GitOps

Reproduzierbare Deployments durch GitOps

Agenda

- 09-12 - Workshop* GitOps
- 12-13 - Mittag
- 13-16/17 - Workshop* GitOps

*... Pausen werden frei gewählt

Ziele der Schulung

Scope

Was ist GitOps?

Was ist Kustomize/Helmchart?

Was ist FluxCD?

Was ist ArgoCD?

Was sind GitOps Repository Strukturen?

Out of Scope

Secrets Handling

Keine Git Schulung (wird vorausgesetzt)

Keine Kubernetes Schulung (wird vorausgesetzt)

Keine Helmchart Schulung (wird vorausgesetzt)

Keine Docker Schulung (wird vorausgesetzt)

Jan Winter

- 48 Jahre, aus Leipzig
- Seit ~20 Jahren Software Entwickler
  - Seit 6 Jahren Freiberufler
  - 7 Jahre Berater bei Itemis AG
  - 5 Jahre in einem Startup
  - 3 Jahre Freiberufler während des Studiums
- Seit ~10 Jahren Trainer (Nebentätigkeit)
  - Kubernetes/Docker, GitOps/FluxCD/ArgoCD
  - Git, Gitlab CI, Bitbucket CI
  - OSGi

Welches Problem adressiert GitOps?

1. **Reproduzierbare Kubernetes Deployments**
2. Continuous Deployment (CD) in Kubernetes Cluster
3. **Change Management von Yaml Artefakten**
4. Continuous Integration (CI) in Multi Cluster Environment

Brief History of GitOps (Git, Docker, Kubernetes, FluxCD, ArgoCD)

2005 - Linux Kernel search a new SCM and decide to develop Git

2006 - cgroups for linux introduced by google

2013 - Docker first official release based on cgroups

2015/16 - Google übergibt Kubernetes Cloud Native Computing Foundation (CNCF)

2016 - ArgoCD was born (Company: Intuit)

2016/2017 - Flux1 v.1.0.0 release (Company: Weaveworks)

