GitOps

“in a nutshell”

Was passiert hier?

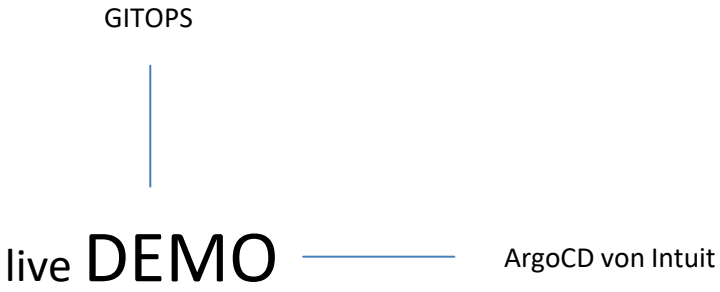
live DEMO

Was passiert hier?

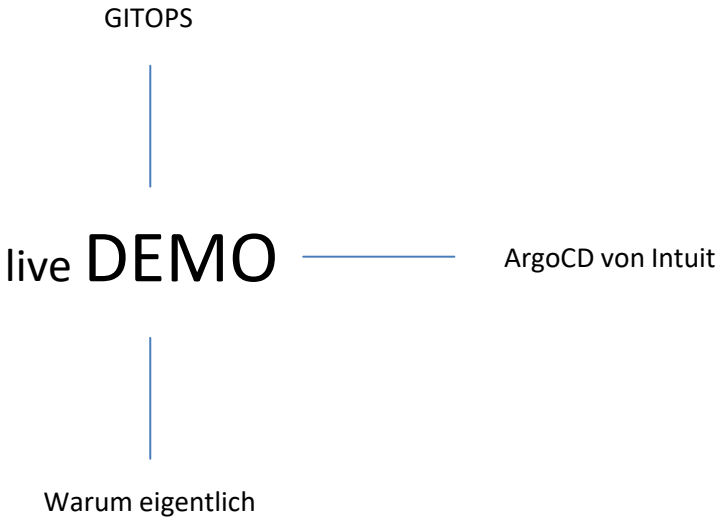
GITOPS

live DEMO

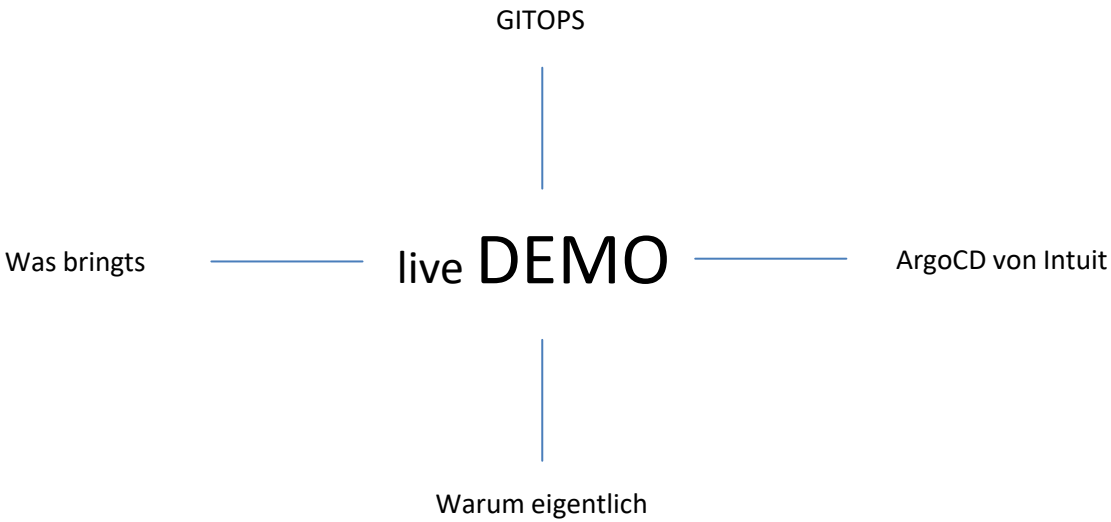
Was passiert hier?



Was passiert hier?



Was passiert hier?



Warum darf ich hier stehen?

Willkommen

@cedricboesiger

huma

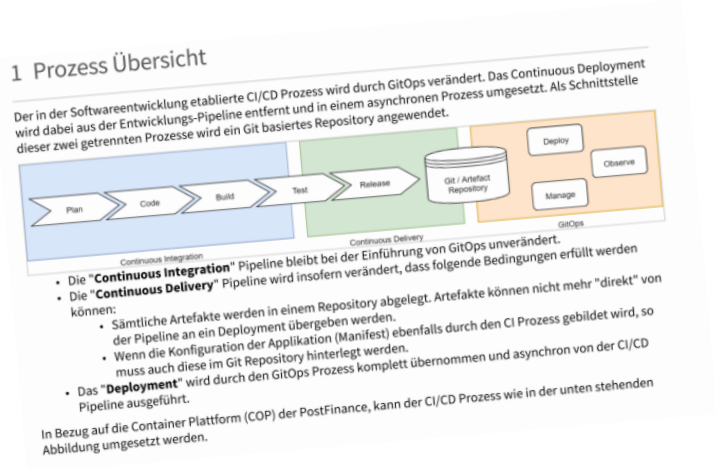
PostFinance

Willkommen

huma

PostFinance

@cedricboesiger



1 Prozess Übersicht

Der in der Softwareentwicklung etablierte CI/CD Prozess wird durch GitOps verändert. Das Continuous Deployment wird dabei aus der Entwicklungs-Pipeline entfernt und in einem asynchronen Prozess umgesetzt. Als Schnittstelle dieser zwei getrennten Prozesse wird ein Git basiertes Repository angewendet.

