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 Kubernetes Cluster
2. **Change Management von Kubernetes Yaml Artifakten**
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 Unterstüzung

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 <<<<<<<<<

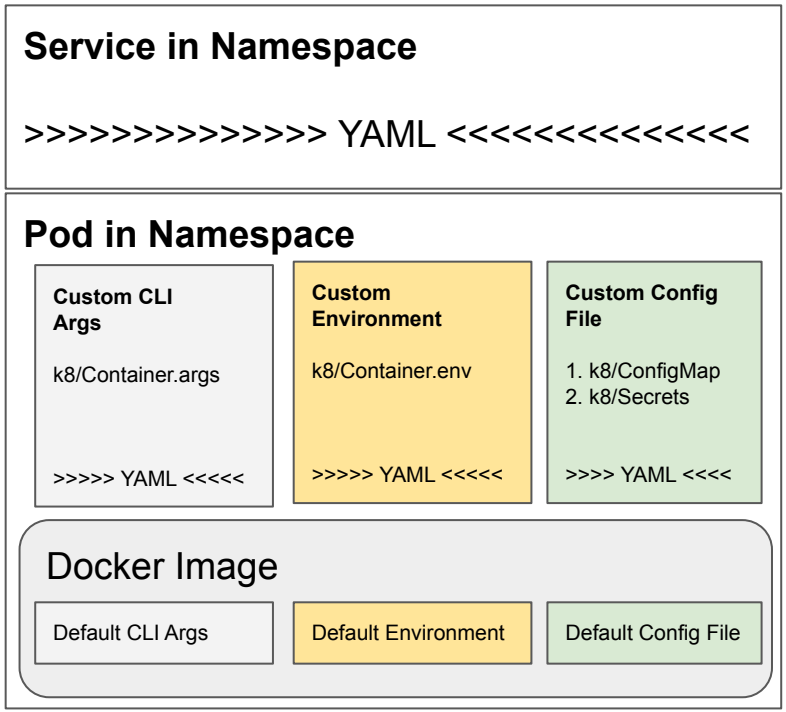
Deployment Artefakte (**Blau**) - Bsp.Spring Application

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?

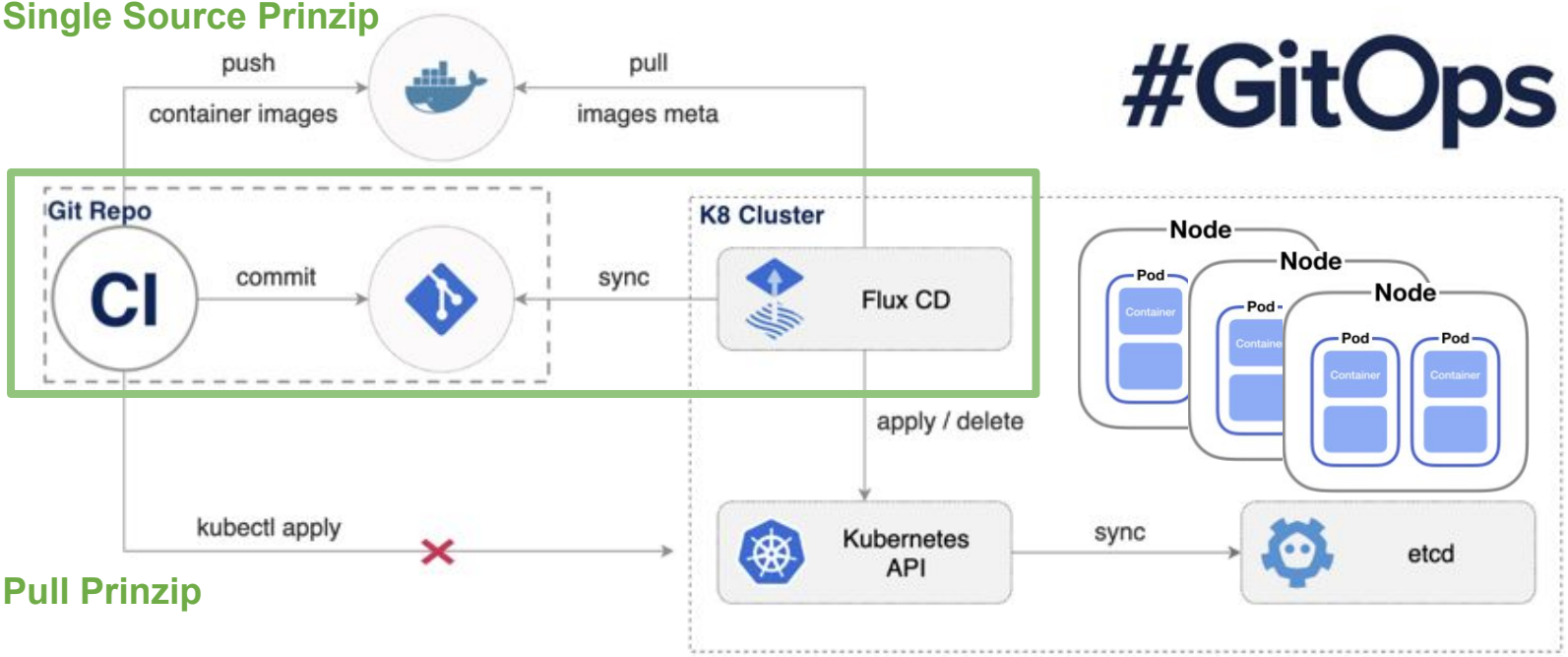
1. Docker Image
   a. Binaries
   b. Default CLI Args
   c. Default Environment
   d. Default Config File
2. Deployment/Service Konfiguration
   a. Custom CLI
   b. Custom Environment
   c. Custom Config File
3. ConfigMap's
4. Secret's
5. Container.args
6. Container.env



