



REMOTE
KONFERENZ



Johannes Schnatterer, Clouddogu GmbH

// GITOPS: HANDS-ON CONTINUOUS OPERATIONS WITH KUBERNETES

Version: 202112081726-8938e47



Agenda

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

The background is a dark gray surface covered with numerous 3D question marks. Most are black and slightly out of focus, while three are bright orange and in sharp focus. One orange question mark is at the top right, another is in the middle left, and the largest one is at the bottom center. A white rectangular box with rounded corners is centered horizontally, containing the text 'What is GitOps?'.

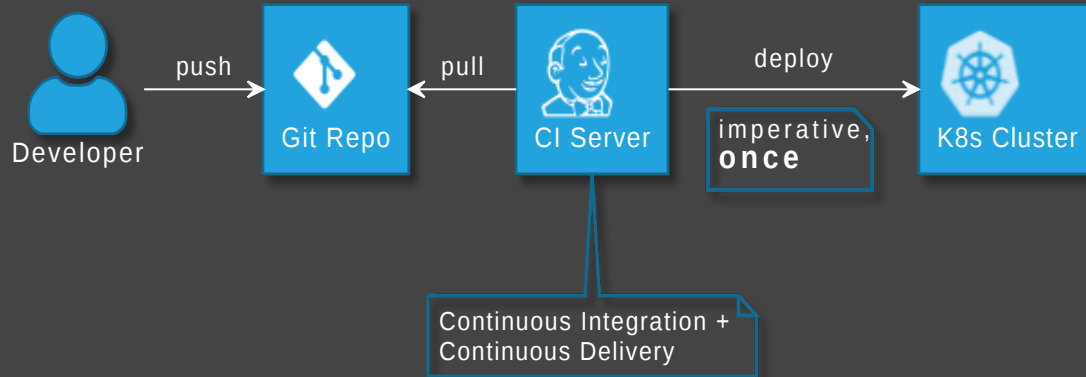
What is GitOps?

Origin: blog post by Weaveworks, August 2017

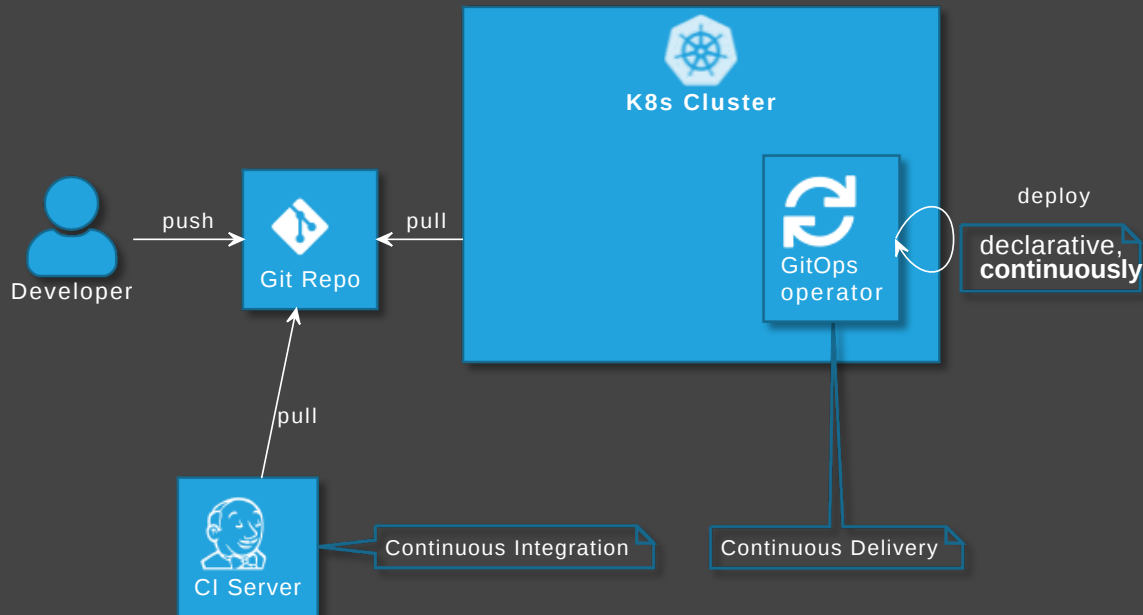
Use developer tooling to drive operations

