



# // GITOPS - IS THIS SOMETHING FOR ME?

Johannes Schnatterer, Cloudogu GmbH

 @jschnatterer

Version: 202211111412-9d3c206



# Agenda

- Basics
- Tools
- Challenges
- Demo

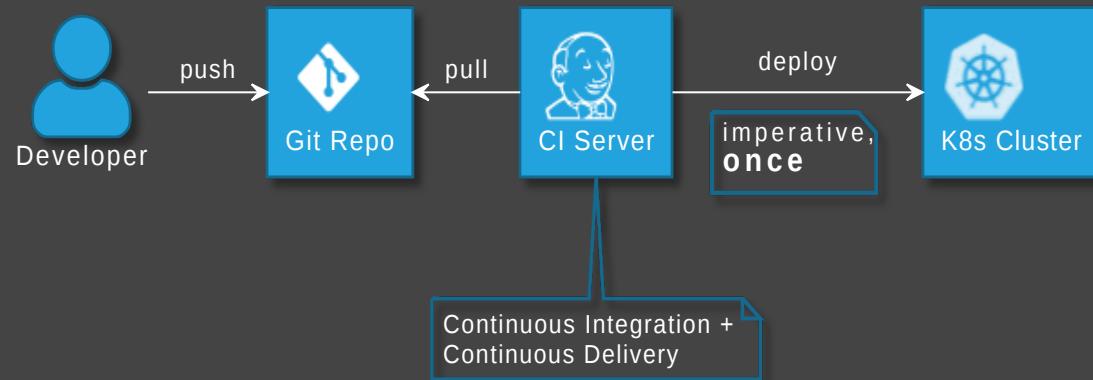
# GitOps basics

Origin: blog post by Weaveworks, August 2017

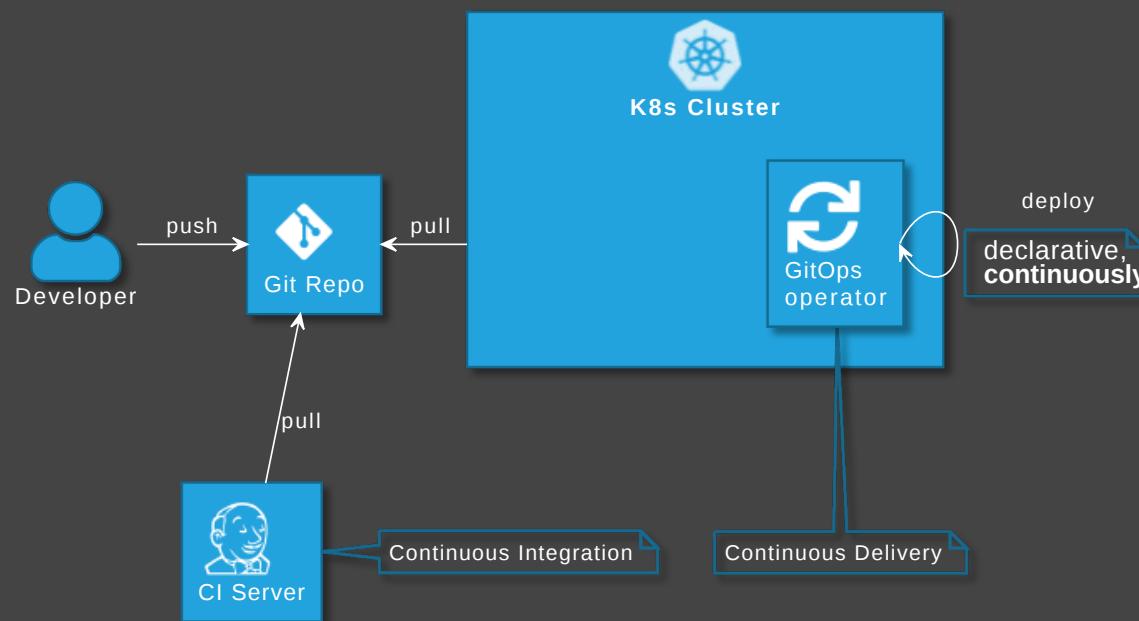
Use developer tooling to drive operations

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

# "Classic" Continuous Delivery ("CIOps")



# GitOps



# GitOps Principles

The desired state of a GitOps managed system must be:

- 1 Declarative**
- 2 Versioned and Immutable**
- 3 Pulled Automatically**
- 4 Continuously Reconciled**



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

# GitOps vs DevOps

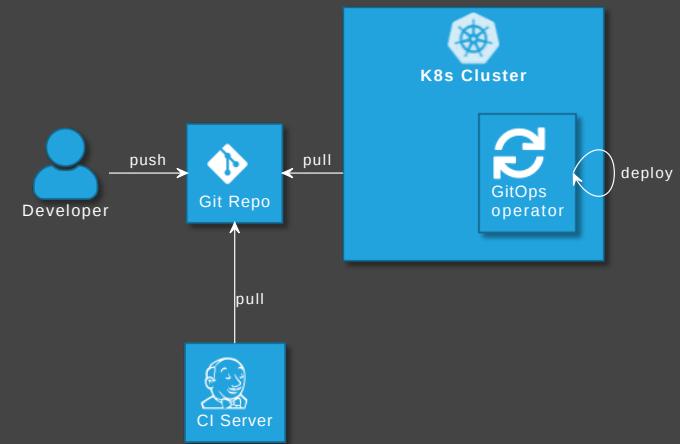
- DevOps is about collaboration of formerly separate groups (mindset)
- GitOps focuses on ops (operating model)
- GitOps could be used with or without DevOps and vice versa
- Still, GitOps might be...

