Strategies for Promoting Releases in GitOps Environments

Thomas Stadler, BSc
FH-Burgenland, Eisenstadt, Österreich

ABSTRACT:
Git is becoming the new home for information technology (IT) operations. Following a core concept of DevOps - reducing friction between engineering teams within the software development lifecycle (SDLC) - a new practice has emerged, which leverages the revision control system Git for IT operations. GitOps is a set of principles for operating and managing software systems. The desired state of the managed system is - in its entirety - defined declaratively as code, which is continuously reconciled with the actual state by a controller. 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 aims at addressing the promotion of releases in GitOps environments. Abstract models of deployment environments as well as promotion workflows will be designed. Based on these models, a standardized solution for the promotion of releases will be developed, in the form of a Kubernetes operator, according to the GitOps principles. Finally, the research is evaluated by comparing the proposed prototype to the research objectives. The results of this research will address the given problem by providing a vendor-neutral solution for modeling environments and promoting releases between them in a "GitOps"-native approach.

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 to narrow the gaps between teams and thus accelerate the development process. In order to reduce friction between engineering teams who are involved in the software development lifecycle (SDLC), a new practice called GitOps has emerged. It allows developers who are already familiar with the revision control system Git, to easily deploy their applications to target environments in a self-service model. System administrators and operators can also manage IT infrastructure purely by interfacing with declarative state definitions stored in Git.

GitOps as a practice for releasing software has many advantages, but like other solutions, GitOps also has some shortcomings. One of the unresolved problems is the process of promoting releases between multiple deployment environments (illustrated in Figure 1.1).
Current GitOps tools do not provide an integrated solution for this process, nor do they provide any sort of abstraction for defining environments. Users currently need to rely on separation of duties on a file and folder level within the Git repository, for modeling different environments. Promotions are

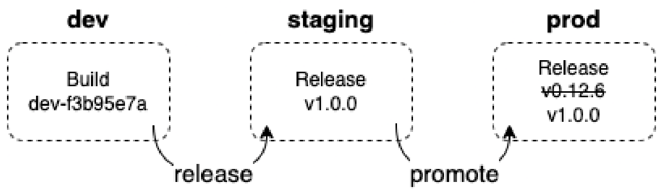

Figure 1.1: release promotion process.

often achieved via hard-coded file copy operations, which is done manually or by a Continuous Integration (CI)/Continuous Delivery (CD) system. Due to the nature of Git, a certain flow of promotion can not currently be enforced (e.g. QA –> Staging –> Production). Furthermore, for each configuration or templating tool which is used (e.g. kustomize, helm, jsonnet, etc.), the modeling of different deployment environments, as well as the process of promotion, is unique. This results in the process of promoting releases with GitOps not being a streamlined task. Clear guidelines and best practices, as well as tools which implement them, are missing in the GitOps ecosystem.

The given problem could be addressed by providing standardised models for defining deployment environments and promotion processes. An application programming interface (API) extension for Kubernetes could provide custom resource definitions for these models. This would allow users to define abstract representations of their environments, and how they want releases to be promoted between them. Additional logic could be introduced into the promotion process, like specifying a rule which ensures that new releases must first pass certain environments or other objectives before being promoted to production. The abstraction would also enable transparent replacement of the configuration or templating tool, while keeping the desired state definition intact. Following the principles of GitOps, an operator would ensure the continuous reconciliation between the desired and the actual state of the resources.

The proposed solution of the problem should present a possible way of defining environments and promotion processes abstractly, onto which future work could build upon. Additionally the solution should provide a protoype of a toolkit, which could serve as an optional component in addition to existing tooling within the Cloud Native Computing Foundation (CNCF). Solving the problem of release promotion natively within the GitOps toolkit, would make the adoption of GitOps more appealing, especially for organisations, which have the need for many different environments. As a result this could generally accelerate the widespread use of GitOps and thus enable more organisations to develop higher quality software.

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 author clearly highlights the benefits of adoption of GitOps, however does not mention any negative aspects. 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 Internet of Things (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; and focuses on the advantages compared to push-based deployment models.

Hitherto, there have not been any pulications in academic journals or conferences 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.