- Die "**Continuous Integration**" Pipeline bleibt bei der Einführung von GitOps unverändert.
- Die "**Continuous Delivery**" Pipeline wird insofern verändert, dass folgende Bedingungen erfüllt werden können:
  - Sämtliche Artefakte werden in einem Repository abgelegt. Artefakte können nicht mehr "direkt" von der Pipeline an ein Deployment übergeben werden.
  - Wenn die Konfiguration der Applikation (Manifest) ebenfalls durch den CI Prozess gebildet wird, so muss auch diese im Git Repository hinterlegt werden.
- Das "**Deployment**" wird durch den GitOps Prozess komplett übernommen und asynchron von der CI/CD Pipeline ausgeführt.

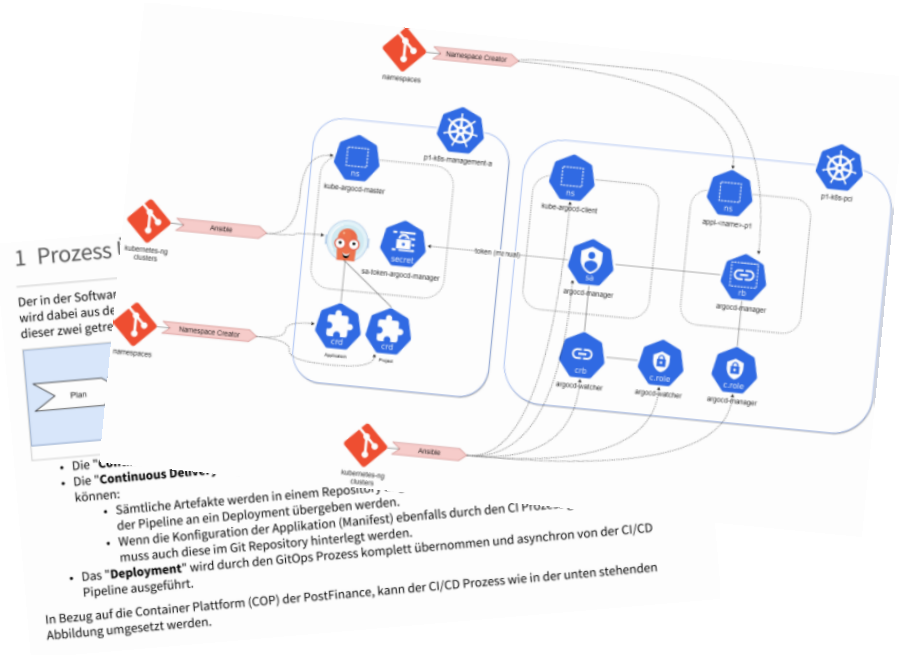
In Bezug auf die Container Plattform (COP) der PostFinance, kann der CI/CD Prozess wie in der unten stehenden Abbildung umgesetzt werden.

Willkommen

@cedricboesiger

huma

PostFinance

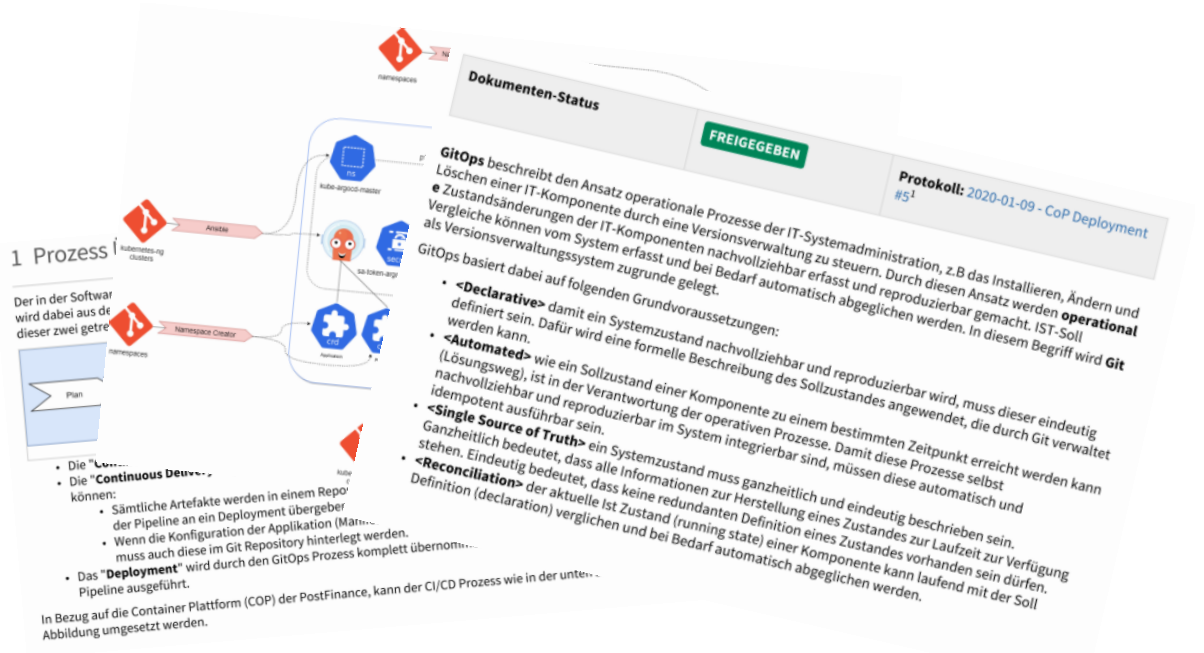


Willkommen

@cedricboesiger

xhuma

PostFinance



Ok, und was ist GitOps?

