GitOps

IT-Systeme mit Git-basierten Ansätzen automatisieren

Die konkreten Tätigkeiten eines Betriebsteams sind oft nur für Eingeweihte nachvollziehba und häufig laufen Dinge auf ein oder zwei Personen zu. Das GitOps-Konzept soll das mit konsequenter Versionierung, aus der Entwicklung übernommene kooperative Arbeitsweisen und ein Bekenntnis zu deklarativer Konfiguration von Systemen.

Definition

GitOps rückt die in einem Versionskontrollsystem abgelegte Beschreibung eines Systemzustands in den Fokus. Sie ist Ausgangspunkt jeder Aktion. Änderungen der Beschreibung lösen im Zielsystem automatisch einen Soll-Ist-Vergleich aus, durchgeführt vom *GitOps-Operator*. Der Operator ermittelt die konkreten Schritte die zum beschriebenen Zielzustand führen, setzt sie um und prüft das Ergebnis automatisch. Die Regelschleife greift ebenso, wenn sich der Zustand des Systems ändert, etwa bei einem Komponentenausfall.

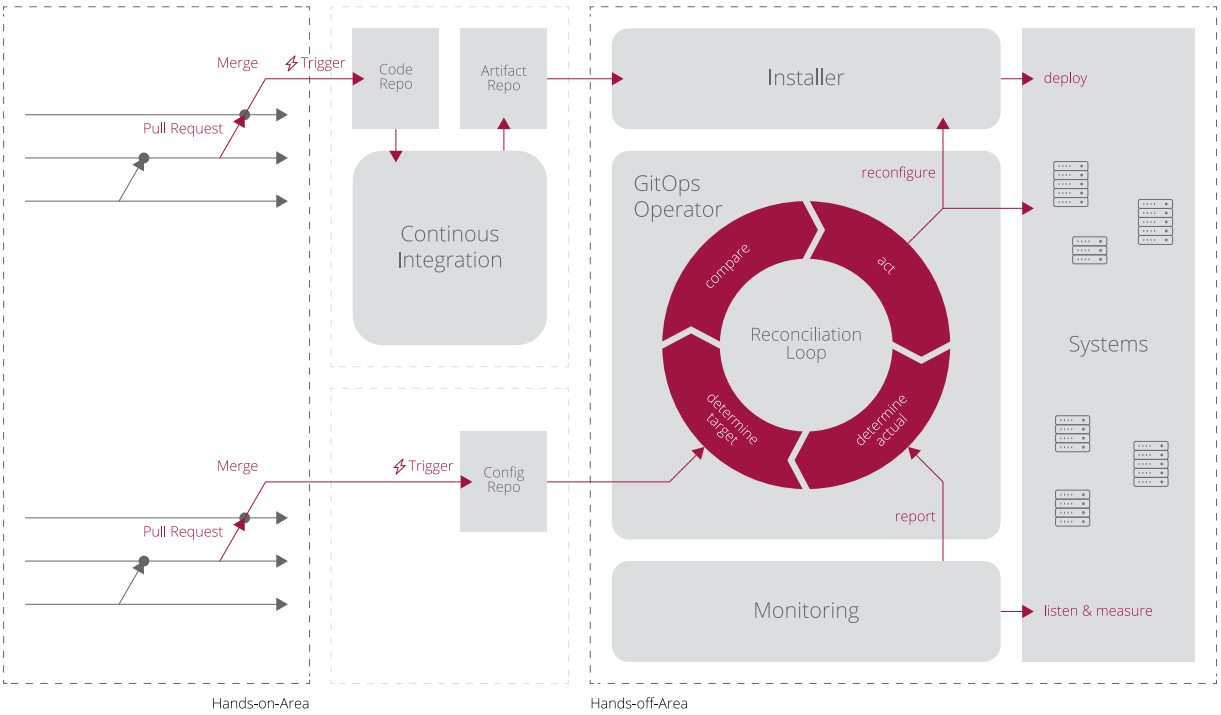
GitOps' deklarativer Ansatz stellt das *Was* dar, der GitOps-Operator übernimmt das *Wie*. Das unterscheidet sich markant von imperativen Vorgehen, die jeden Schritt zum Ziel vorgeben. Zudem greift GitOps in der Software-Entwicklung erprobte kooperative Techniken wie Pull Requests und Approved Merges auf.

Referenzszenario

Manuell auszuführende Schritte bremsten jeden Prozess aus, der bei Continuous Delivery und Deployment auf schnelle Änderungen in der Laufzeitumgebung ausgelegt war. Mit dem Wechsel auf GitOps wird das Versionskontrollsystem zum einzigen Zugangspunkt für das Laufzeitsystem. Das Laufzeitsystem wiederum wird zur Hands-off-Area, in dem nur noch der GitOps-Operator agiert. Dadurch hält gleichzeitig eine stringente Sicherheitsarchitektur Einzug.

