



// GITOPS - IS THIS SOMETHING FOR ME?

Johannes Schnatterer, Cloudogu GmbH

 [@jschnatterer](https://twitter.com/jschnatterer)

Version: 202209261757-398b514



Agenda

- Basics
- Tools
- Challenges
- Demo

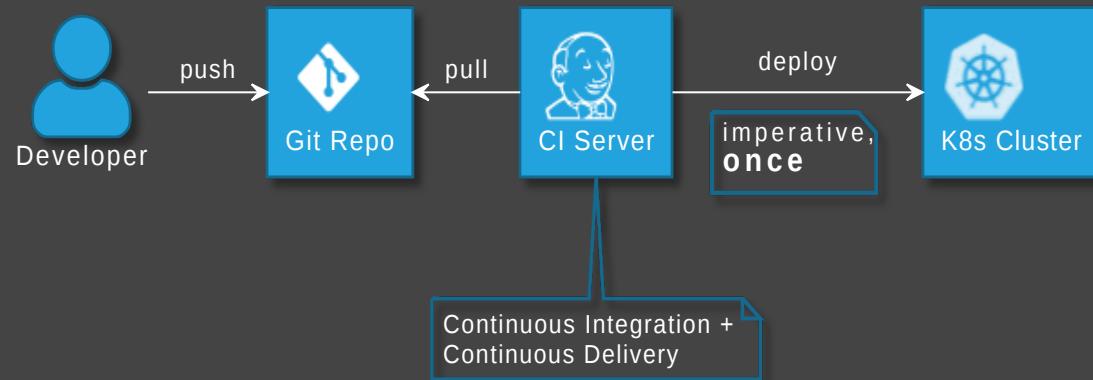
GitOps basics

Origin: blog post by Weaveworks, August 2017

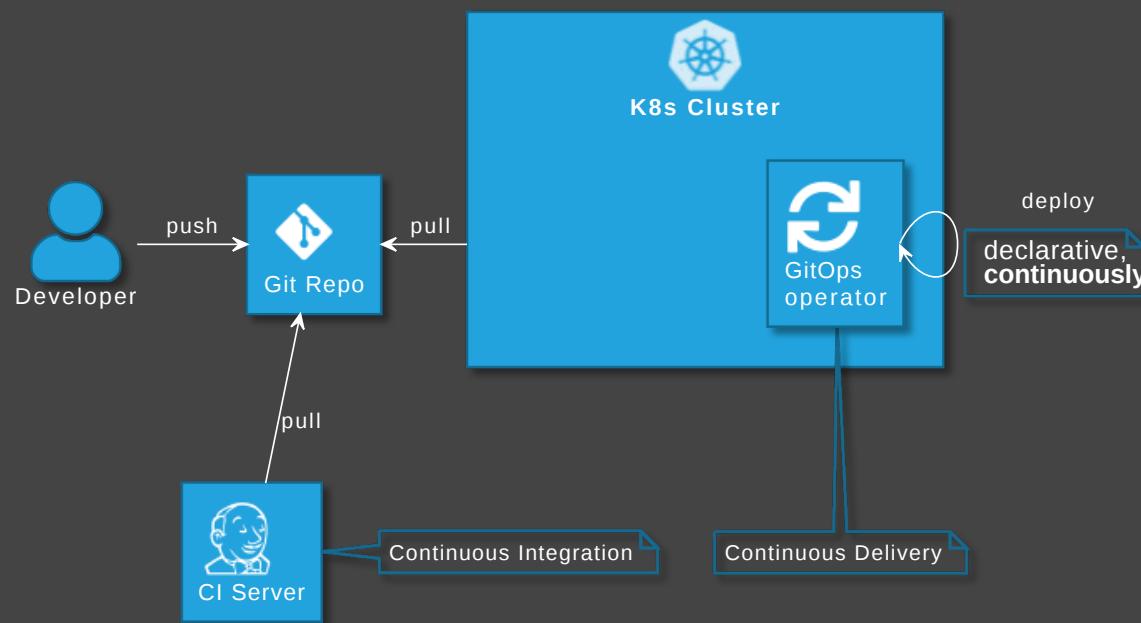
Use developer tooling to drive operations

