// GITOPS - IS THIS SOMETHING FOR ME?

Johannes Schnatterer, Cloudogu GmbH

@jschnatterer

Version: 202204051711-c0d5fc3

Agenda

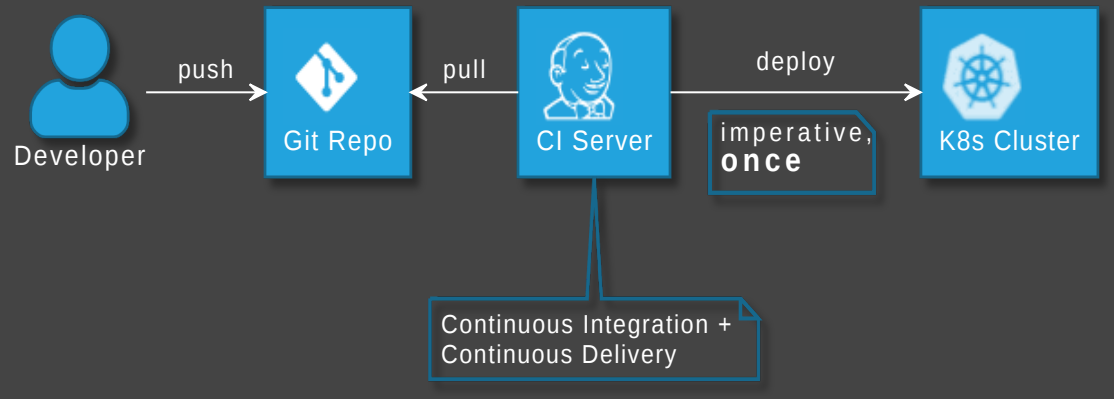
- Basics
- Tools
- Challenges
- Maturity

GitOps basics

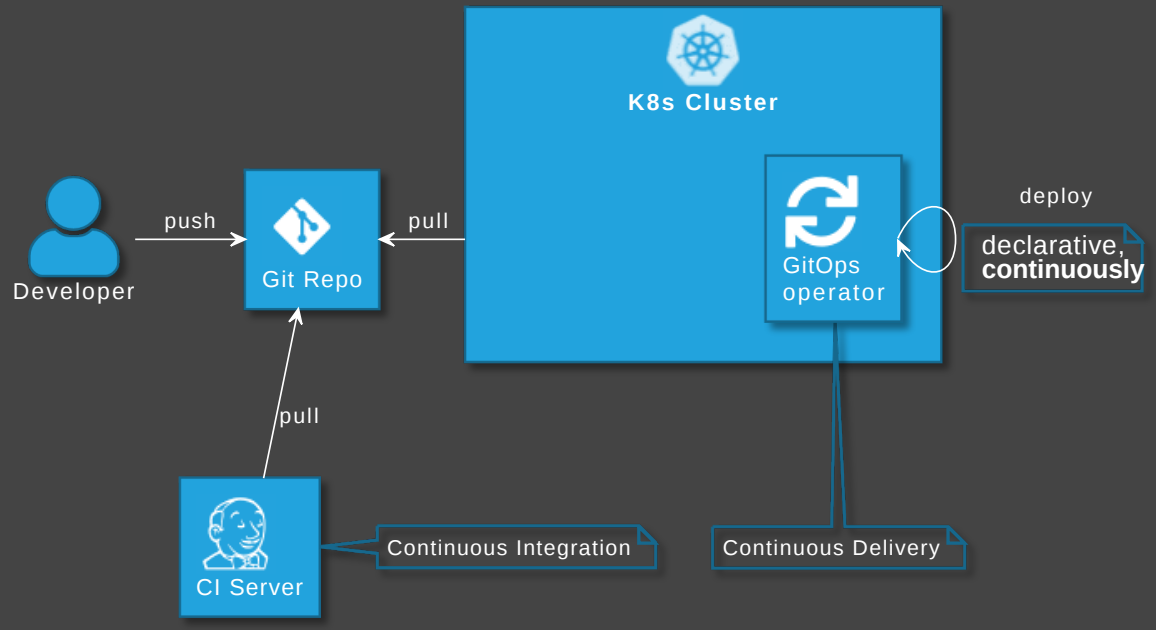
Origin: blog post by Weaveworks, August 2017

> 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

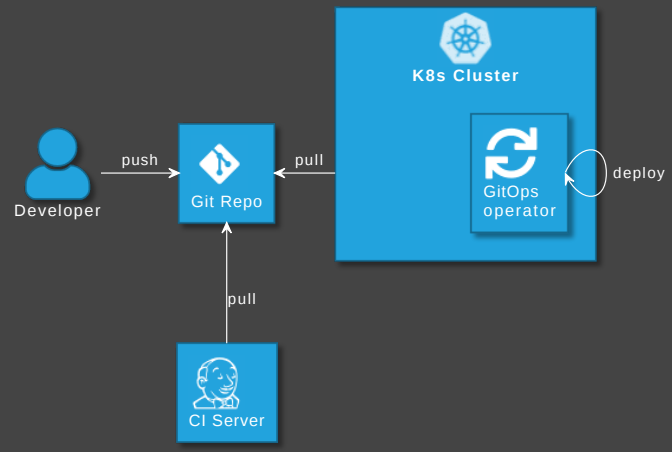
GitOps vs DevOps

- 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...

