

GitOps

“in a nutshell”

Worum gehts hier?

nutshell

Worum gehts hier?

GITOPS



nutshell

Worum gehts hier?

GITOPS



nutshell



Beispiel

Worum gehts hier?

GITOPS



nutshell

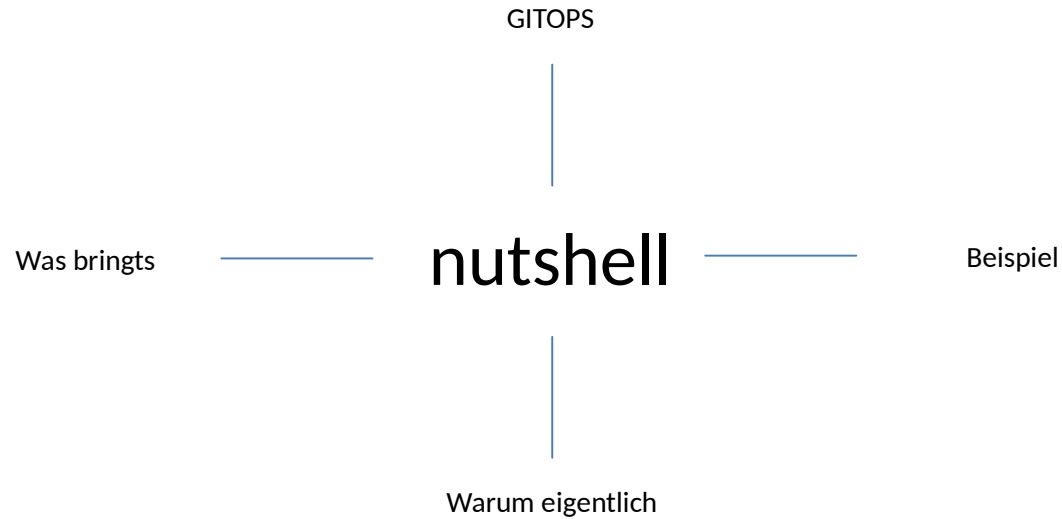


Beispiel



Warum eigentlich

Worum gehts hier?



Warum darf ich hier stehen?

Willkommen



@cedricboesiger

✕huma



Willkommen

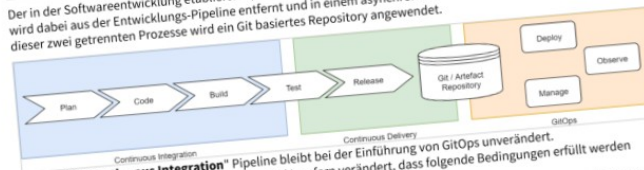


@cedricboesiger



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.



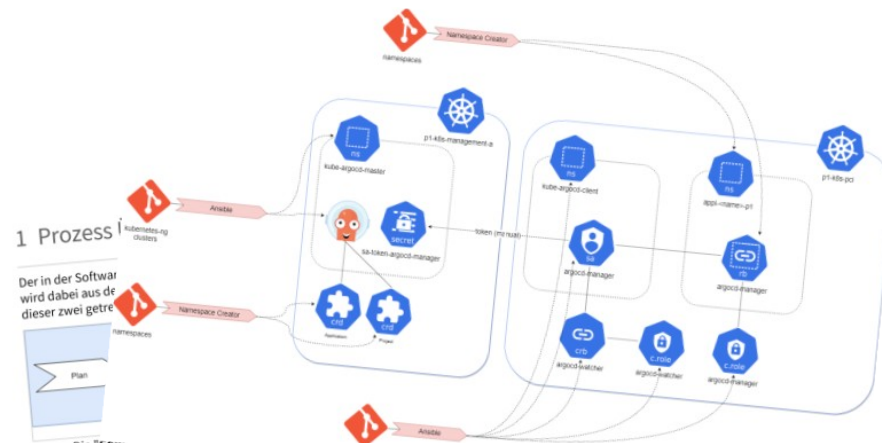
- Die "**Continuous Integration**" Pipeline bleibt bei der Einführung von GitOps unverändert.
- Die "**Continuous Delivery**" Pipeline wird insofern verändert, dass folgende Bedingungen erfüllt werden können:
 - Sämtliche Artefakte werden in einem Repository abgelegt. Artefakte können nicht mehr "direkt" 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



