



// GITOPS: HANDS-ON CONTINUOUS OPERATIONS WITH KUBERNETES

Johannes Schnatterer, Cloudogu GmbH

 @jschnatterer

Version: 202111161046-7884fb2



Agenda

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

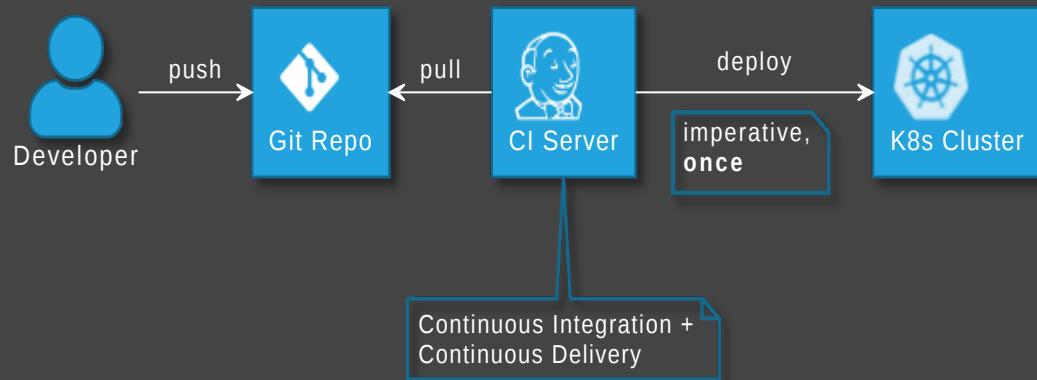
What is GitOps?

- Operating model
- Origin: blog post by Weaveworks, August 2017

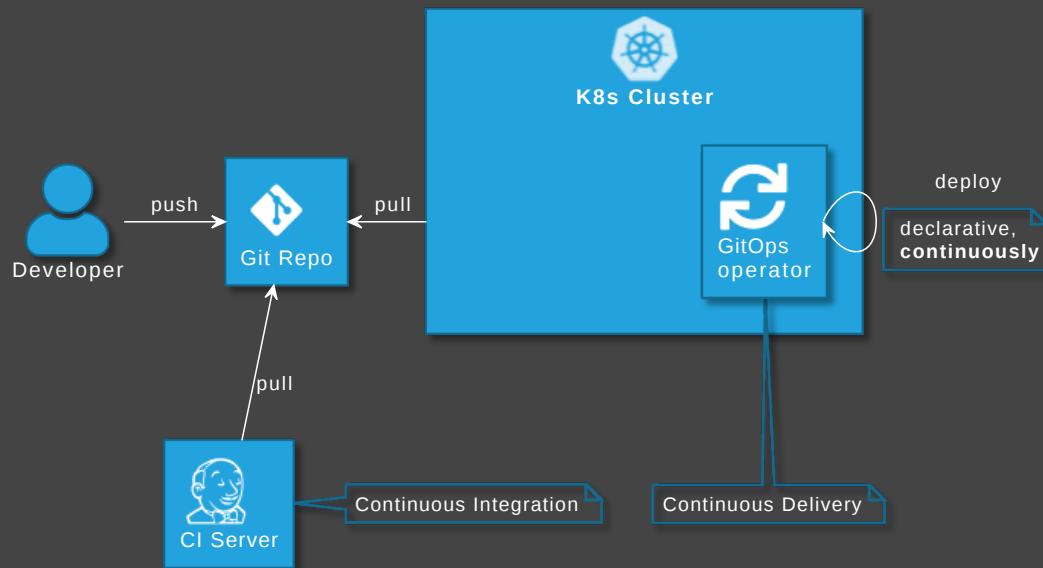
Use developer tooling to drive operations

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

"Classic" Continuous Delivery ("CIOps")