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

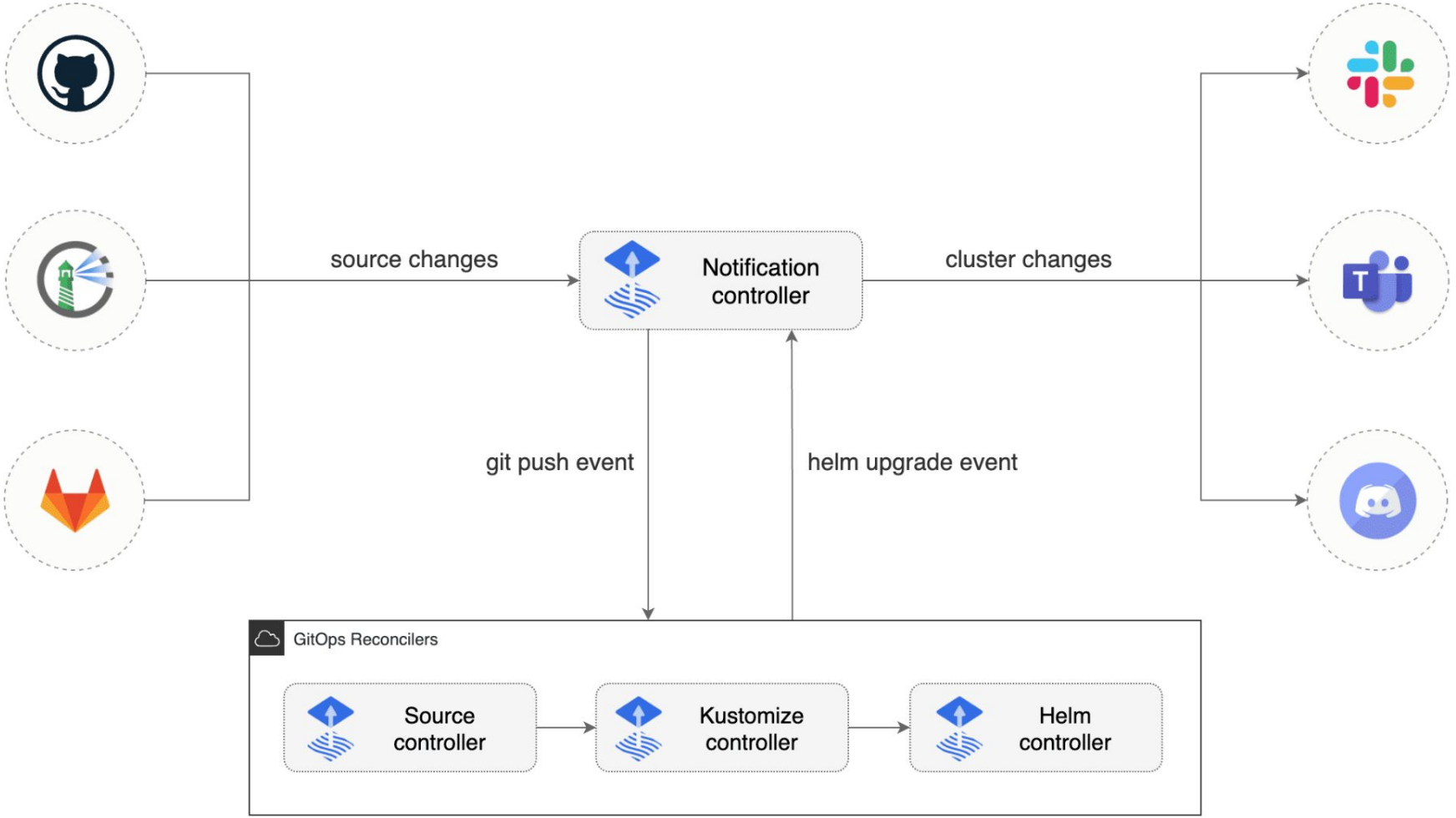
Was ist GitOps mit FluxCD?

