



# // GITOPS: HANDS-ON CONTINUOUS OPERATIONS WITH KUBERNETES

Johannes Schnatterer, Cloudogu GmbH

 @jschnatterer

Version: 202111181137-aa3096d



# Agenda

- What is GitOps?
- How can it be used?
- What challenges arise?
- Demo

The background of the slide is a dark gray surface covered with numerous 3D question marks. Most of these question marks are black and appear to be recessed into the surface. Three question marks are highlighted in a bright orange color and stand out as if they are raised from the surface. One orange question mark is located in the upper right, another in the middle left, and a larger one in the lower center. The text 'What is GitOps?' is written in a clean, white, sans-serif font on the left side of the slide.

# What is GitOps?

(Operating) model Pattern Way

Approach (good) practice

methodology Philosophy Technique Framework

Standardized Workflow Principle Cloud-  
native continuous delivery

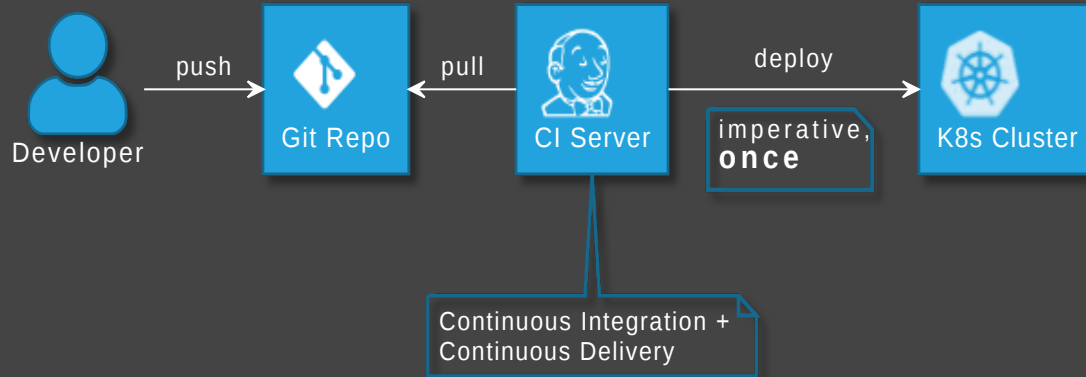


Origin: blog post by Weaveworks, August 2017

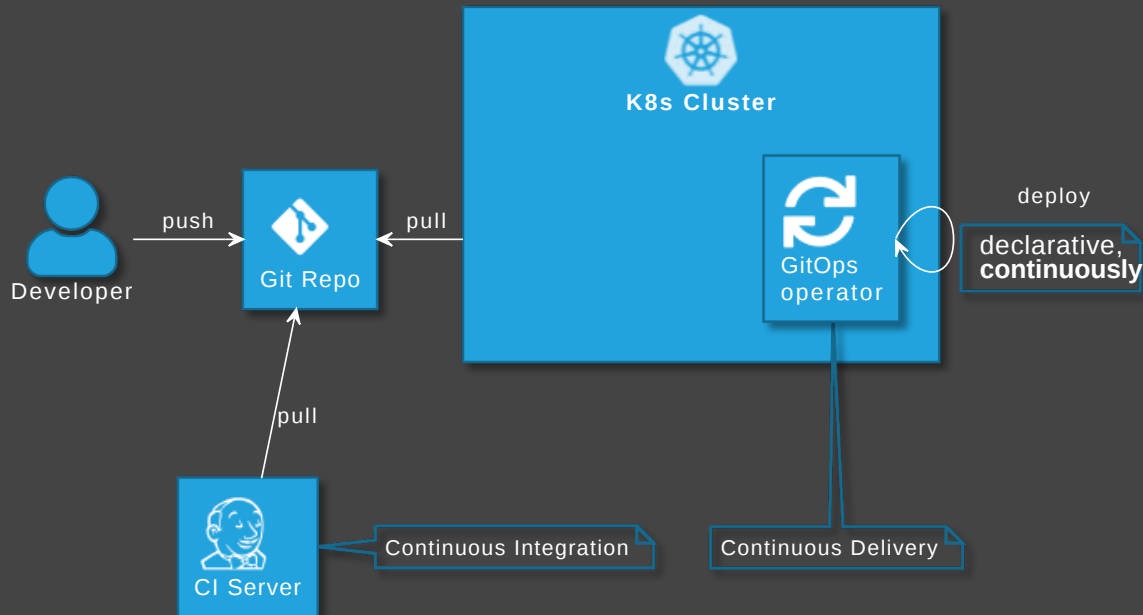
Use developer tooling to drive operations

 [weave.works/blog/gitops-operations-by-pull-request](https://weave.works/blog/gitops-operations-by-pull-request)

# "Classic" Continuous Delivery ("CICDs")



## GitOps



# GitOps Principles



- 1 The principle of declarative desired state
- 2 The principle of immutable desired state versions
- 3 The principle of continuous state reconciliation
- 4 The principle of operations through declaration

