Auswahl des richtigen GitOps Tools für Ihren Amazon EKS-Cluster

AWS Prescriptive Guidance

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Prescriptive Guidance: Auswahl des richtigen GitOps Tools für Ihren Amazon EKS-Cluster

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Handelsmarken und die Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Einführung 1
Gezielte Geschäftsergebnisse 2
Nahtlose Integration mit Amazon EKS 2
Skalierbarkeit und Leistung 2
Sicherheits und Compliance 2
Benutzerfreundlichkeit und Lernkurve 3
Community- und Netzwerkunterstützung 3
Verwaltungsfunktionen für mehrere Cluster 4
Beobachtbarkeit und Überwachung 4
Flexibilität und Anpassung 4
Kontinuierliche Bereitstellung und Unterstützung bei der schrittweisen Bereitstellung 5
Kosteneffektivität und Ressourcennutzung 5
GitOps Tools für EKS-Cluster 6
Argo CD 6
GitOps Unterstützung 6
Architektur 9
Flux 10
GitOps Unterstützung 10
Architektur 13
Weben GitOps 14
GitOps Unterstützung 14
Architektur 17
Jenkins X 18
GitOps Unterstützung 19
Architektur 22
GitLab CI/CD 23
GitOps Unterstützung 23
Spinnaker 26
GitOps Unterstützung 27
Architektur 30
Rancher-Flotte 31
GitOps Unterstützung 31
Architektur 35
Codefresh 35

GitOps Unterstützung 35
Pulumi 40
GitOps Unterstützung 40
GitOps Vergleich der Tools 45
Benutzerfreundlichkeit 45
Kubernetes-Integration 45
CI/CD-Funktionen 45
GitOps Reinheit 45
Multi-Cloud-Unterstützung 46
Multi-Cluster-Support 46
Integration 46
Gemeinschaft und Unterstützung 46
Funktionen für Unternehmen 46
Flexibilität und Erweiterbarkeit 47
Skalierbarkeit 47
Verwaltung der Infrastruktur 47
Programmiermodell und Sprachunterstützung 47
Anwendungsfälle für Argo CD und Flux 48
Allgemeine Überlegungen 48
Anwendungsfälle für Argo CD 48
Flux-Anwendungsfälle 49
Funktionsvergleich 51
Bewährte Methoden für die Auswahl eines GitOps Tools 53
Häufig gestellte Fragen 59
Ressourcen 62
Dokumentverlauf 63
..... lxiv

Auswahl des richtigen GitOps Tools für Ihren Amazon EKS-Cluster

Pradip Kumar Pandey und Pratap Kumar Nanda, Amazon Web Services ()AWS

April 2025 (Geschichte der Dokumente)

In der sich schnell entwickelnden Landschaft cloudnativer Technologien GitOps hat sich diese Methode zu einer leistungsstarken Methode für die Verwaltung und Bereitstellung von Anwendungen und Infrastruktur entwickelt. Wenn Sie Amazon Elastic Kubernetes Service (Amazon EKS) verwenden, können GitOps Implementierungsprinzipien Ihre Bereitstellungsprozesse erheblich verbessern, die Zuverlässigkeit verbessern und den Betrieb rationalisieren. Es stehen eine Vielzahl von GitOps Tools zur Verfügung, und die Auswahl des richtigen Tools für Ihren EKS-Cluster ist eine wichtige Entscheidung, die sich auf die Effizienz Ihres Teams und den Gesamterfolg Ihrer Verfahren auswirken kann. DevOps

Bei der Auswahl eines geeigneten GitOps Tools für Ihre Amazon EKS-Umgebung müssen verschiedene Faktoren sorgfältig berücksichtigt werden, darunter Ihre spezifischen Anforderungen, das Fachwissen Ihres Teams, die Skalierbarkeitsanforderungen und die Integrationsmöglichkeiten mit bestehenden Tools AWS-Services. Jedes Tool hat seine eigenen Funktionen, Stärken und potenziellen Einschränkungen. Daher ist es wichtig, dass Sie Ihre Wahl an den Zielen und dem betrieblichen Kontext Ihres Unternehmens ausrichten.