GitOps?

Git

Ops

GitOps?

Git

Laufzeitumgebung

GitOps?

Git

SOLL Zustand des Services

Laufzeitumgebung

IST Zustand des Services

GitOps?

Git

Agent

Laufzeitumgebung

SOLL Zustand des Services

IST Zustand des Services

GitOps?

Git

Agent

Laufzeitumgebung

SOLL Zustand des Services

Abgleich Soll/Ist

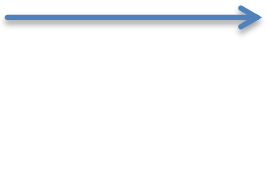
IST Zustand des Services

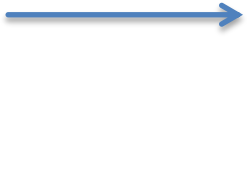
GitOps?

Git

Agent

Laufzeitumgebung





SOLL Zustand des Services

Abgleich Soll/Ist

IST Zustand des Services

Single Source of Truth

Deploy

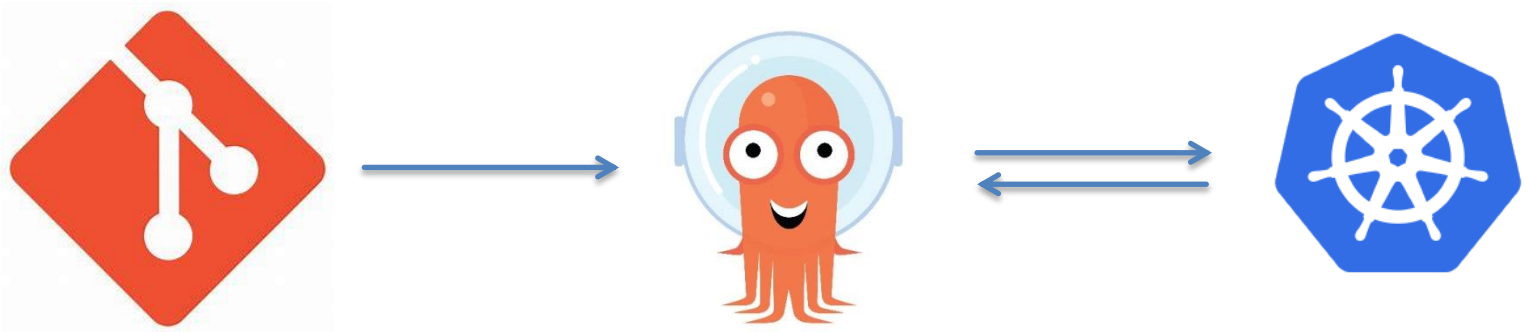
Run

GitOps?

Git

Agent

Laufzeitumgebung



SOLL Zustand des Services

Abgleich Soll/Ist

IST Zustand des Services

Single Source of Truth
Declarative

Deploy

Run
Observe

GitOps?