GitOps



GitOps Principles

- 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

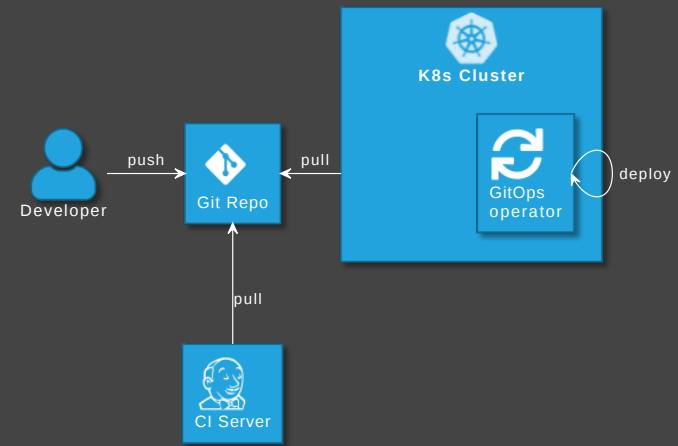


GitOps vs DevOps

- 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

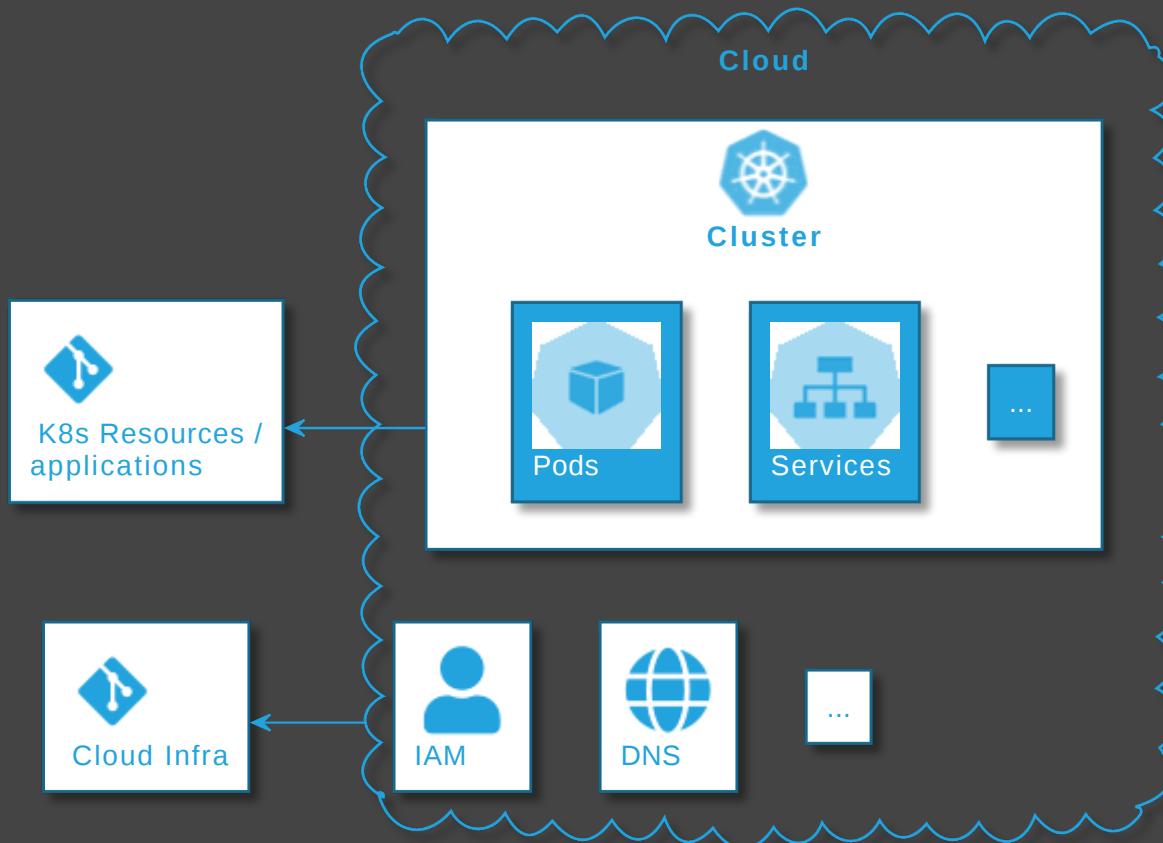
- No access to cluster from outside
 - ➔ No credentials on CI server
- Forces declarative description
- IaC is auditable
- Scalability - one repo many applications
- Self-healing / Hands-off ops





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](#)
-  [Soluto/kamus](#)
-  [mozilla/sops + K8s integration](#)
- Operators for Key Management Systems