Kapelonis (2021) discusses the idea of modeling different deployment environments by using Git branches. He explains thoroughly why this approach is an anti-pattern and should not be used (Kapelonis, 2021). Kapelonis (2022) shares a multitude of suggestions and best practices about modeling 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, like a CI/CD 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) presented the "Argo-CD Autopilot"-tool (argocd-autopilot, 2023) to help new GitOps adopters with structuring their Git repositories, 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.

Prior research on the concrete problem is focused on presenting best practices and suggestions which users need to manually implement themselves. In addition it is suggested to let an external CI/CD system handle the promotion process. Conversely, this thesis will bring forward abstract models of environments and promotion processes, which are implemented in the proposed prototype tooling. The prototype will assess the feasibility of defining deployment environments and promotion processes declaratively, following the GitOps principles.

3 RESEARCH QUESTION

Large organizations in particular typically have many non-production and production environments such as: Development (Dev), Quality Assurance (QA), Staging-US, Staging-EU, Production-US, Production-EU. Usually new releases are automatically deployed to an environment, such as QA, by a CI/CD system. 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.

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

- RQ 1: How can the promotion of releases in GitOps environments be designed?
  - RQ 1.1: What possibilities do existing tools offer for the promotion of releases with multiple deployment environments?
  - RQ 1.2: How can deployment environments, as well as promotion processes be modeled abstractly?
  - RQ 1.3: How can abstract modeling be used to implement a standardized solution for promoting releases?

4 METHODOLOGY

To achieve the main goal of the thesis and answer the identified RQs, a mix of different scientific methods will be used. In order to help with the recognition and legitimization of the conducted research, a commonly accepted framework, namely the methodology for conducting design science (DS) research in information systems (IS) (Peffers et al., 2007) will be applied. It consists of six activities (illustrated in Figure 4.1): Identify Problem & Motivate (activity 1), Define Objectives of a Solution (activity 2), Design & Development (activity 3), Demonstration (activity 4), Evaluation (activity 5), and Communication (activity 6). The process is structured in a nominally sequential order. For this concrete research, the problem-centered initiation is chosen as the entry point, thus the process will begin with activity 1. Afterwards the research will proceed sequentially, because the idea of the research resulted from observation of the problem (Peffers et al., 2007).

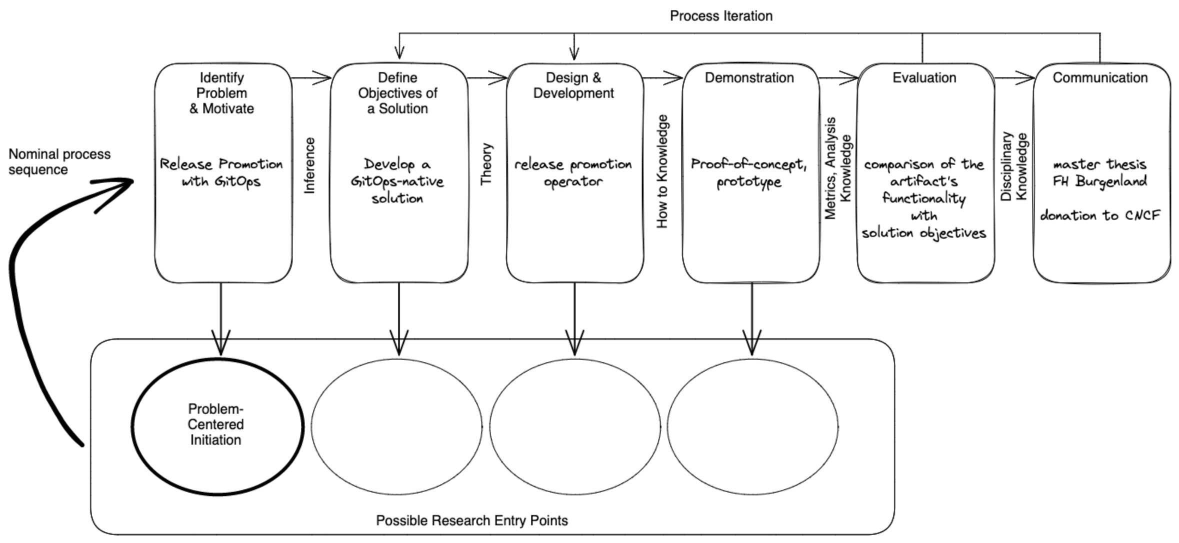
Process Iteration
Identify Problem & Motivate
Define Objectives of a Solution
Design & Development
Demonstration
Evaluation
Communication
Nominal process sequence
Release Promotion with GitOps
Inference
Develop a GitOps-native solution
Theory
release promotion operator
How to Knowledge
Proof-of-concept, prototype
Metrics, Analysis Knowledge
comparison of the artifact's functionality with solution objectives
Disciplinary Knowledge
master thesis FH Burgenland
donation to CNCF
Problem-Centered Initiation
Possible Research Entry Points

Figure 4.1: DSRM Process for this thesis.