The right way to do DevOps

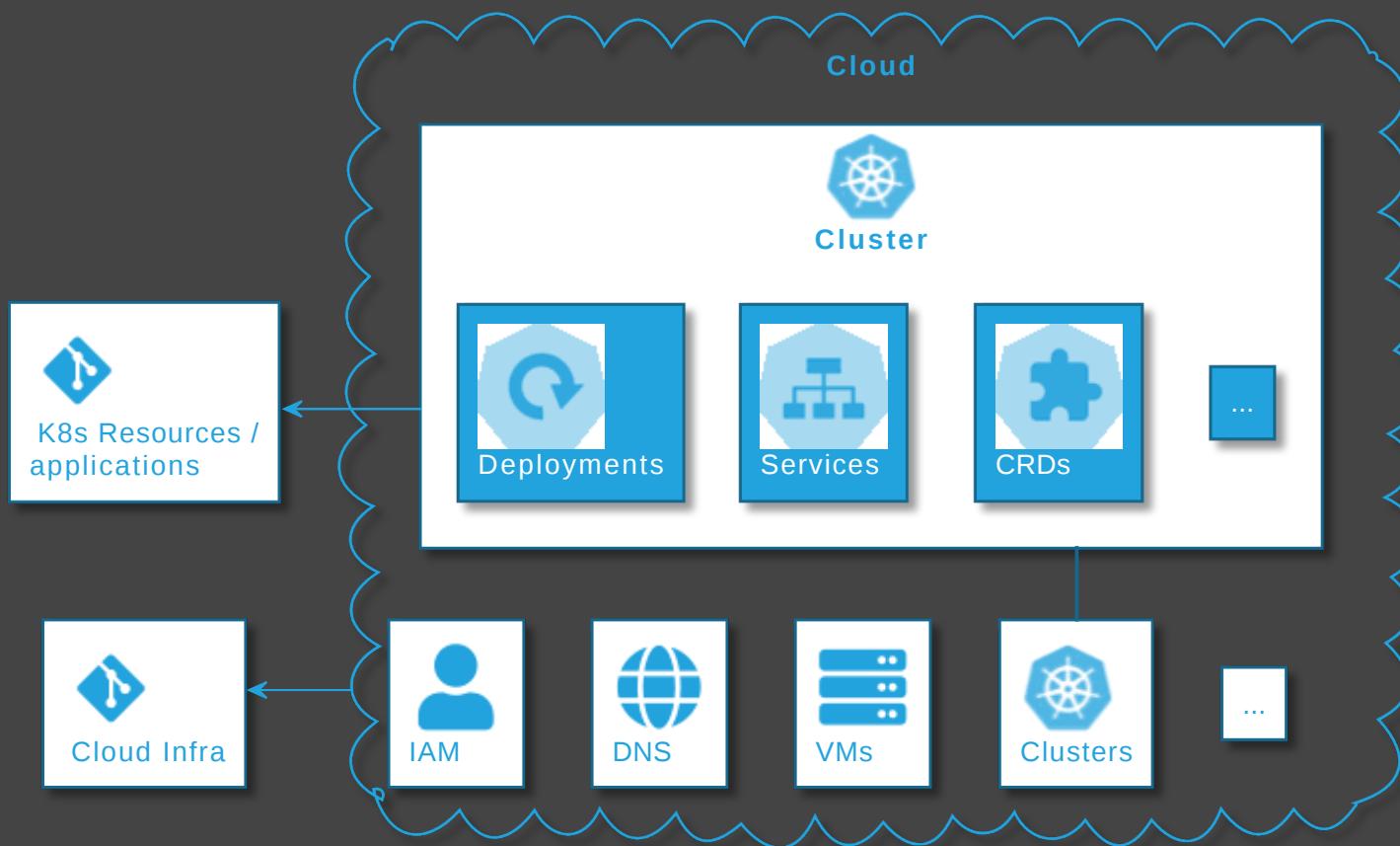
 Alexis Richardson

# 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



# What can GitOps be used for?





# GitOps tools

# GitOps tool categories

- GitOps operators/controllers
- Supplementary GitOps tools
- Tools for operating cloud infra

# GitOps operators/controllers



# Supplementary GitOps tools

- Secrets
  - KMS, e.g.    ...
  - + K8s Integration
    - Operator
    - Container Storage Interface (CSI) driver
    - Side car (injector)
    - Helm/Kustomize plugin
    - GitOps Operator: native support or plugin
- Backup / **restore**
- Deployment Strategies - Progressive Delivery



- ...