Git | Agent | Laufzeitumgebung

SOLL Zustand des Services

Abgleich Soll/Ist

IST Zustand des Services

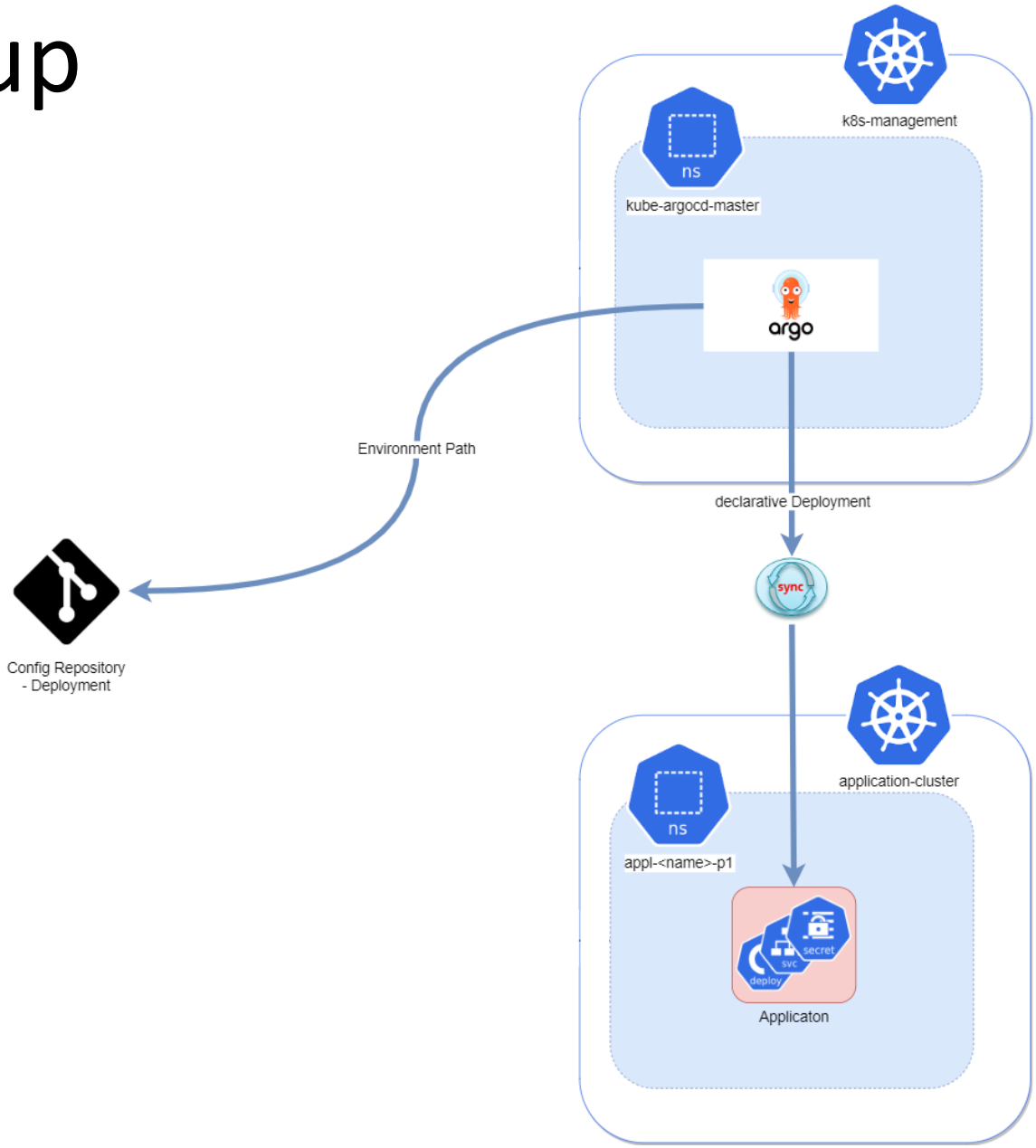
Single Source of Truth
Declarative

Deploy
Converge / Reconcile

Run
Observe

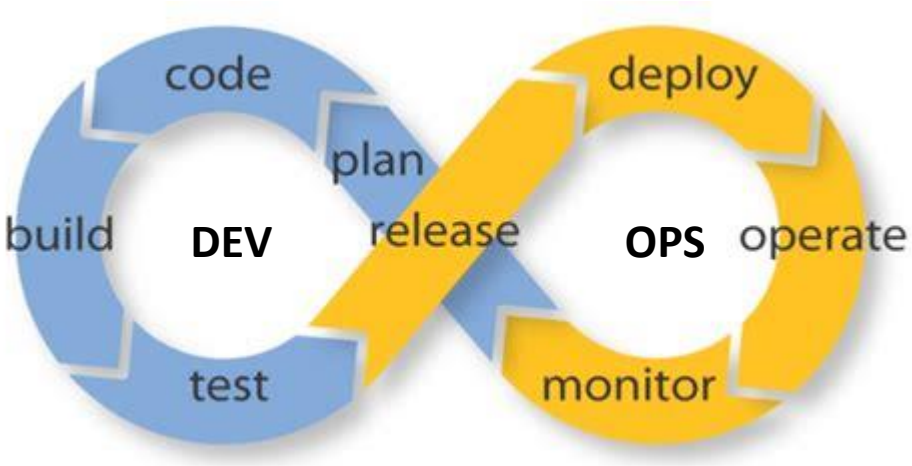
Und wann kommt die
LIVE Demo ?

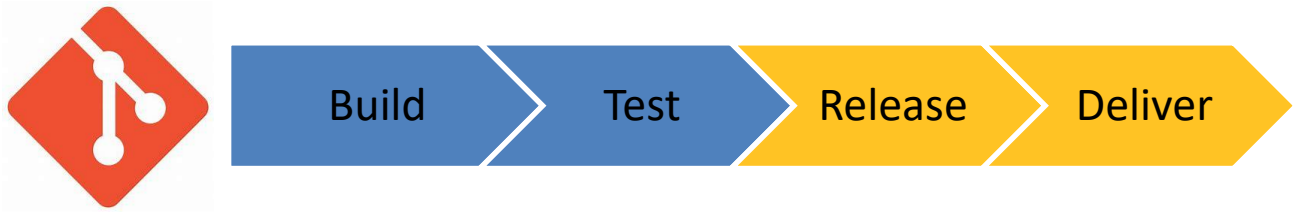
Setup


k8s-management
ns
kube-argocd-master
argo
Environment Path
declarative Deployment
sync
Config Repository
- Deployment
application-cluster
ns
appl-<name>-p1
secret
svc
deploy
Applicaton

Warum eigentlich GitOps?