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

"Classic" Continuous Delivery ("CLOps")



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**



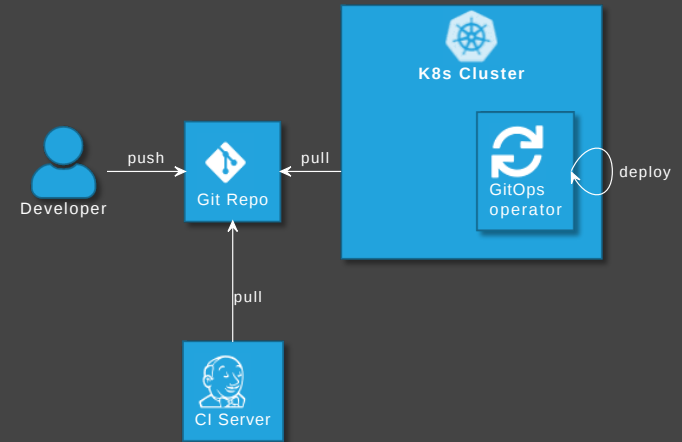
 github.com/open-gitops/documents/blob/main/PRINCIPLES.md

GitOps vs DevOps

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

Advantages of GitOps

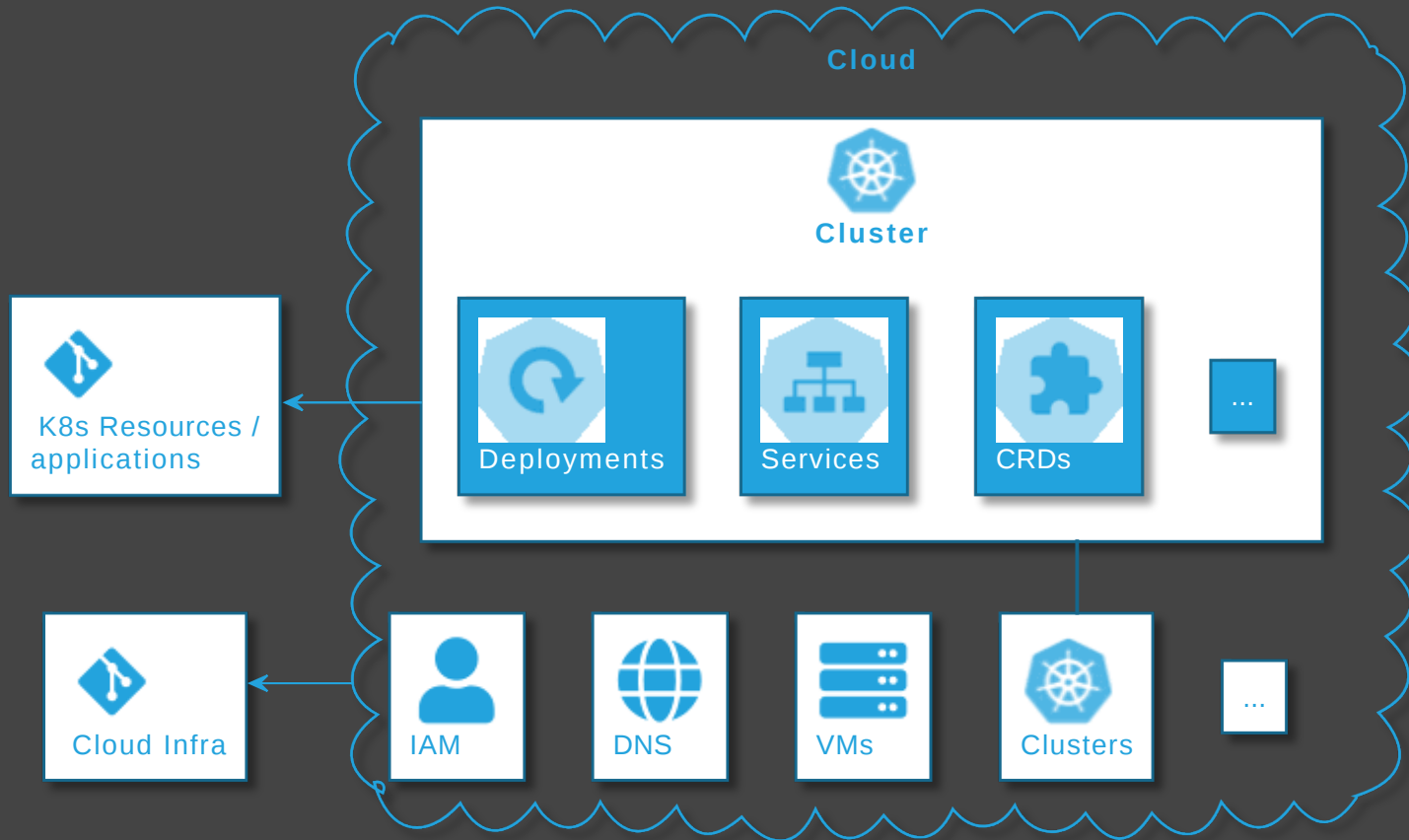
- No access to cluster from outside (might also solve firewall/zone issues)
- No credentials on CI server (neither cluster access nor for apps)
- Forces declarative description
- IaC is auditable
- Scalability - one repo many applications
- Self-healing / Hands-off ops





