

GitOps - FluxCD

Continues Multi Cluster Deployment

Was gibt es gleich zu sehen?

1. Motivation
2. Problem von Kubernetes Deployments
3. GitOps - FluxCD als mögliche Lösung
4. Demo
5. Zusammenfassung

Welches Problem adressiert GitOps?

1. Continuous Deployment (CD) in Cluster
2. **Change Management von Yaml Artefakten**
3. Continuous Integration (CI) in Multi Cluster Environment
4. Technologiebrüche bei der Beschreibung/Konfiguration der Cluster Deployments (Wiki vs Yaml vs Developer Gehirn)

>>>>>>>>>>> REPRODUZIERBARE DEPLOYMENTS <<<<<<<<<<<<

Welches PROS/CONS hat Kubernetes/Yaml?

CONS

- Verbose
- Keine Möglichkeit Variablen zu definieren
- Keine referenzielle Integrität (Abhängigkeiten werden zur Entwicklungszeit nicht überprüft)
- Schlechte IDE Unterstützung

CONS - Runtime

- Fail-late (Syntaxfehler werden erst zur Installationszeit bemerkt)
- Fail-late 2 (Semantikfehler werden erst zur Bootzeit bemerkt)

PROS

- Deklarativ
- Lesbar
- Kommentierbar (im Gegensatz zu Json)

PROS - Runtime

- Stabile Deployments werden erst deinstalliert, wenn nächste Version stabil läuft (Readiness Probe)

>>>>>>> Grosse YAML`ei <<<<<<<<<<

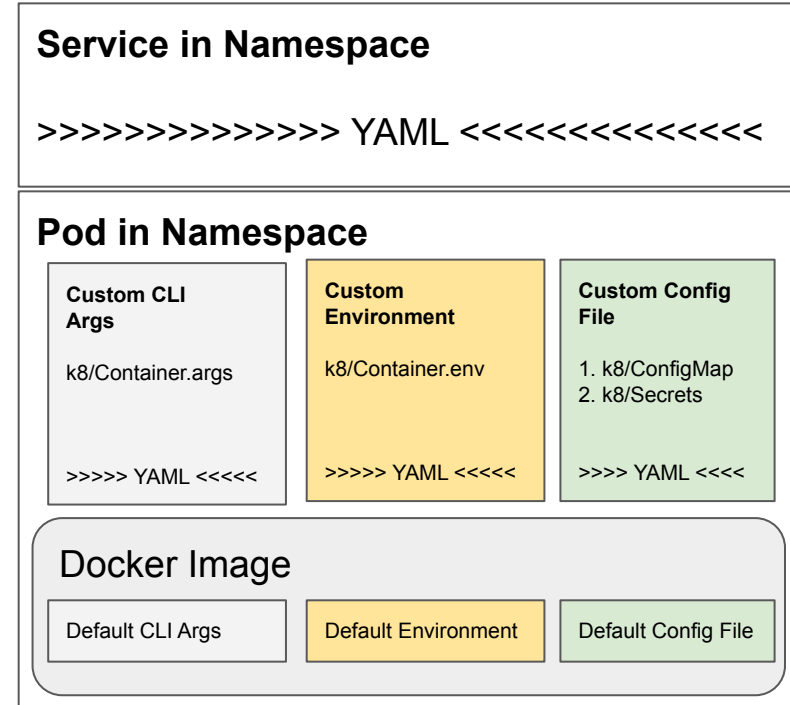
Deployment Artefakte (**Blau**) - Bsp.Spring Application

Development Artefakte	Command	Cluster Artefakte
Jar → application.yaml	mvn package → src/main/..	Environment ConfigMap Secret
Image	docker build&push → Dockerfile	PullPolicy ImagePullSecrets
HelmChart → values.yaml	helm create helm template	Git-Repository Controller HelmRepository Controller
Deployment-/Service.yaml HelmRelease → HelmRelease.yaml	kubectl apply helm install	Kustomize Controller HelmRelease Controller • create/update POD

>>>>>>> Grosse YAML`ei <<<<<<<<<<

Was ist ein Kubernetes Deployment?