- Die "Continuous Delivery" können:
 - Sämtliche Artefakte werden in einem Repository der Pipeline an ein Deployment übergeben werden.
 - Wenn die Konfiguration der Applikation (Manifest) ebenfalls durch den CI Prozess 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



1 Prozess

Der in der Software wird dabei aus dieser zwei getrennt

Plan

- Die "Continuous Delivery" können:
 - Sämtliche Artefakte werden in einem Repository der Pipeline an ein Deployment übergeben
 - Wenn die Konfiguration der Applikation (Manifeste) muss auch diese im Git Repository hinterlegt werden.
- Das "Deployment" wird durch den GitOps Prozess komplett übernommen Pipeline ausgeführt.

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

Dokumenten-Status

FREIGEgeben

Protokoll: 2020-01-09 - CoP Deployment #51

GitOps beschreibt den Ansatz operationale Prozesse der IT-Systemadministration, z.B. das Installieren, Ändern und Löschen einer IT-Komponente durch eine Versionsverwaltung zu steuern. Durch diesen Ansatz werden **operational** Zustandsänderungen der IT-Komponenten nachvollziehbar erfasst und reproduzierbar gemacht. IST-Soll Vergleiche können vom System erfasst und bei Bedarf automatisch abgeglichen werden. In diesem Begriff wird **Git** als Versionsverwaltungssystem zugrunde gelegt.

GitOps basiert dabei auf folgenden Grundvoraussetzungen:

- <Declarative>** damit ein Systemzustand nachvollziehbar und reproduzierbar wird, muss dieser eindeutig definiert sein. Dafür wird eine formelle Beschreibung des Sollzustandes angewendet, die durch Git verwaltet werden kann.
- <Automated>** wie ein Sollzustand einer Komponente zu einem bestimmten Zeitpunkt erreicht werden kann (Lösungsweg), ist in der Verantwortung der operativen Prozesse. Damit diese Prozesse selbst nachvollziehbar und reproduzierbar im System integrierbar sind, müssen diese automatisch und idempotent ausführbar sein.
- <Single Source of Truth>** ein Systemzustand muss ganzheitlich und eindeutig beschrieben sein. Ganzheitlich bedeutet, dass alle Informationen zur Herstellung eines Zustandes zur Laufzeit zur Verfügung stehen. Eindeutig bedeutet, dass keine redundanten Definition eines Zustandes vorhanden sein dürfen.
- <Reconciliation>** der aktuelle Ist Zustand (running state) einer Komponente kann laufend mit der Soll Definition (declaration) verglichen und bei Bedarf automatisch abgeglichen werden.

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



SOLL Zustand des Services

Single Source of Truth
Declarative

Agent



Abgleich Soll/Ist

Deploy

Laufzeitumgebung



IST Zustand des Services

Run
Observe



GitOps?