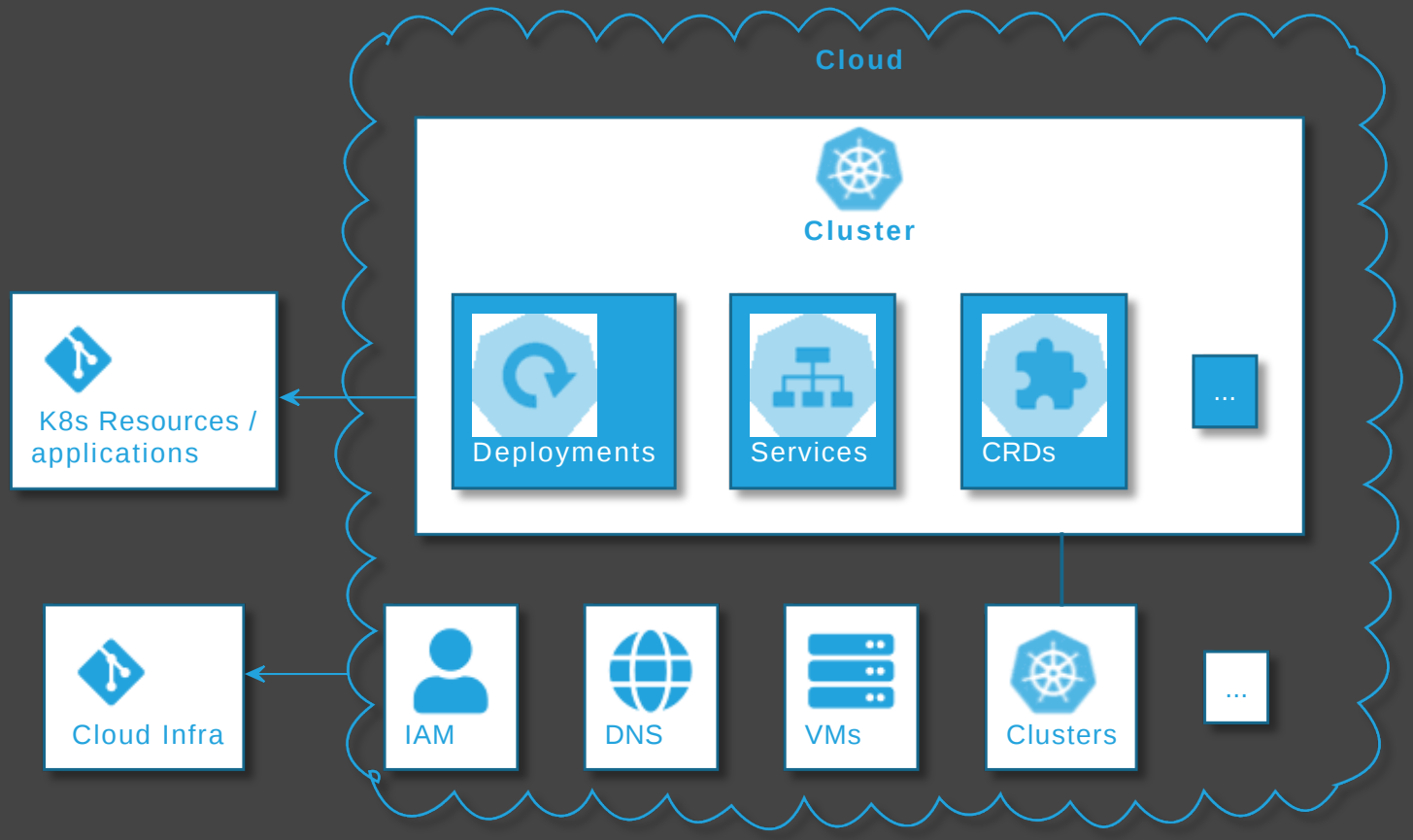
> The right way to do DevOps
> ☒ Alexis Richardson