2019 - ArgoCD v1.0.0 released

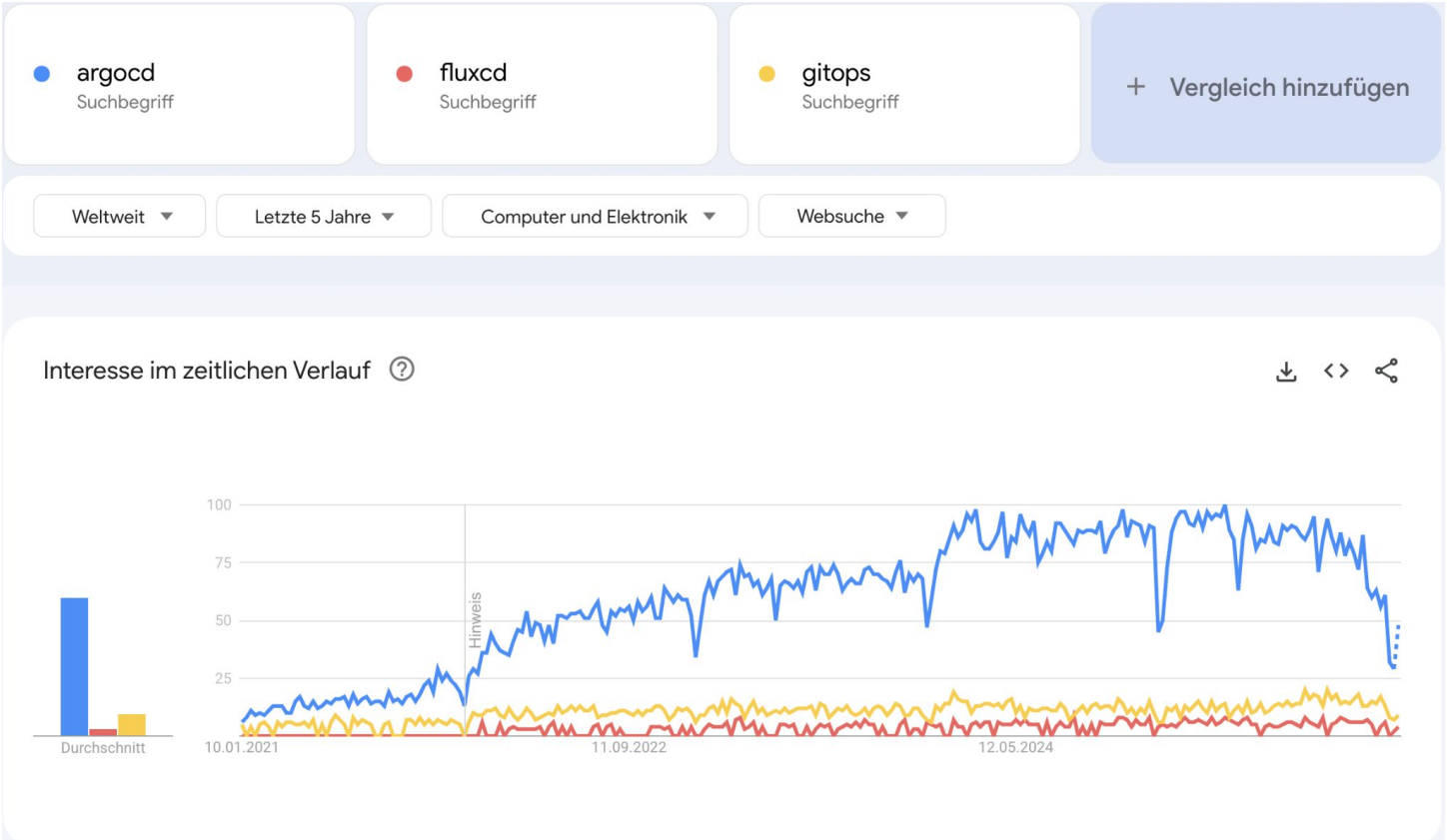
2019 - Mirantis kauft Docker Enterprise

2020 - Flux2 was born

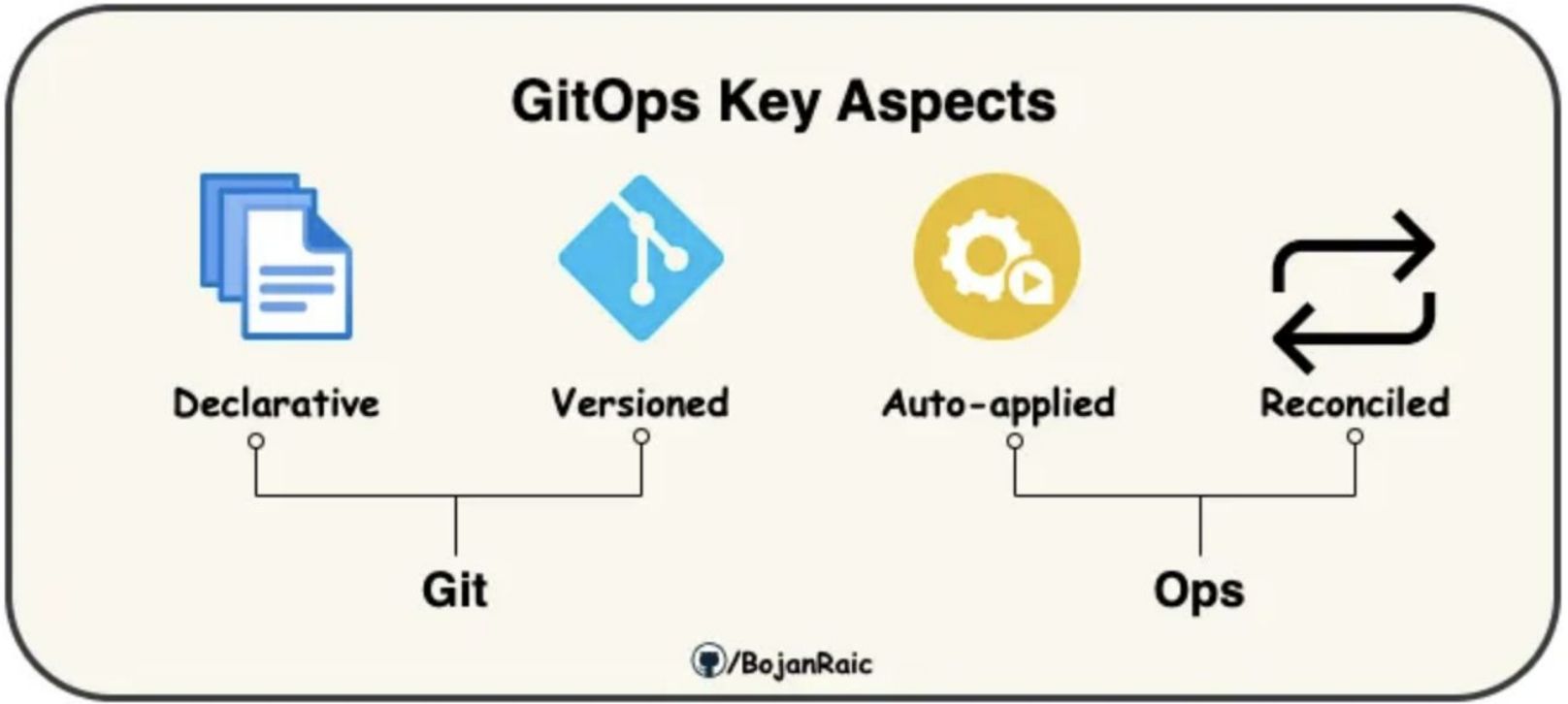
2023 - Flux2 v2.0.0 released

2024 - Weaveworks übergibt Flux2 an die Community

Google Trend - ArgoCD vs FluxCD

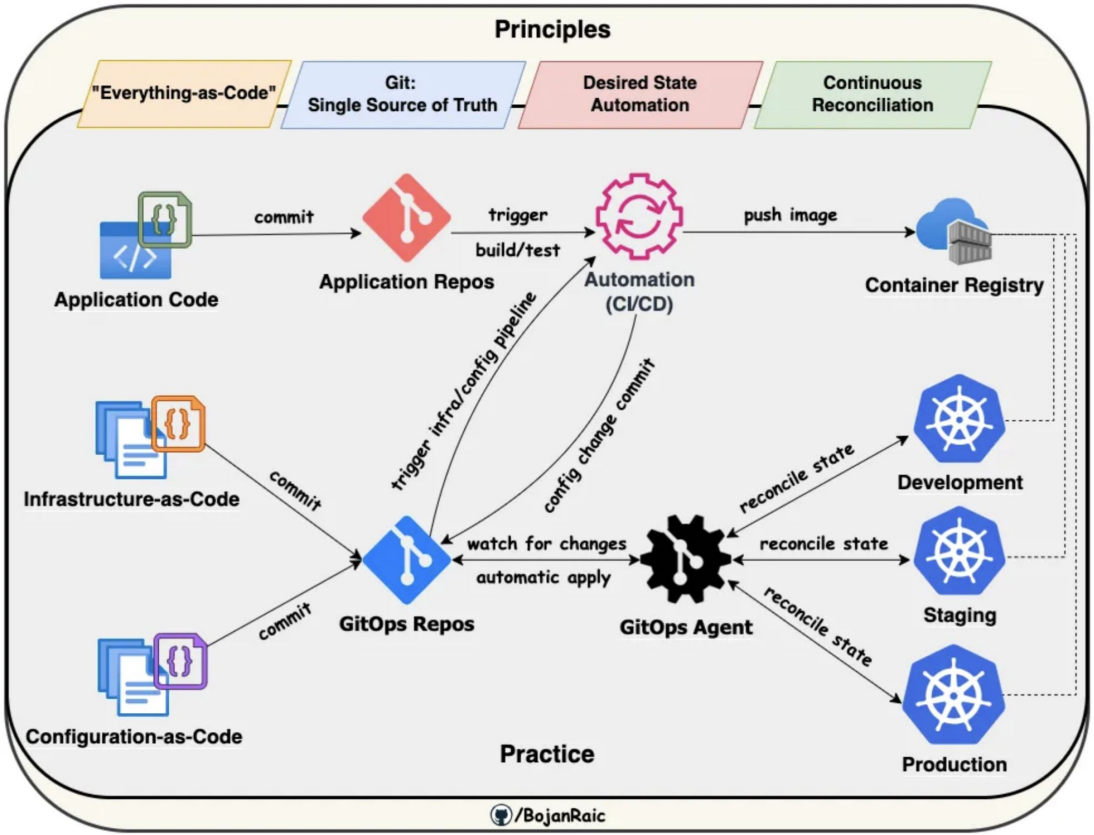