1. **Single Source Prinzip**



2. **Pull Prinzip**

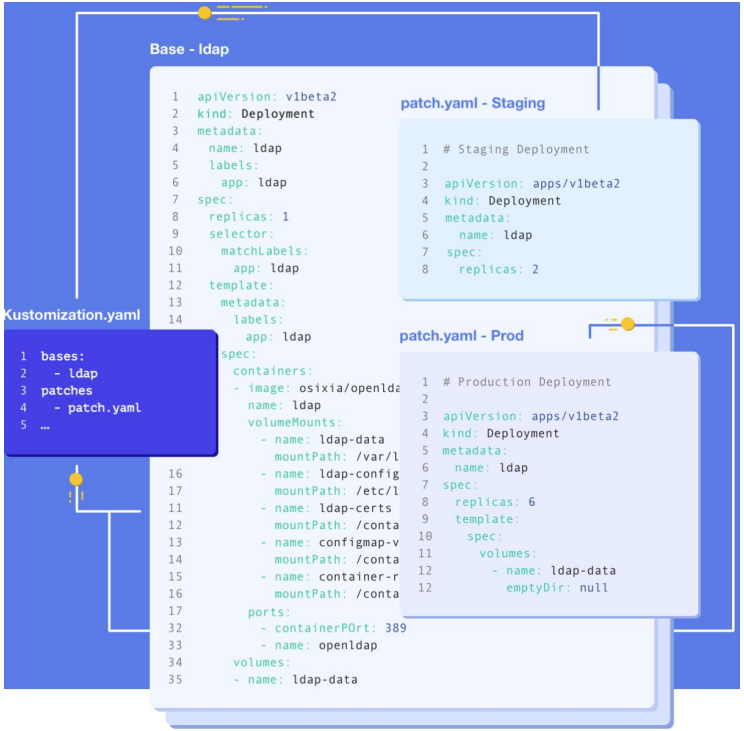
https://fluxcd.io/legacy/flux/



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

FluxCD Technologien um Variablen zu verwenden

Kustomize (https://kustomize.io/)



Helm (https://helm.sh/)

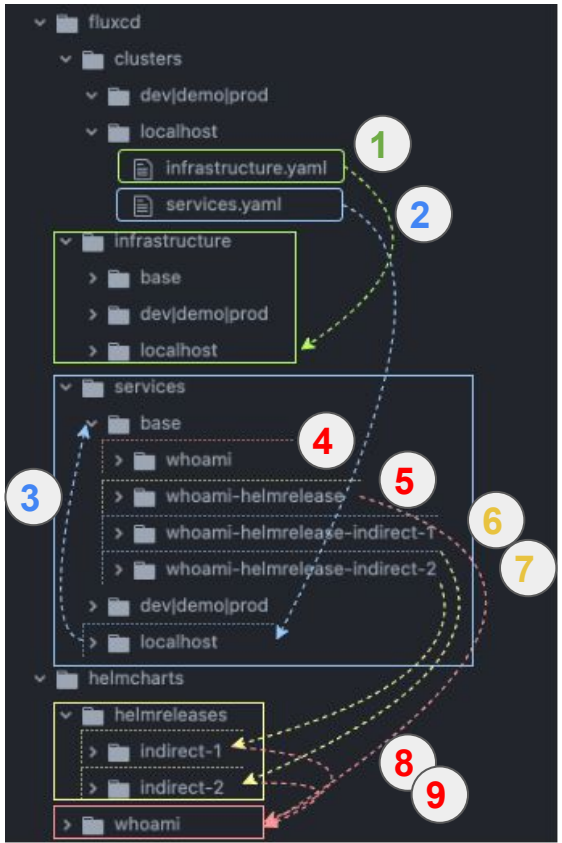
```
$ tree -a
.
├── .helmignore
├── Chart.yaml
├── charts
├── helm_vars
│   ├── .sops.yaml
│   ├── secrets.yaml
│   └── values.yaml
├── templates
│   ├── NOTES.txt
│   ├── _helpers.tpl
│   ├── configmap.yaml
│   ├── deployment.yaml
│   ├── ingress.yaml
│   ├── secret.yaml
│   ├── service.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml
```

HELM

The package manager for Kubernetes

Demo

FluxCD - Beispiel (4xWhoami)



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

Zusammenfassung

- Kein heiliger Gral
- Vendor Login ist hoch
- Reproduzierbare Deployments sind die Regel
- Der Preis dafür ist noch mehr Yaml

Missing features:

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

Fragen