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

Secrets - Ways of storing secrets

- Store Secrets in Repo (encrypted/sealed)
- Store Secrets in Key Management System (KMS)
  - Different KMS
    - Proprietary KMS: ...
    - Hashicorp Vault
  - Different K8s Integrations
    - Operator
    - Container Storage Interface (CSI) driver
    - Side car (injector)
    - Helm/Kustomize plugin
    - GitOps Operator: native support or plugin

Secrets - Tools

- bitnami-labs/sealed-secrets
- mozilla/sops + K8s integration
  - isindir/sops-secrets-operator
  - jkroepke/helm-secrets (plugin)
  - viaduct-ai/kustomize-sops (plugin)
  -  flux v2 (native support)
-  argoproj-labs/argocd-vault-plugin
-  kubernetes-sigs/secrets-store-csi-driver
- external-secrets/external-secrets
- hashicorp/vault-k8s (sidecar injector)

Others

- Backup / **restore**
- Deployment Strategies - Progressive Delivery

- ...

➡ GitOps ♥ operators

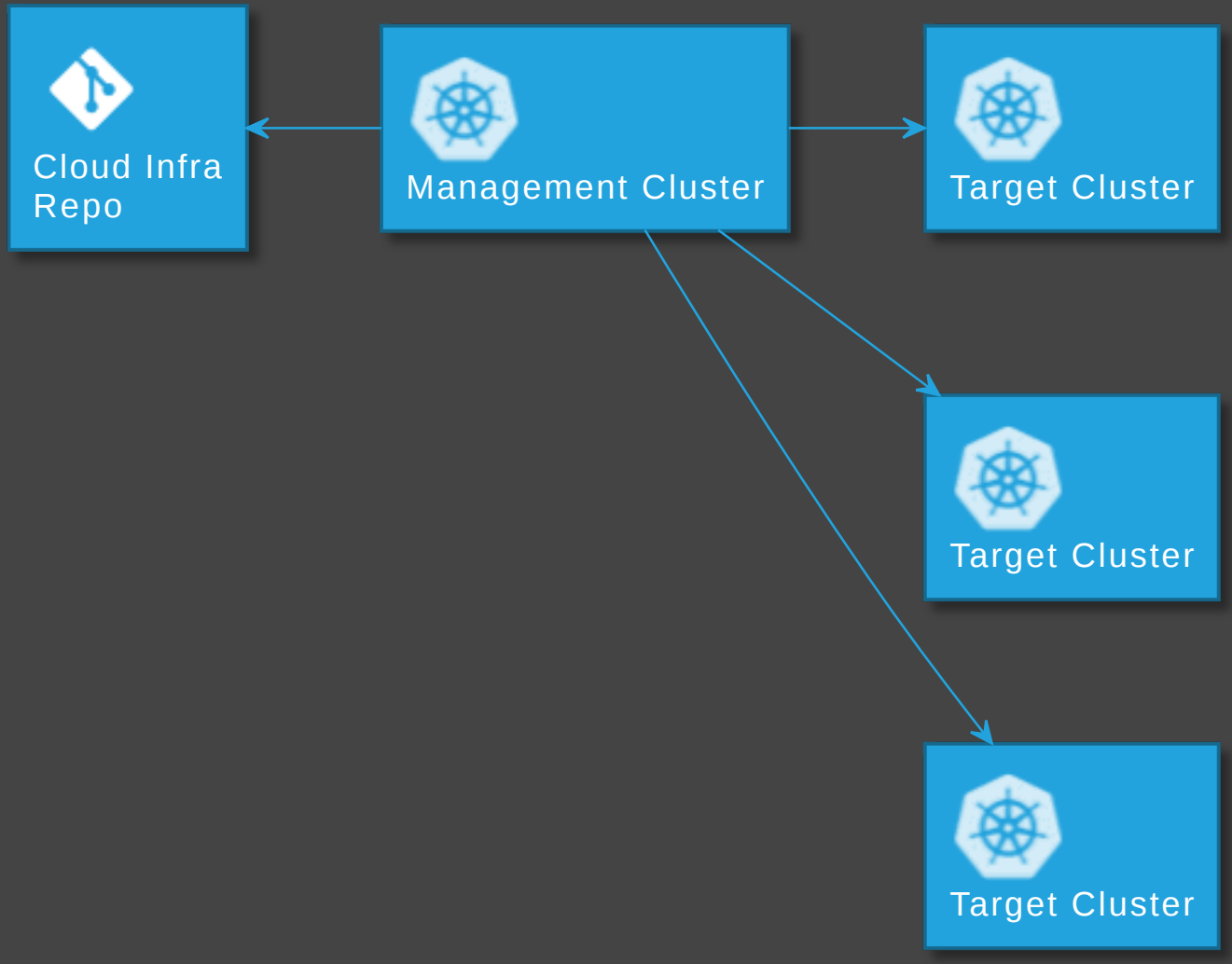
+

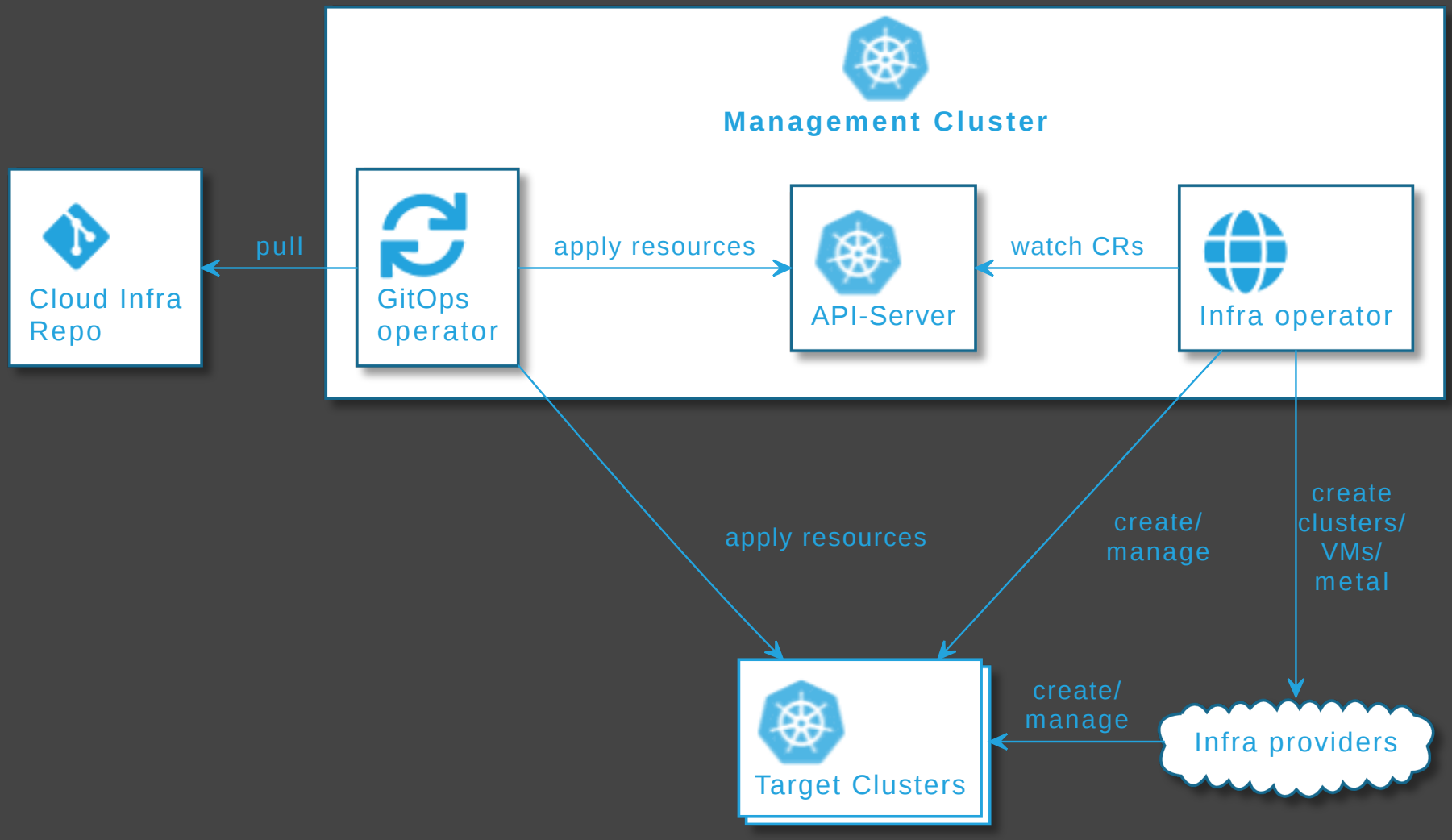
Infra Operator

=

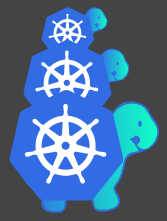
Operate cloud infra with GitOps

Operate Kubernetes with Kubernetes




Management Cluster
Cloud Infra Repo
pull
GitOps operator
apply resources
API-Server
watch CRs
Infra operator
apply resources
create/ manage
create clusters/ VMs/ metal
Target Clusters
create/ manage
Infra providers

Tools for operating cloud infra




Terraform + GitOps

Terraform Cloud or K8s Operator

-
- weaveworks/tf-controller
- 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 and Flux