Git



SOLL Zustand des Services

Single Source of Truth
Declarative

Agent



Abgleich Soll/Ist

Deploy
Converge / Reconcile

Laufzeitumgebung



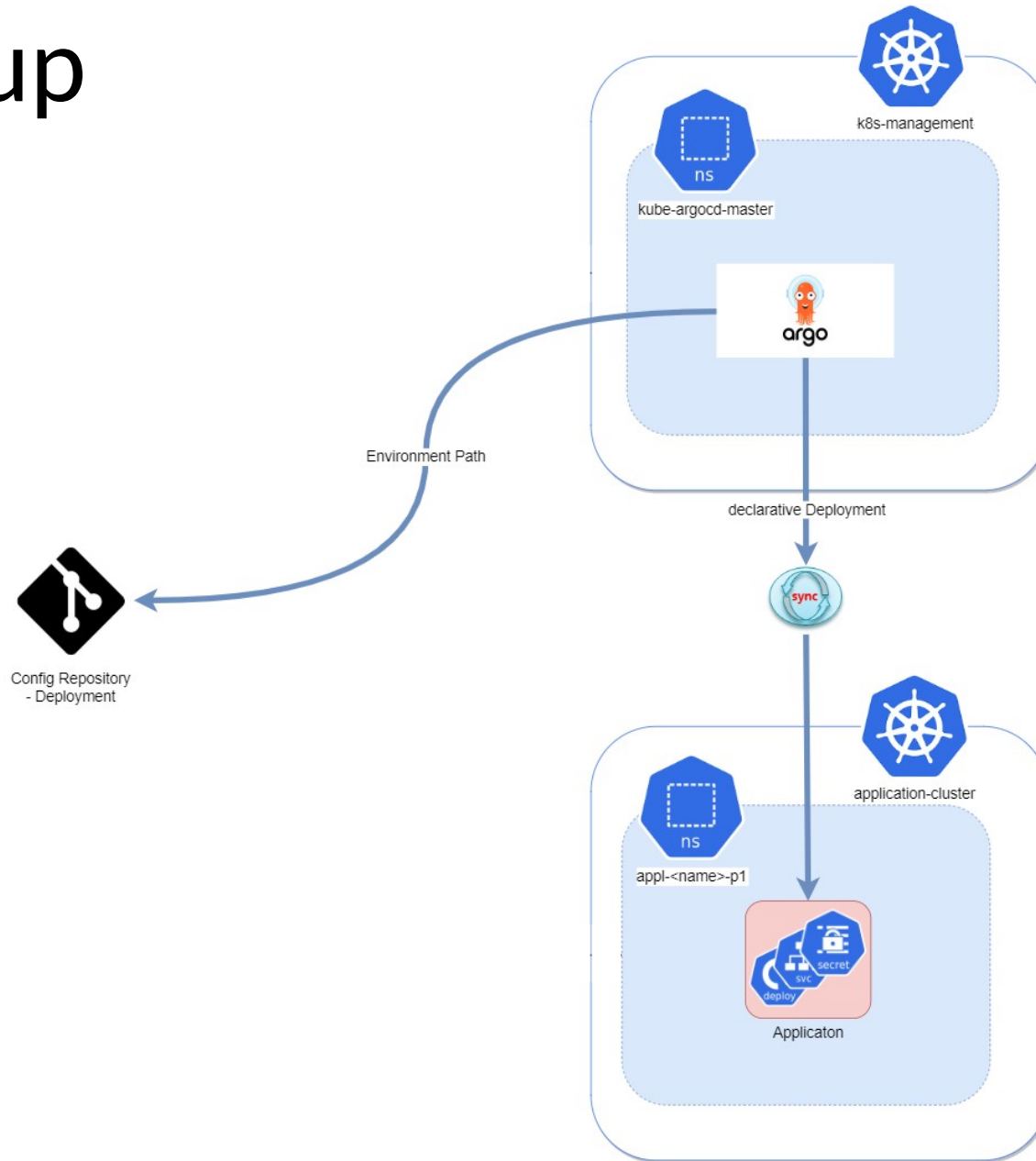
IST Zustand des Services

Run
Observe

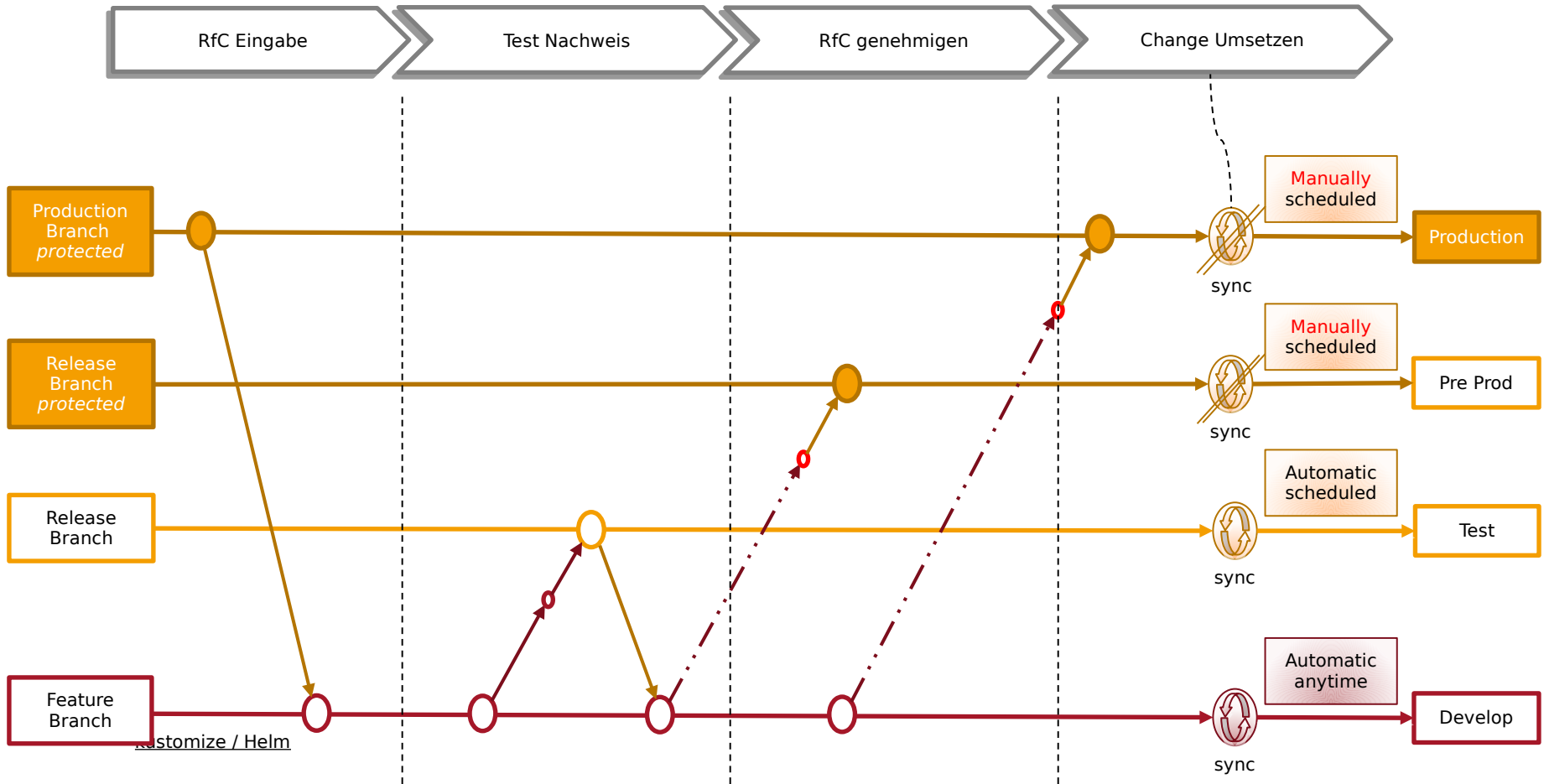


Und wie könnte ein
echtes Beispiel aussehen ?

