Strategies for Promoting Releases in GitOps Environments

Thomas Stadler, BSc
FH-Burgenland, Eisenstadt, Österreich

ABSTRACT:
From the principles of DevOps came GitOps, as a collection of principles and best practices for the operation of software systems. At the center, or to be precise, the source - as the single source of truth - stands thereby the Git Repository as the revision control system. The state of the managed system is completely defined declaratively as code. A GitOps controller takes care of the continuous reconcilation between the declared state and the current state. The promotion of new software releases between several environments shows itself to be an unresolved problem at present. Uniform standard practices, as well as necessary tools are missing in the open source ecosystem. This thesis has the goal, to address the promotion of releases in GitOps environments. Abstract models of deployment environments as well as promotion workflows will be defined. Based on these defined models, a standardized solution for the promotion of releases will be implemented according to the GitOps principles, and compared to existing solutions.

1 INTRODUCTION

Increasingly more organizations are adopting a DevOps culture to develop new applications and services at high velocity. After all, a culture that encourages shared responsibility, transparency and rapid feedback, helps narrow the gaps between teams and thus accelerate processes. GitOps was born out of the need for rapid innovation.

GitOps is a set of principles for operating and managing software systems. These principles are derived from modern software operations, but also have their roots in existing and widely adopted best practices. The primary four principles, which serve as principles for the desired state of a system managed by GitOps are the following:

- **Declarative**
  A system managed by GitOps must have its desired state expressed declaratively.
- **Versioned and Immutable**
  Desired state is stored in a way that enforces immutability, versioning and retains a complete version history.
- **Pulled Automatically**
  Software agents automatically pull the desired state declarations from the source.

- **Continuously Reconciled**
  Software agents continuously observe actual system state and attempt to apply the desired state.

(opengitops.dev, 2023b)

These principles are defined in OpenGitOps version 1.0.0. OpenGitOps is a Cloud Native Computing Foundation (CNCF) sandbox project under the GitOps Working Group. The GitOps Working Group is a working group under the CNCF App Delivery TAG, with the goal of establishing a clear vendor-neutral, principle-driven meaning of GitOps. This will provide a foundation for interoperability between tools, conformance and certification through enduring programs, documents and code (opengitops.dev, 2023a). It should be noted that the term GitOps is not exclusively limited to the principles defined in OpenGitOps, but goes beyond that.

GitOps as a practice for releasing software has several advantages, but like other solutions, GitOps also has several shortcomings. Currently, there are several unresolved problems with GitOps. On the one hand, these are problems that often were a problem in an organization already before GitOps was implemented, on the other hand, a GitOps strategy brings with it some limitations, which lead to new problems. Depending on the level of adoption of GitOps, and how strictly the principles are followed, some problems may or may not be an issue for an organization.

When considering GitOps at scale, adopters of GitOps and its current tools and practices soon realize the problem of modeling multiple environments and promoting releases between them. Current tools do not come with an out-of-the-box solution for describing multiple environments. It is to decide by the user how to model their multiple environments in a GitOps fashion. There have emerged a couple of models for describing environments like branch-per-environment, folder-per-environment, repo-per-environment, which all seem to have their individual upsides and downsides. For every approach, the process of promotion is different. Clear guidelines and best practices, as well as tools which implement them, are missing in the GitOps ecosystem.

2 RELATED WORK

Since the introduction of the term GitOps by Richardson (2017) at Weaveworks, there has been insufficient scientific literature written about the topic. The following paragraph highlights prior scientific research done on the subject "GitOps", which has mainly been targeted at evaluating the concept of GitOps in general.

Limoncelli (2018) draws attention to GitOps in an article, where the concept is brought to the reader in an easily digestible way. The benefits of adoption of GitOps are presented well, however there are not any possible problems discussed. Salecha (2023) discusses the definition of the term in the chapter "What Is GitOps?" of the book "Practical GitOps" (Salecha, 2023). Beetz und Harrer (2021) discuss the idea of GitOps as an evolution of DevOps. They conduct research on the definition of both terms. López-Viana et al. (2022) publish a conference paper about implementing GitOps in IoT Edge Computing to achieve Continuous Deployment. They present a proof of concept to check the feasibility of applying GitOps in IoT Edge Computing solutions. Ramadoni et al. (2021) present an analysis of declarative and pull-based deployment models following GitOps principles by using the tool ArgoCD. The research focuses on the advantages compared to push-based deployment models.