Others

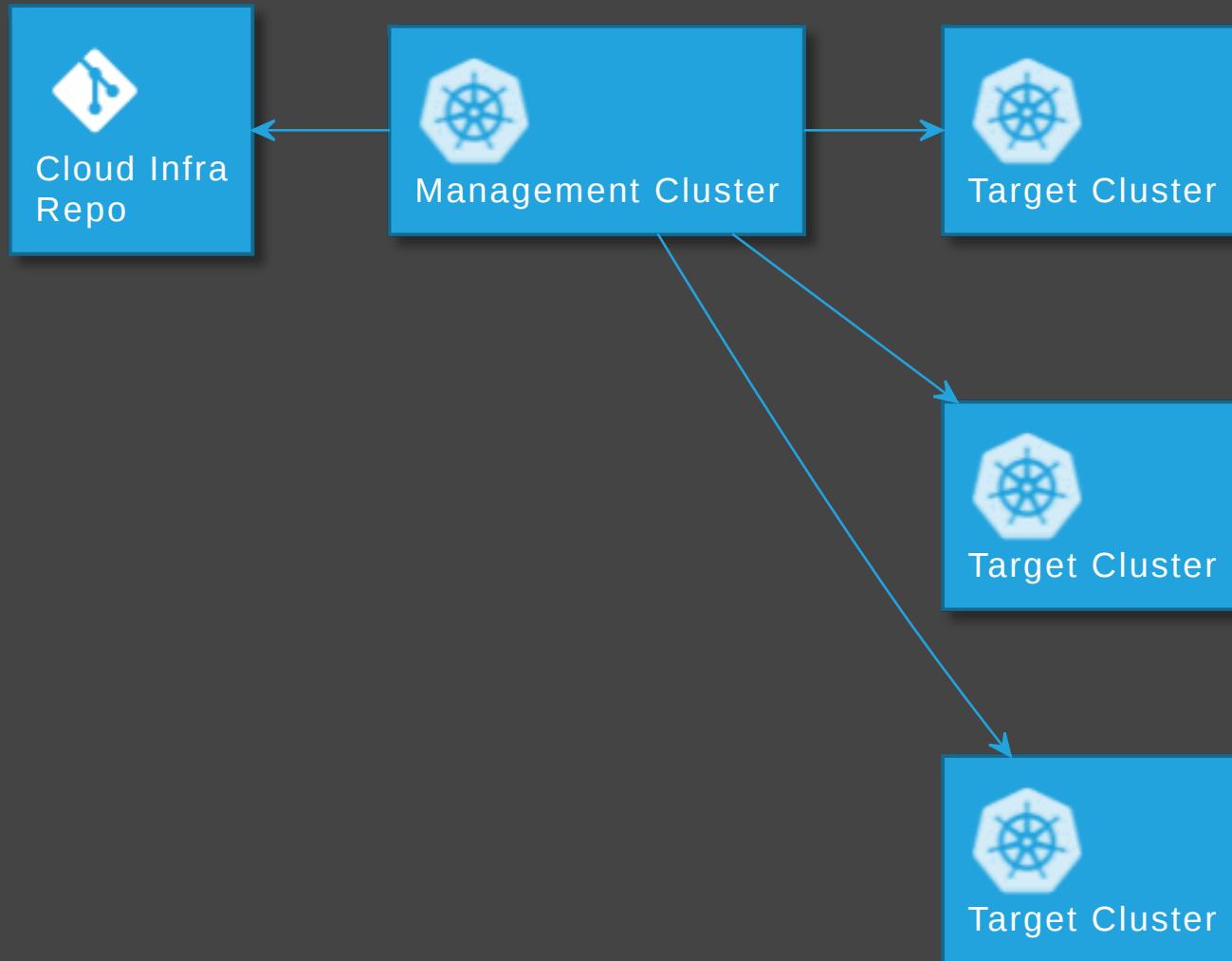
- Backup
- Deployment Strategies - Progressive Delivery