Challenges 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

- Logic for branching complicated (merges)
- Gets even more difficult with more stages

Idea 2: Staging folders

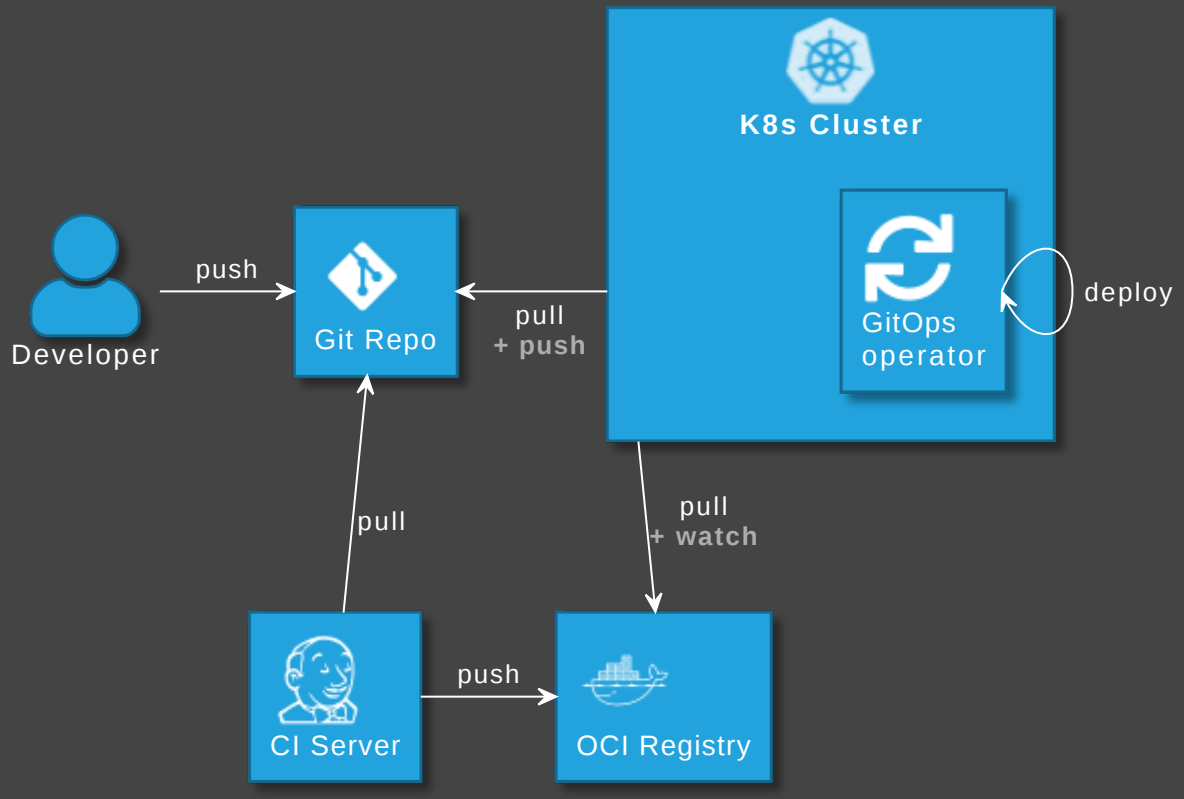
- On the same branch: One folder per stage

```
├── 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 automized

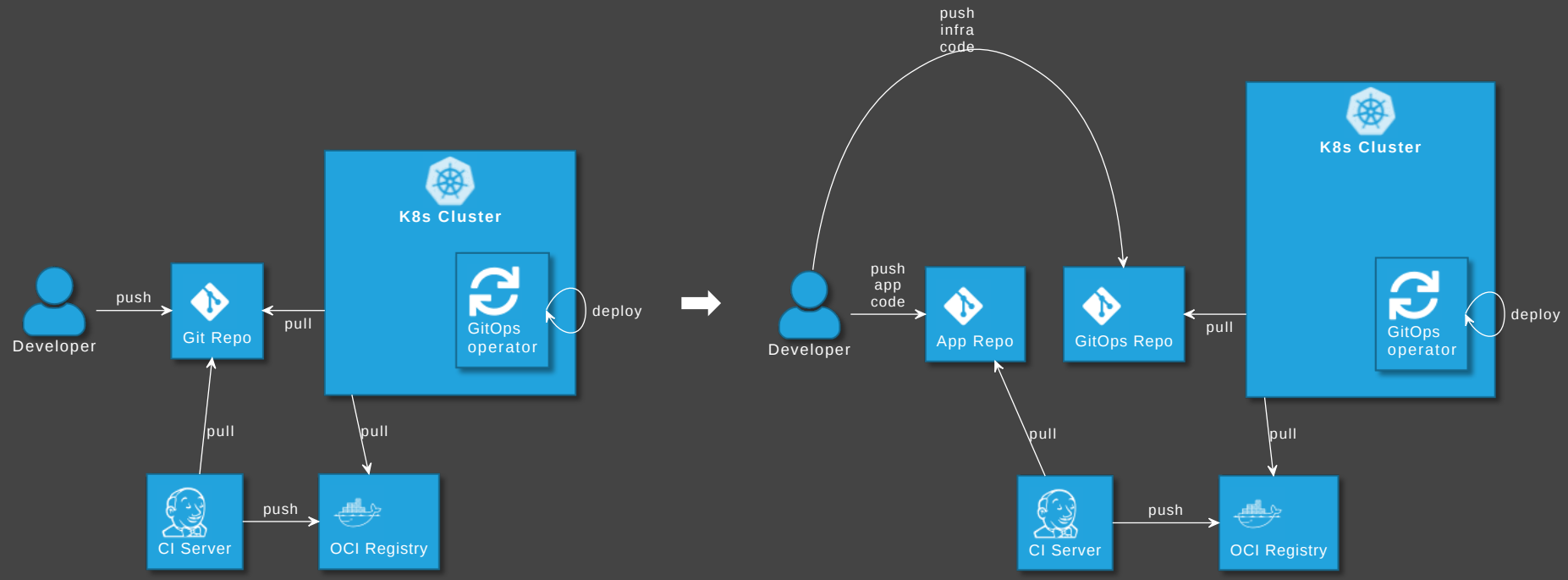
- Logic for branching simpler
- Supports arbitrary number of stages

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

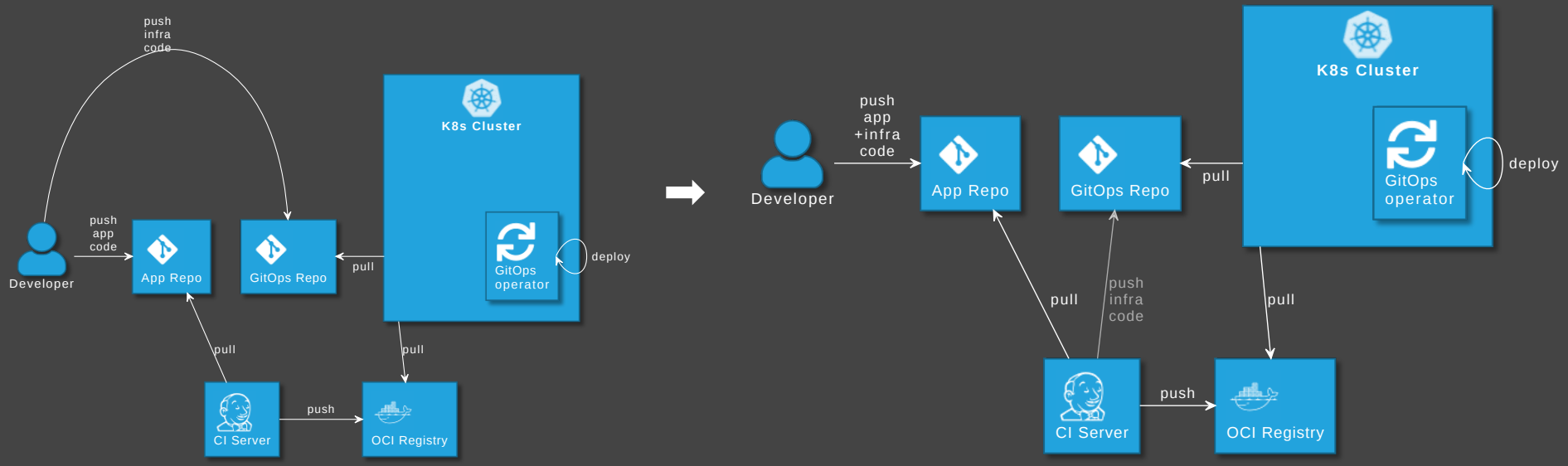