Setup



Git Flow



ArgoCD in Action

The screenshot displays the ArgoCD web interface for the application 'vetted-life-webapp'. At the top, there is a navigation bar with the following buttons: APP DETAILS, APP DIFF, SYNC, SYNC STATUS, HISTORY AND ROLLBACK, DELETE, and REFRESH. Below the navigation bar, the application's overall status is shown as 'Healthy'. The 'Synced' status is highlighted, indicating that the application is up-to-date with the target revision 'HEAD (2f1611f)'. The 'Sync OK' status is also shown, indicating that the application is in a healthy state and has been successfully synchronized. The main content area shows a tree view of the application's resources, including a namespace, a service, a deployment, a horizontal pod autoscaler, and two ingress controllers. Each resource is represented by an icon and a status indicator (heart or checkmark).

Applications / vetted-life-webapp

APP DETAILS APP DIFF SYNC SYNC STATUS HISTORY AND ROLLBACK DELETE REFRESH

Healthy

Synced
To HEAD (2f1611f)
Authored by razvanonet <53819454+razvan...>
Merge pull request #30 from lolaent/feature...

Sync OK
To 2f1611f
Succeeded 13 days ago (Wed Jul 22 2020 09:26:33 GMT+0300)
Authored by razvanonet <53819454+razvanonet@users.noreply.github.com>
Merge pull request #30 from lolaent/feature/VL-211

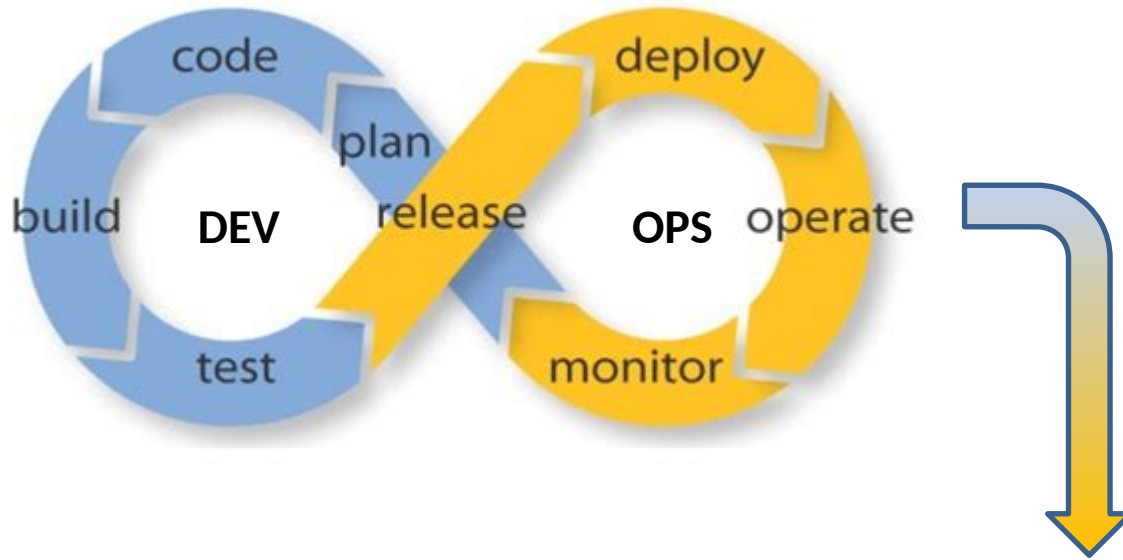
Refreshing

vetted-life-webapp

- vetted-life-webapp (ns)
- vetted-life-webapp (svc)
- vetted-life-webapp (deploy)
- vetted-life-webapp (hpa)
- arena-admin-tool (ing)
- vetted-life-webapp (ing)

Warum eigentlich
GitOps?