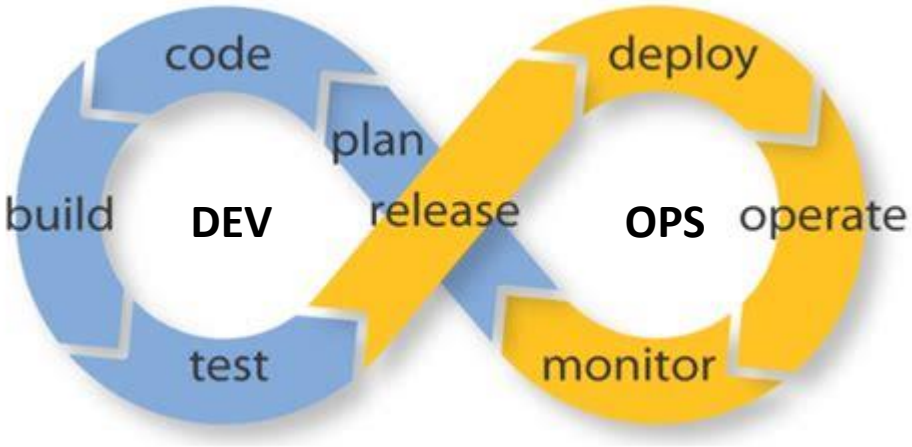
Von DevOps zu CI/CD





Continuous Integration Continuous Delivery

Von DevOps zu CI/CD



Continuous Integration Continuous Delivery Continuous Deployment

Von CI/CD Pipelines zu CI-OPS

(Alexis Richardson)

CI-OPS

Es funktioniert, kann aber schwierig sein

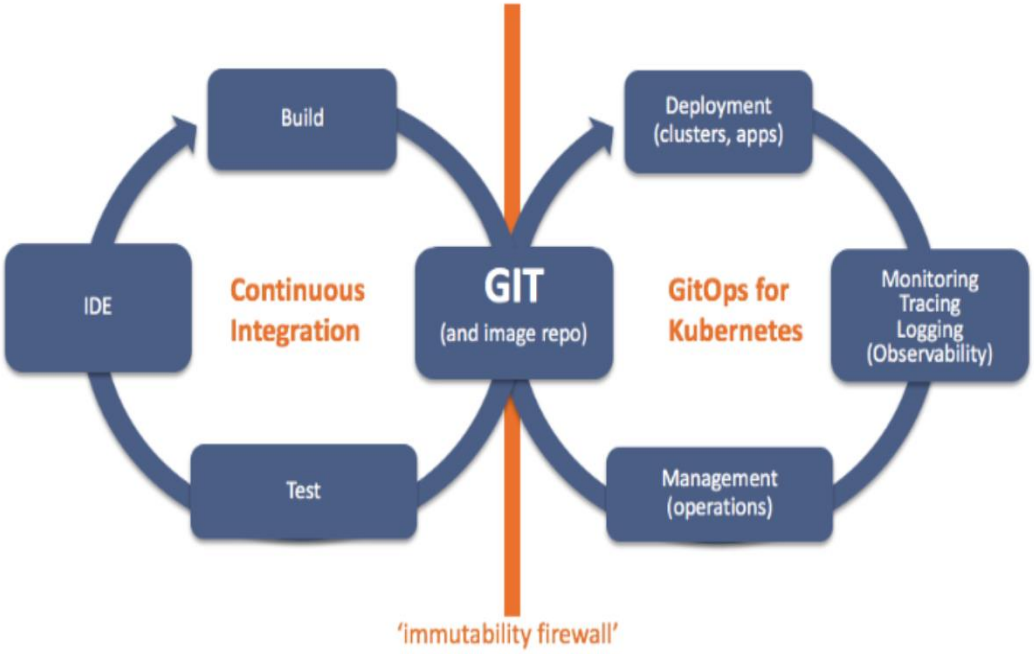
weil

die Hardware häufig am falschen Ort ist
die Verfügbarkeit teils nicht stimmt
die “Segregation of duty” schwierig ist
die Nachvollziehbarkeit teils nicht gegeben ist

und es oft nur scheinbar “Infrastructure as Code” ist

Der GitOps Ansatz

https://www.weave.works/blog/what-is-gitops-really



Git as the single source of truth of a system's desired state

GitOps Diffs compare desired state with observed state (eg Kubediff, Terradiff, Canary..)

ALL intended operations are committed by pull request, for all environments

ALL diffs between GIT and observed state lead to (auto) convergence using tools like K8s

ALL changes are observable, verifiable and audited indisputably, with rollback & D/R