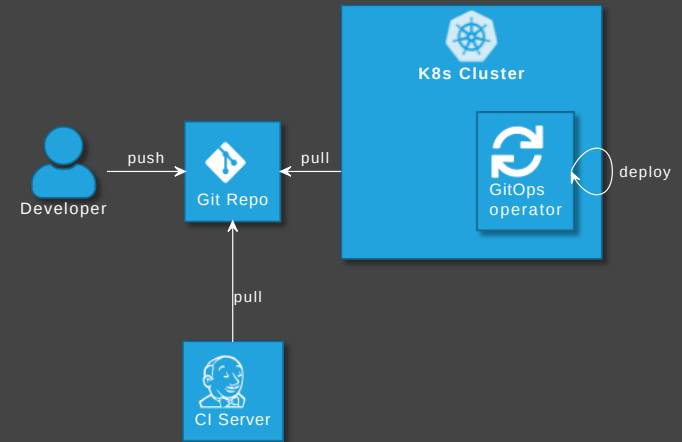
 [github.com/open-gitops/documents/blob/main/PRINCIPLES.md](https://github.com/open-gitops/documents/blob/main/PRINCIPLES.md)

# GitOps vs DevOps

- DevOps is about collaboration of formerly separate groups (mindset)
- GitOps focuses on ops (operating model)
- GitOps can be used with or without DevOps

# Advantages of GitOps

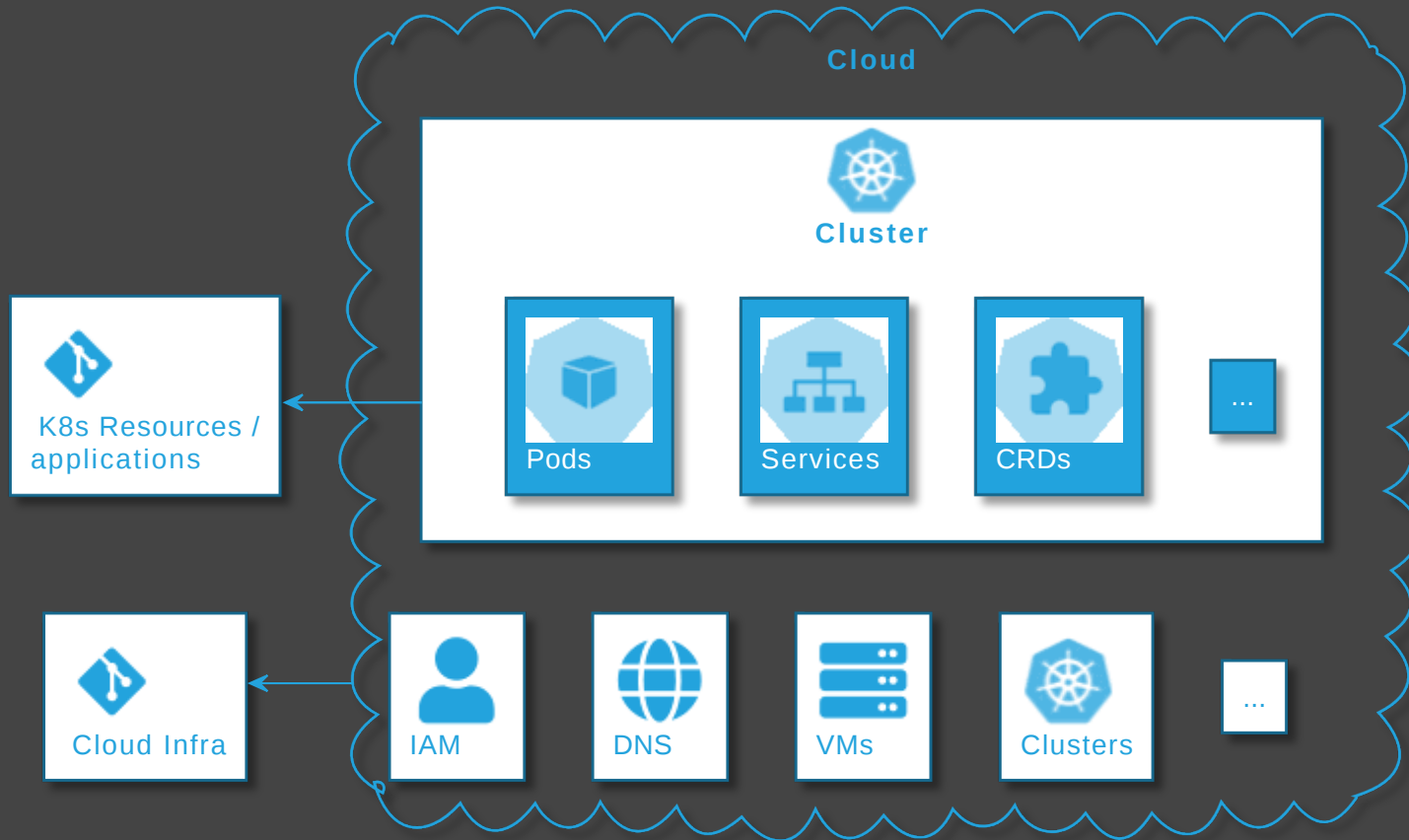
- No access to cluster from outside (might also solve firewall/zone issues)
- No credentials on CI server (neither cluster access nor for apps)
- Forces declarative description
- IaC is auditable
- Scalability - one repo many applications
- Self-healing / Hands-off ops