 weave.works/blog/gitops-operations-by-pull-request

"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

GitOps vs DevOps

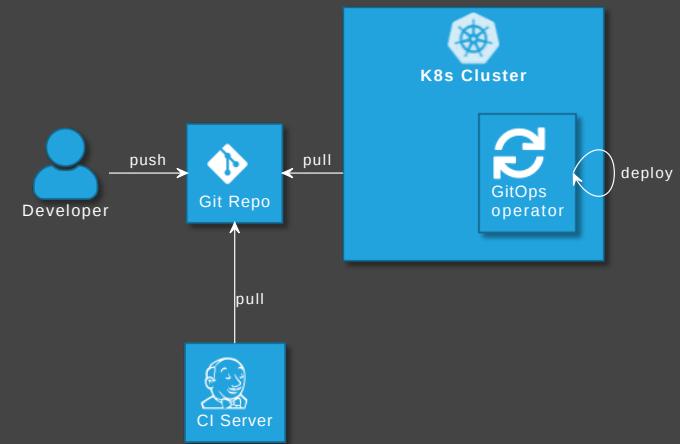
- DevOps is about collaboration of formerly separate groups (mindset)
- GitOps focuses on ops (operating model)
- GitOps could be used with or without DevOps and vice versa
- Still, GitOps might be...