argo-cd.readthedocs.io/en/release-2.0/user-guide/best_practices

Disadvantages

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

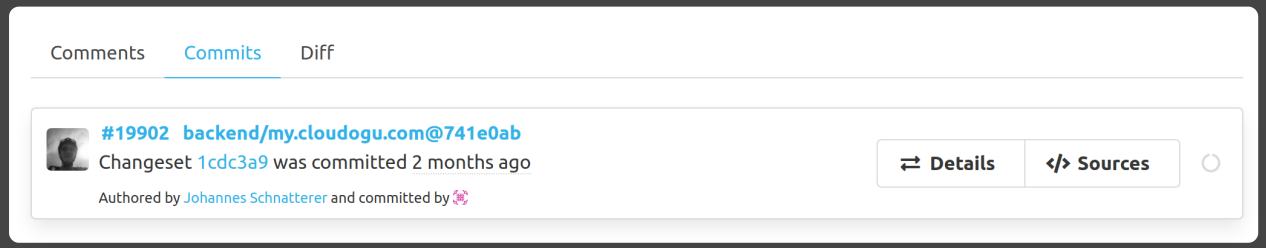
How to avoid those?

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


GitOps maturity

TECHNOLOGY RADAR

Download Subscribe Search Build your Radar About

Techniques

GitOps

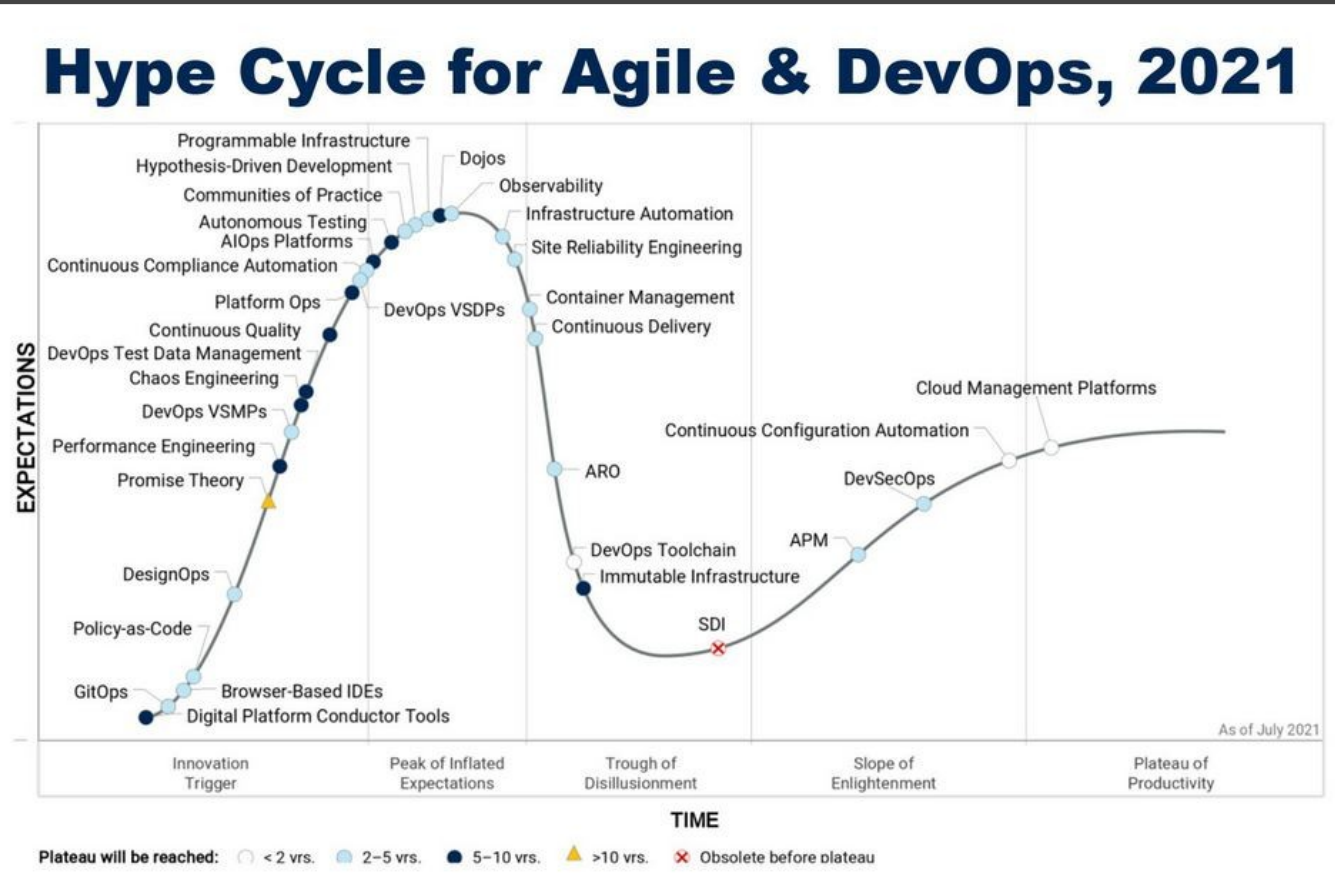
Published: Apr 13, 2021

APR 2021

HOLD

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



linkedin.com/pulse/hype-cycle-agile-devops-2021-joachim-herschmann/