# How can GitOps be used?

# What can GitOps be used for?



# GitOps tool categories

- GitOps operators/controllers
- Supplementary GitOps tools
- Tools for operating k8s clusters + cloud infra with GitOps






# GitOps operators/controllers



# Supplementary GitOps tools

## Secrets

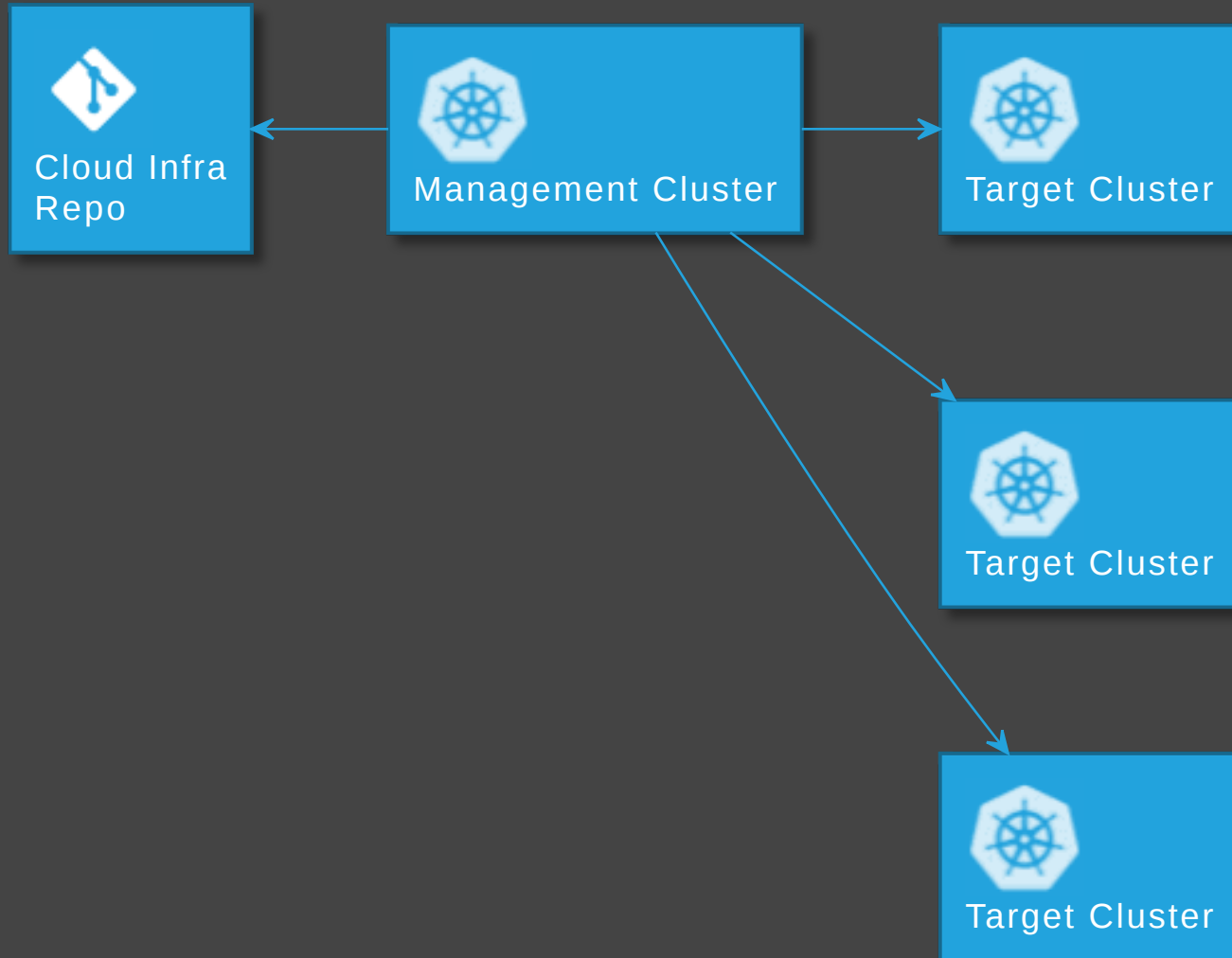
-  [bitnami-labs/sealed-secrets](#)
-  [Solutio/kamus](#)
-  [mozilla/sops](#) + K8s integration
- Operators for Key Management Systems