Was ist die Idee von GitOps?



Was ist die Idee von GitOps?

1. Everything-as-Code
2. Single Source of Truth
3. State Automation
4. Continuous Reconciliation



Was ist die Idee von GitOps?

Hollywood principle

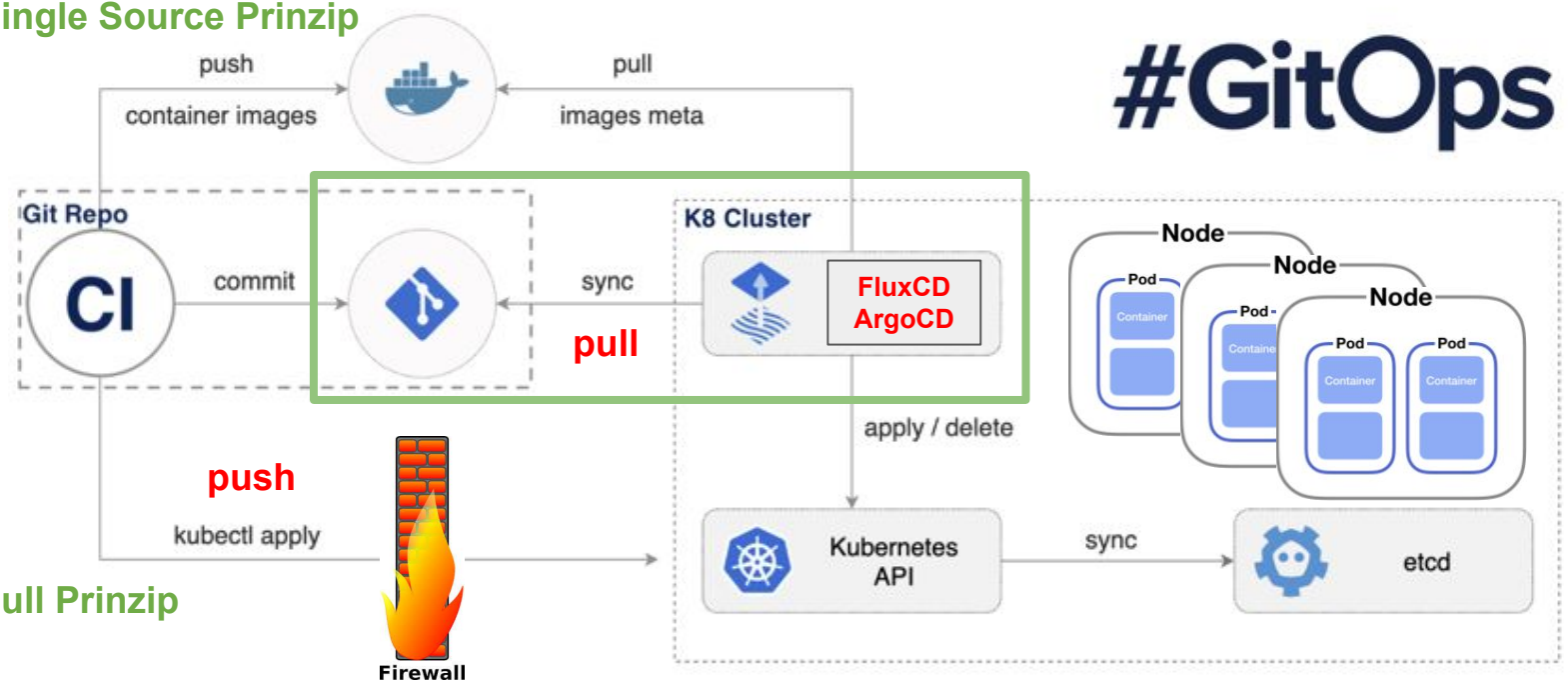
- Don't call me
- I call You

Divide and Conquer

- CI vs CD

Was ist die Idee von GitOps?