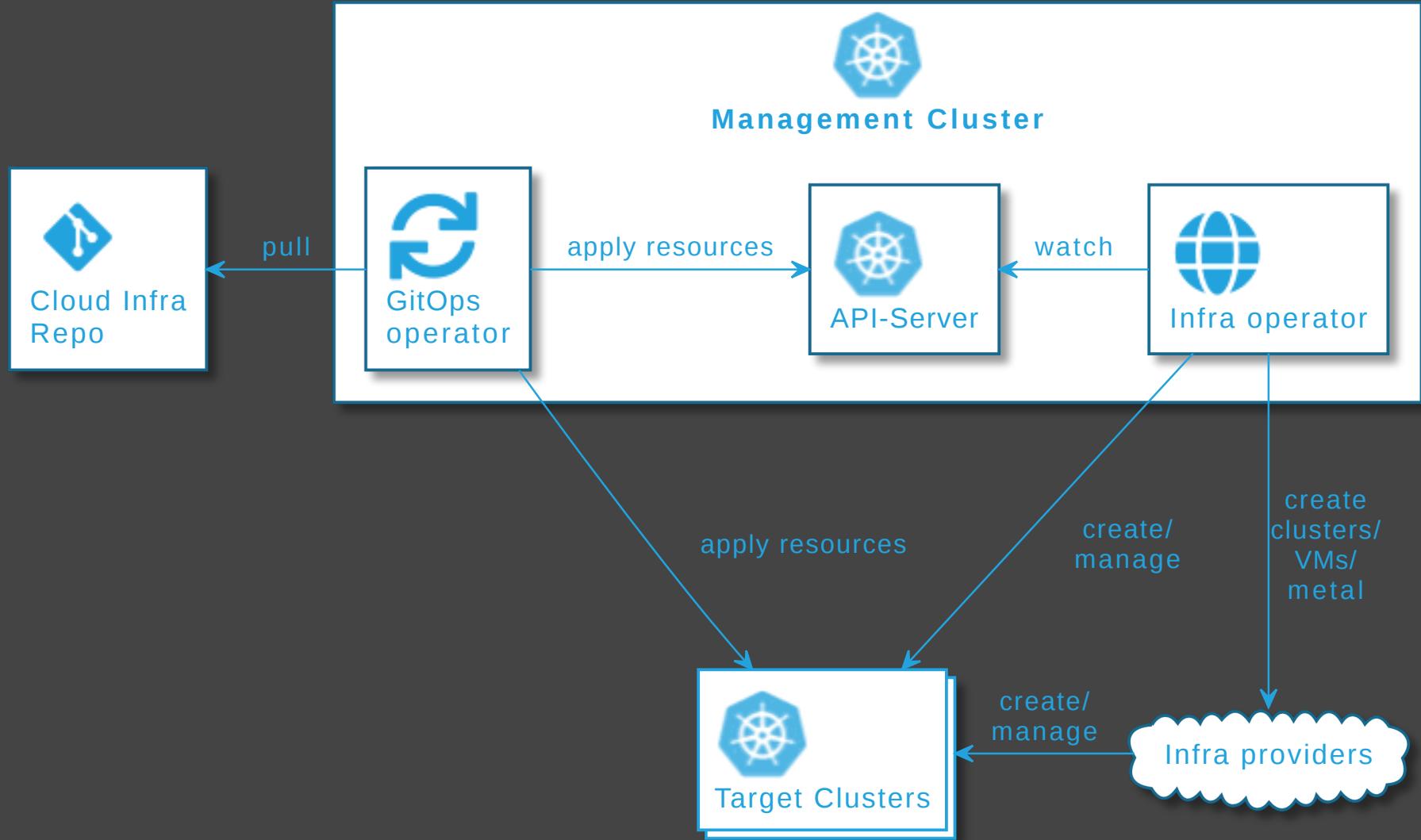


- ...