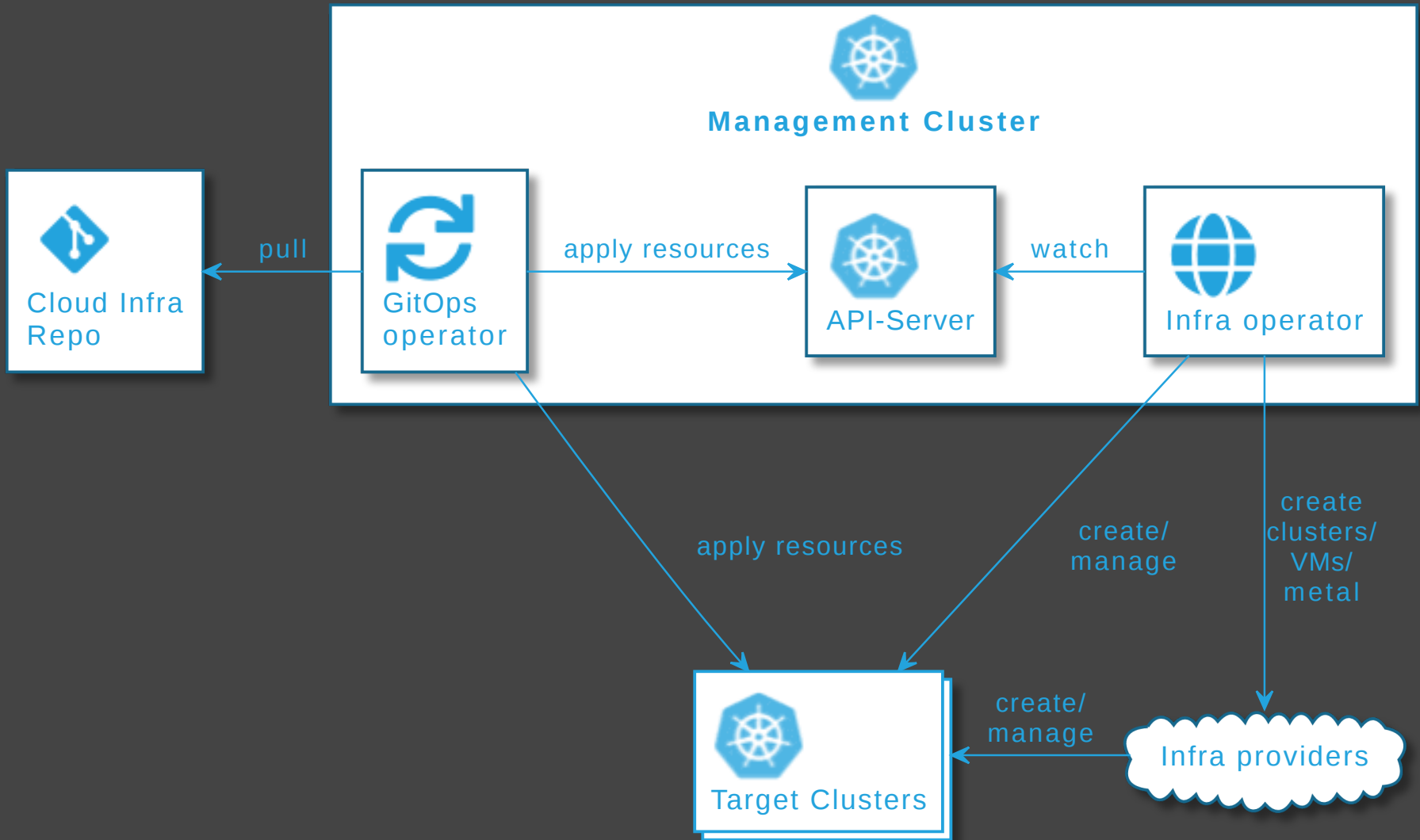
## Others

- Backup / **restore**
- ~~Horizontal Pod Autoscaler~~  
 [argo-cd.readthedocs.io/en/release-2.0/user-guide/best\\_practices](https://argo-cd.readthedocs.io/en/release-2.0/user-guide/best_practices)
- Deployment Strategies - Progressive Delivery  
 
- ...

 **GitOps loves operators**

# Operate Kubernetes with Kubernetes





# Tools for operating k8s clusters + cloud infra



+



Cloud or Operator

- 
-  [rancher/terraform-controller](#)
- 

## See also

 [clouddogu.com/blog/gitops-tools](https://clouddogu.com/blog/gitops-tools) (iX 4/2021)

- General tool comparison,
- tips on criteria for tool selection,
- comparison of ArgoCD v1 and Flux v2

# What challenges arise with GitOps?





## More Infra ...

- GitOps Operator: One or more custom controllers
- Helm, Kustomize Controllers
- Operators for Supplementary tools (secrets, etc.)
- Monitoring/Alerting systems
- ...

