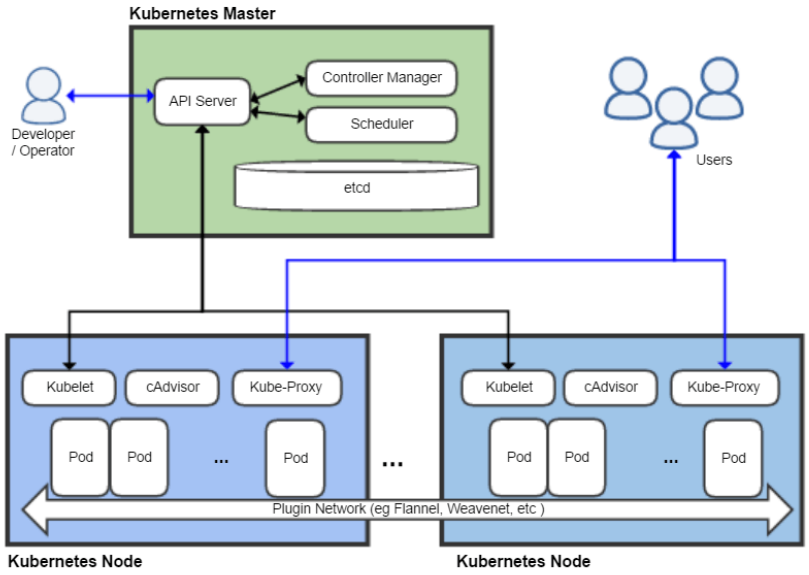

Figure 3: Image showing a Kubernetes cluster. Image from Wikimedia Commons.

2.2.2 The benefits & challenges of GitOps

The automation of application deployment and changes to the existing infrastructure results in faster feedback loops. The usage of Git, which keeps a complete history of all changes, also means that there is complete transparency of when and by whom a change was made to the infrastructure, thanks to digital signing. Git also supports reverts and rollbacks. [Beetz and Harrer, 2021].

Some of the challenges with GitOps would be that it requires use of a VCS, with the name implying that it should also be Git, that it strictly requires a CI/CD provider unlike DevOps where it is only a common practice, and it is is made for use with Kubernetes and other container orchestrating tools [Beetz and Harrer, 2021].

3 Comparison & Discussion

Firstly, I will consider the aspects in which they share similarities. It is no surprise that they are similar in that CI/CD are critical components of both, as GitOps enforces application of many DevOps best practices. Thus the benefit of getting faster feedback loops which implies higher reliability applies to both of them. GitOps also makes use of Kubernetes and containers, i.e. it has a microservice architecture design approach which is yet another part of DevOps culture. GitOps also makes use of IaC like commonly in DevOps. Automation is also a core principle for both.

When it comes to the differences between the two, a notable one is that DevOps is a lot more loosely defined and abstract, the best practices are akin to recommendations to follow, whereas GitOps guidelines are relatively stricter and required, for example with CI/CD. A lot of the times, DevOps is described as a culture, and that is very fitting description. GitOps on the other hand, is strongly tied to specific tools, Git and Kubernetes, the latter of which makes its use case more niche. It is also less of a culture, but more of an applied approach. There are over 100 tools that can be used for DevOps, some of which are specialized for usage in certain portions of the pipeline, which also a drawback as it can lead to confusion and overchoice. Picking the tools that are ideal for a certain project, would require understanding the possibilities and limitations of each tool.

Some of the ways in which GitOps differ from DevOps has great benefits. While the strong tie to Git in some ways can be a detriment, it provides extra security and reliability as it keeps a complete history of changes containing the identity of the person that changed the infrastructure, when it was done and what was done with digital signing as proof. The rollback feature also adds reliability because it allows easy reverts to an earlier working state in case of a problem. It is also likely cheaper and easier to implement, as with DevOps, one of the challenges with its implementation is that there are too many tools to choose from and to learn to use, while most developers are already quite comfortable with Git, requiring most to only get comfortable with Kubernetes or another container orchestration platform.

GitOps can not exactly be considered an evolution of Devops as it is far narrower in scope, being stricter with requirements and tied down by specific tooling. The similarities between them is only due to the fact that GitOps embraces the same practices that are common in Devops, so it can be stated that GitOps is closer to a subset of DevOps, with improvements in certain aspects. It would not be far-fetched to consider it a step forward in some ways, as some of the differences add reliability and security, but it is not likely replace DevOps as the next evolution of it.