The right way to do DevOps

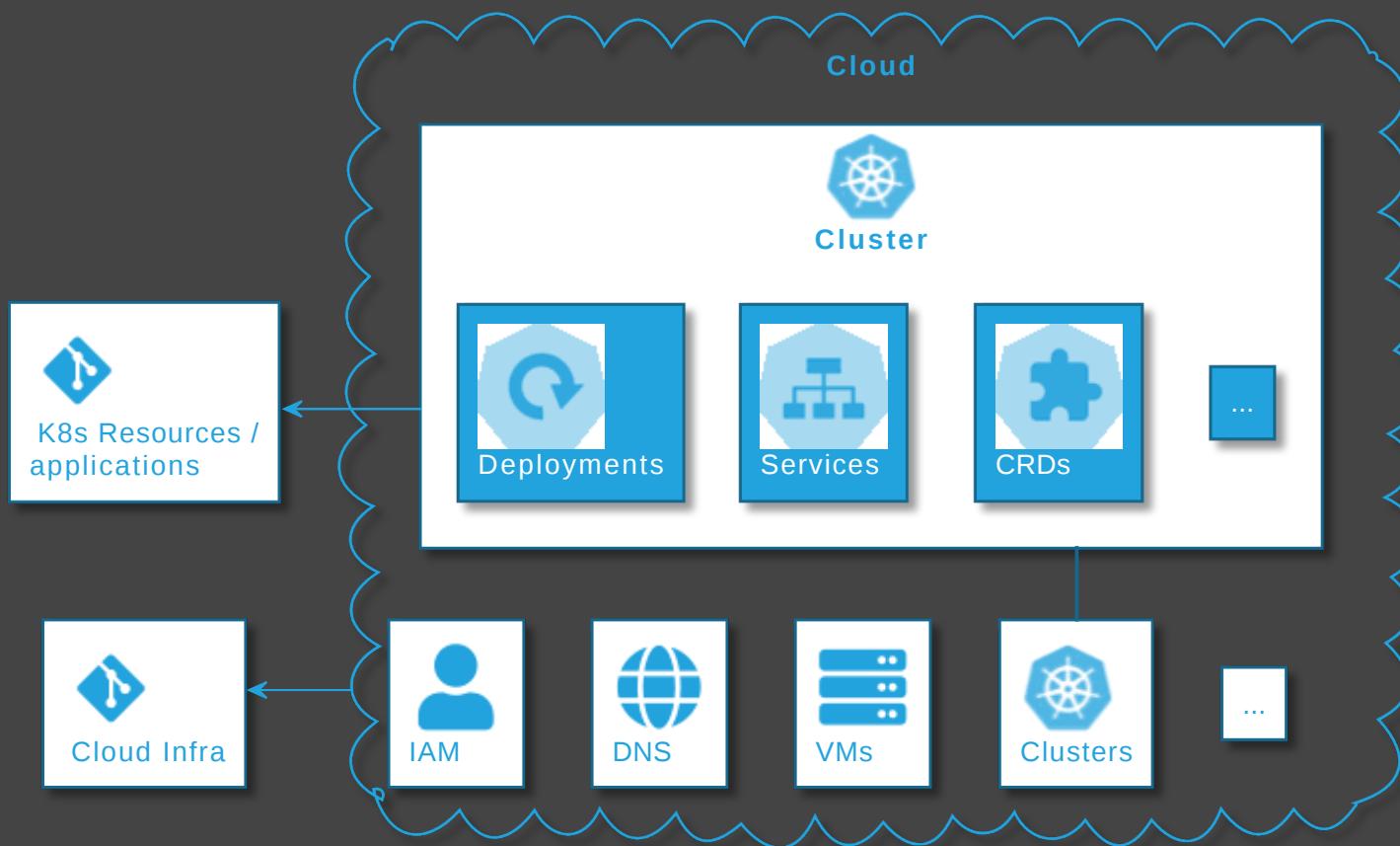
 Alexis Richardson

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

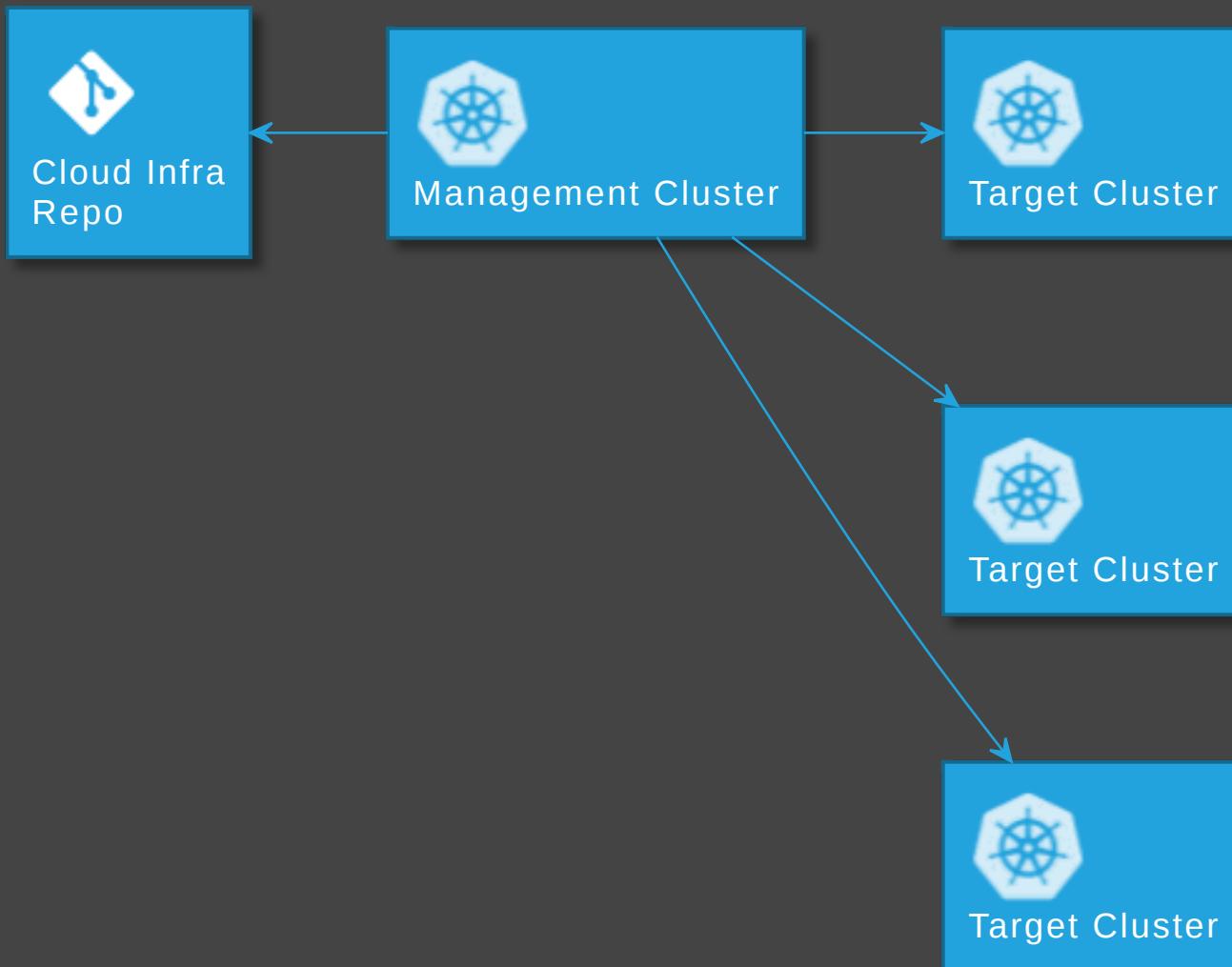
+

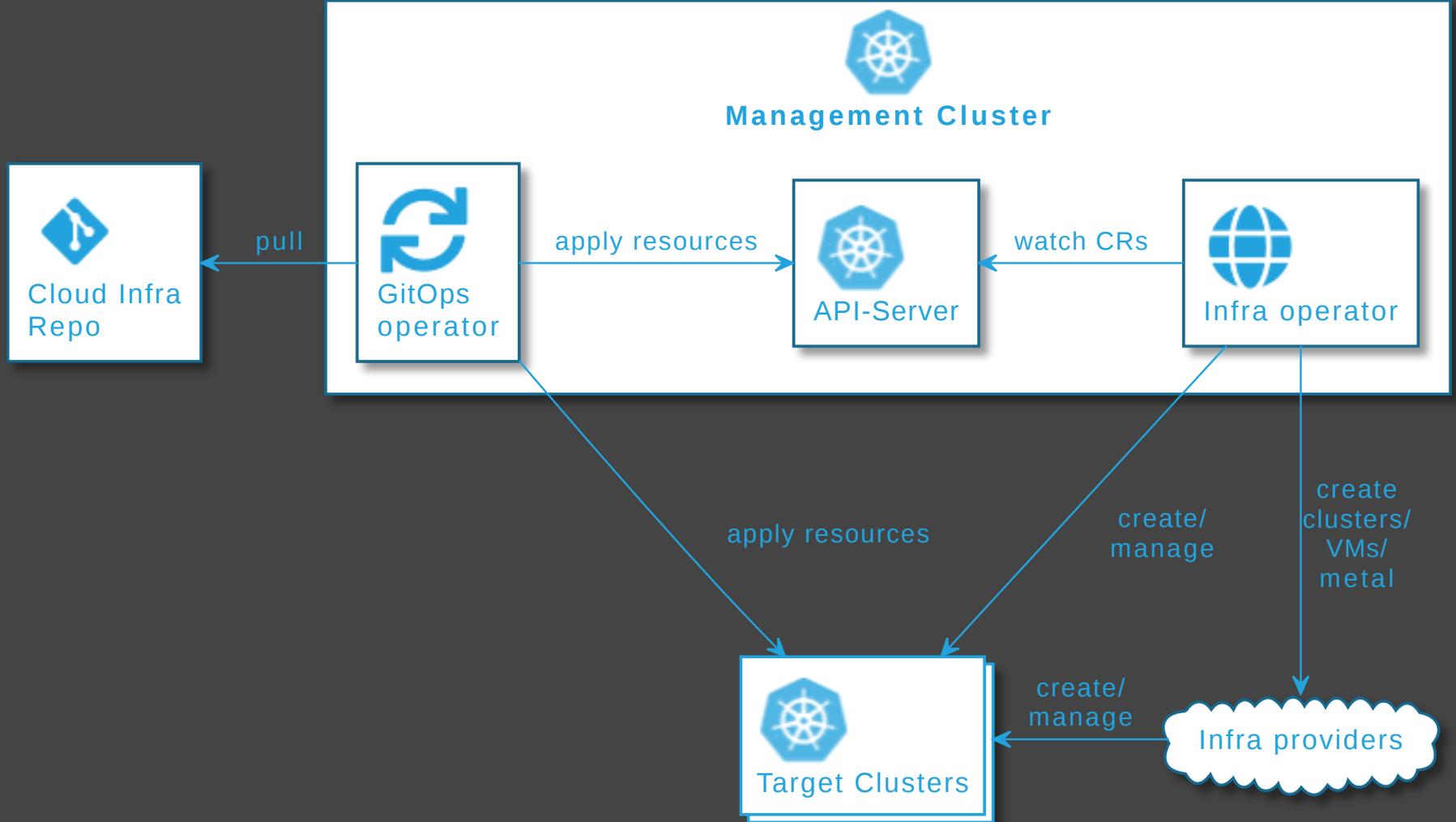
Infra Operator

=

Operate cloud infra with GitOps

Operate Kubernetes with Kubernetes





Tools for operating cloud infra



See also

 cloudogu.com/blog/gitops-tools (iX 4/2021)

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

Challenges with GitOps



More Infra ...

- GitOps Operator: several applications
- 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 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