Dieser Leitfaden untersucht die wichtigsten Überlegungen bei der Auswahl von GitOps Tools für Amazon EKS, vergleicht häufig verwendete Optionen und bietet Einblicke, die Ihnen helfen, eine fundierte Entscheidung zu treffen. Es behandelt neun beliebte GitOps Tools:

- Argo-CD
- Flux
- Weben GitOps
- Jenkins X
- GitLab CI/CD
- Spinnaker
- Rancher-Flotte
- Codefresh
- Pulumi

Gezielte Geschäftsergebnisse

In der folgenden Liste werden mögliche Ziele und Ergebnisse beschrieben, wenn Sie sich für ein Tool zur Implementierung von GitOps Prinzipien in Ihren Entwicklungs- und Betriebsprozessen entscheiden.

Nahtlose Integration mit Amazon EKS

Ihr GitOps Tool sollte sich problemlos in Amazon EKS integrieren lassen und Kompatibilität mit Amazon EKS-spezifischen Funktionen und Optimierungen bieten.

- Native Amazon EKS-Unterstützung: Suchen Sie nach Tools, die integrierte Unterstützung für Amazon EKS bieten, einschließlich einfacher Clusterverbindung und -verwaltung.
- AWS-Service Integration: Stellen Sie sicher, dass das Tool mit anderen Tools AWS-Services wie AWS Identity and Access Management (IAM), Amazon Elastic Container Registry (Amazon ECR) und Amazon interagieren kann. CloudWatch
- Amazon EKS-Add-On-Kompatibilität: Vergewissern Sie sich, dass das Tool Amazon EKS-Add-Ons unterstützt und diese effektiv verwalten kann.

Skalierbarkeit und Leistung

Ihr GitOps Tool sollte in der Lage sein, den Umfang Ihrer Amazon EKS-Operationen zu bewältigen, von kleinen Clustern bis hin zu großen Umgebungen mit mehreren Clustern.

- Ressourceneffizienz: Bewerten Sie den Ressourcenverbrauch des Tools und seine Auswirkungen auf die Cluster-Leistung.
- Umfangreicher Betrieb: Beurteilen Sie, ob das Tool in der Lage ist, zahlreiche Anwendungen und Cluster gleichzeitig zu verwalten.
- Leistung unter Last: Überlegen Sie, wie das Tool bei häufig aktualisierten Updates und umfangreichen Bereitstellungen abschneidet.

Sicherheits und Compliance

Sicherheitsfunktionen und Compliance-Funktionen sind von entscheidender Bedeutung, insbesondere in regulierten Branchen oder beim Umgang mit sensiblen Daten.

- Zugriffskontrolle: Halten Sie Ausschau nach robusten Funktionen für die rollenbasierte Zugriffskontrolle (RBAC), die sich in IAM integrieren lassen.
- Verwaltung von Geheimnissen: Evaluieren Sie, wie das Tool mit vertraulichen Informationen umgeht und sich in unsere anderen Lösungen integrieren lässt. AWS Secrets Manager
- Prüfprotokolle: Stellen Sie sicher, dass das Tool umfassende Protokollierungs- und Prüffunktionen zur Einhaltung von Vorschriften und zur Fehlerbehebung bietet.
- Sicherheitsscans: Ziehen Sie Tools in Betracht, die integrierte Sicherheitsscans nach Sicherheitslücken in Bereitstellungen bieten.

Benutzerfreundlichkeit und Lernkurve

Das Tool sollte benutzerfreundlich sein und den Fähigkeiten Ihres Teams entsprechen, um eine schnelle Einführung und effiziente Nutzung zu gewährleisten.

- Benutzeroberfläche: Beurteilen Sie die Intuitivität der Funktionen der Befehlszeilenschnittstelle (CLI) und der grafischen Benutzeroberfläche (GUI).
- Qualität der Dokumentation: Halten Sie Ausschau nach umfassenden up-to-date Dokumentationen und Tutorials.
- Lernressourcen: Berücksichtigen Sie die Verfügbarkeit von Schulungsmaterialien, Kursen und Community-Ressourcen.

Community- und Netzwerkunterstützung

Eine starke Community und ein starkes Netzwerk können wertvolle Ressourcen, Plugins und langfristige Nachhaltigkeit bieten.

- Aktive Entwicklung: Überprüfe die Häufigkeit der Updates und die Reaktionsfähigkeit der Betreuer.
- Größe der Community: Berücksichtigen Sie die Größe und Aktivität der Benutzergemeinschaft für Support und Wissensaustausch.
- Integrationen von Drittanbietern: Evaluieren Sie die Verfügbarkeit von Plugins und Integrationen mit anderen Tools in Ihrem Stack.

Verwaltungsfunktionen für mehrere Cluster

Wenn Sie über mehrere EKS-Cluster verfügen, ist die Fähigkeit, diese effizient zu verwalten, von entscheidender Bedeutung.