→ **GitOps loves operators**

Operate Kubernetes with Kubernetes





Tools for operating k8s clusters + cloud infra



+



-

Cloud or Operator

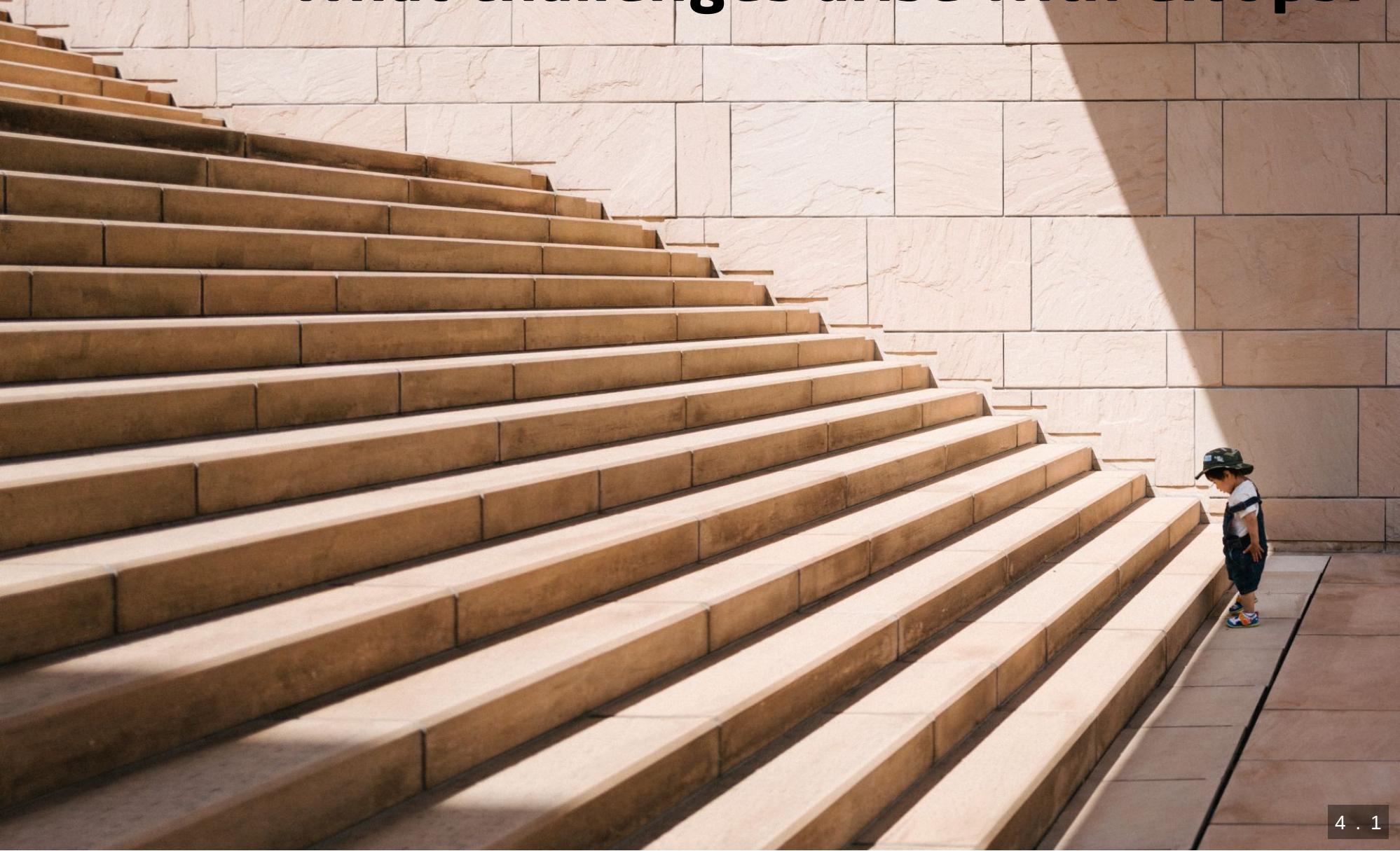
-
- [rancher/terraform-controller](#)
-

See also

 cloudogu.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 staging?
 - How to structure repos and how many of them?
 - Role of CI server?
 - How to realize local dev env?