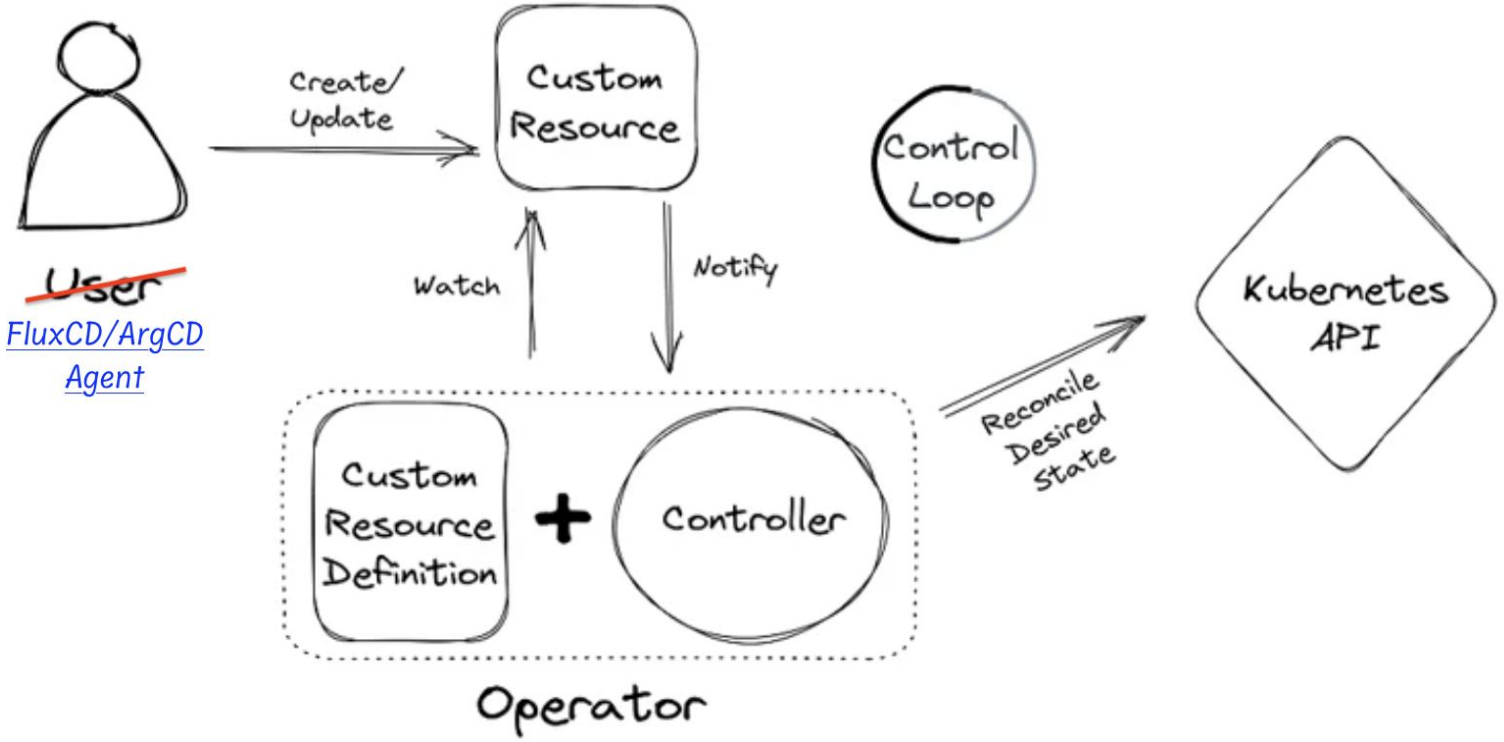
1. **Single Source Prinzip**



2. **Pull Prinzip**

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

Was ist die Idee von GitOps?



Warum ist GitOps entstanden?

Grosse YAML`ei

Kubernetes API Datenformate sind

- Yaml - "application/x-yaml"
- Json - "application/json” (no comments)

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)