In activity 1, the research problem of release promotion with GitOps is defined. This is accomplished, by seeking knowledge of the state of the problem from practicing professionals, as well as analysing prior written literature. To assist later evaluation, the problem is conceptually broken down into distinct items. The value of a solution is highlighted, in order to help the audience of the research understand the reasoning associated with the researcher's understanding of the problem (Peffers et al., 2007).

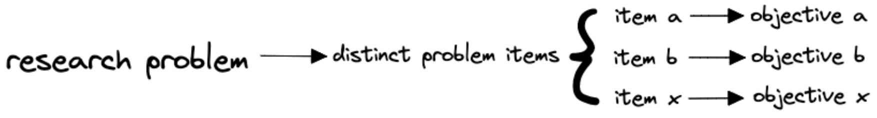

Figure 4.2: Inference of objectives from problems.

In activity 2, research objectives are inferred from the problem definition in activity 1. Each objective maps to a distinct item from the problem specification (illustrated in Figure 4.2), which helps with later evaluation in activity 5. In practice, a research objective will be a qualitative description of how a new artifact is expected to support a solution to the problem definition.

In activity 3, solutions for the previously defined objectives are designed and developed by means of producing an artifact. This is achieved by determining the artifact's desired functionality and its architecture, followed by actually creating the artifact (Peffers et al., 2007). In practice this means that: Abstract model definitions of deployment environments, as well as promotion processes are designed; With the specification of the model definitions in place, the "release promotion operator" is developed as an artifact.

In activity 4, the in-context use of the artifact is demonstrated. After outlining how to use the artifact to effectively provide a solution to the problem definition, the artifact is implemented in a proof-of-concept-level prototype.

In activity 5, the implementation of the artifact, and how well it supports a solution to the problem, is evaluated. This is achieved by comparing the objectives of a solution to actual observed results from use of the artifact in the demonstration (Peffers et al., 2007). In practice this means, that the functionality of the artifact implemented in the prototype in activity 4, is compared with the solution objectives from activity 2.

In activity 6, as a final step, the whole conducted research is communicated by means of disclosing the problem and its importance, the artifact and its utility and novelty, and the demonstration accompanied by the evalution results (Peffers et al., 2007), within the publication of a master thesis at the University of Applied Sciences Burgenland (FH Burgenland). In addition, it may be communicated to relevant audiences such as practicing professionals; as well as the CNCF.

5 RESULTS

The overall goal of the thesis is to provide a solution to the problem of release promotion in GitOps environments. In this regard a prototype will be developed and its functionality demonstrated in a proof-of-concept evaluation. The goal for the prototype is to implement it as a modular extension to Kubernetes and existing GitOps tooling. By doing so, users can promote releases between multiple deployment environments following the GitOps principles. The main contribution of this thesis is a proof of concept for the developed prototype which can be used as an extension to the GitOps toolkit within the CNCF.

6 CONCLUSION AND FUTURE WORK

The currently available GitOps tools do not provide an integrated solution to the problem of promoting releases between environments. This thesis aims at addressing the given problem. This is achieved by clearly defining the problem in distinct items, from which research objectives are then inferred. Following in designing and developing abstract models for environments and promotion processes; which are implemented in the produced artifact "release promotion operator". The artifact is demonstrated in a prototype serving as a proof-of-concept. Finally the research is evaluated by means of comparing the artifact's functionality with the solution objectives.

The developed "release promotion operator" prototype should serve as a modular extension to Kubernetes and existing GitOps tooling within the CNCF. It should provide a more streamlined and GitOps-native approach to the process of release promotion. For subsequent research projects the use of the prototype could be observed in a case study or tested in a field experiment, and adapted in another iteration of the applied methodology process model. The main goal for future work is the extensive adaption of the prototype to enable use in production by organizations or other projects.

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*.

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

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*, 45–77.

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