 - How to delete resources?
 - ...

Implementing stages

Idea 1: Staging Branches

- Develop → Staging
- Main → Production



Logic for branching complicated and error prone (merges)

Idea 2: Staging folders

- On the same branch: One folder per stage

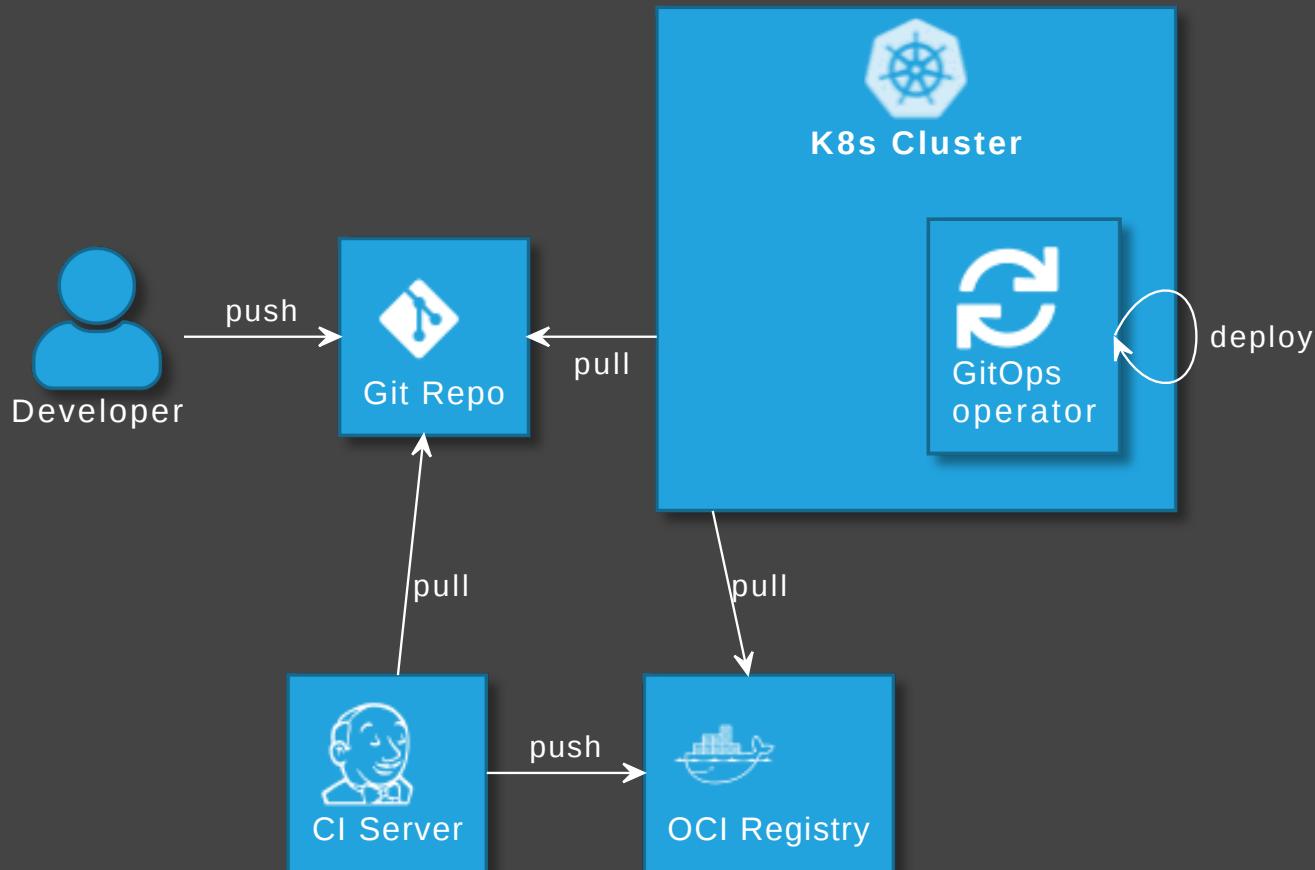
```
└── production
    └── application
        └── deployment.yaml
└── staging
    └── application
        └── deployment.yaml
```