## ... higher cost

- Maintenance/patching (vendor lock-in)
- Resource consumption
- Learning curve
- Error handling
  - failing late and silently
  - monitoring/alerting required
  - reason might be difficult to pinpoint
  - operators cause alerts (OOM errors, on Git/API server down, etc.)

# Day two questions

- POC is simple
- Operations in prod has its challenges
  - How to realize local dev env?
  - How to delete resources?
  - How to realize staging?
  - How to structure repos and how many of them?
  - Role of CI server?
  - ...

# Local development

- Option 1: Deploy GitOps operator and Git server on local cluster  
➡ complicated
- Option 2: Just carry on without GitOps.  
Easy, when IaC remains in app repo 🤔

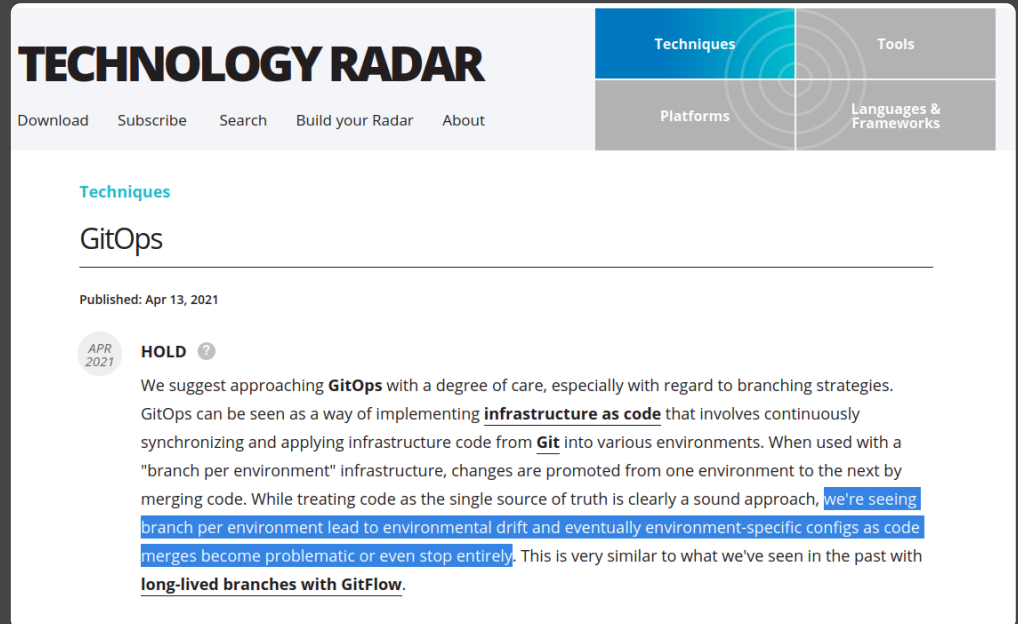
# How to delete resources?

- `garbage collection` (Flux) / `resource pruning` (ArgoCD) disabled by default
- 📌 Enable from beginning ➡ avoid manual interaction
- Unfortunately, still often unreliable / too defensive (?) 😞

# Implementing stages

## Idea 1: Staging Branches

- Develop → Staging
- Main → Production



The screenshot shows the 'TECHNOLOGY RADAR' website. The top navigation bar includes 'Download', 'Subscribe', 'Search', 'Build your Radar', and 'About'. The main content area is titled 'Techniques' and features a 'GitOps' article. The article is marked with a 'HOLD' status and a date of 'APR 2021'. The text discusses the challenges of implementing GitOps, particularly regarding branching strategies and environmental drift.

**TECHNOLOGY RADAR**

Download Subscribe Search Build your Radar About

Techniques Tools Platforms Languages & Frameworks

Techniques

### GitOps

Published: Apr 13, 2021

**HOLD** ?

APR 2021

We suggest approaching **GitOps** with a degree of care, especially with regard to branching strategies. GitOps can be seen as a way of implementing **infrastructure as code** that involves continuously synchronizing and applying infrastructure code from **Git** into various environments. When used with a "branch per environment" infrastructure, changes are promoted from one environment to the next by merging code. While treating code as the single source of truth is clearly a sound approach, we're seeing branch per environment lead to environmental drift and eventually environment-specific configs as code merges become problematic or even stop entirely. This is very similar to what we've seen in the past with long-lived branches with GitFlow.