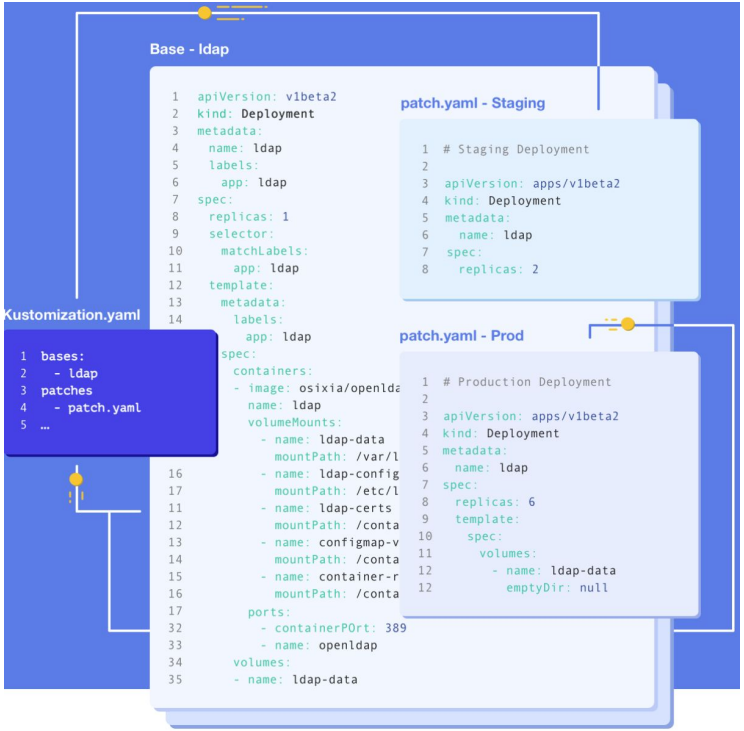
PROS - Runtime

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

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

GitOps Basis 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

GitOps Basis Technologien Kustomize/Helm

	ArgoCD	**FluxCD**
Deployment Helmchart	Yes	No
Basistechnologie	Kustomize	Kustomize
Kustomize Support	Application.kustomize	Kustomization
Helmchart Support	Application.helm	Helmrelease
HelmRepository Support	Application.chart	HelmRepository
UI	Integrated UI	Optional: Weave UI
Easy to use	Easier for GitOps beginners	CLI Focused

Demo - Setup

```
git clone https://jwausle-demo1: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@github.com/jwausle-demo1/gitops.git
```

```
git clone https://jwausle-demo2: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@github.com/jwausle-demo2/gitops.git
```

Demo - Required Tools

Required:

Docker[Desktop] - https://docs.docker.com/desktop/setup/install/

Kubectl - https://kubernetes.io/de/docs/tasks/tools/install-kubectl/#installation-der-kubectl-anwendung-mit-curl

Git - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

FluxCD **v2.7.5** - https://github.com/fluxcd/flux2/releases/tag/v2.7.5

Optional:

Freelens - https://freelensapp.github.io/

K9s - https://k9scli.io/topics/install/

ArgoCD - https://argo-cd.readthedocs.io/en/stable/cli_installation/

Kustomize - https://github.com/kubernetes-sigs/kustomize/releases/tag/kustomize%2Fv5.8.0

Demo - Setup preconfigured AWS1 (demo1)

ec2-18-192-53-212.eu-central-1.compute.amazonaws.com

```
cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3N...5PSgP97yxHQ== You@YourMachine

# When: No such file or directory
ssh-keygen      # follow the instructions and remember the passphrase

cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3N...5PSgP97yxHQ== You@YourMachine
```

1. Send ‘ssh-rsa … ‘ to me

```
aws1>  echo “ssh-rsa … ” > ~/.ssh/authorized_keys
```

2. Login into AWS1

```
ssh ubuntu@ec2-18-192-53-212.eu-central-1.compute.amazonaws.com
aws1> cd schulung/gitops
aws1> git status
```

Demo - Setup preconfigured AWS2 (demo2)

ec2-18-159-224-241.eu-central-1.compute.amazonaws.com

```
cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3N...5PSgP97yxHQ== You@YourMachine

# When: No such file or directory
ssh-keygen      # follow the instructions and remember the passphrase

cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3N...5PSgP97yxHQ== You@YourMachine
```

1. Send ‘ssh-rsa … ‘ to me

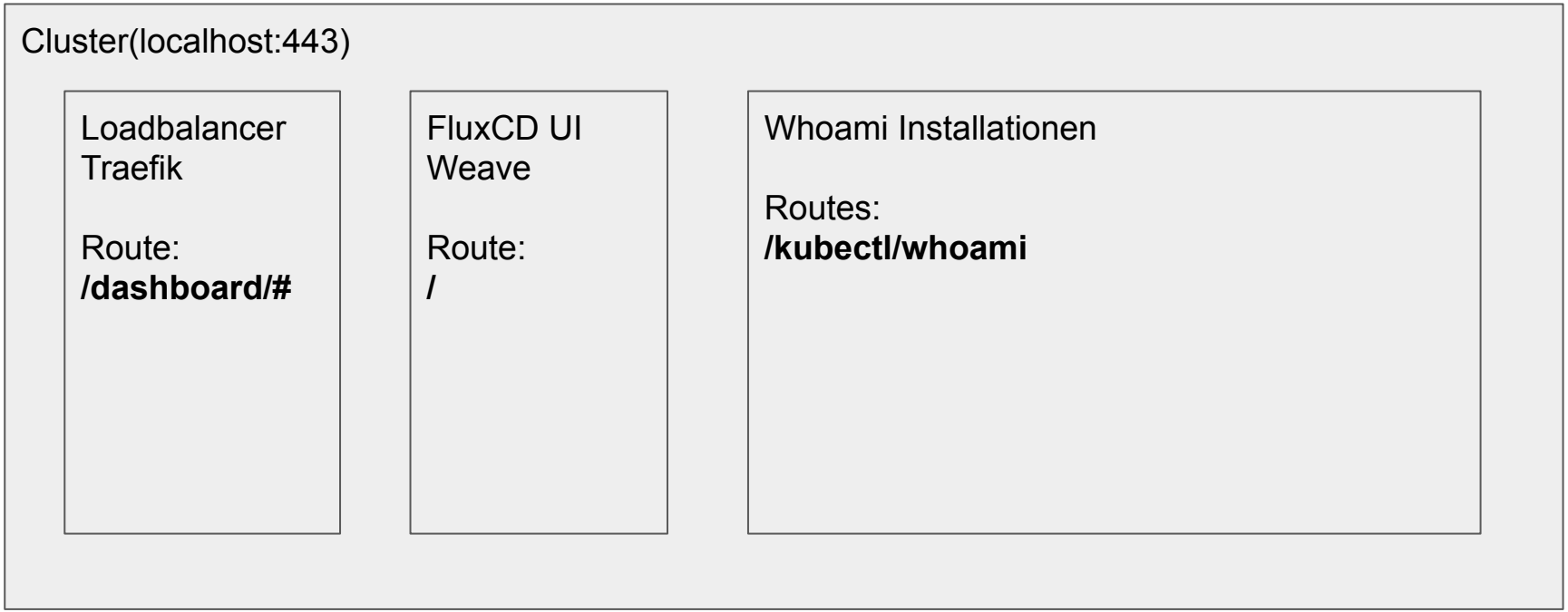
```
aws2>  echo “ssh-rsa … ” > ~/.ssh/authorized_keys
```