GitOps ❤ operators

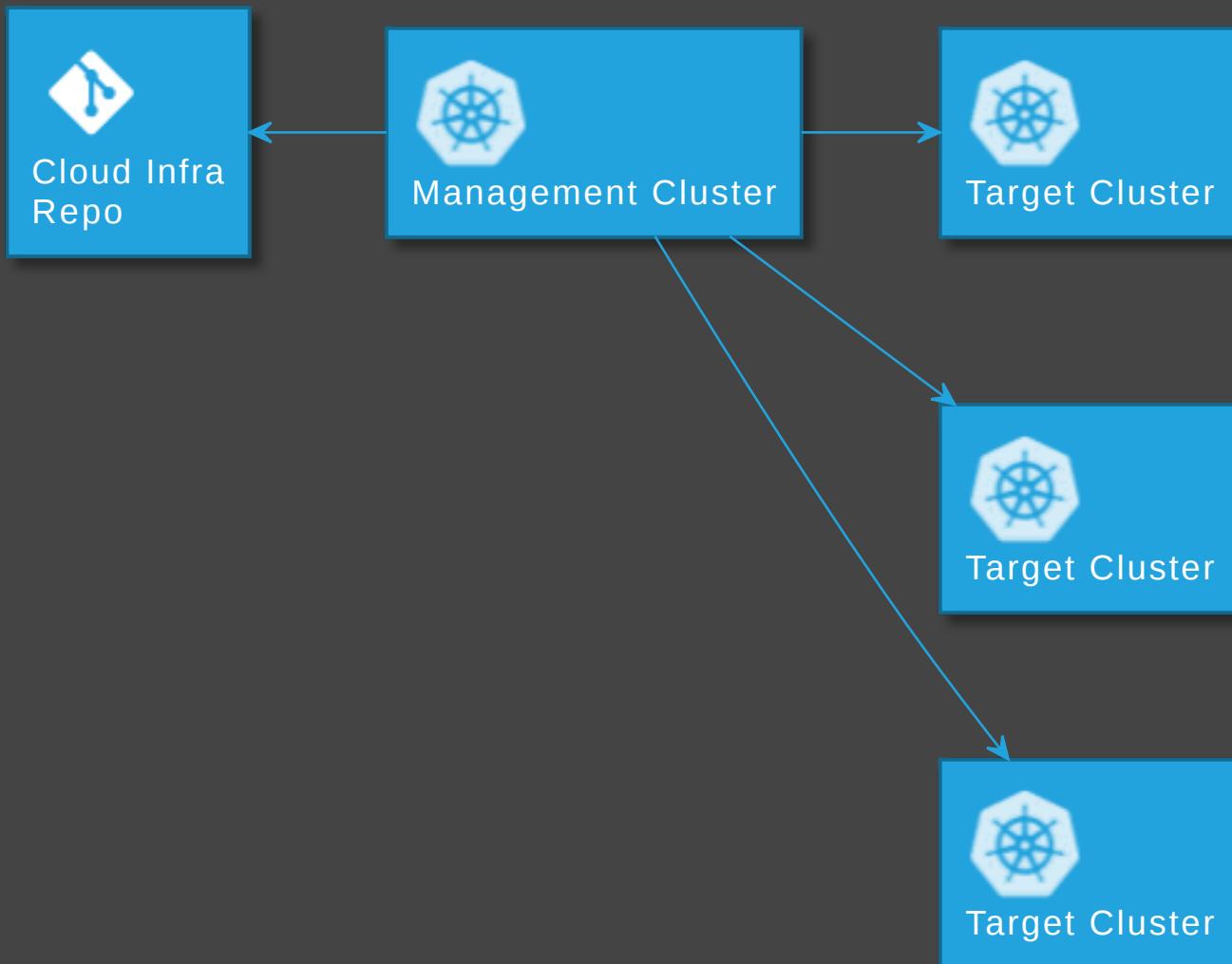
+

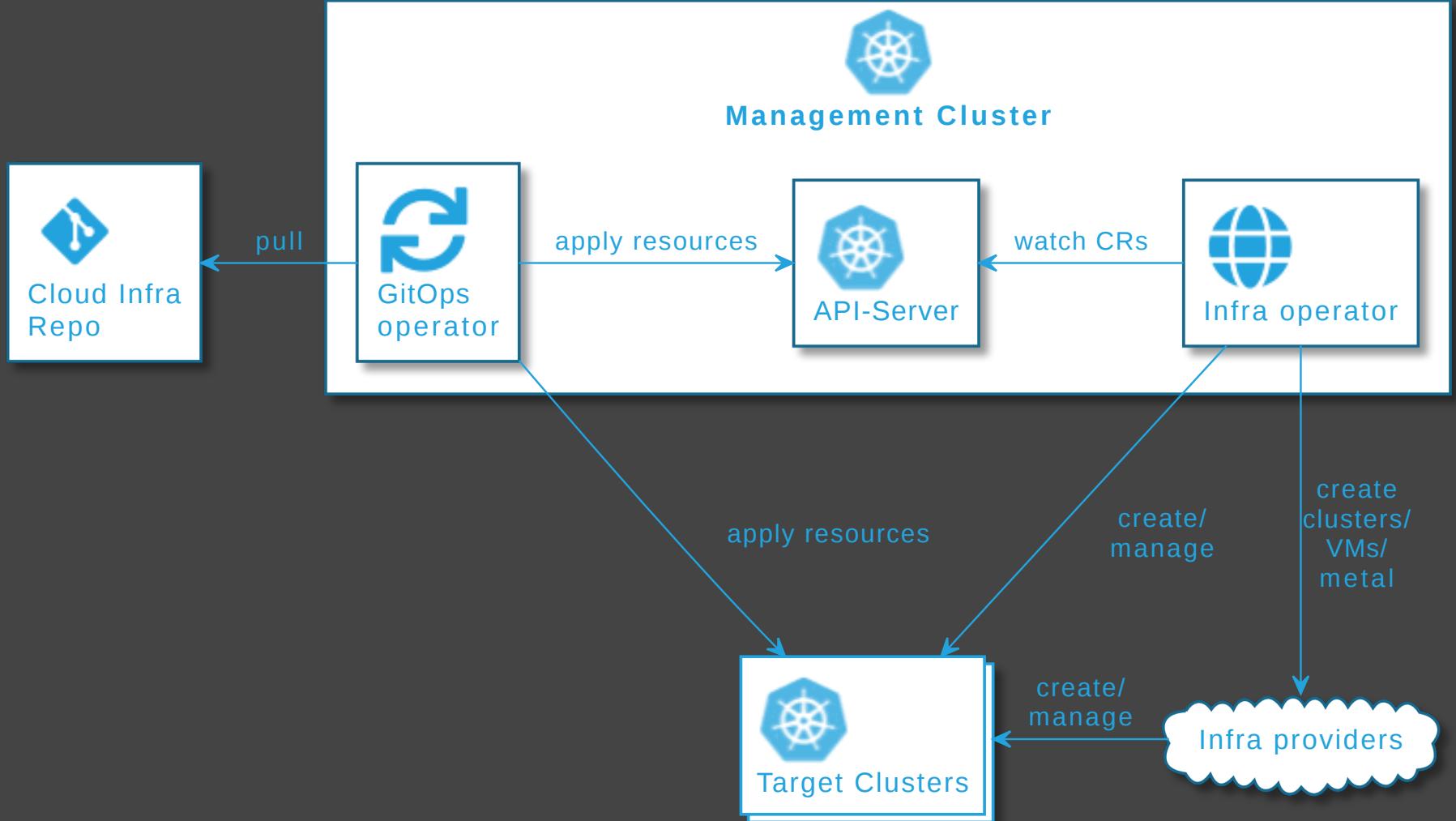
Infra Operator

=

Operate cloud infra with GitOps

# Operate Kubernetes with Kubernetes





# Tools for operating cloud infra



## See also

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

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

# Challenges with GitOps

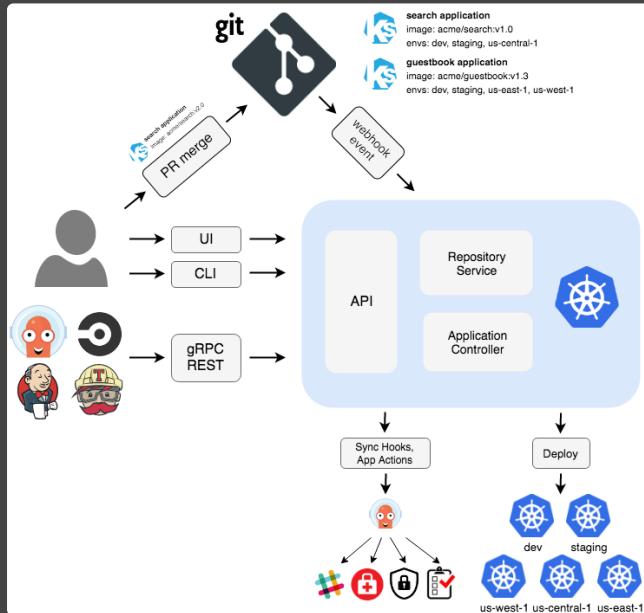


# Downsides

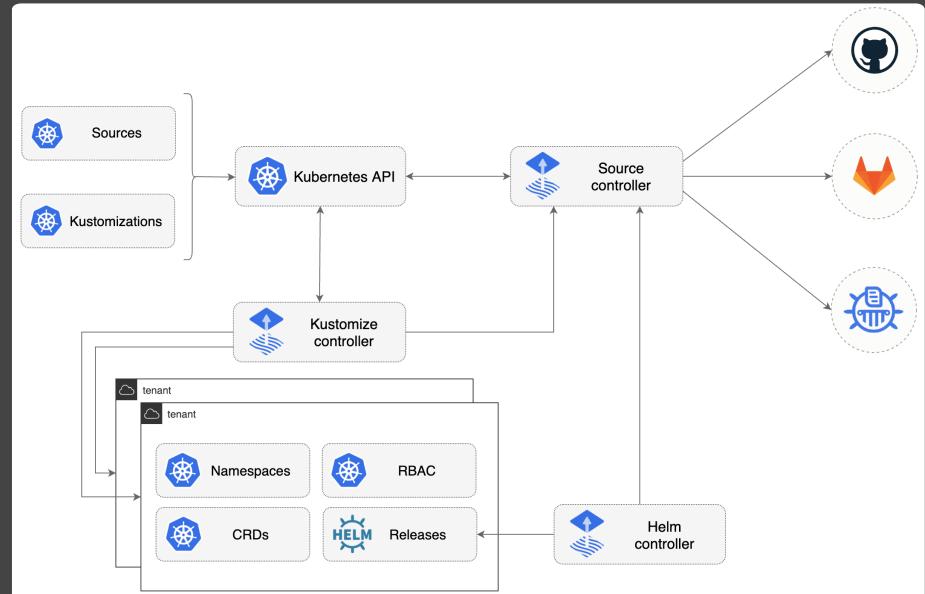
- More infra necessary
- Steep Learning curve

# GitOps infra

- GitOps Operator comprises several applications;
- Cause ops efforts: maintenance, alerts



[argo-cd.readthedocs.io/en/stable/assets/argocd\\_architecture.png](https://argo-cd.readthedocs.io/en/stable/assets/argocd_architecture.png)



[fluxcd.io/img/diagrams/gitops-toolkit.png](https://fluxcd.io/img/diagrams/gitops-toolkit.png)

## Learning curve

- New concepts and tools for developers and platform teams
- Adapt deployment process
- Migrate applications
- Adapt error handling and alerting
  - avoid failing late and silently
  - accustom to new notification mechanism
  - still, reason might be difficult to pinpoint

## Day two questions

- How to realize local dev env?
- How to delete resources?
- How to structure repos and folders?
- How to realize staging?
- 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.

## 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 (?) 😞

## Repo and folder structure

- No standard for structures (intentionally) → Conway's law
- Repo patterns: Monorepo vs Polyrepo (per app, team, stage) 🏢
- Within repo: folder/~~branch~~ structure for stage, team, app
- More options:
  - Topology: GitOps controller (s) ↔ Cluster(s) / Namespaces
  - GitOps controller-specific config