How can GitOps be used?

What can GitOps be used for?



GitOps tool categories




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

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
- Deployment Strategies - Progressive Delivery



- ...

➔ GitOps ❤ operators

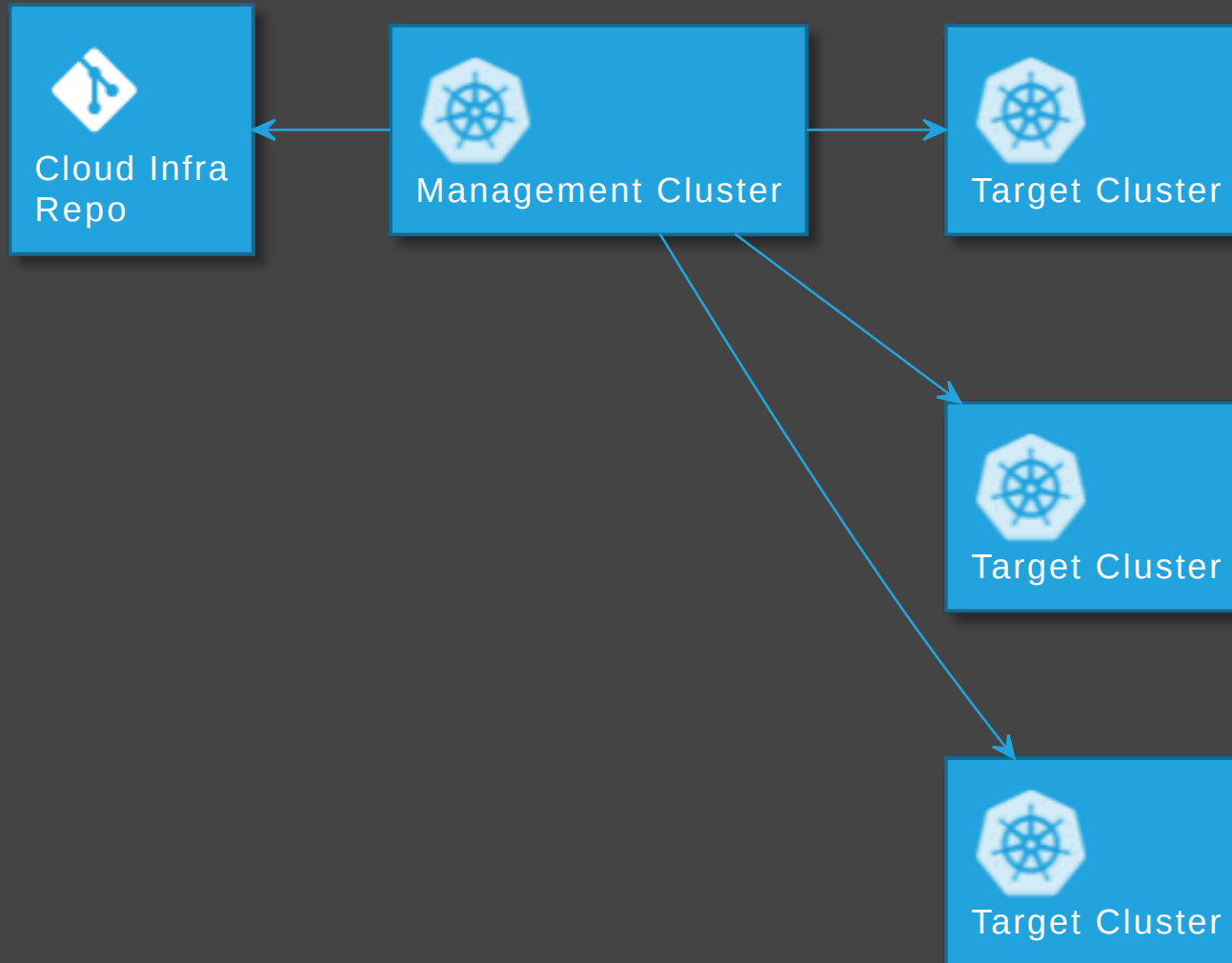
+

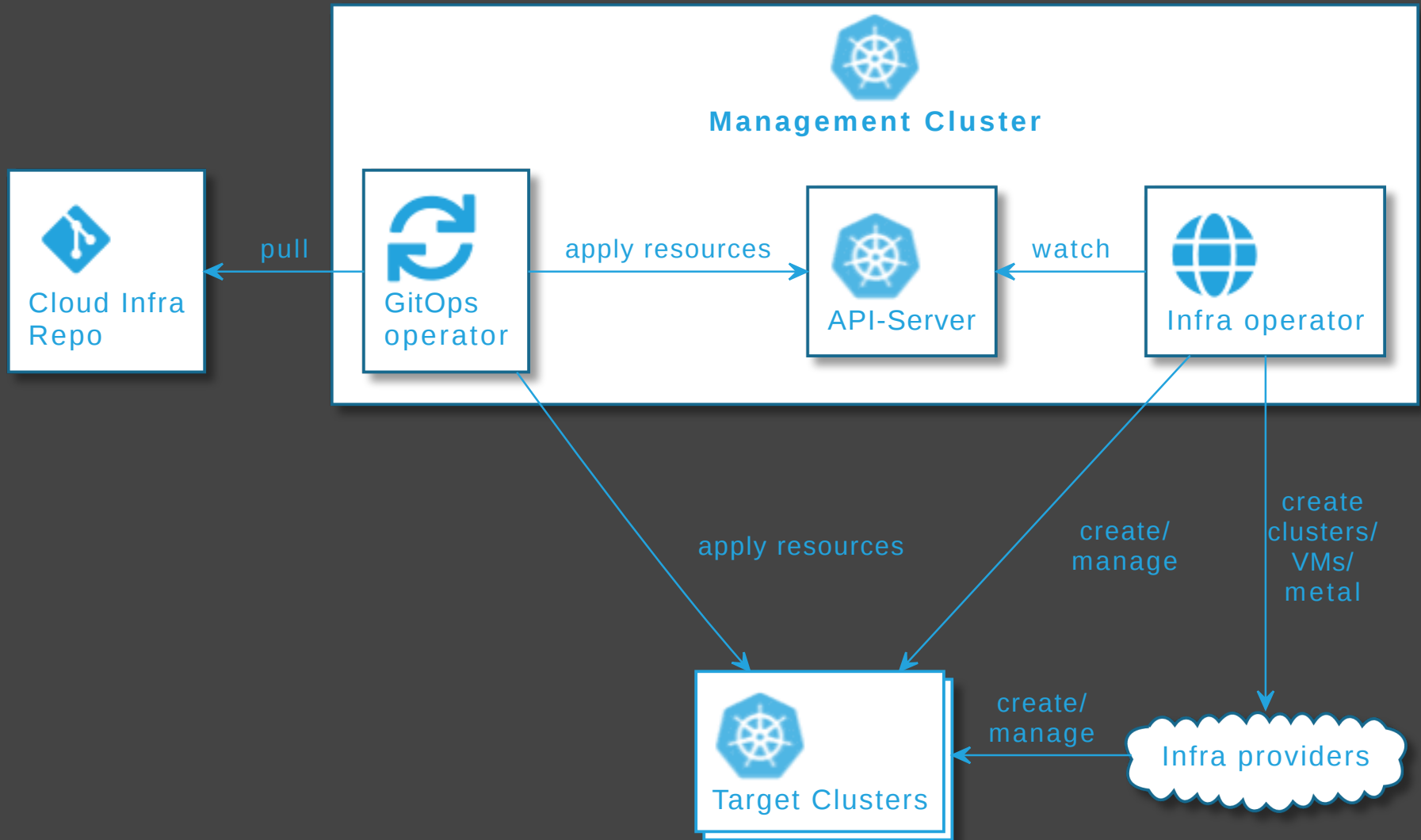
Infra Operator

=

Operate cloud infra with GitOps

Operate Kubernetes with Kubernetes

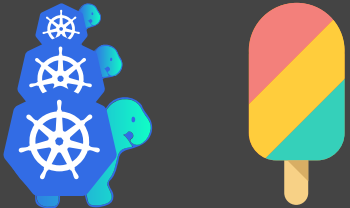




Tools for operating cloud infra



+



Cloud or Operator

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

See also

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

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

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 is stored 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

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



Logic for branching complicated and error prone (merges)