There has not been any scientific research conducted yet on the specific topic of modeling multiple deployment environments with GitOps and promoting releases between them. However, Kapelonis (2023) at Codefresh published a couple of blog posts in recent years, where some best practices are presented on the topic. Codefresh is the organization behind the Argo Project (argoproj.io, 2023), and therefore a major driving force in the GitOps ecosystem.

In "Stop Using Branches for Deploying to Different GitOps Environments" Kapelonis (2021) discusses the thought of modeling different GitOps environments by using Git branches. Kapelonis (2021) explains thoroughly why this approach is an anti pattern and should not be used (Kapelonis, 2021). In "How to Model Your Gitops Environments and Promote Releases between Them" Kapelonis (2022) shares a multitude of suggestions and best practices about modeling GitOps environments and promoting releases between them. Different environments are modeled by customizing (kustomize.io, 2023) configuration in separate files and folders or Git repositories. For promoting between environments, basic file copy operations are suggested. It is noted, that these simple file copy operations can easily be automated by an external system. Kapelonis (2022) suggests four categories of environment configuration. *The application version*, *Kubernetes specific settings*, *mostly static business settings*, and *non-static business settings*. While *the application version* and *non-static business settings* are promoted, *Kubernetes specific settings* and *mostly static business settings* are generally not promoted between environments (Kapelonis, 2022).

In 2021 the Argo Project (argoproj.io, 2023) has brought forward the tool "Argo-CD Autopilot" (argocd-autopilot, 2023) to help new GitOps adopters with structuring their Git Repsitories, and promoting applications between environments. This command line interface (CLI) tool, which includes some of the earlier presented best practices, helps with the initial bootstrap process for ArgoCD.

3 RESEARCH QUESTION

The goal of this thesis is to address the problem of promoting releases between different environments in GitOps.

Large organizations in particular usually have many non-production and production environments such as: dev, qa, staging-us, staging-eu, production-us, production-eu. Usually new releases are automatically rolled out in an environment such as qa - by means of a CI pipeline. Now the task is to promote new changes, which are brought about by a new release, into subsequent or other environments. Current GitOps tools do not have a simple answer to the question about what is the right approach for the process of promotion.

In particular, this thesis aims to explore existing strategies for the solution of the problem with existing tools. Furthermore, a prototype of a newly proposed strategy will be developed. Subsequently, the prototype will be tested in a laboratory experiment and compared to currently existing strategies.

To achieve the goal of the thesis, the following research questions were identified:

- How can the promotion of releases in GitOps environments be designed?

– What possibilities do existing tools offer for the promotion of releases with multiple deployment environments?
– How can deployment environments, as well as promotion workflows be modeled abstractly?
– How can abstract modeling be used to implement a standardized solution for promoting releases according to GitOps principles?

4 METHODOLOGY

In the beginning, existing literature on the topic, as well as the concrete research questions are discussed. Furthermore, various opinions and approaches of pioneering organizations in the GitOps community are analyzed.

Next, existing software solutions, which are available under open source licenses, will be evaluated for their appropriate use for the present research. The aim is to find out the extent to which the defined research questions can be addressed with existing solutions. Finally, a baseline is defined as state-of-the-art.

Furthermore, abstract models for deployment environments and promotion workflows will be defined. These serve as a basis for the design of the prototype in the next step.

As far as possible and reasonable, existing toolkits and best practices will be used for the prototype. It is desirable to solely extend the Kubernetes-API and the GitOps-Toolkit to integrate the prototype natively into the existing ecosystem around GitOps within the CNCF.

According to Houde und Hill (1997) a prototype is any representation of a design idea, regardless of the medium. A prototype is something that serves as a model or inspiration for later developments (Houde & Hill, 1997).

The developed prototype will be compared to existing approaches to solving the problem in a laboratory experiment.

Montgomery (2017) defines an experiment as a test or series of runs in which intentional changes are made to the input variables, in order to then observe and identify reasons for changes in the output result (Montgomery, 2017).

5 RESULTS

The results of the literature review so far show, that there is no standard practice for the promotion of releases between multiple environments in GitOps. The existing GitOps tools do not provide a native solutions for this. As a best practice, it is suggested that the promotion operations should be outsourced to the CI system - in the form of simple file copy operations. According to the literature as well as current observations, any kind of standardization is missing, when it comes to defining multiple environments with GitOps and promoting releases between them.