# 💩 GitOps Chasm

## 🌐 🐙 🚧 Infra

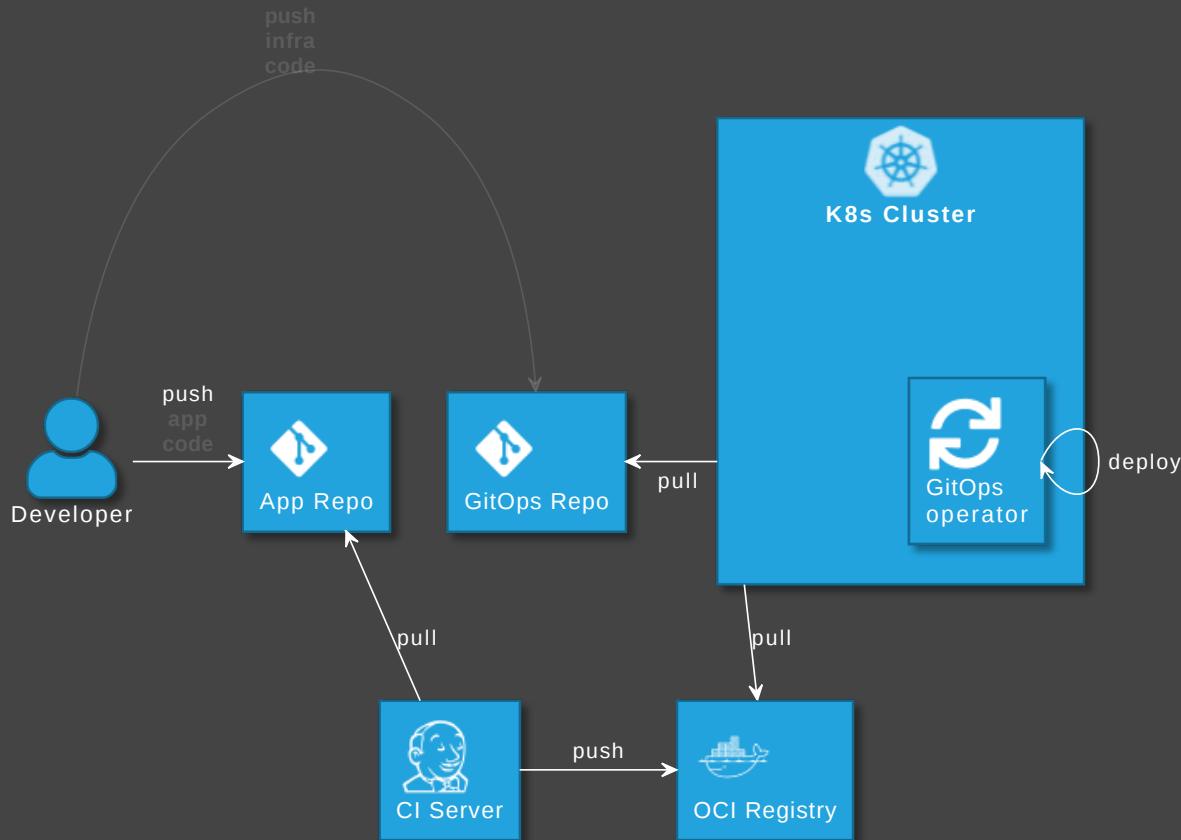
- repos
- folders
- branches
- clusters
- namespaces
- controller instances
- controller-specific config

↔  
Mapping?  
🤔

## 🌐 Real-world

- company/departments
- teams
- projects
- applications
- microservices
- stages/environments
- customers
- tenants
- etc.

# App vs GitOps repo



GitOps tools: Put infra in separate repo! See



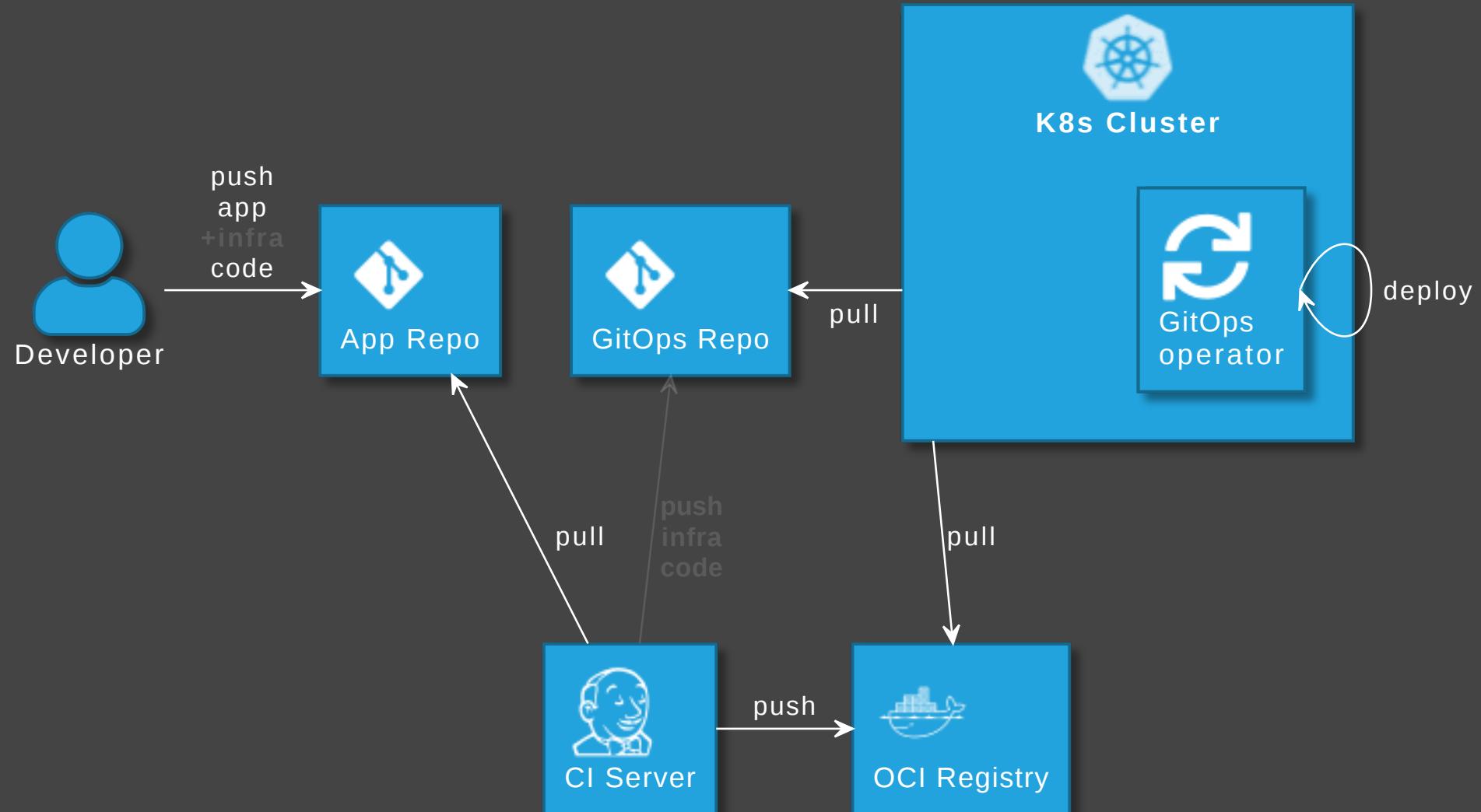
[argo-cd.readthedocs.io/en/release-2.5/user-guide/best\\_practices](https://argo-cd.readthedocs.io/en/release-2.5/user-guide/best_practices)