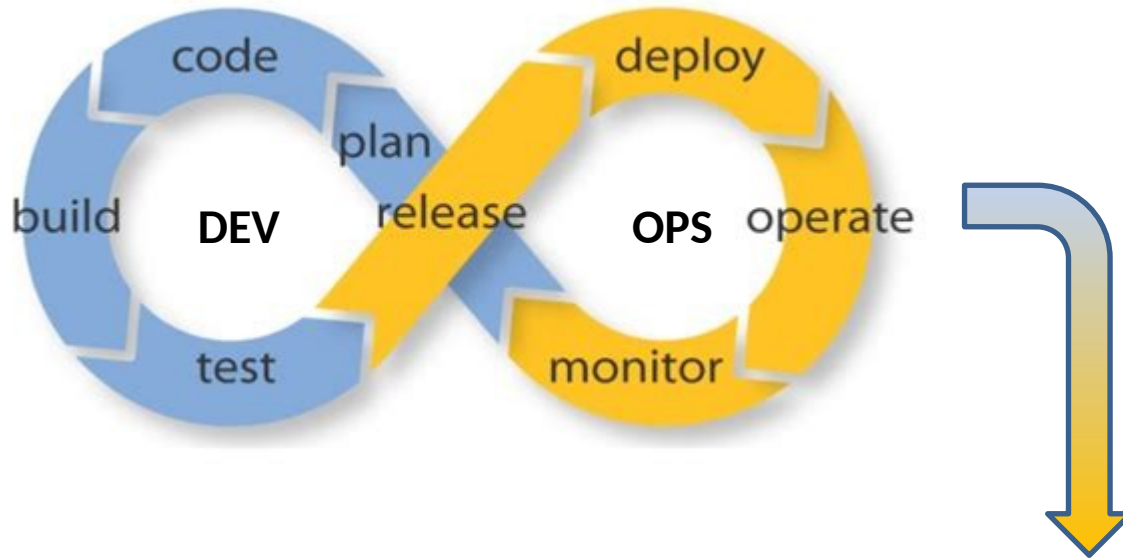
Von DevOps zu CI/CD



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

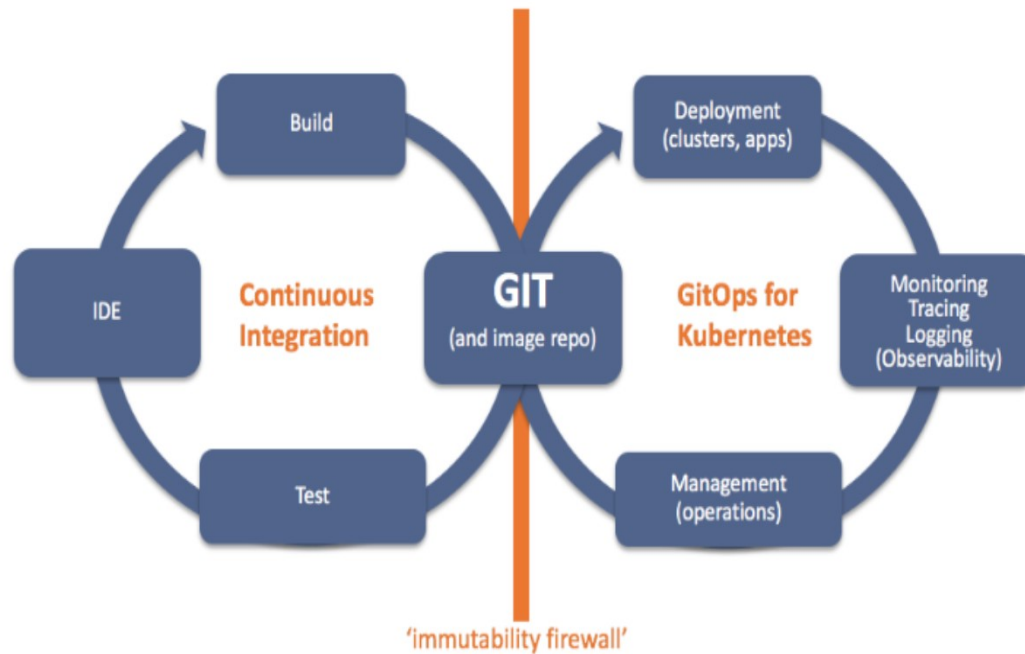
weil

- 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 Kurediff, 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