2. Login into AWS2

```
ssh ubuntu@ec2-18-159-224-241.eu-central-1.compute.amazonaws.com
aws2> cd schulung/gitops
aws2> git status
```

Demo - Kubectl

Demo - Kubectl



Demo - Kubectl

Login into AWS(1|2)

```
ssh ubuntu@aws…
aws ~> cd schulung/gitops
aws ~ schulung/gitops> git status
```

Start Cluster

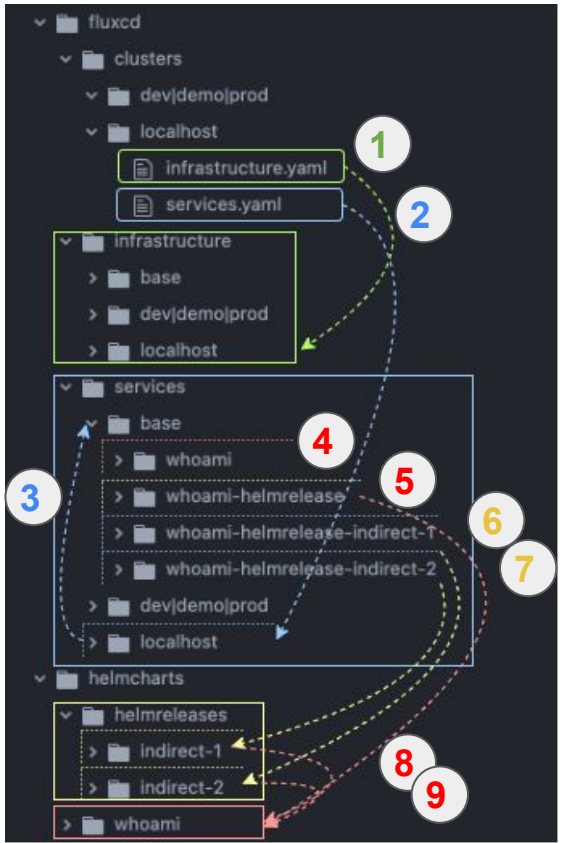
```
aws ~ schulung/gitops> bash scripts/run-cluster-local.sh

export KUBECONFIG=$(pwd)/.k3s/kubconfig.yaml
```

Deploy Workload

```
aws ~ schulung/gitops> bash kubectl/deploy-traefik.sh     # /dashboard/#
aws ~ schulung/gitops> bash kubectl/deploy-weave.sh     # /
aws ~ schulung/gitops> bash kubectl/deploy-whoami.sh  # /kubectl/whoami/test
```

Demo - Flux CD

Demo - FluxCD



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)

https://github.com/fluxcd/flux2-multi-tenancy

Demo - Kubectl

Login into AWS(1|2)

```
ssh ubuntu@aws…
aws ~> cd schulung/gitops
aws ~ schulung/gitops> git status
```