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



- Logic for branching simpler
- Supports arbitrary number of stages

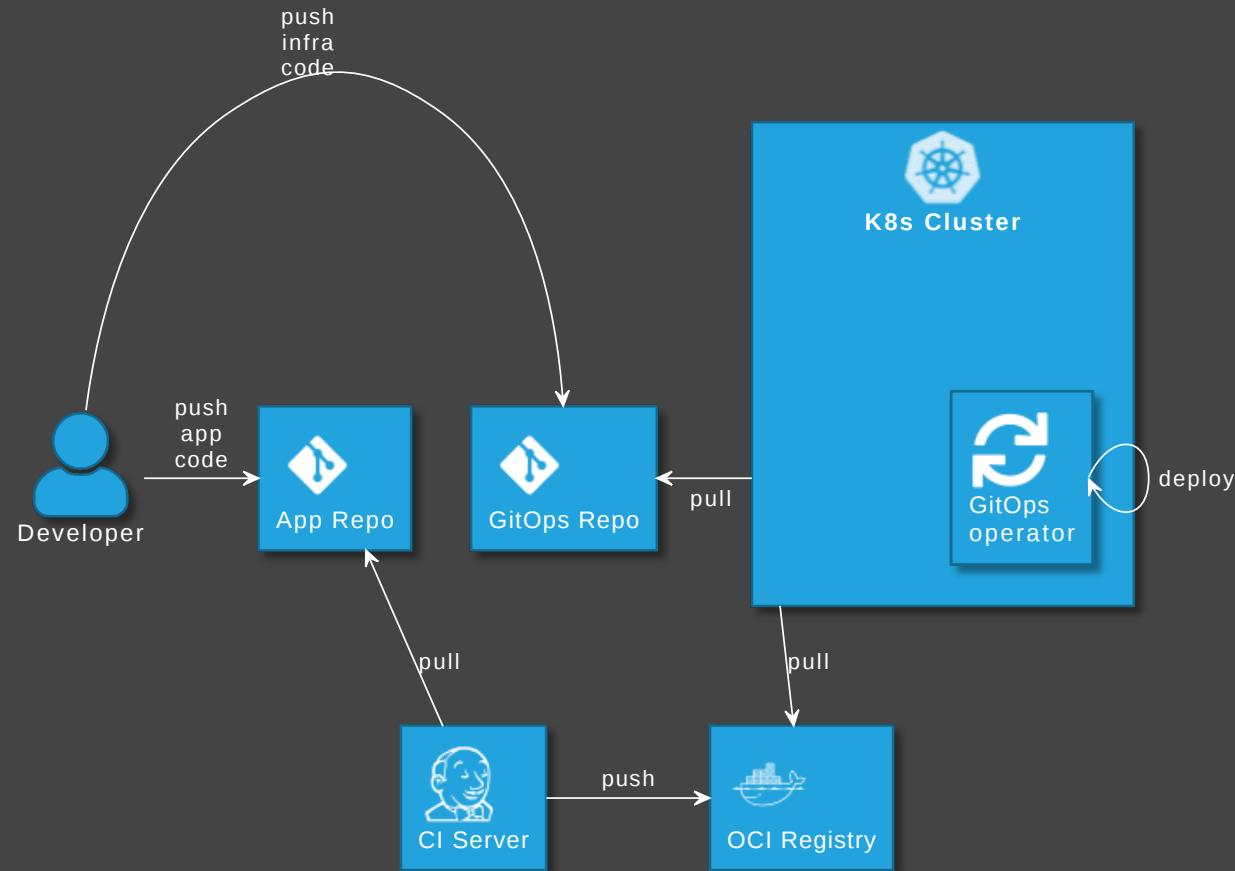
Basic role of CI server



Number of repositories: application vs GitOps repo

GitOps tools: Put infra in separate repo! See

argo-cd.readthedocs.io/en/release-2.0/user-guide/best_practices

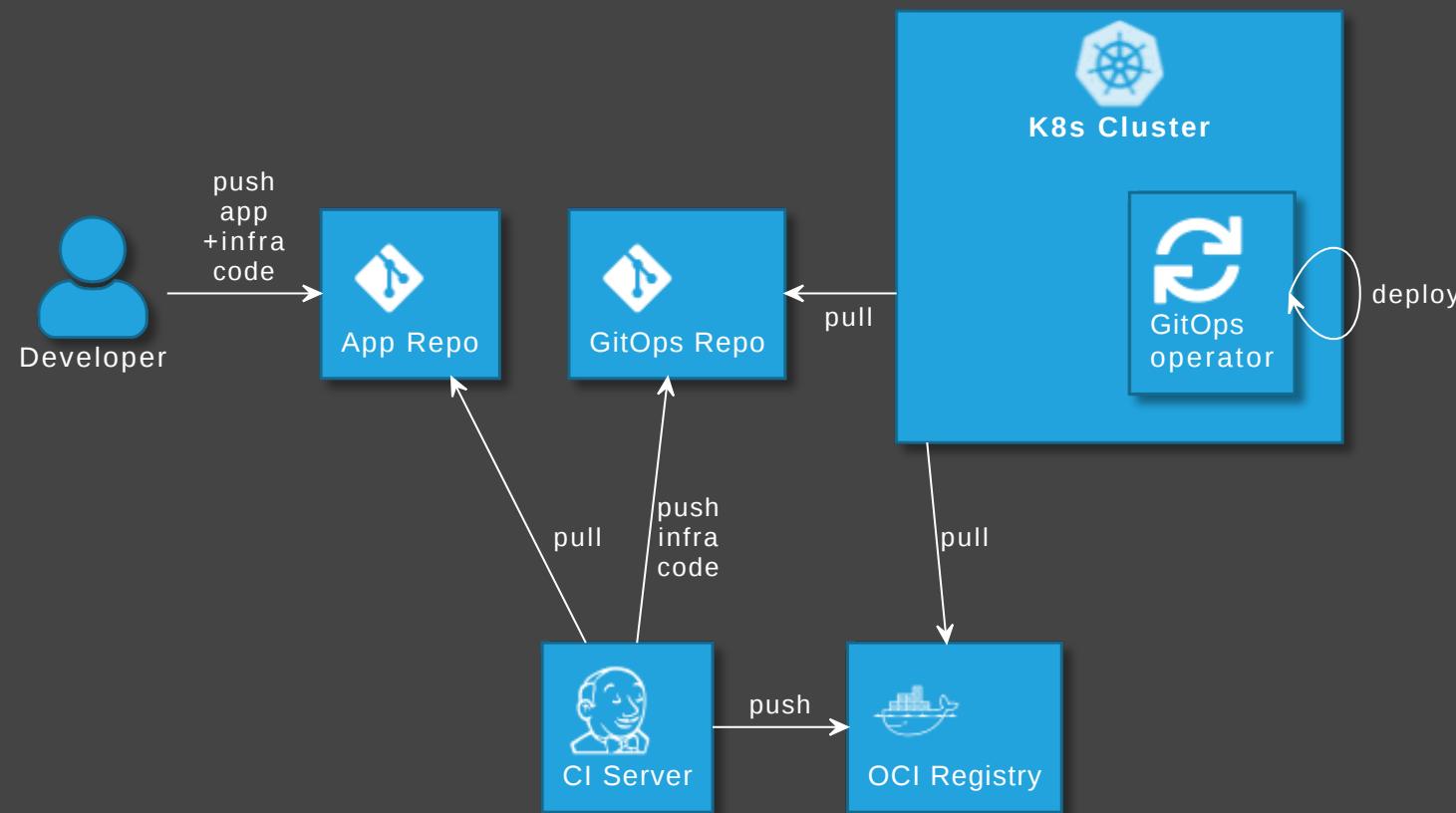


Disadvantages

- Separated maintenance & versioning of app and infra code
- Review spans across multiple repos
- Local dev more difficult