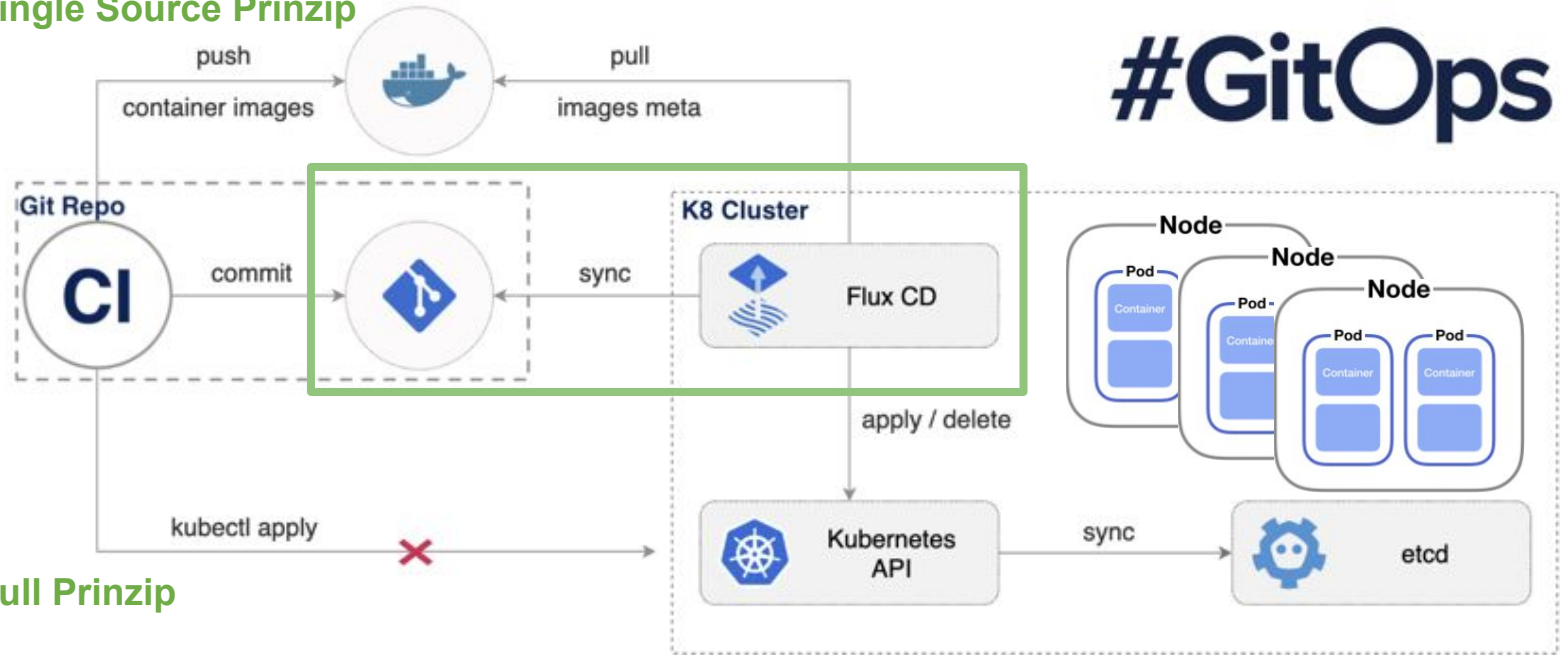
1. Docker Image
 - a. Binaries
 - b. Default CLI Args
 - c. Default Environment
 - d. Default Config File
2. Deployment/Service Konfiguration
 - a. Custom CLI
 - b. Custom Environment
 - c. Custom Config File
3. ConfigMap's
4. Secret's
5. Container.args
6. Container.env