Potenzial

GitOps ersetzt potenziell riskante, manuelle Eingriffe durch tiefgreifende Automatisierung des Betriebs von IT-Anwendungen. Gleichzeitig schafft es Klarheit. Alle Betriebshandlungen sind nachvollziehbar im Versionskontrollsystem pro-



Git

- überall verfügbar
- mensch- und maschinengeeignet
- Single Source of Truth
- übernimmt Workflow-Mustern aus der Entwicklung

Everything-as-Code

- beschrieben mit Textdateien
- einfach versionierbar
- geskriptete Automatisierung
- versionierte Konfiguration und Software

GOP

Devops

- vereint Entwicklung und Betrieb
- alles automatisieren
- mehr Interaktion von Entwicklung mit der Betriebsumgebung

Kubernetes

- deklarativer Ansatz
- dominiert Cloud-Betrieb
- komplexe, kleinteilige Konfiguration

tokolliert, mit Urheber, Zeitpunkt, Änderung und Begründung. Anstelle komplexer, grafischer Bedienschnittstellen kommen Textdateien zum Einsatz, ganz gemäß des Infractructure-as-Code-Ansatzes. Damit öffnet sich GitOps auch für Generatoren, Reporting-Tools und andere Werkzeuge. Die Beschreibung des Zielzustands dokumentiert zudem von selbst das Laufzeitsystem.

Reduzierte manuelle Routinetätigkeiten senken Personalkosten und verringern Risikofaktoren. Gleichzeitig ist Self-Service durch andere Teams in festgelegtem Rahmen möglich, ohne diesen Zugriff auf das Laufzeitsystem gewähren zu müssen. Zun kritischen Faktor wird der GitOps-Operator. Im Falle einer Fehlfunktion oder eines fehlerhaften Soll-Ist-Vergleichs muss ein Expertenteam eingreifen, die Situation analysieren und beheben. Zudem sind deklarativ konfigurierbare Umgebungen nicht die Norm, weshalb es sie sich hauptsächlich noch auf Container-Orchestration beschränkt.

Reifegrad

Im Bereich Container-Orchestration ist GitOps einfach einzuführen sowie einzusetzen und etabliert sich zunehmend als Standard-Betriebsverfahren. In anderen Umgebungen ist derzeit nur ein reduzierter Einsatz denkbar, der auf deklarative Konfigurationen verzichtet sowie sehr einfache GitOps-Operatoren koordiniert. Das gewährt die Vorteile der Versionskontrolle und des Reviewprozesses, schränkt aber den Automatisierungsgrad des Prozesses ein.

Marktübersicht

Für Kubernetes existiert eine nennenswerte Anzahl von GitOps-Werkzeugen, etwa Flux und Argo CD. Hinzu kommen Rancher Fleet, PipeCD und Jenkins X.

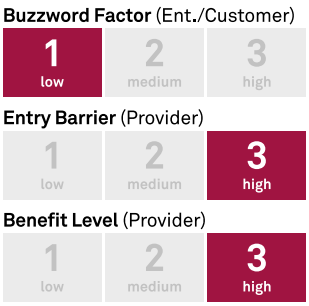
Außerhalb der Container-Orchestration existieren vor allem Werkzeuge zur Konfiguration von Infrastruktur, etwa Terraform, dessen Derivat Atlantis oder Ansible Automation Controller.

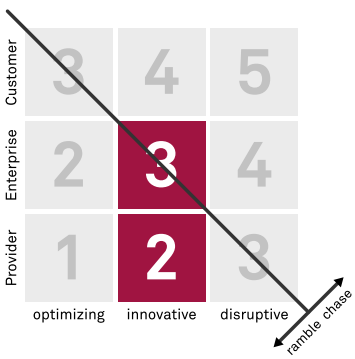
Alternativen

CI-Ops, die letzte Stufe einer Build-Pipeline, ermöglicht automatisiertes Deployment, allerdings ohne Hands-off-Area oder Dokumentation der Systemkonfiguration in einem Versionskontrollsystem. Zudem lassen sich die meisten Ziele von GitOps durch Regeln und Konventionen erreichen, falls sie vom Betriebsteam zuverlässig umgesetzt werden: keine Änderung am System ohne Versionierung, Vier-Augen-Prinzip für Modifikationen und Dokumentationen, und Vermeiden von Betriebs-Heldentum.

Fazit

- \+ protokolliert Betriebshandlungen
- \+ verringert Wissenssilos
- \+ Automatisierung entlastet Betrieb
- \+ sicherere Betriebsumgebung
- \+ integriert QA-Workflows
- \+ ermöglicht Self-Service
- \+ deklarativer Ansatz
- \- textuell-deklarative Konfiguration
- \- komplizierter GitOps-Operator
- \- herausfordernd bei zustandsbehafteten Systemen
- \- ersetzt weder Wissen noch Reaktion
- \- wenig Routine bei Störungen





https://msg.direct/techrefresh

Stand: Oktober 2022

msg systems ag

Robert-Bürkle-Straße 1 | 85737 Ismaning/München | Telefon: +49 89 96101-0 | Fax: +49 89 96101-1113 | www.msg.group | info@msg.group