Idea 2: Staging folders

- On the same branch: One folder per stage

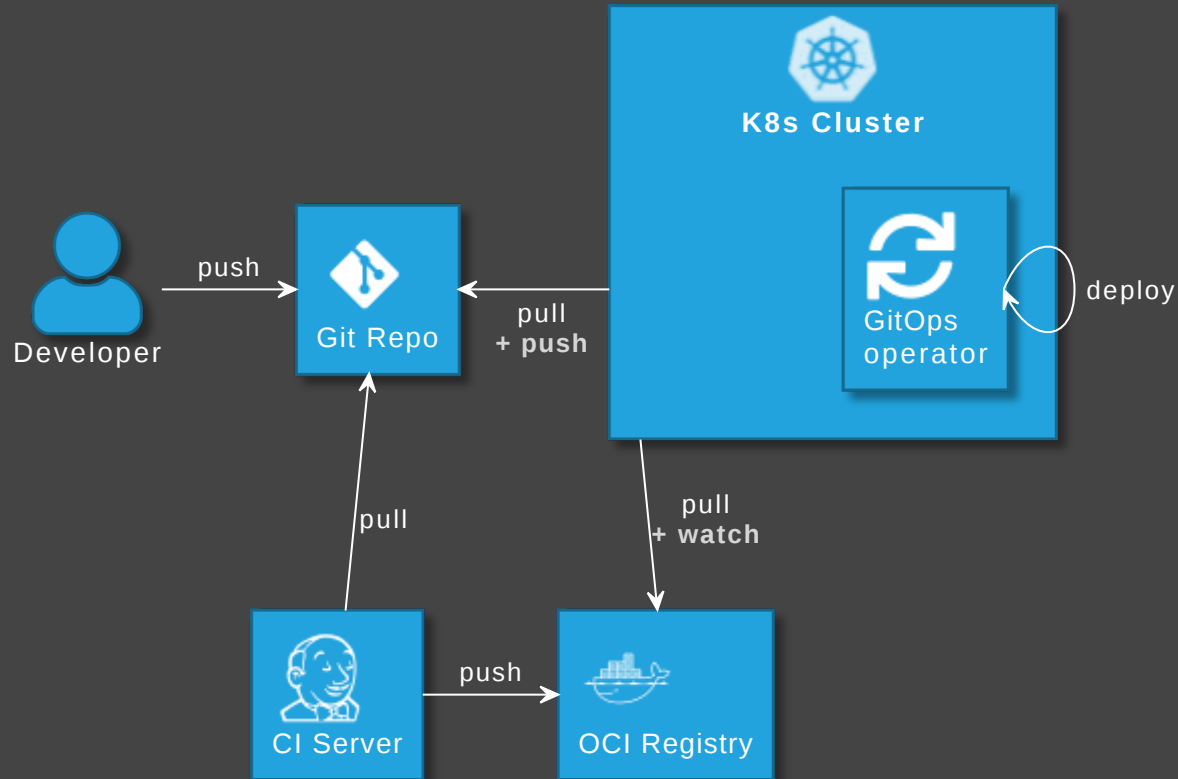
```
├── production
│   ├── application
│   └── deployment.yaml
└── staging
    ├── application
    └── deployment.yaml
```