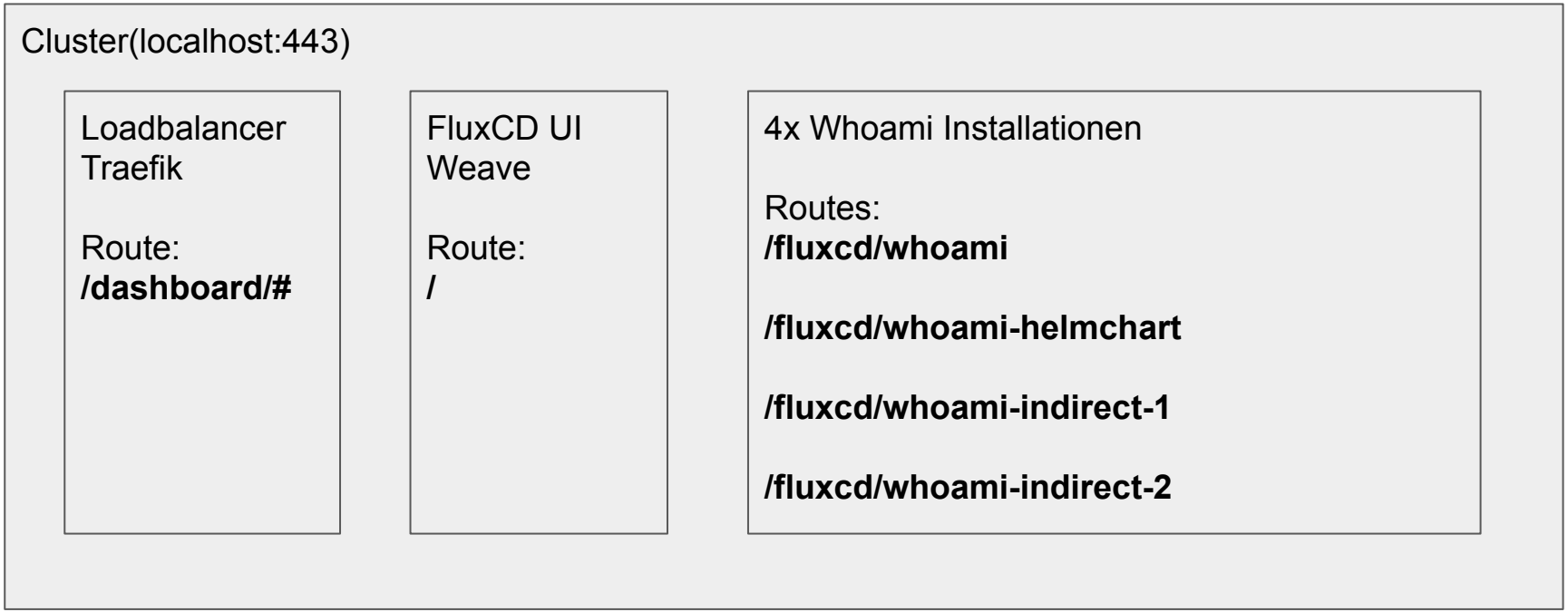
Start Cluster

```
aws ~ schulung/gitops> bash scripts/run-cluster-local.sh

export KUBECONFIG=$(pwd)/.k3s/kubconfig.yaml
```

Deploy FluxCD

```
export GITHUB_USER=jwausle-demo(1|2)
export GITHUB_TOKEN=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      # demo1
export GITHUB_TOKEN=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 # demo2

aws ~ schulung/gitops> bash scripts/deploy-fluxcd.sh
aws ~ schulung/gitops> fluxcd get kustomization –watch
```

Demo - FluxCD


Cluster(localhost:443)
Loadbalancer
Traefik
Route:
/dashboard/#
FluxCD UI
Weave
Route:
/
4x Whoami Installationen
Routes:
/fluxcd/whoami
/fluxcd/whoami-helmchart
/fluxcd/whoami-indirect-1
/fluxcd/whoami-indirect-2

Demo - Argo CD

Demo - Kubectl

Login into AWS(1|2)

```
ssh ubuntu@aws…
aws ~> cd schulung/gitops
aws ~ schulung/gitops> git status
```

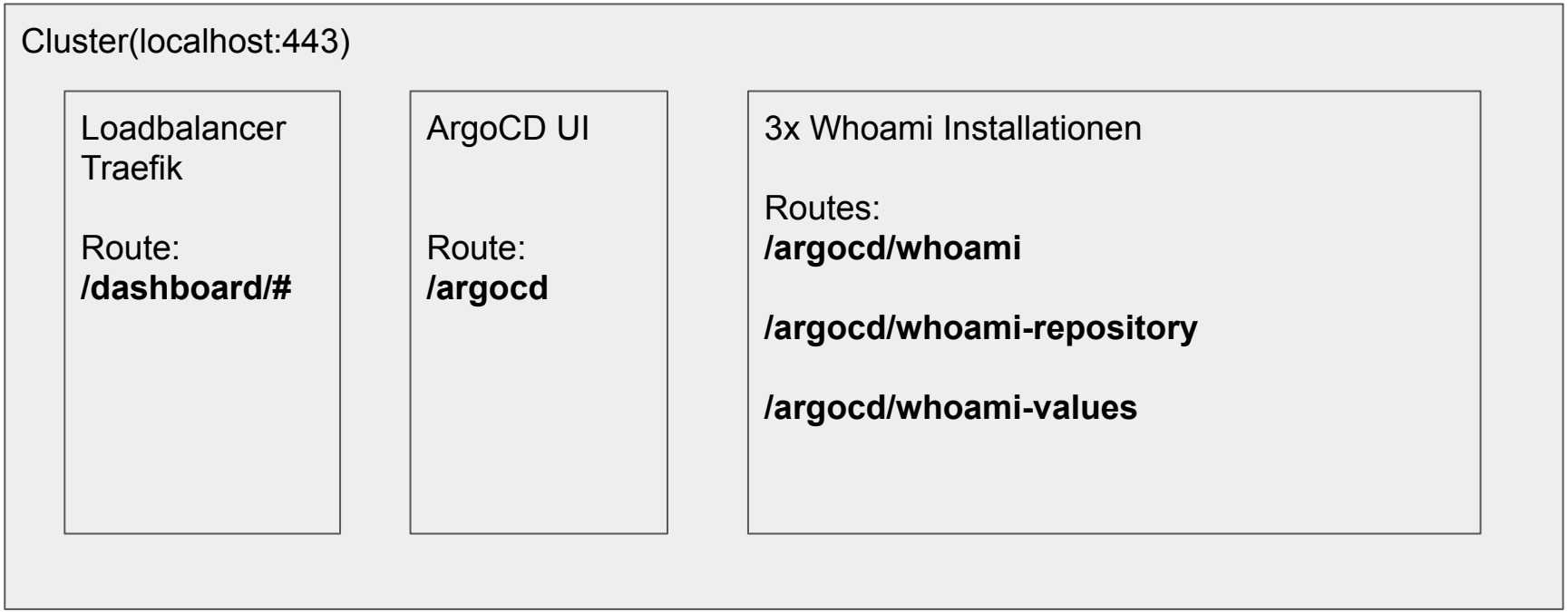
Start Cluster

```
aws ~ schulung/gitops> bash scripts/run-cluster-local.sh

export KUBECONFIG=$(pwd)/.k3s/kubconfig.yaml
```