 [thoughtworks.com/radar/techniques/gitops](https://thoughtworks.com/radar/techniques/gitops)



Logic for branching complicated and error prone (merges)

## Idea 2: Staging folders

- On the same branch: One folder per stage

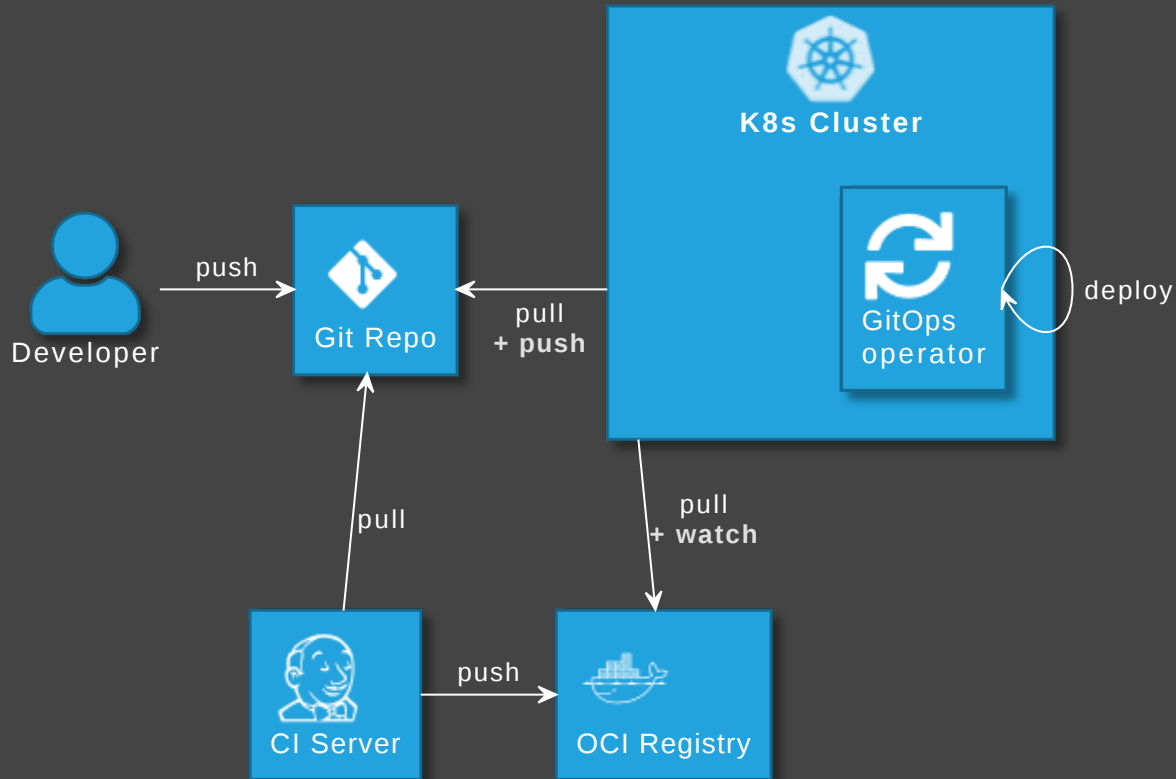
```
├── production
│   ├── application
│   └── deployment.yaml
└── staging
    ├── application
    └── deployment.yaml
```

- Process:
  - commit to staging folder only (📌 protect prod),
  - create short lived branches and pull requests for prod
- Duplication is tedious, but can be automatized