## 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

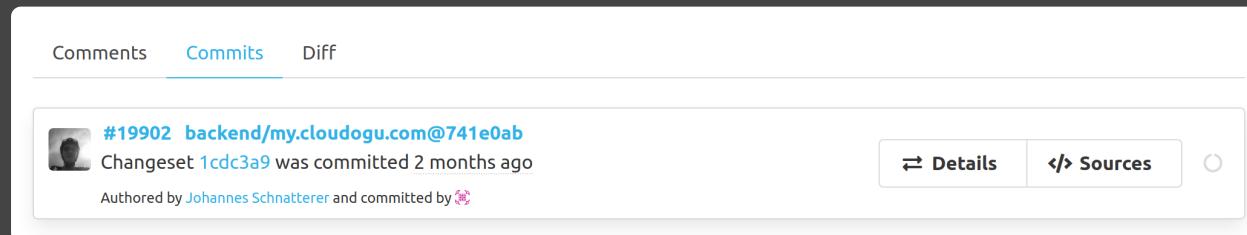
# How to avoid those?

# Using CI-Server with GitOps part 1



## Advantages

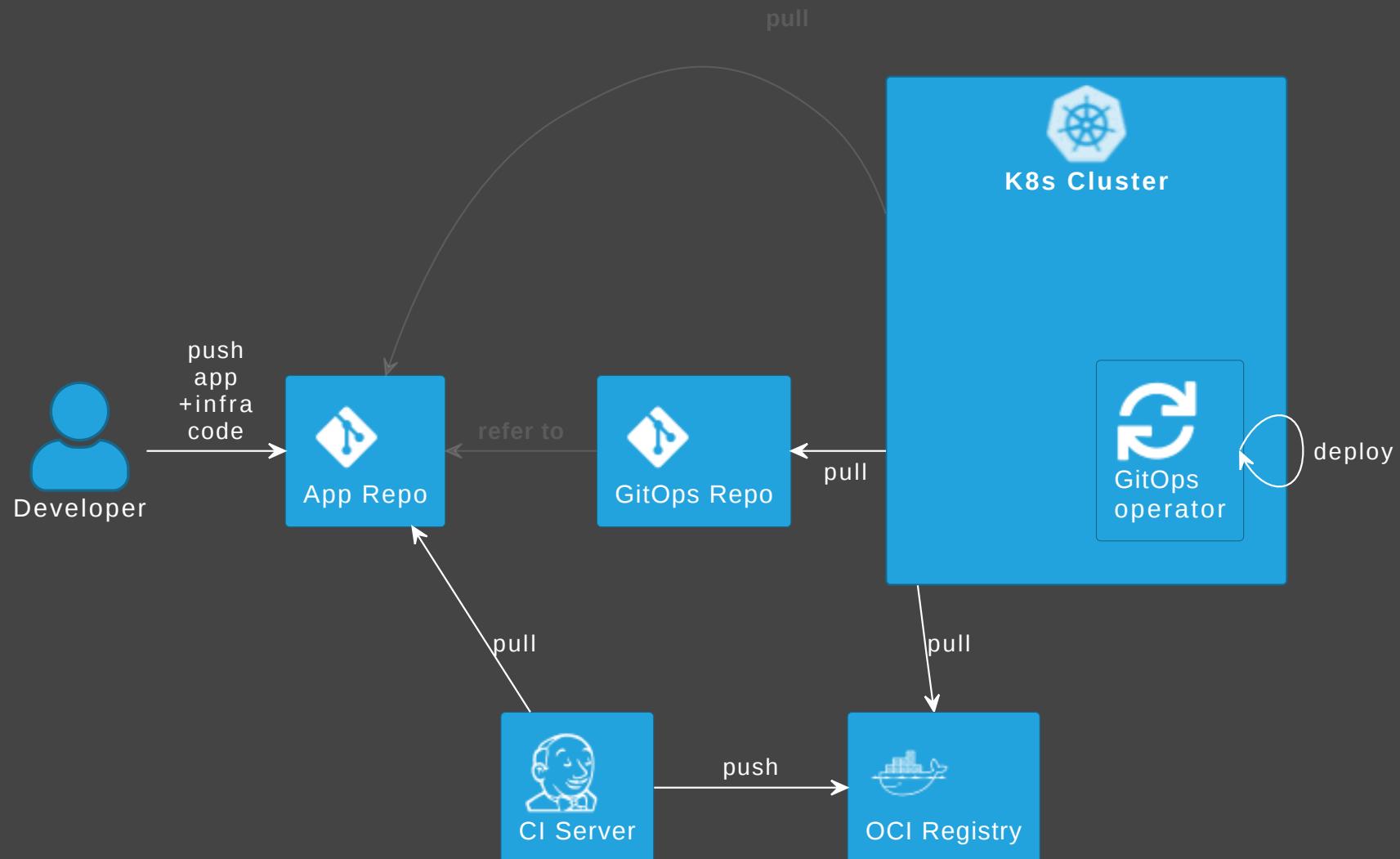
- Single repo for development: higher efficiency
- Shift left: static code analysis + policy check on CI server, e.g. yamlint, kubeval, helm lint, conftest
- Automated staging (e.g. PR creation, namespaces)
- Simplify review by adding info to PRs