- Zentralisiertes Management: Suchen Sie nach Funktionen, mit denen Sie mehrere Cluster von einer einzigen Steuerungsebene aus verwalten können.
- Clusterverbund: Ziehen Sie Tools in Betracht, die den Kubernetes-Verbund für Multi-Cluster-Anwendungen unterstützen.
- Umgebungsparität: Beurteilen Sie, wie gut das Tool die Konsistenz in verschiedenen Umgebungen wie Entwicklung, Staging und Produktion gewährleistet.

Beobachtbarkeit und Überwachung

Das Tool sollte klare Einblicke in den Status Ihrer Bereitstellungen und den Zustand Ihres Clusters bieten.

- Sichtbarkeit der Bereitstellung: Suchen Sie nach Funktionen, die einen klaren Überblick über den Status und den Verlauf der Bereitstellung bieten.
- Integration mit Überwachungstools: Überlegen Sie, wie gut sich das Tool in beliebte Überwachungslösungen wie Prometheus und Grafana integrieren lässt.
- Warnfunktionen: Beurteilen Sie, ob das Tool in der Lage ist, Warnmeldungen für Bereitstellungsprobleme oder Abweichungen einzurichten und zu verwalten.

Flexibilität und Anpassung

Die Fähigkeit, das Tool an Ihre spezifischen Arbeitsabläufe und Anforderungen anzupassen, ist wichtig für die langfristige Zufriedenheit.

- Erweiterbarkeit: Suchen Sie nach Plugin-Architekturen oder nach Plugin-Architekturen APIs , mit denen Sie die Funktionalität des Tools erweitern können.
- Unterstützung für benutzerdefinierte Ressourcen: Stellen Sie sicher, dass das Tool benutzerdefinierte Kubernetes-Ressourcen effektiv verarbeiten kann.
- Workflow-Anpassung: Beurteilen Sie, wie einfach Sie die GitOps Workflows an die Bedürfnisse Ihres Teams anpassen können.

Kontinuierliche Bereitstellung und Unterstützung bei der schrittweisen Bereitstellung

Fortschrittliche Bereitstellungsstrategien sind oft entscheidend, um Risiken zu minimieren und reibungslose Updates zu gewährleisten.

- Bereitstellungen auf Canary: Halten Sie Ausschau nach integrierter Unterstützung für Canary-Versionen.
- Blue/green deployments: Assess the tool's capabilities for blue/greenBereitstellungsstrategien.
- Rollback-Mechanismen: Sorgen Sie für robuste easy-to-use Rollback-Funktionen für eine schnelle Wiederherstellung nach fehlgeschlagenen Bereitstellungen.

Kosteneffektivität und Ressourcennutzung

Berücksichtigen Sie die Gesamtkosten für die Einführung und Wartung des Tools, einschließlich der direkten und indirekten Kosten.

- Lizenzkosten: Vergleichen Sie Open-Source-Optionen mit kommerziellen Lösungen und berücksichtigen Sie Support- und Unternehmensfunktionen.
- Betriebskosten: Beurteilen Sie die zusätzlichen Betriebskosten in Bezug auf Verwaltung und Wartung.
- Ressourcenverbrauch: Bewerten Sie die Effizienz des Tools in Bezug auf die benötigten Rechen- und Speicherressourcen.

Wenn Sie diese Ergebnisse und ihre Aspekte sorgfältig abwägen, können Sie eine fundierte Entscheidung über das am besten geeignete GitOps Tool für Ihren EKS-Cluster treffen und sicherstellen, dass das Tool den Bedürfnissen, Fähigkeiten und der langfristigen Strategie Ihres Unternehmens entspricht.

GitOps Tools für EKS-Cluster

Es gibt mehrere GitOps Tools für Kubernetes, die derzeit auf dem Markt erhältlich sind. Hier ist eine Liste der am häufigsten verwendeten Optionen:

- Argo-CD
- Flux
- Weben GitOps
- Jenkins X
- GitLab CI/CD
- Spinnaker
- Rancher-Flotte
- Codefresh
- Pulumi

Folgen Sie den Links, um detaillierte Informationen darüber zu erhalten, wie diese Tools GitOps Praktiken implementieren. Jedes Tool hat Stärken und Anwendungsfälle. Die Wahl hängt von Faktoren wie Ihren spezifischen Anforderungen, der vorhandenen Infrastruktur, dem Fachwissen des Teams und den gewünschten Funktionen ab. Es ist wichtig, dass Sie diese Tools auf der Grundlage der Bedürfnisse Ihres Unternehmens und der Komplexität Ihrer Kubernetes-Umgebung bewerten.