- 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

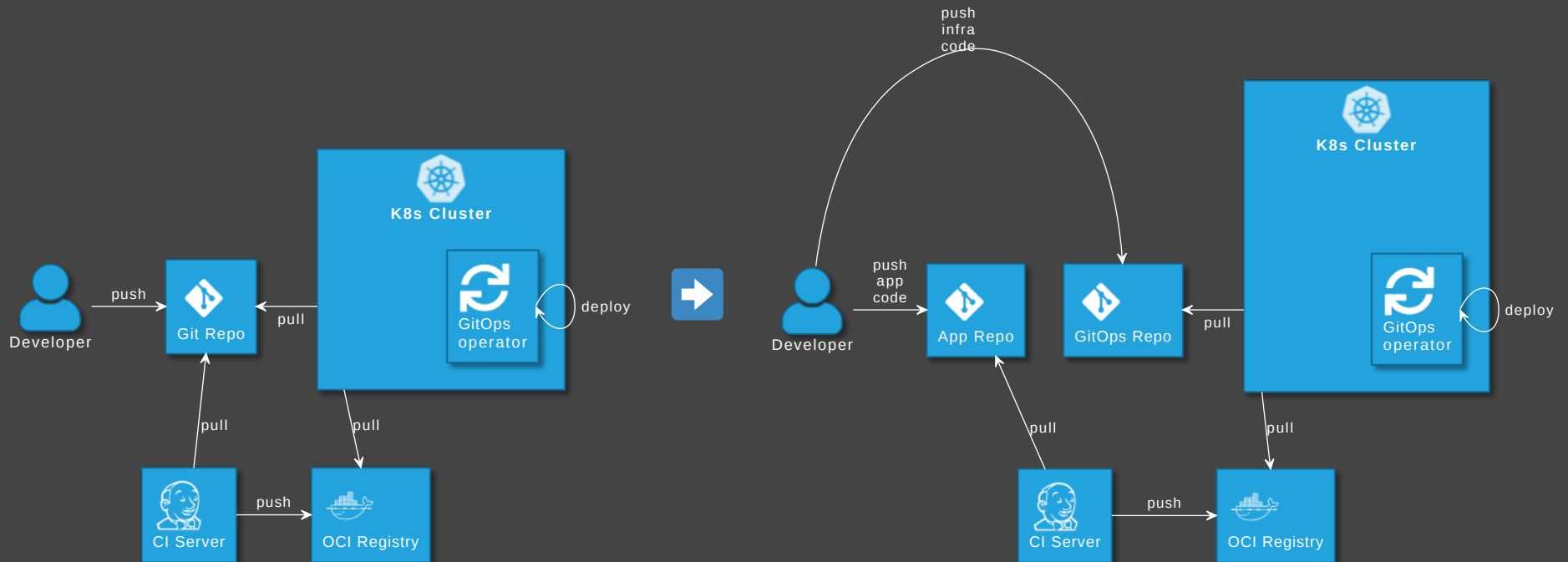
Basic role of CI server



📌 Optional: GitOps operator updates image version in Git

- 🧑 github.com/argoproj-labs/argocd-image-updater
- 📡 fluxcd.io/docs/guides/image-update