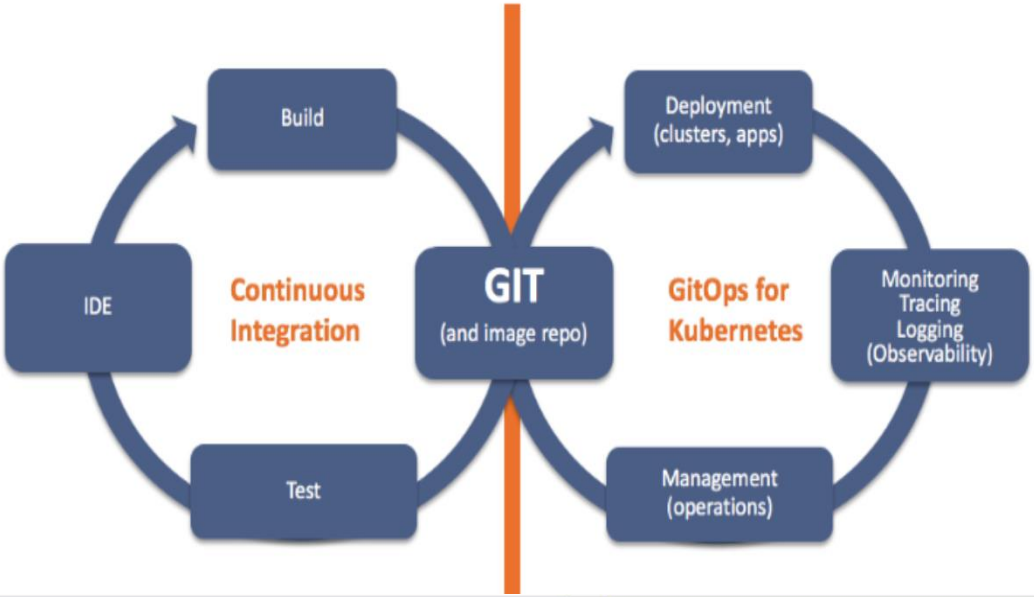
Git-Push
DEV

merge →

Git-Pull
OPS

Der GitOps Ansatz

https://www.weave.works/blog/what-is-gitops-really



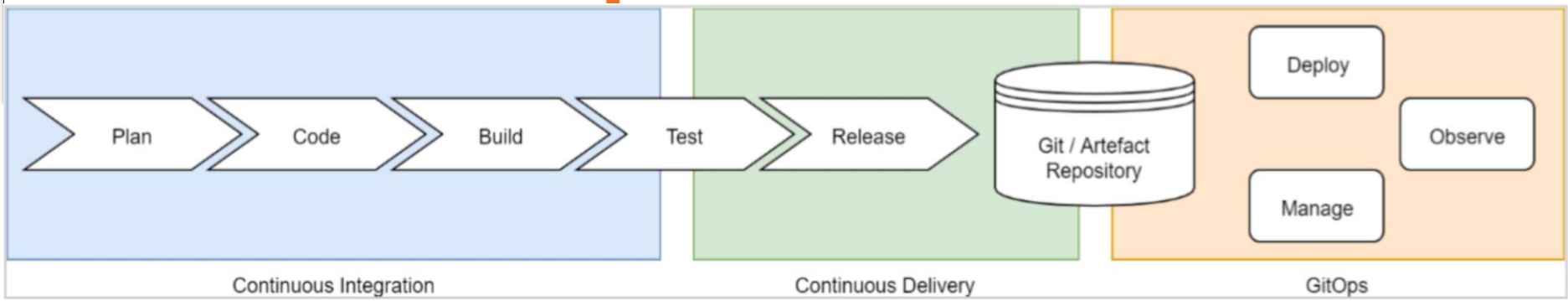
Git as the single source of truth of a system's desired state

GitOps Diffs compare desired state with observed state (eg Kubediff, Terradiff, Canary..)

ALL intended operations are committed by pull request, for all environments

ALL diffs between GIT and observed state lead to (auto) convergence using tools like K8s

ALL changes are observable, verifiable and audited indisputably, with rollback & D/R



Und noch ein paar Statements

Ein paar Statements zu GitOps

GitOps baut auf DEVOPS auf

Ein paar Statements zu GitOps

Git ist “single source of truth” und deklarativ

Ein paar Statements zu GitOps

Der “running State” ist “observable” und kann “reconciled” werden

Ein paar Statements zu GitOps

Alles ist unter Versionskontrolle,

Nachvollziehbar,

“Rollback”-bar

und “Restore”-bar

Ein paar Statements zu GitOps

Das Modell ist “nice and easy”