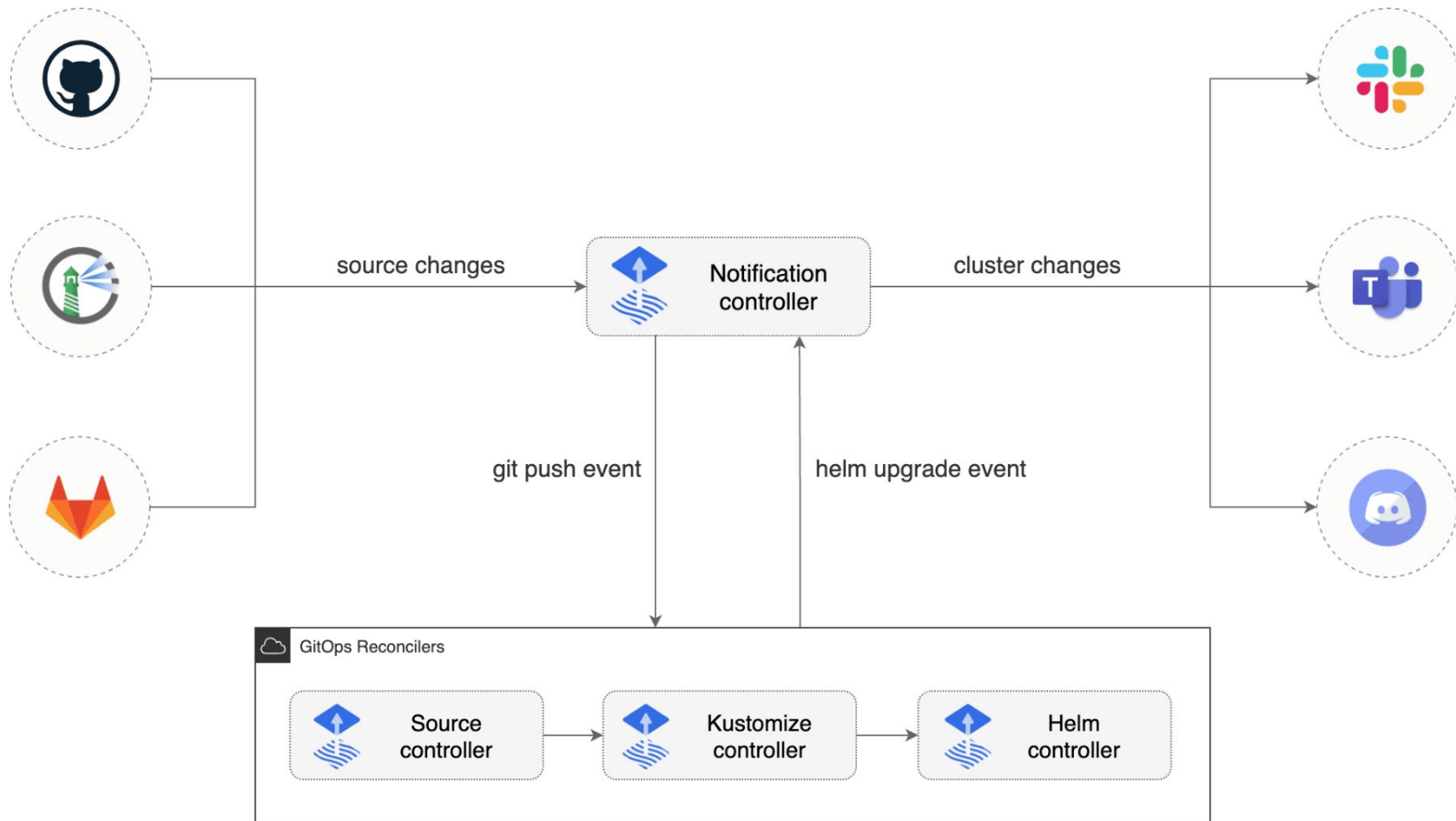
>>>>>>> Grosse YAML`ei <<<<<<<<<<

Was ist GitOps mit FluxCD?