Number of repositories: application vs GitOps repo



GitOps tools: Put infra in separate repo! See



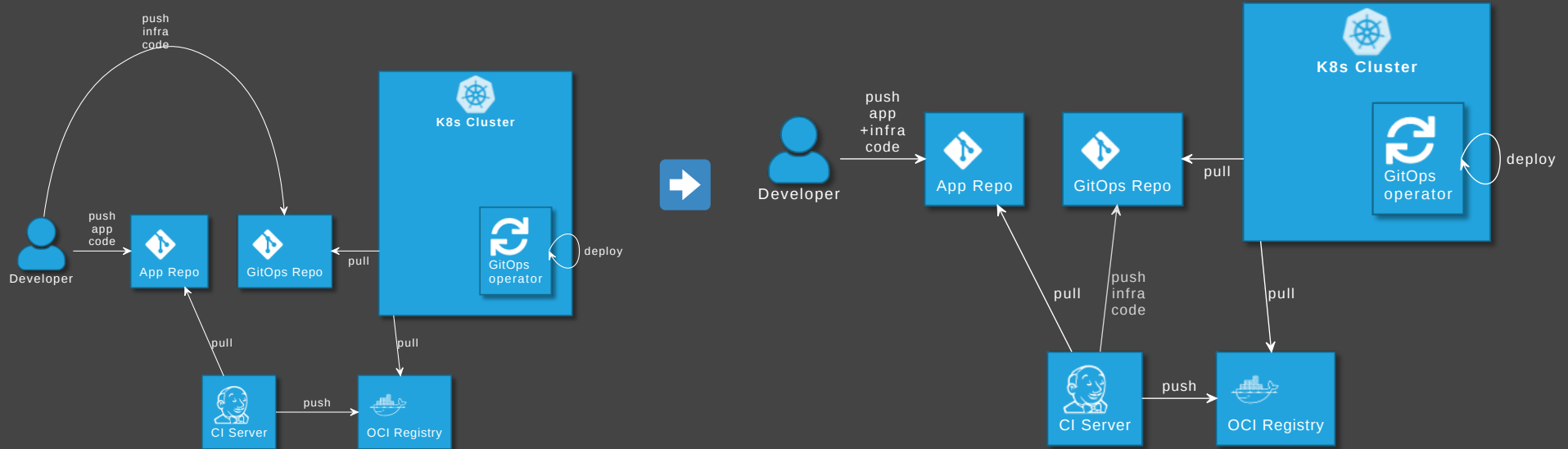
argocd.readthedocs.io/en/release-2.0/user-guide/best_practices

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

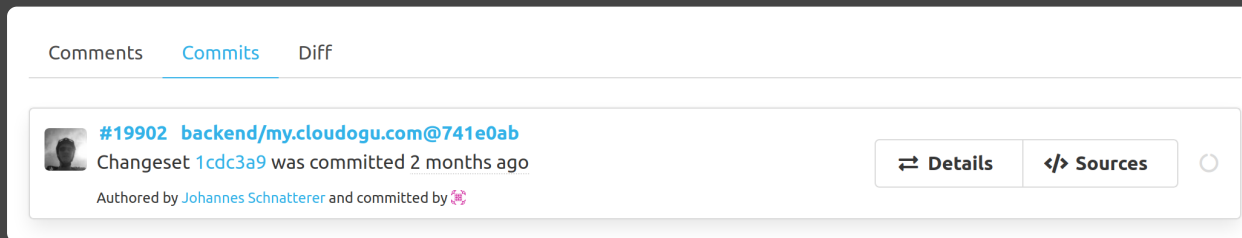
How to avoid those?