Argo CD

Argo CD ist ein weit verbreitetes GitOps Continuous Delivery (CD) -Tool für Kubernetes, das mehreren wichtigen Prinzipien entspricht. GitOps

GitOps Unterstützung

Flächen	Funktionen des Tools
Deklarative Konfiguration	Argo CD verwendet deklarative Konfigura tionen, die in Git-Repositorys gespeichert sind. Der gewünschte Status der Anwendung und Infrastruktur ist in YAML-Dateien definiert

Flächen	Funktionen des Tools
	. Diese Konfigurationen beschreiben, was bereitgestellt werden sollte, nicht, wie sie bereitgestellt werden sollen.
Versionskontrollsystem als zentrale Informationsquelle	Git-Repositorien dienen als zentrale Informationsquelle für das gesamte System. Alle Änderungen an der Anwendung und Infrastruktur werden über Git vorgenommen. Dies gewährleistet einen vollständigen Prüfpfad und die Möglichkeit, zu einem beliebigen vorherigen Status zurückzukehren.
Automatisierte Synchronisation	Argo CD überwacht das Git-Repository kontinuierlich auf Änderungen. Wenn Änderungen erkannt werden, synchronisiert es automatisch den aktuellen Status des Clusters mit dem gewünschten Status, der in Git definiert ist. Dadurch wird sichergestellt, dass der Cluster immer den Status widerspiegelt, der im Repository beschrieben ist.
Kubernetes-native	Argo CD wurde speziell für Kubernetes-Umgebungen entwickelt. Es nutzt den deklarativen Charakter und die benutzerdefinierten Ressourcen in Kubernetes für die Verwaltung von Anwendungen.
Self-healing und Drift-Erkennung	Argo CD vergleicht regelmäßig den Live-Status des Clusters mit dem gewünschten Status in Git. Wenn es Abweichungen (Unterschiede zwischen dem tatsächlichen und dem gewünschten Zustand) feststellt, kann es diese Abweichungen automatisch korrigieren.

Flächen	Funktionen des Tools
Multi-cluster und Unterstützung für mehrere Mandanten	Argo CD kann mehrere Kubernetes-Cluster von einer einzigen Instanz aus verwalten. Es unterstützt Mehrmandantenfähigkeit, sodass verschiedene Teams ihre Anwendungen unabhängig voneinander verwalten können.
Definition der Anwendung	Anwendungen in Argo CD werden mithilfe der Anwendungs-CRD (benutzerdefinierte Ressourcendefinition) definiert. Auf diese Kubernetes-native Weise kann definiert werden, was und wie bereitgestellt werden soll.
Trennung von Bereitstellung und Veröffent lichung	Argo CD trennt die Bereitstellung von Code von seiner Veröffentlichung für Benutzer. Dies wird durch verschiedene Bereitstellungsstrategien erreicht, z. B. blue/green durch kanarische Bereitstellungen.
Beobachtbarkeit und Überprüfbarkeit	Argo CD bietet eine Web-Benutzeroberfläche und CLI zur Beobachtung des Status von Anwendungen und Clustern. Alle Aktionen werden protokolliert, um einen klaren Prüfpfad für Änderungen und Implementierungen zu gewährleisten.
Sicherheit und RBAC	Argo CD ist in die rollenbasierte Zugriffsk ontrolle (RBAC) von Kubernetes integriert. Es unterstützt die Single-Sign-On-Integration für Authentifizierung und Autorisierung.
Steckbare Architektur	Argo CD unterstützt verschiedene Quellcode verwaltungssysteme, Helm-Charts, Kustomize und andere Kubernetes-Manifestformate. Diese Flexibilität ermöglicht es, sich in verschiedene Umgebungen und Workflows einzufügen.

Flächen	Funktionen des Tools
Kontinuierliche Bereitstellung (CD)	Obwohl sich Argo CD auf Continuous Delivery konzentriert, kann es in CI-Tools (Continuous Integration) integriert werden, um eine vollständ ige CI/CD Pipeline zu erstellen.