3.1 Reflections

DevOps is a concept that I have been familiar with for a while, but has always been hard to define. After having to read up on it from different sources it has

become clearer to me what DevOps is, but it is still very vaguely defined. It is a buzzword that is thrown around a lot, but because it is so loosely defined it is hard to know what falls under DevOps and what does not. It was also a bit difficult to find reliable information on GitOps, as it is relatively new and not as established as DevOps. There are not many articles about it, e.g. on IEEE and ACM there was only one article I could find, and there are very few that discuss or study its shortcomings and challenges in practice. I would have liked to source more information about GitOps from somewhere that is not Weaveworks that pioneered it, but the information was too sparse.

4 Conclusion

To conclude this essay, DevOps and GitOps share quite a few similarities due to GitOps embracing DevOps common best practices, but there are also some notable differences between them as GitOps is tied to specific tools that provide security and reliability and requires certain practices, whereas DevOps allows for more freedom of choice as the guidelines are more loosely defined, with DevOps best practices just being common and recommended. GitOps has some improvements in regards to reliability and security as the usage of Git provides transparency of what changes have been done, and by whom, and allows for rollback. It also is likely easier to implement into a lot of workplaces due to requiring less tools, with having to choose from, which also lowers the cost, comapared to DevOps. GitOps can not be considered to be the next evolution of DevOps, as it is far narrower in scope and more restrictive. It is closer to a subset of DevOps, rather than an evolution. Considering the improvements in certain regards, calling it a step forward would not be implausible, however.

References

[Beetz and Harrer, 2021] Beetz, F. and Harrer, S. (2021). Gitops: The evolution of devops? *IEEE Software*, pages 0–0.

[Díaz et al., 2018] Díaz, J., Almaraz, R., Pérez, J., and Garbajosa, J. (2018). Devops in practice: An exploratory case study. In *Proceedings of the 19th International Conference on Agile Software Development: Companion*, XP '18, New York, NY, USA. Association for Computing Machinery.

[Ebert et al., 2016] Ebert, C., Gallardo, G., Hernantes, J., and Serrano, N. (2016). Devops. *IEEE Software*, 33(3):94–100.

[GitLab, 2022] GitLab (2022). What is gitops? `https://about.gitlab.com/topics/gitops/`. [Online; accessed 3-May-2022].

[Gottesheim, 2015] Gottesheim, W. (2015). Challenges, benefits and best practices of performance focused devops. In *Proceedings of the 4th International Workshop on Large-Scale Testing*, LT '15, page 3, New York, NY, USA. Association for Computing Machinery.

[Khan et al., 2022] Khan, M. S., Khan, A. W., Khan, F., Khan, M. A., and Whangbo, T. K. (2022). Critical challenges to adopt devops culture in software organizations: A systematic review. *IEEE Access*, 10:14339–14349.

[Kubernetes, 2022] Kubernetes (2022). Nodes. `https://kubernetes.io/docs/concepts/architecture/nodes/`. [Online; accessed 3-May-2022].

[Leite et al., 2019] Leite, L., Rocha, C., Kon, F., Milojicic, D., and Meirelles, P. (2019). A survey of devops concepts and challenges. *ACM Comput. Surv.*, 52(6).

[Limoncelli, 2018] Limoncelli, T. A. (2018). Gitops: A path to more self-service it. *Commun. ACM*, 61(9):38–42.

[Tyson Matthew, 2022] Tyson Matthew (2022). How kubernetes works. `https://www.infoworld.com/article/3617008/how-kubernetes-works.html`. [Online; accessed 3-May-2022].

[Weaveworks Blog, 2022] Weaveworks Blog (2022). Devops: The next evolution - gitops. `https://www.weave.works/blog/devops-the-next-evolution-gitops`. [Online; accessed 3-May-2022].

[Weaveworks Limited, 2022] Weaveworks Limited (2022). Guide to gitops. `https://www.weave.works/technologies/gitops/`. [Online; accessed 3-May-2022].

[Zhu et al., 2016] Zhu, L., Bass, L., and Champlin-Scharff, G. (2016). Devops and its practices. *IEEE Software*, 33(3):32–34.