There are the challenges, but

- Mature tools
  - very active tool development
  - ArgoCD and Flux CNCF graduation ahead
  - Lots of new tools and integrations emerging, including platforms
- Vibrant community
  - increasing adoption
  - several dedicated GitOps conferences:
    GitOps Days, GitOps Con, GitOps Summit, Mastering GitOps
- I have used GitOps successfully in production for years

My GitOps experience distilled

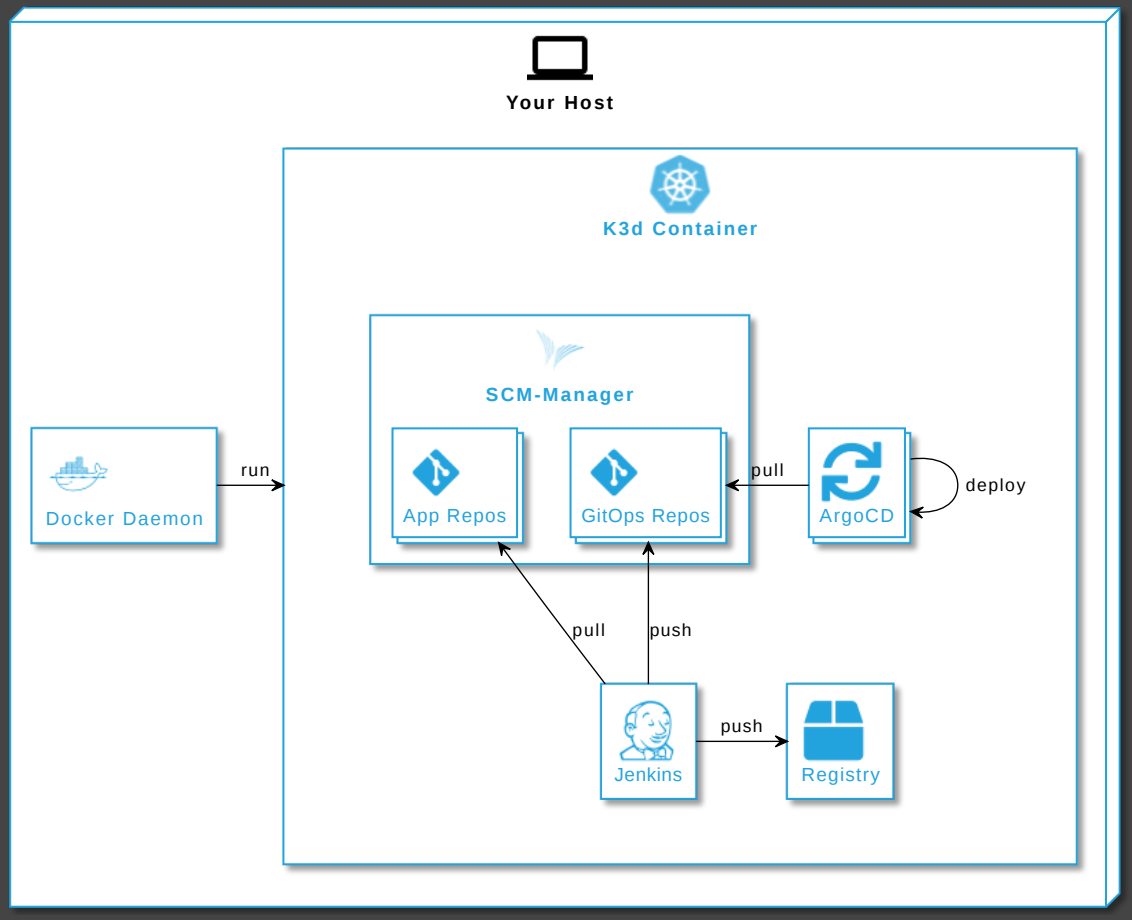
Has advantages, once established

Mileage for getting there may vary

Adopt GitOps?

- Greenfield: Definitely
- Brownfield: Depends

Hands-on



cloudogu/gitops-playground

Johannes Schnatterer, Cloudogu GmbH

 cloudogu.com/gitops

- ℹ GitOps Resources:
  articles, videos, projects, eBook
- Community
- Trainings / Consulting
- Jobs

Slides

Image sources

- Basics: https://pixabay.com/illustrations/question-mark-important-sign-1872665/
- Tools: https://pixabay.com/photos/tools-knives-wrenches-drills-1845426/
- Challenges: https://unsplash.com/photos/bJhT_8nbUA0
- GitMaturity: https://pixabay.com/photos/age-bacteria-bio-biology-blue-1238283/