Ein paar Statements zu GitOps

“Compliance” kommt in vielen Bereichen einfach so mit

Ein paar Statements zu GitOps

“Collaboration” ist jetzt auch für OPS

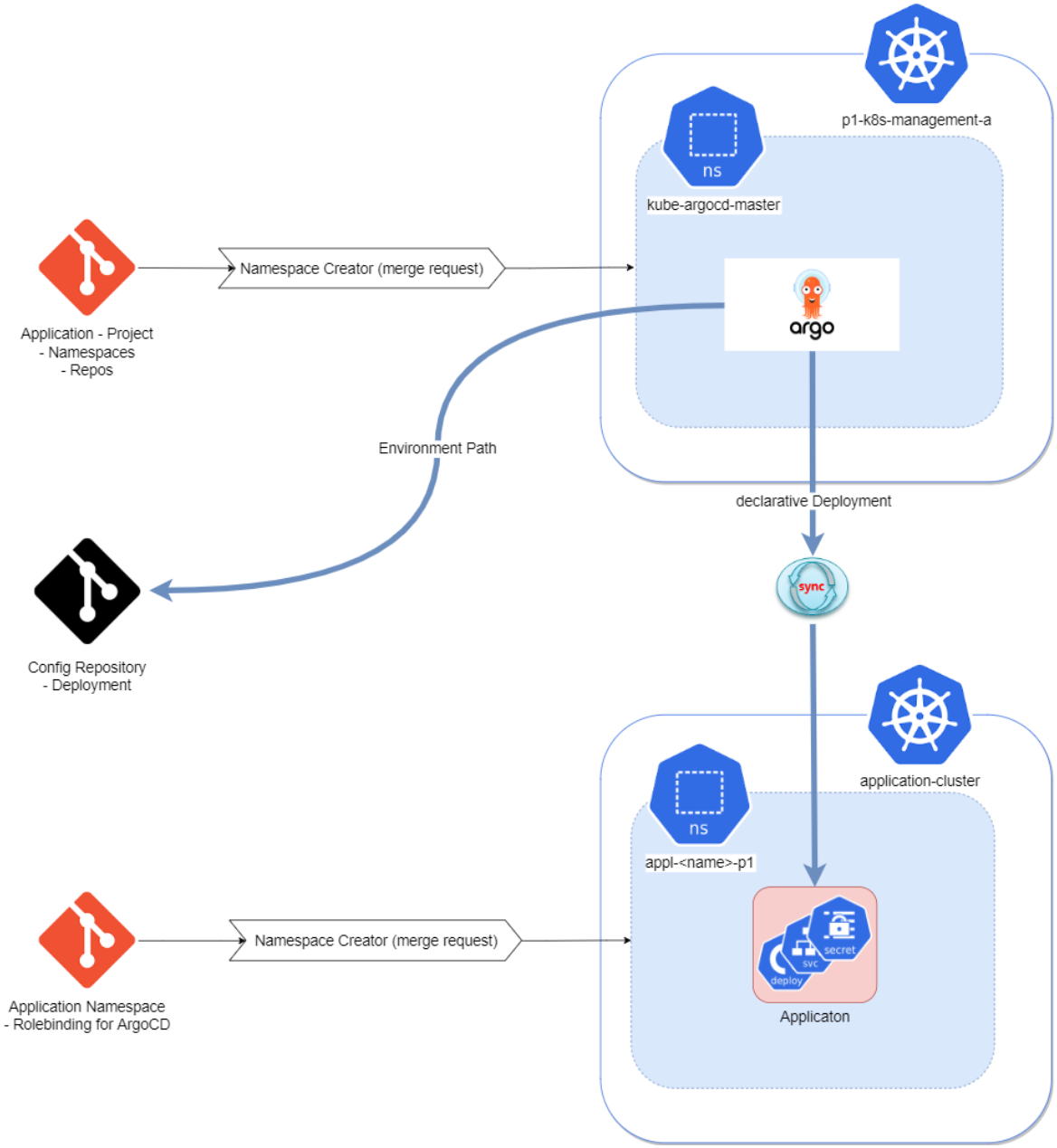
zum Schluss…

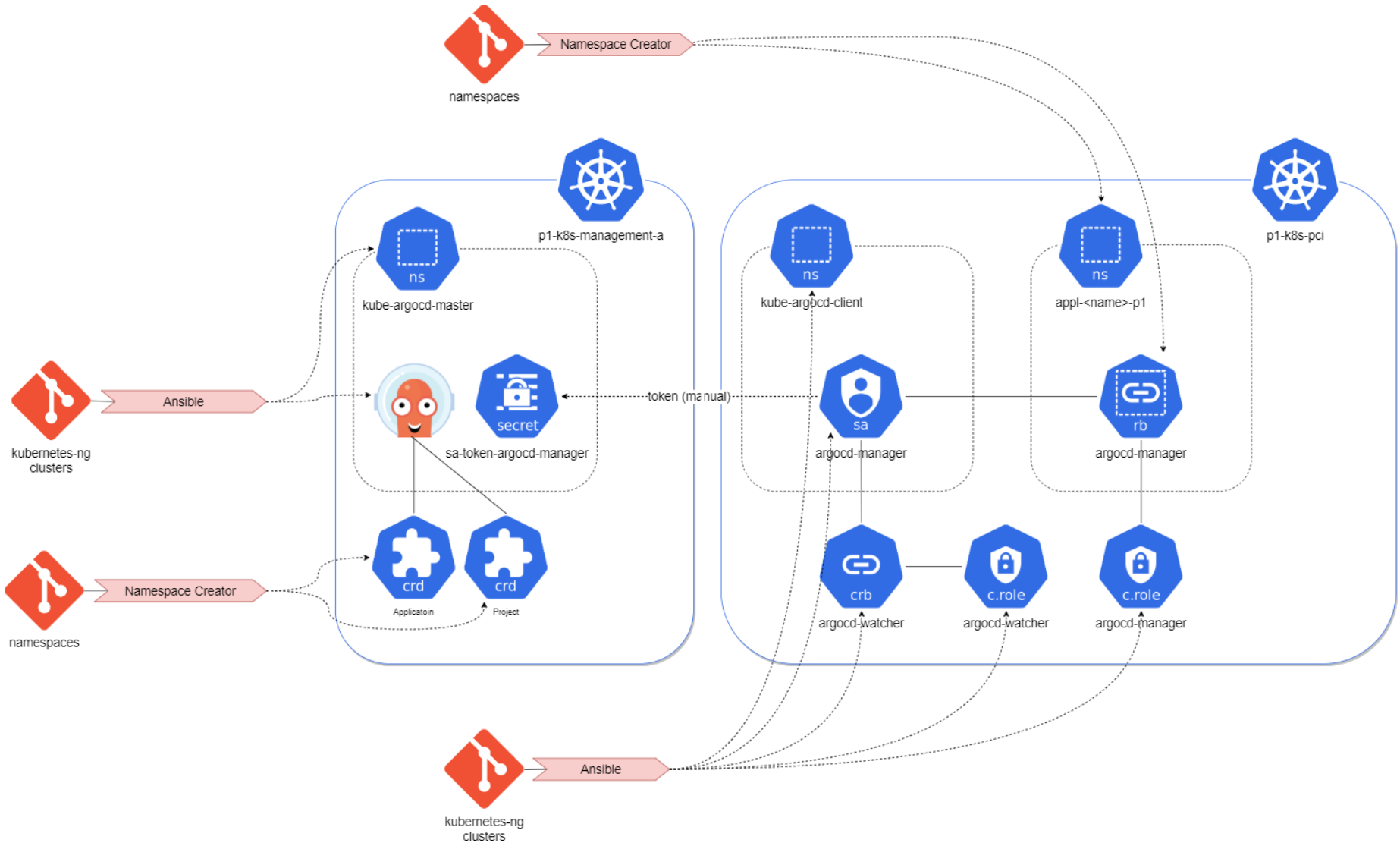
Es hilft die “Security” und “Velocity” zu erhöhen

zum Schluss…

Es hilft die “Security” und “Velocity” zu erhöhen

…vielen Dank!