1. Single Source Prinzip



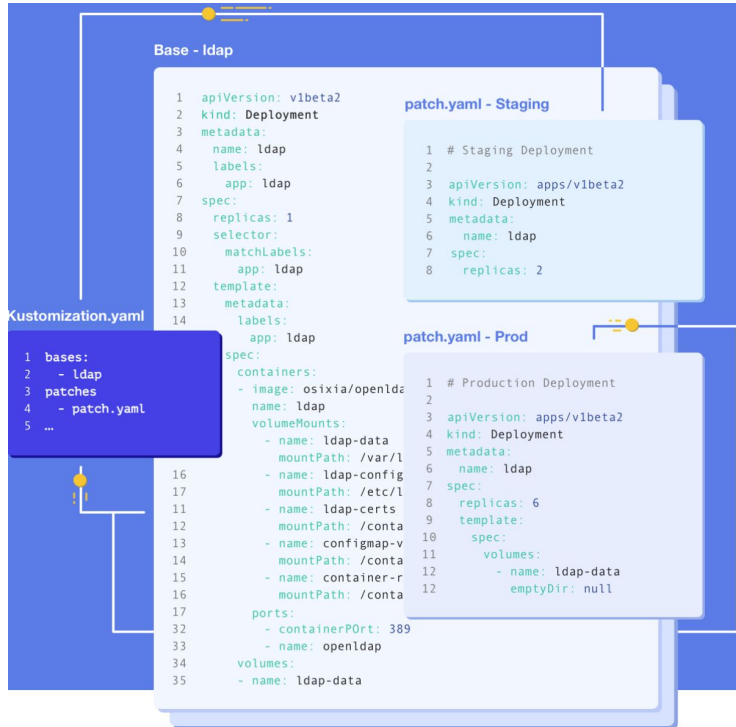
2. Pull Prinzip



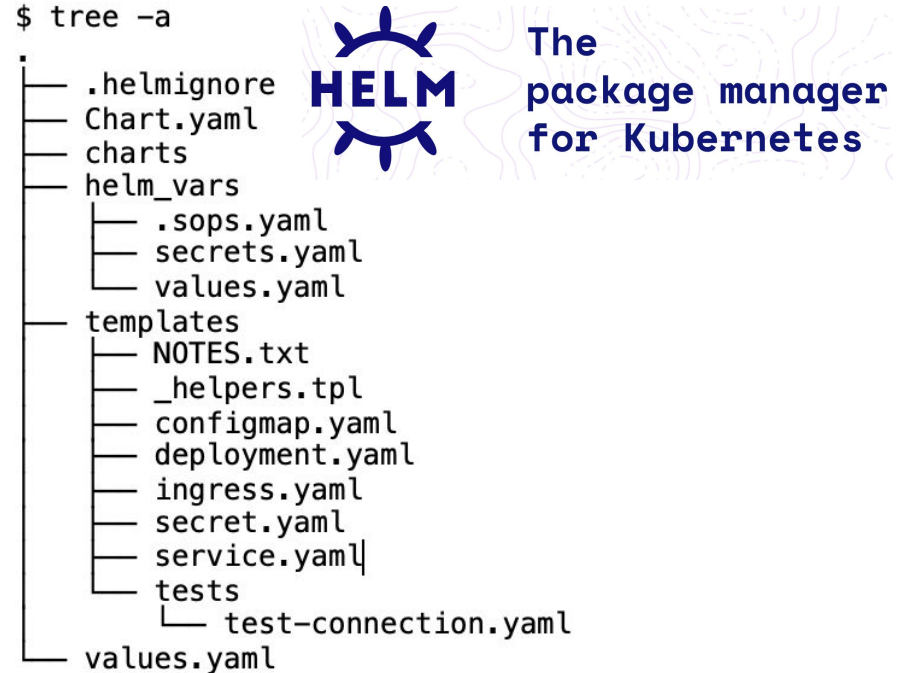
<https://fluxcd.io/docs/components/notification/>

FluxCD Technologien um Variablen zu verwenden

Kustomize (<https://kustomize.io/>)



Helm (<https://helm.sh/>)



Demo

Cluster(localhost:443)

Loadbalancer
Traefik

FluxCD UI
Weave

Route:
/

4x Whoami Installationen

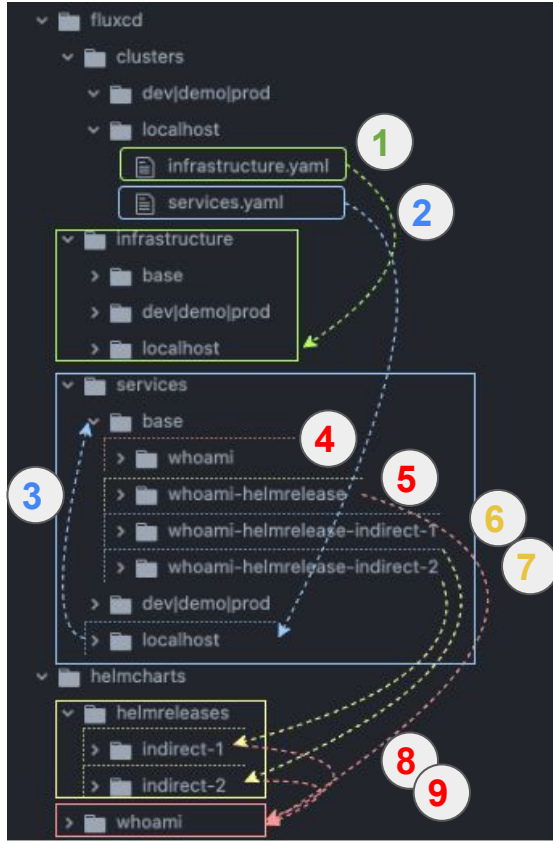
Routes:
/whoami

/whoami-helm2

/whoami-helm3

/whoami-helm4

FluxCD - Beispiel (4xWhoami)



1. Customize/Link to deploy Infrastructure
2. Customize/Link to deploy Services
3. Customize/Include from Service Deployments
4. **Deploy** Raw Whoami Yaml (1)
5. **Deploy** Helmrelease Whoami (2)
6. Customize/Link to Helmrelease Deployment (3)
7. Customize/Link to Helmrelease Deployment (4)
8. **Deploy** Helmrelease Whoami (3)
9. **Deploy** Helmrelease Whoami (4)

Zusammenfassung

- Reproduzierbare Deployments sind die Regel
- Der Preis dafür ist noch mehr Yaml
- Kein heiliger Gral
- Vendor Login
- Abhängigkeiten und Variablen existieren

Missing features:

- Interactive UI wie bei ArgoCD
- Support für andere DSLs (z.B. dhall)

Fragen

Links

demo repo <https://github.com/jwausle/gitops.fluxcd>

slides <https://docs.google.com/presentation/d/1nFYMIjM28u0hPfmOMB0TP2D9Mj3IRZ6Wc83VBjx6tmM/edit?usp=sharing>

fluxcd doc <https://fluxcd.io/>

traefik loadbalancer <https://doc.traefik.io/traefik/>

weave ui <https://github.com/weaveworks/weave-gitops>