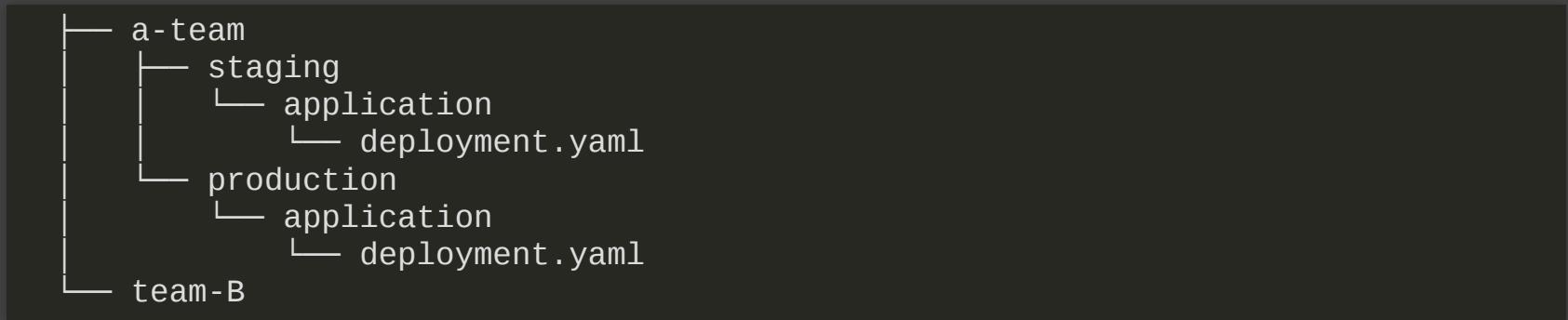
↔
Mapping?
🤔

🌐 Real-world

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

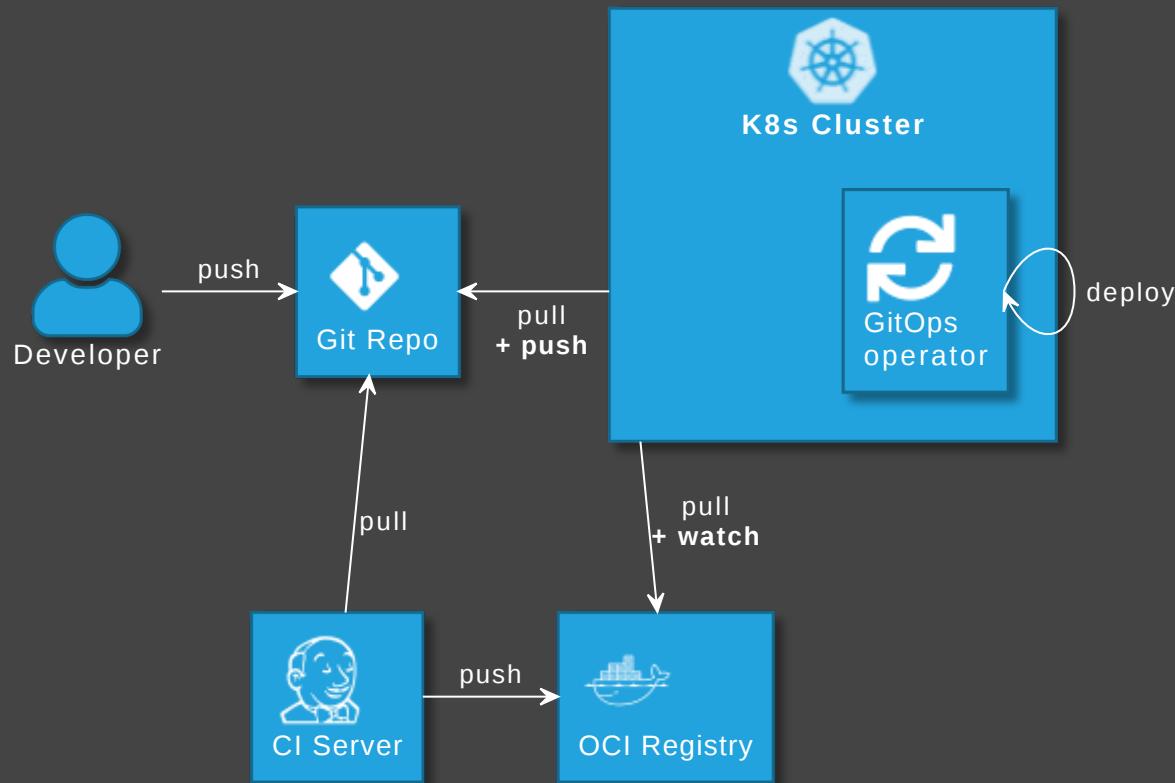
As example: Our approach

- Trunk-based monorepo, folder per team, stage, app pattern



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

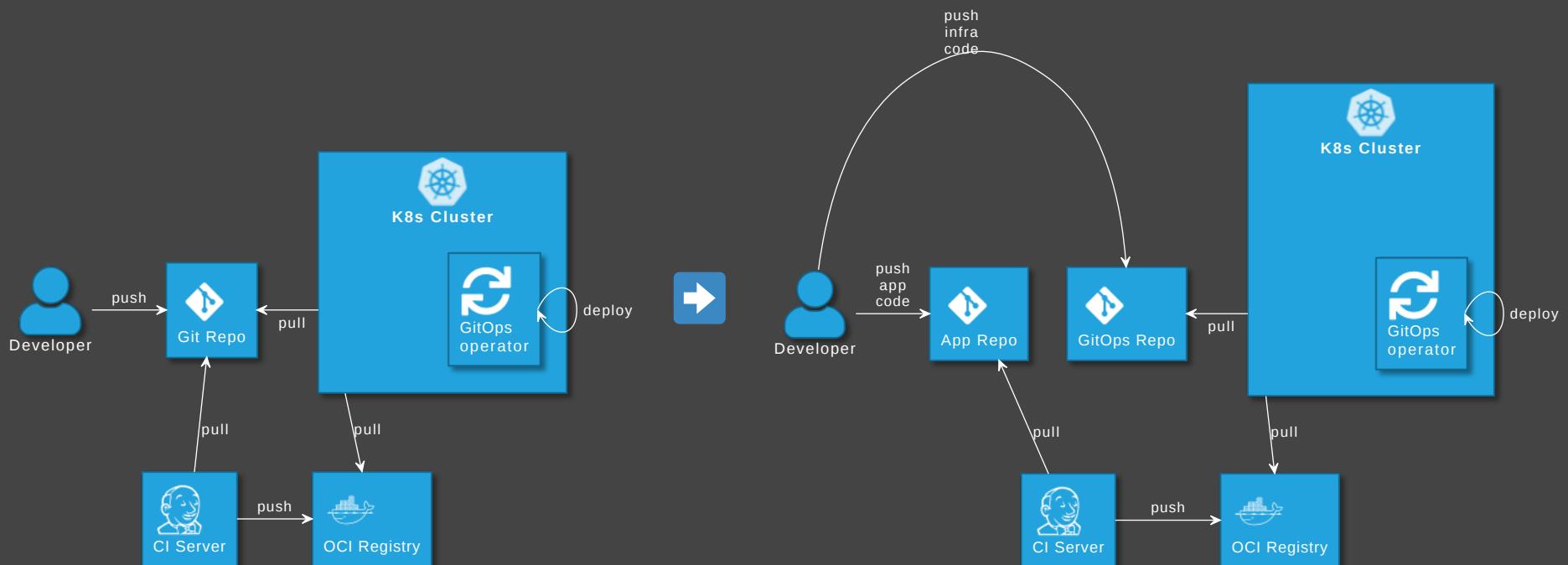
Basic role of CI server