Deploy ArgoCD

```
export GITHUB_USER=jwausle-demo(1|2)
export GITHUB_TOKEN=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      # demo1
export GITHUB_TOKEN=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 # demo2

aws ~ schulung/gitops> bash scripts/deploy-argocd.sh
aws ~ schulung/gitops> kubectl get apps -n –watch
```

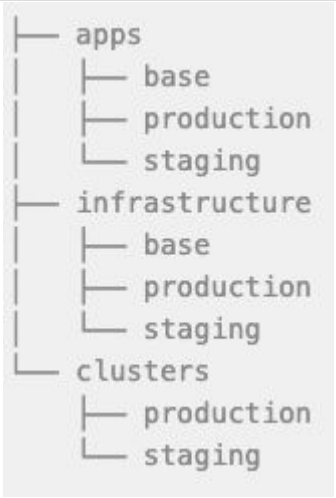
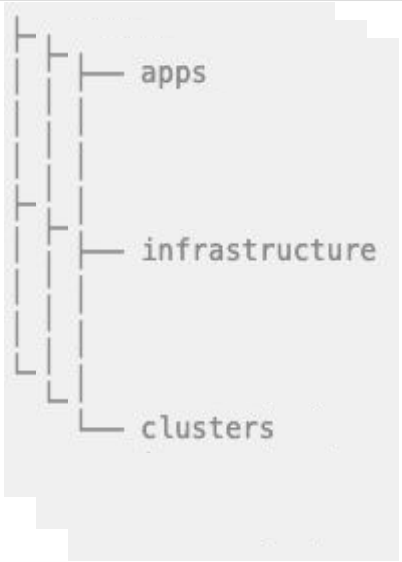
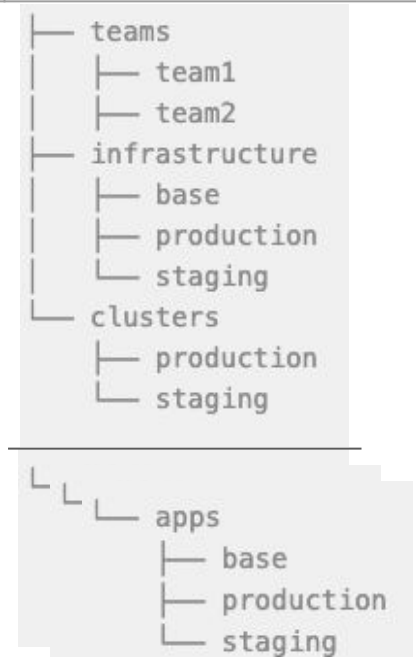
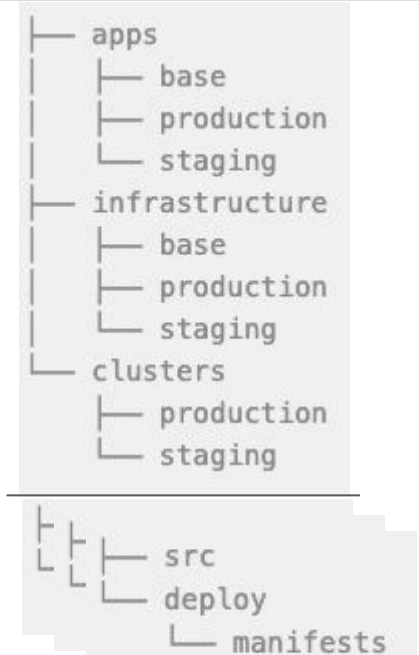
Demo - ArgoCD


Cluster(localhost:443)
Loadbalancer Traefik
Route:
/dashboard/#
ArgoCD UI
Route:
/argocd
3x Whoami Installationen
Routes:
/argocd/whoami
/argocd/whoami-repository
/argocd/whoami-values

Demo - Ende

Repository Strukturierung

https://fluxcd.io/flux/guides/repository-structure/

Mono repo	Repo per environment	Repo per team	Repo per application
			

Zusammenfassung

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

Missing features:

- Support für andere DSLs (z.B. https://github.com/dhall-lang/dhall-kubernetes)

Es ist schwer den Überblick zu behalten im Kubernetes Zoo:

- Kubernetes Tool-eritis

Kubernetes Tool-eritis

Fragen?