As the primary outcome of this thesis, an answer to the research questions is expected, by proposing a software solution as a prototype. The developed prototype should ideally serve as a modular extension to Kubernetes and existing GitOps tooling.

In the end, the results of the research, including the developed prototype, can be donated to the CNCF.

6 CONCLUSION AND FUTURE WORK

Although the adoption of GitOps brings some advantages on the one hand, on the other hand, it creates new problems that did not exist before, or were better solved with previously followed strategies. In addition, with GitOps the demands of good software have increased - as so often - and new problems have been identified as a result. These newly identified problems may currently turn out to be merely luxury problems for many organizations. However, as an organization grows, these current luxury problems with the increasing growth of an organization and the increasing demands that come with it, may emerge as pressing problems.

The higher the complexity and the needed degree of automation of an organization, as well as the adoption of DevOps and the principles like Continuous Integration (CI), Continuous Delivery (CD), Continuous Deployment (CDP) - the more important it becomes to fully automate the promotion of releases in GitOps environments.

References

argocd-autopilot. (2023). Argo-CD Autopilot [(Accessed on 01/01/2023)]. https://argocd-autopilot.readthedocs.io/en/stable/

argoproj.io. (2023). Get More Done with Kubernetes [(Accessed on 01/01/2023)]. https://https://argoproj.io/

Beetz, F., & Harrer, S. (2021). GitOps: The Evolution of DevOps? *IEEE Software*.

Houde, S., & Hill, C. (1997). Chapter 16 - What do Prototypes Prototype? In M. G. Helander, T. K. Landauer & P. V. Prabhu (Hrsg.), *Handbook of Human-Computer Interaction (Second Edition)* (Second Edition, S. 367–381). North-Holland. https://doi.org/https://doi.org/10.1016/B978-044481862-1.50082-0

Kapelonis, K. (2022). How to Model Your Gitops Environments and Promote Releases between Them [(Accessed on 01/01/2023)]. https://codefresh.io/blog/how-to-model-your-gitops-environments-and-promote-releases-between-them/

Kapelonis, K. (2023). Kostis Kapelonis [(Accessed on 01/01/2023)]. https://codefresh.io/meet-a-codefresher/kostis-kapelonis/

Kapelonis, K. (2021). Stop Using Branches for Deploying to Different GitOps Environments [(Accessed on 01/01/2023)]. https://codefresh.io/blog/stop-using-branches-deploying-different-gitops-environments/

kustomize.io. (2023). Kubernetes native configuration management [(Accessed on 01/01/2023)]. https://kustomize.io/

Limoncelli, T. A. (2018). GitOps: A Path to More Self-Service IT. *Commun. ACM*, *61*(9), 38–42. https://doi.org/10.1145/3233241

López-Viana, R., Díaz, J., & Pérez, J. E. (2022). Continuous Deployment in IoT Edge Computing : A GitOps implementation. *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–6. https://doi.org/10.23919/CISTI54924.2022.9820108

Montgomery, D. C. (2017). *Design and analysis of experiments*. John wiley & sons.

opengitops.dev. (2023a). GitOps Documents v1.0.0 [(Accessed on 01/01/2023)]. https://github.com/open-gitops/documents/blob/v1.0.0/README.md

opengitops.dev. (2023b). GitOps Principles v1.0.0 [(Accessed on 01/01/2023)]. https://github.com/open-gitops/documents/blob/v1.0.0/PRINCIPLES.md

Ramadoni, Utami, E., & Fatta, H. A. (2021). Analysis on the Use of Declarative and Pull-based Deployment Models on GitOps Using Argo CD. *2021 4th International Conference on Information and Communications Technology (ICOIACT)*, 186–191. https://doi.org/10.1109/ICOIACT53268.2021.9563984

Richardson, A. (2017). GitOps - Operations by Pull Request [(Accessed on 01/01/2023)]. https://www.weave.works/blog/gitops-operations-by-pull-request

Salecha, R. (2023). What Is GitOps? In *Practical GitOps: Infrastructure Management Using Terraform, AWS, and GitHub Actions* (S. 1–30). Apress. https://doi.org/10.1007/978-1-4842-8673-9_1