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

A screenshot of a GitHub pull request details page. The top navigation bar shows 'Comments', 'Commits' (which is underlined in blue), and 'Diff'. Below the navigation, a commit card is displayed for pull request #19902. The commit message is: '#19902 backend/my.cloudogu.com@741e0ab'. It includes the text 'Changeset 1cdc3a9 was committed 2 months ago' and 'Authored by Johannes Schnatterer and committed by 🤖'. To the right of the commit message are two buttons: 'Details' and 'Sources'. A small circular icon is also visible.

 [cloudogu/gitops-build-lib](#) 

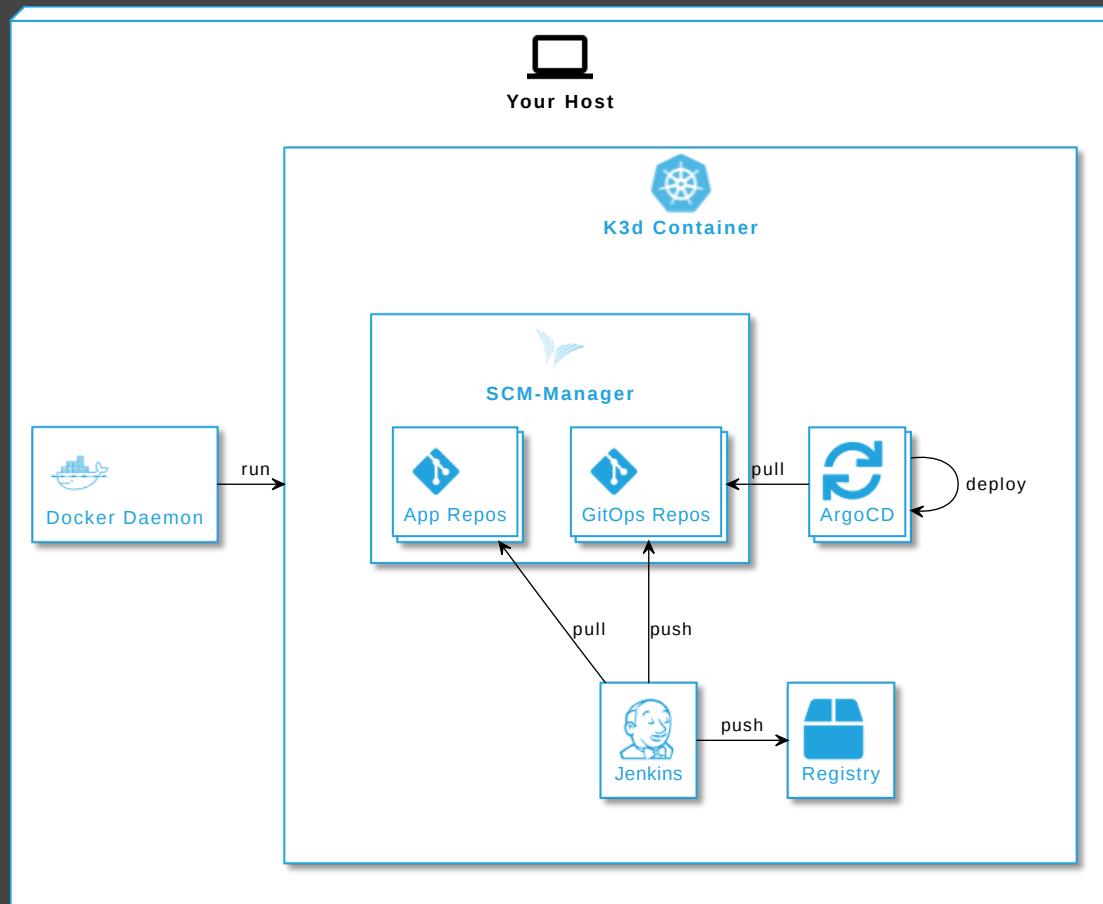
Local development

- 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

Demo



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

CONCLUSION



Personal Conclusion

After migrating to and operating with GitOps in production for > 1 year

- Smoother CI/CD,
 - *everything* declarative
 - faster deployment
- But: security advantages only when finished migration

GitOps experience distilled

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

Adopt GitOps?

- Greenfield
 - AppOps: Definitely
 - ClusterOps: Depends
- Brownfield: Depends

Johannes Schnatterer, Cloudogu GmbH

 cloudogu.com/gitops

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



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