p1-k8s-management-a
ns
kube-argocd-master
argo
Application - Project
- Namespaces
- Repos
Namespace Creator (merge request)
Environment Path
declarative Deployment
sync
Config Repository
- Deployment
application-cluster
ns
appl-<name>-p1
secret
svc
deploy
Applicaton
Application Namespace
- Rolebinding for ArgoCD
Namespace Creator (merge request)


Namespace Creator
namespaces
p1-k8s-management-a
p1-k8s-pci
ns
kube-argocd-master
ns
kube-argocd-client
ns
appl-<name>-p1
Ansible
kubernetes-ng clusters
secret
sa-token-argocd-manager
token (manual)
sa
argocd-manager
rb
argocd-manager
crd
Applicatoin
crd
Project
crb
argocd-watcher
c.role
argocd-watcher
c.role
argocd-manager
Namespace Creator
namespaces
Ansible
kubernetes-ng clusters

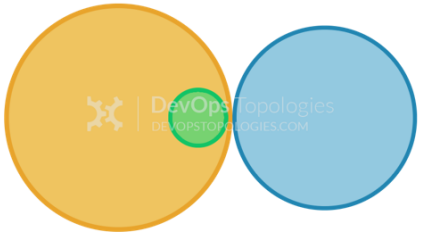
DEVOPS Topologies

Type 3: Ops as Infrastructure-as-a-Service (Platform)

For organisations with a fairly traditional IT Operations department which cannot or will not change rapidly [enough], and for organisations who run all their applications in the public cloud (Amazon EC2, Rackspace, Azure, etc.), it probably helps to treat Operations as a team who simply provides the elastic infrastructure on which applications are deployed and run; the internal Ops team is thus directly equivalent to Amazon EC2, or Infrastructure-as-a-Service.

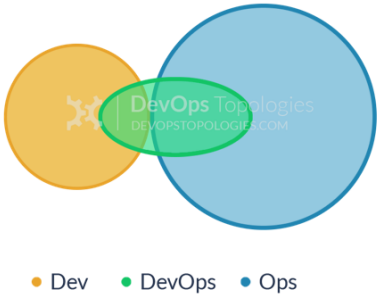
A team (perhaps a virtual team) within Dev then acts as a source of expertise about operational features, metrics, monitoring, server provisioning, etc., and probably does most of the communication with the IaaS team. This team is still a Dev team, however, following standard practices like TDD, CI, iterative development, coaching, etc.

The IaaS topology trades some potential effectiveness (losing direct collaboration with Ops people) for easier implementation, possibly deriving value more quickly than by trying for **Type 1 (Dev and Ops Collaboration)** which could be attempted at a later date.



Type 3 suitability: organisations with several different products and services, with a traditional Ops department, or whose applications run entirely in the public cloud.

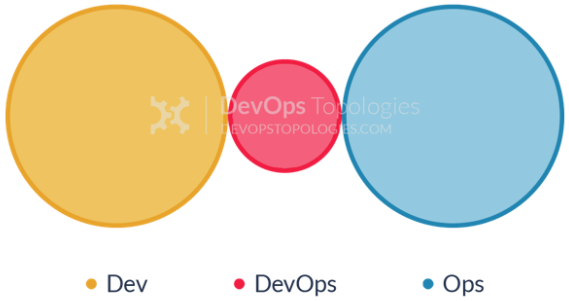
Potential effectiveness: MEDIUM



Type 4: DevOps as an External Service

Some organisations, particularly smaller ones, might not have the finances, experience, or staff to take a lead on the operational aspects of the software they produce. The Dev team might then reach out to a service provider like Rackspace to help them build test environments and automate their infrastructure and monitoring, and advise them on the kinds of operational features to implement during the software development cycles.

DEVOPS Anti-Types



Anti-Type B: DevOps Team Silo

The DevOps Team Silo (Anti-Type B) typically results from a manager or exec deciding that they "need a bit of this DevOps thing" and starting a 'DevOps team' (probably full of people known as 'a DevOp'). The members of the DevOps team quickly form another silo, keeping Dev and Ops further apart than ever as they defend their corner, skills, and toolset from the 'clueless Devs' and 'dinosaur Ops' people.

The only situation where a separate DevOps silo really makes sense is when the team is temporary, lasting less than (say) 12 or 18 months, with the express purpose of bringing Dev and Ops closer together, and with a clear mandate to make the DevOps team superfluous after that time; this becomes what I have called a **Type 5 DevOps Topology**.

Anti-Type C: Dev Don't Need Ops

This topology is borne of a combination of naivety and arrogance from developers and development managers, particularly when starting on new projects or systems. Assuming that Ops is now a thing of the past ("we have the Cloud now, right?"), the developers wildly underestimate the complexity and importance of operational skills and activities, and believe that they can do without them, or just cover them in spare hours.