- Logic for branching simpler
- Supports arbitrary number of stages

# Basic role of CI server

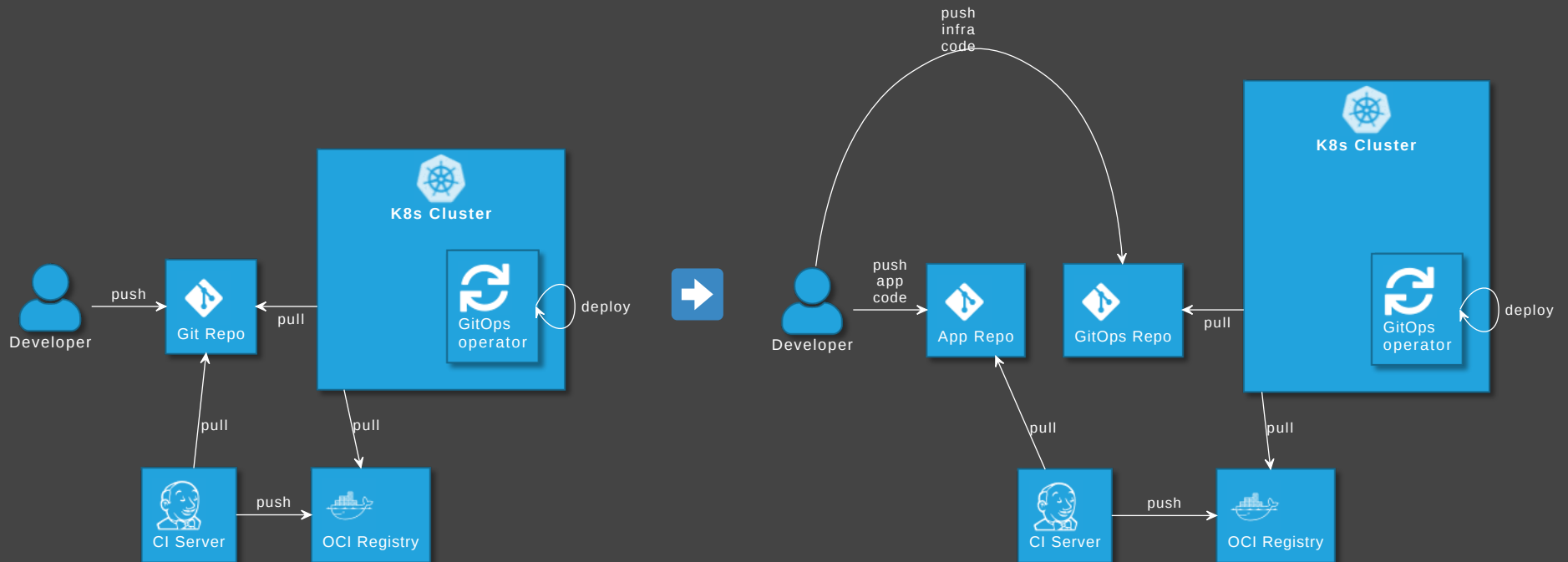


📌 Optional: GitOps operator updates image version in Git

- 🧑 [github.com/argoproj-labs/argocd-image-updater](https://github.com/argoproj-labs/argocd-image-updater)
- 📄 [fluxcd.io/docs/guides/image-update](https://fluxcd.io/docs/guides/image-update)



# Number of repositories: application vs GitOps repo



GitOps tools: Put infra in separate repo! See



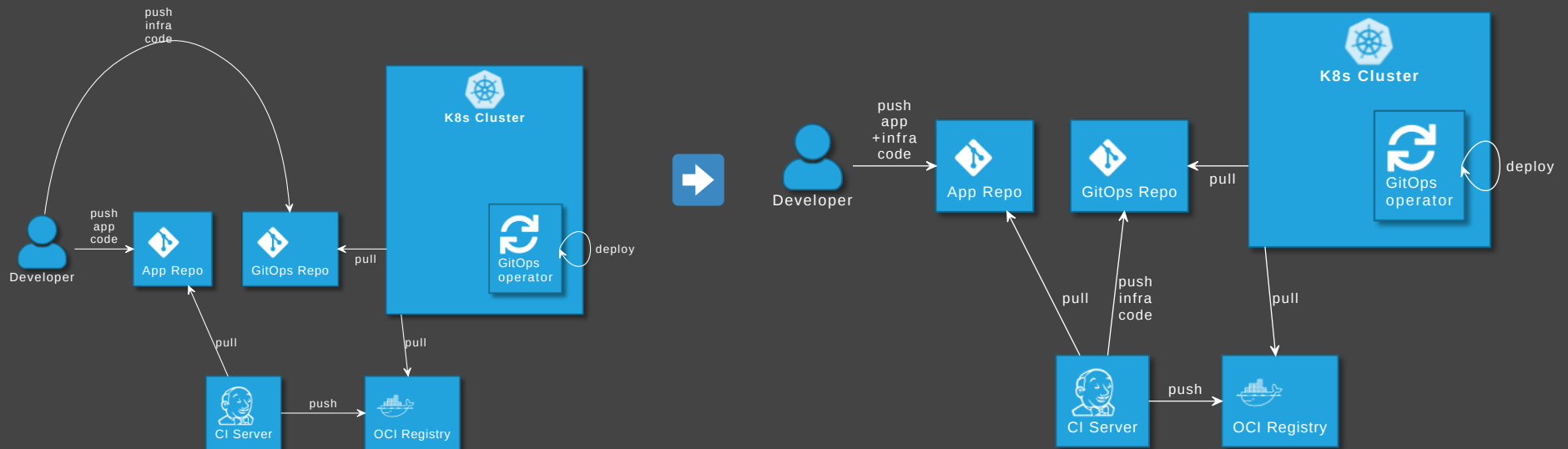
[argocd.readthedocs.io/en/release-2.0/user-guide/best\\_practices](https://argocd.readthedocs.io/en/release-2.0/user-guide/best_practices)

## Disadvantages

- Separated maintenance & versioning of app and infra code
- Review spans across multiple repos
- Local dev more difficult
- Static code analysis for IaC code not possible

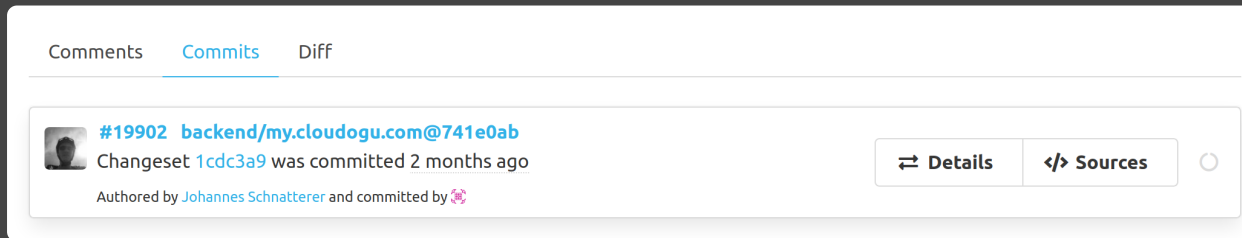
# How to avoid those?