## Disadvantages

- Complexity in CI pipelines
  - ➡ Recommendation: Use a plugin or library, e.g.  [cloudogu/gitops-build-lib](#) 
- Redundant code

## Alternative: Refer to app repo

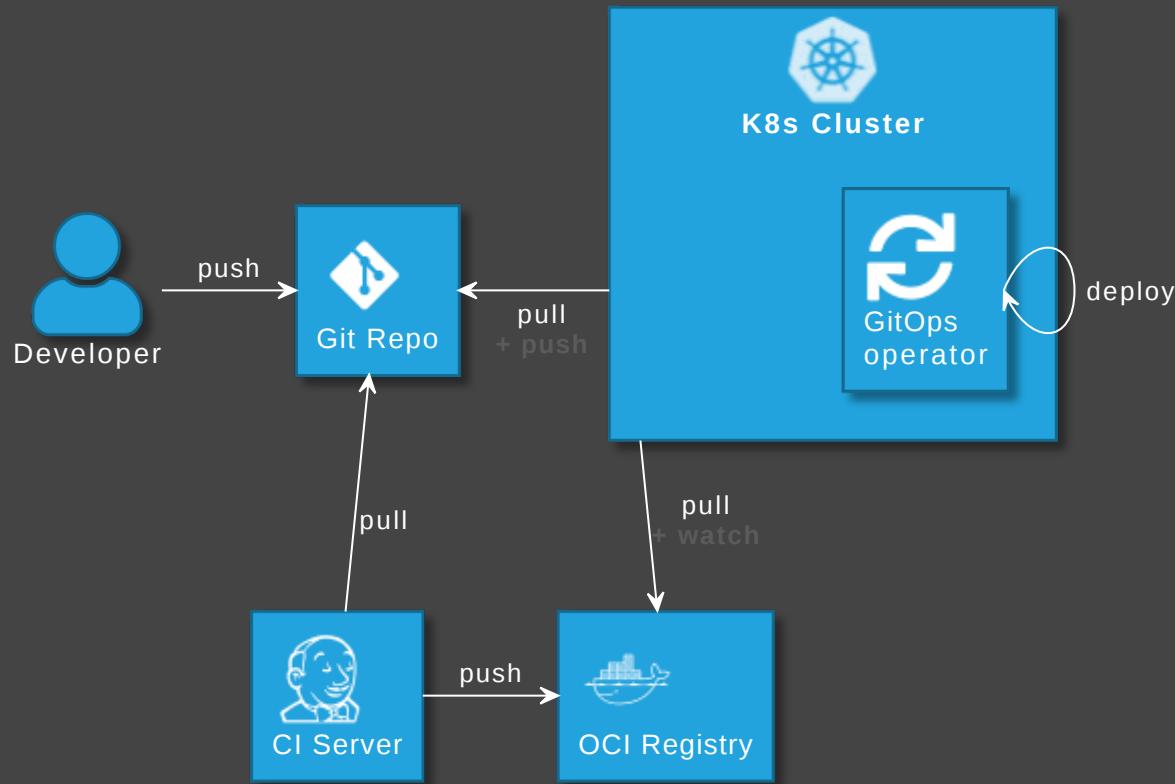


## Stage promotion

How to update bump version in GitOps repo and promote through stages?

- **Manual:** Push to short-lived branches, create PRs manually
- **Image Updater:** Operator pushes branches, create PRs manually
- **CI Server:** Build job pushes branches, creates PRs
- **Renovate Bot:** Bot watches registry pushes branches, create PRs