Optional: GitOps operator updates image version in Git

- 🐙 github.com/argoproj-labs/argocd-image-updater
- ⚓ fluxcd.io/docs/guides/image-update

Application vs GitOps repo

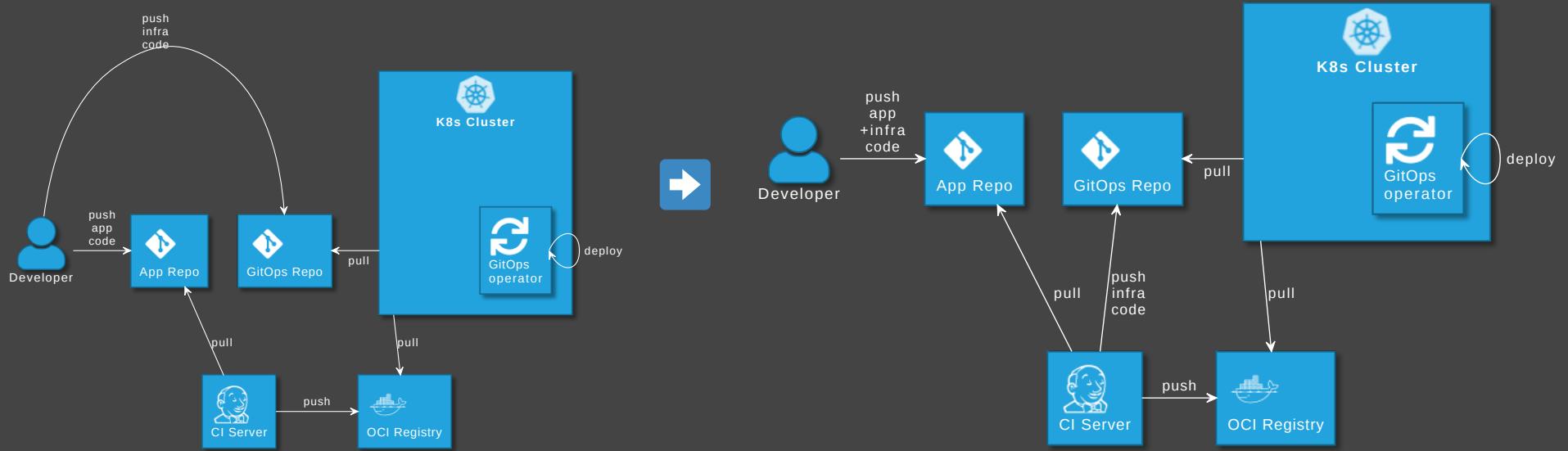


Disadvantages

- 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

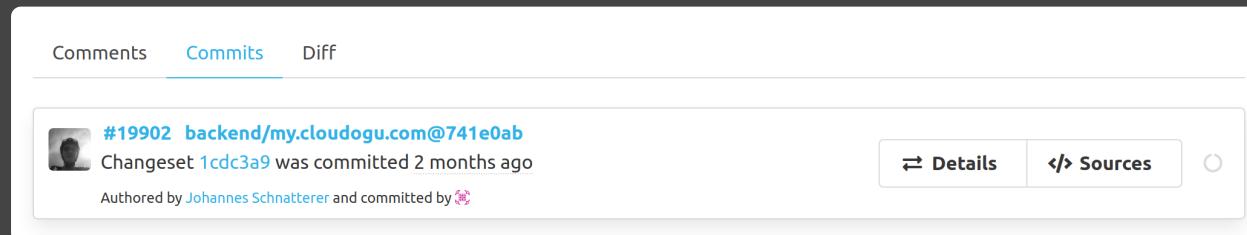
How to avoid those?