# Extended role of CI server



# Advantages

- Single repo for development: higher efficiency
- Automated staging (e.g. PR creation, namespaces)
- Shift left: static code analysis + policy check on CI server, e.g. yamllint, kubeval, helm lint, conftest
- Simplify review by adding info to PRs

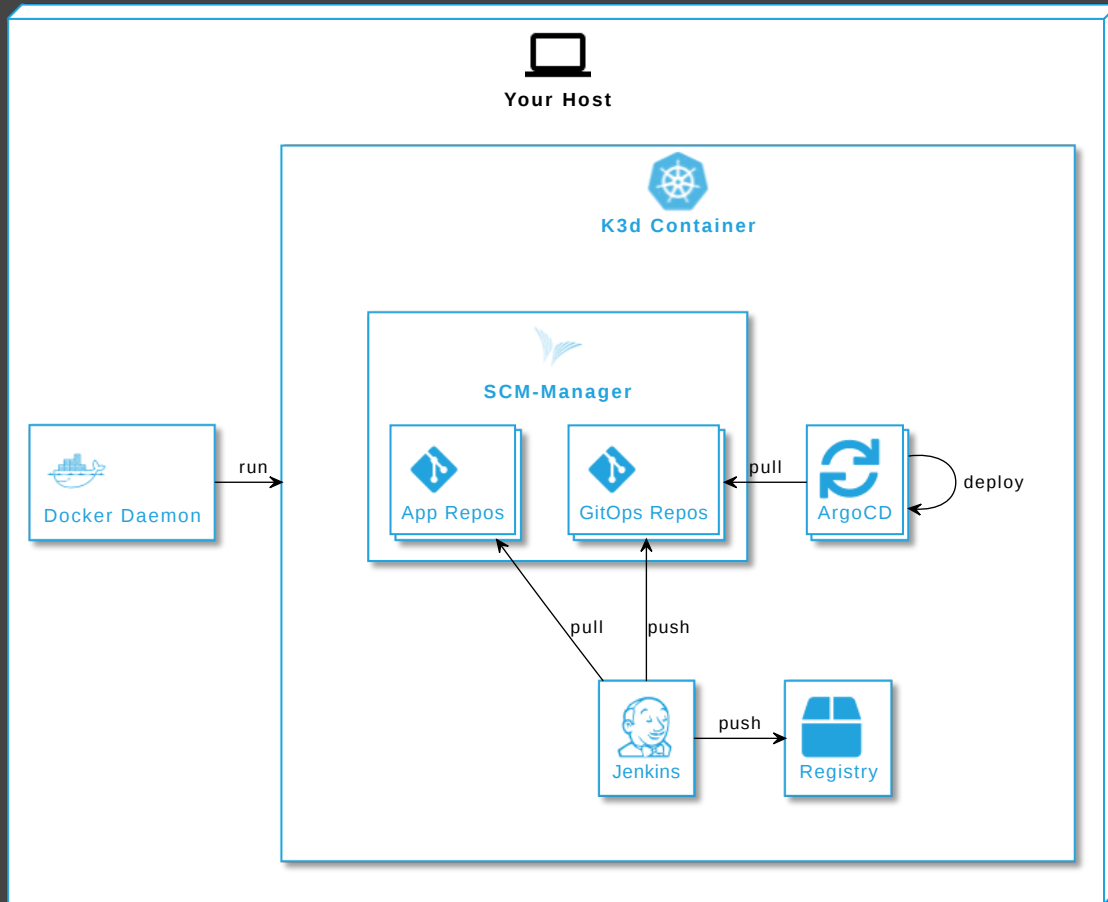


Disadvantage: Complexity in CI pipelines

➡ Recommendation: Use a plugin or library, e.g.

 [cloudogu/gitops-build-lib](https://github.com/cloudogu/gitops-build-lib) 

# Demo



# CONCLUSION

A hand with light-colored nail polish is holding a red marker, drawing a thick red underline beneath the word 'CONCLUSION'. The hand is positioned in the lower right quadrant of the image.

# GitOps experience distilled

- + Has advantages, once established
- Mileage for getting there may vary






# Adopt GitOps?

- Greenfield: Definitely
- Brownfield: Depends



# Johannes Schnatterer, Cloudogu GmbH

 [cloudogu.com/gitops](https://cloudogu.com/gitops)

-  GitOps Resources (intro, our articles, etc.)
-  Links to GitOps Playground and Build Lib
-  Discussions
-  Trainings / Consulting
-  Jobs



Slides



# Image sources

- What is GitOps? <https://pixabay.com/illustrations/question-mark-important-sign-1872665/>
- How can GitOps be used? Tools: <https://pixabay.com/photos/tools-knives-wrenches-drills-1845426/>
- What challenges arise with GitOps?  
[https://unsplash.com/photos/bJhT\\_8nbUA0](https://unsplash.com/photos/bJhT_8nbUA0)