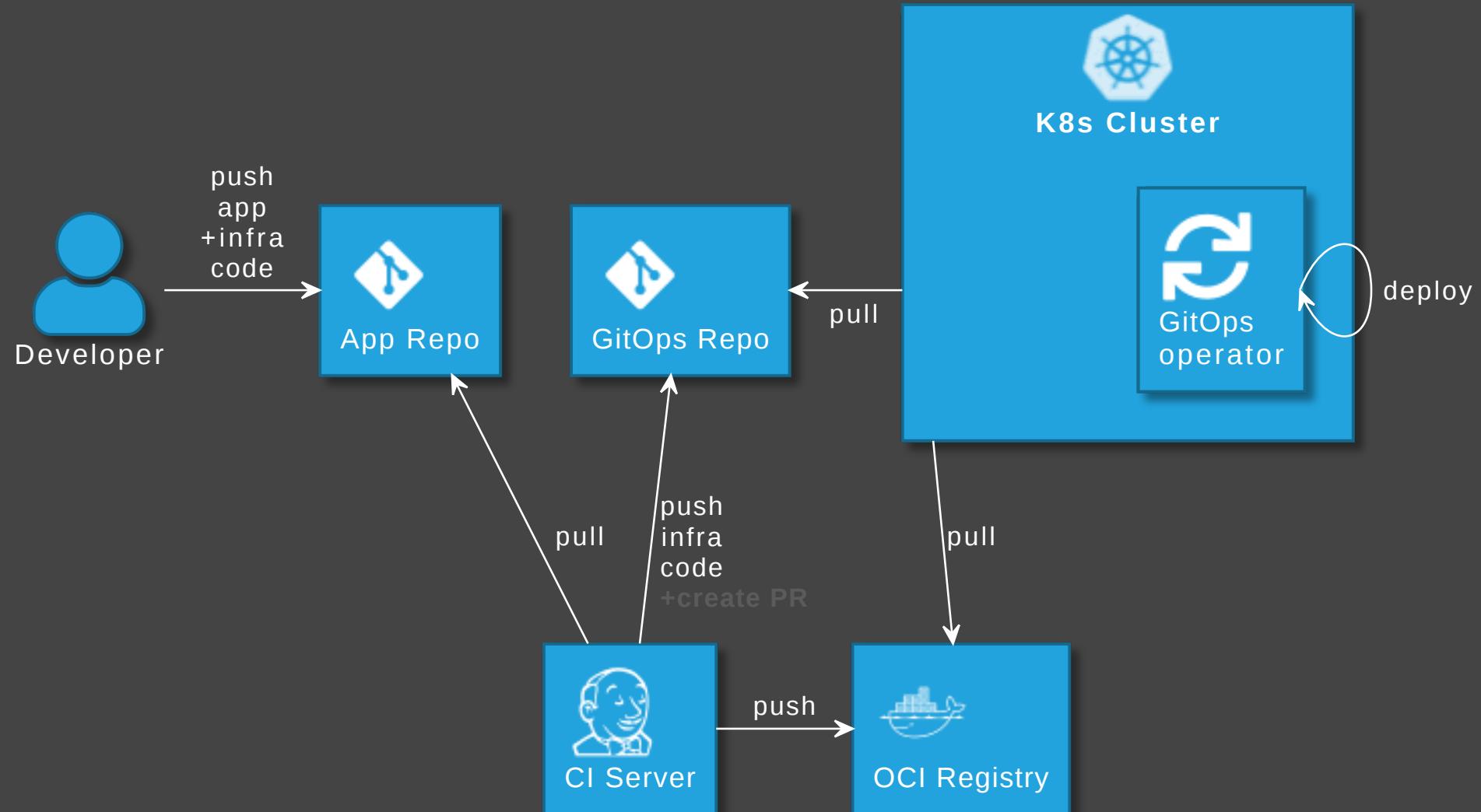
# Image updater



GitOps operator can update 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)

# Using CI-Server with GitOps part 2



## **Stage promotion using renovate bot**

 [github.com/renovatebot/renovate](https://github.com/renovatebot/renovate)

## As example: Our approach

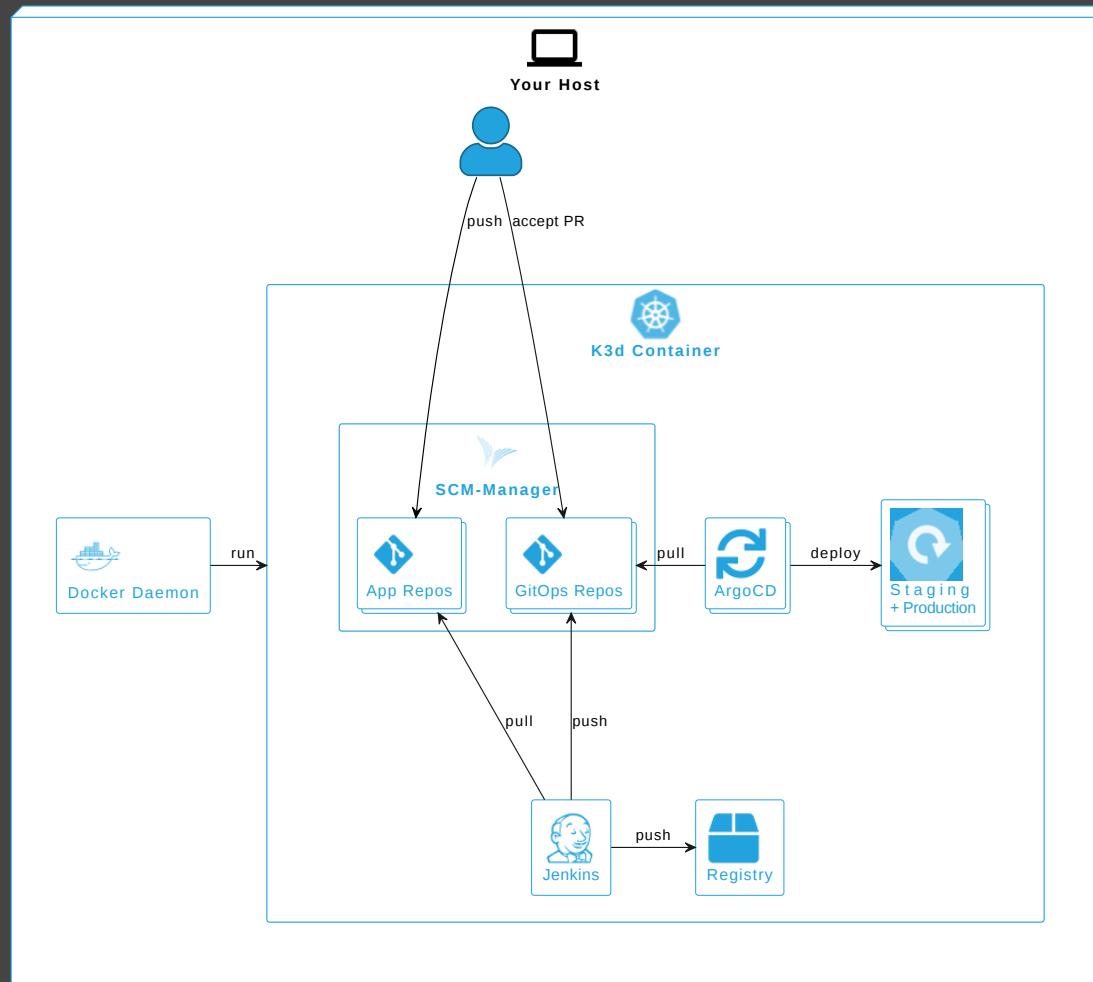
- Repo pattern: Trunk-based monorepo, folder per stage, app

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

- Promotion between stages:
  - commit to staging folder only (💡 protect production),
  - create short lived branches and pull requests for prod
- TODO Duplication is tedious, but can be automated
- TODO CI



# Demo





Johannes Schnatterer, Cloudogu GmbH

 [cloudogu.com/en/gitops](http://cloudogu.com/en/gitops)

Begin your  
**GitOps**  
journey  
with us. 

- GitOps Resources
- Community
- Trainings
- Consulting



Slides + Q&A