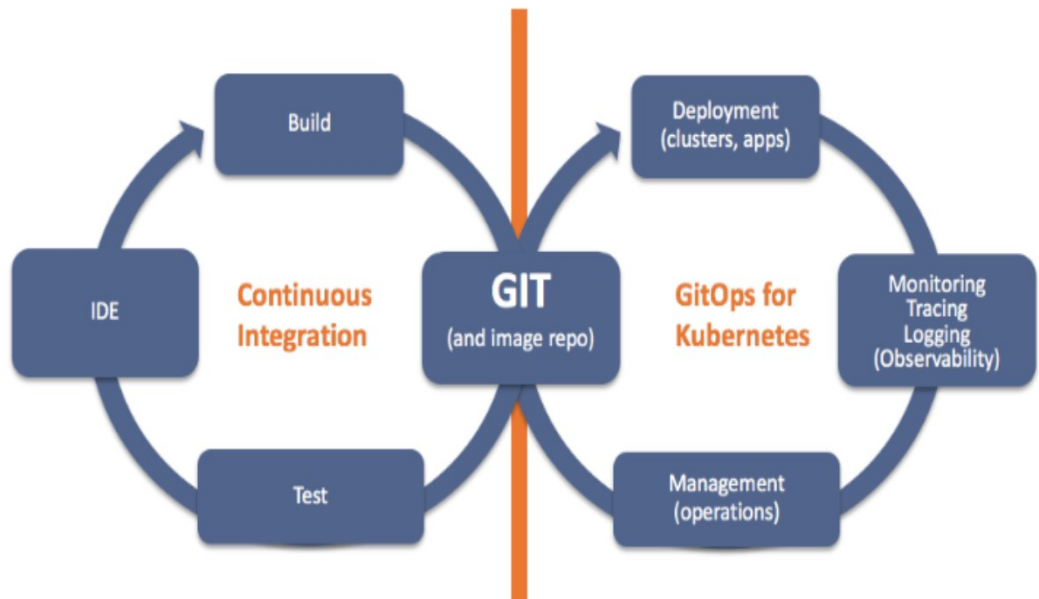
ALL changes are observable, verifiable and audited indisputably, with rollback & D/R

Git-Push
DEV

merge →

Git-Pull
OPS

Der GitOps Ansatz



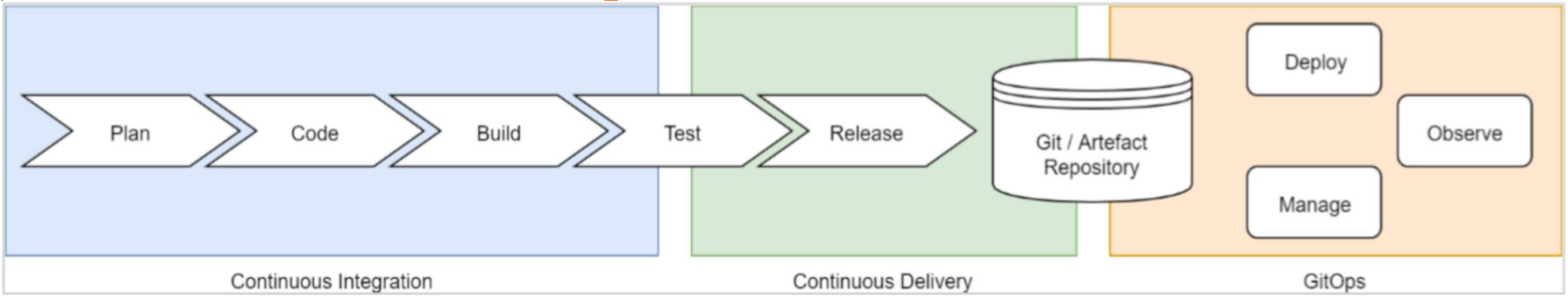
Git as the single source of truth of a system's desired state

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!