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. yamlint, kubeval, helm lint, conftest
- Simplify review by adding info to PRs



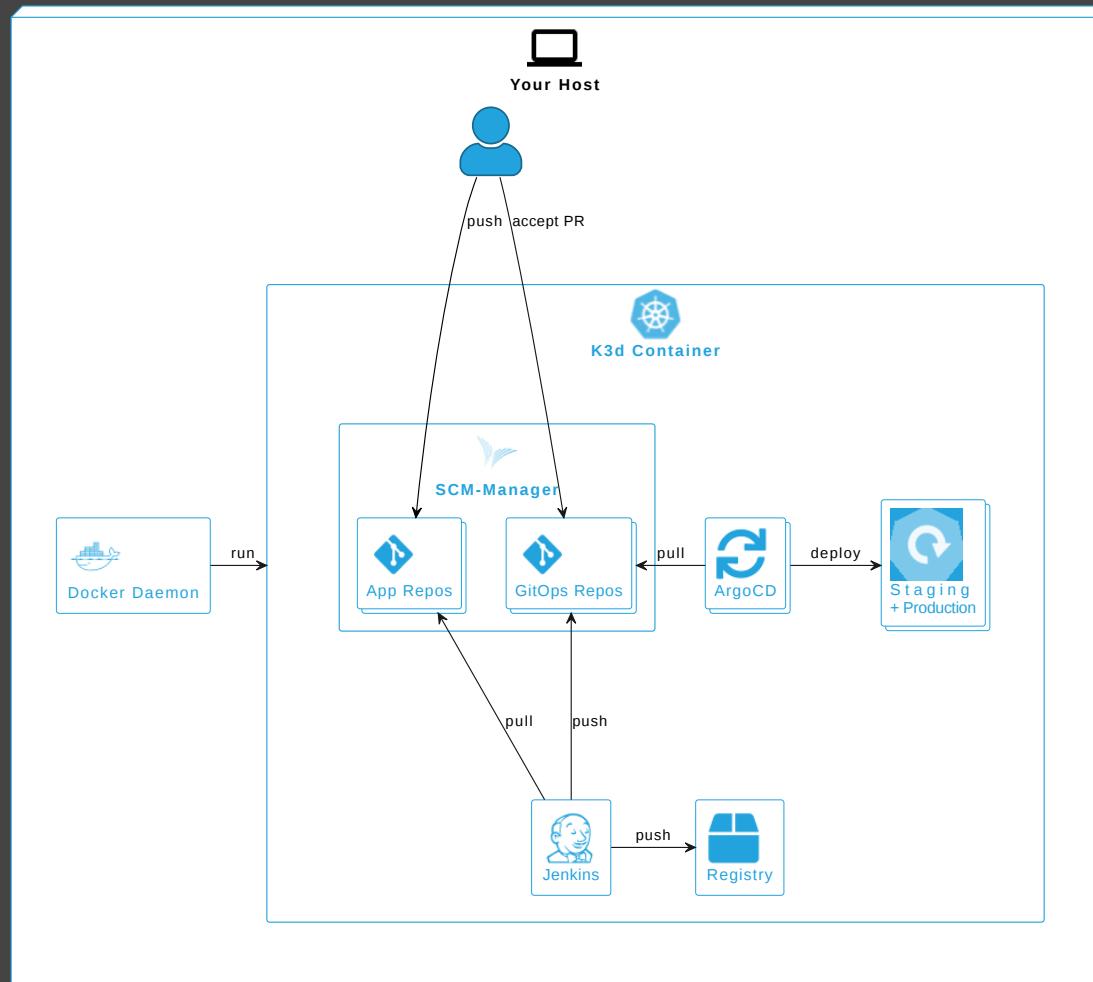
Disadvantage

Complexity in CI pipelines

- Recommendation: Use a plugin or library, e.g.
 [cloudogu/gitops-build-lib](https://github.com/cloudogu/gitops-build-lib) 



Demo



Johannes Schnatterer, Cloudogu GmbH



cloudogu.com/en/gitops

Begin your
GitOps
journey
with us.



- GitOps Resources
- Community
- Trainings
- Consulting



Slides + Q&A