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



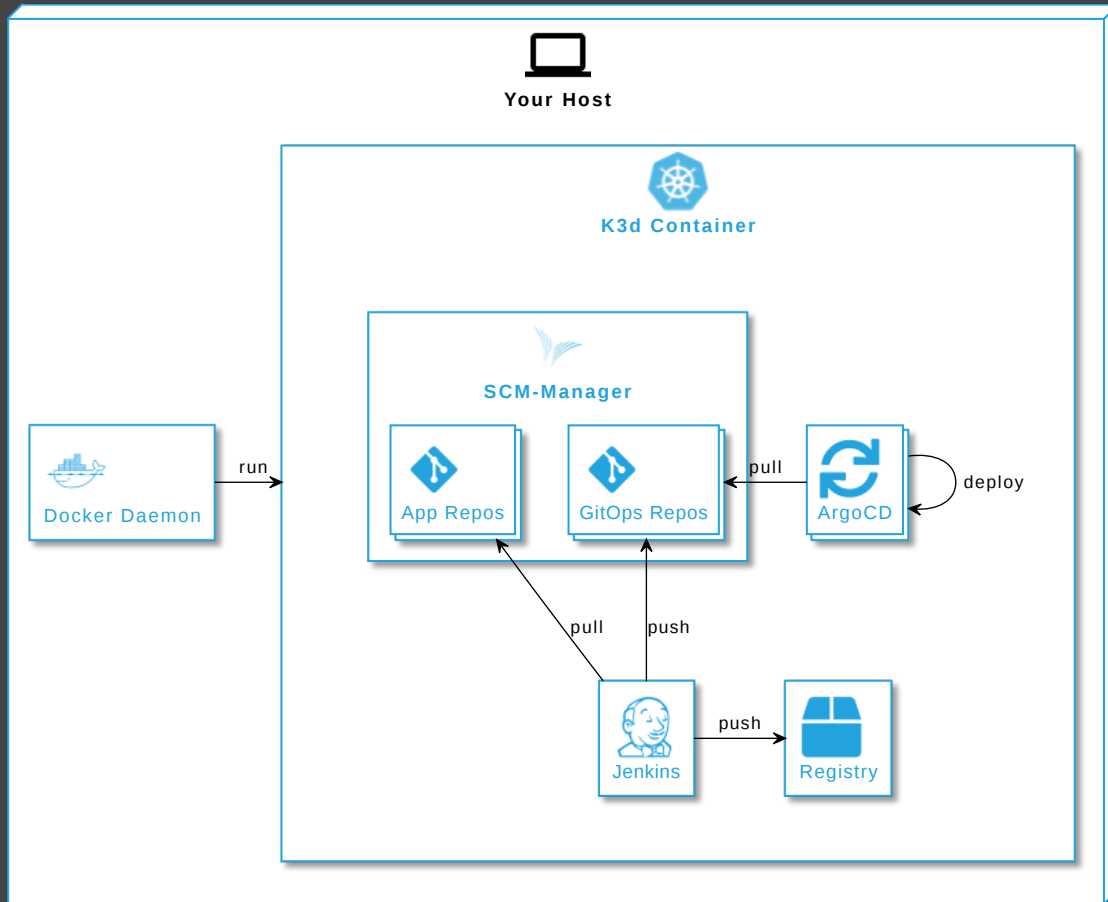
Disadvantage

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

-  GitOps Resources (intro, articles, talks, videos,  projects)
-  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