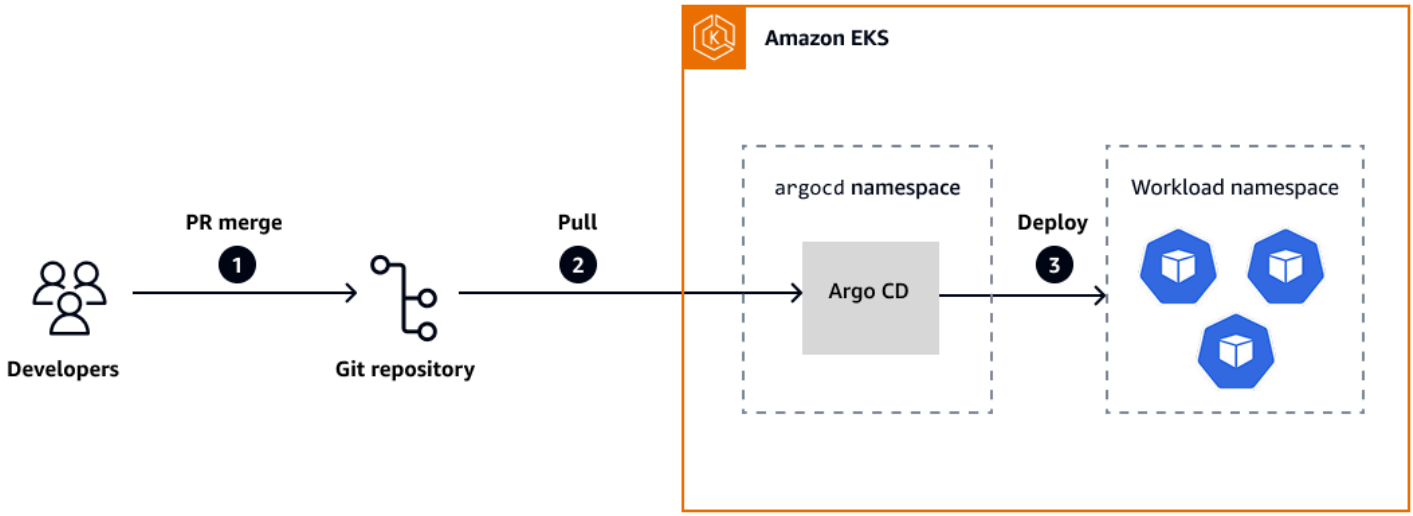
Durch die Einhaltung dieser GitOps Prinzipien bietet Argo CD eine robuste, skalierbare und sichere Methode zur Verwaltung von Kubernetes-Bereitstellungen. Es stellt sicher, dass der Betriebsstatus Ihres Systems immer mit dem gewünschten Status synchronisiert ist, der in Ihrem Git-Repository definiert ist, und sorgt für Konsistenz, Zuverlässigkeit und einfache Verwaltung in komplexen Kubernetes-Umgebungen.

Szenarien und Anforderungen, die Argo CD erfüllen kann, finden Sie weiter unten in diesem Handbuch unter Anwendungsfälle für Argo CD. Einen Vergleich zwischen Argo CD und Flux finden Sie weiter unten in diesem Handbuch unter Funktionsvergleich.

Weitere Informationen finden Sie in der Argo-CD-Dokumentation.

Architektur

Das folgende Diagramm zeigt einen GitOps-driven CD-Workflow, der Argo CD innerhalb eines EKS-Clusters verwendet. Ausführliche Informationen finden Sie in der Argo-CD-Dokumentation.



Wobei:

- Schritt 1: Zusammenführung von Pull-Requests (PR). Ein Entwickler überträgt Änderungen an Kubernetes-Manifesten oder Helm-Diagrammen, die in einem Git-Repository gespeichert sind. Wenn der PR überprüft und mit dem Hauptzweig zusammengeführt wurde, wird der gewünschte Status der Anwendung in der Quellcodeverwaltung aktualisiert.
- Schritt 2: Repository-Synchronisierung. Argo CD läuft in einem dedizierten Namespace (`argocd`) im EKS-Cluster und überwacht kontinuierlich das konfigurierte Git-Repository. Wenn es Änderungen erkennt, ruft es die neuesten Updates ab, um den deklarierten Status abzugleichen.
- Schritt 3: Bereitstellung im Ziel-Namespace. Argo CD vergleicht den gewünschten Status von Git mit dem Live-Status im Cluster. Anschließend werden die erforderlichen Änderungen auf den Ziel-Workload-Namespace angewendet, sodass die Anwendung entsprechend bereitgestellt oder aktualisiert wird. Dazu gehört die Verwaltung von Kubernetes-Ressourcen wie Deployments, Services und Secrets ConfigMaps, um die Cluster-Konsistenz mit der Git Source of Truth aufrechtzuerhalten.

Flux

Flux ist ein weiteres Tool für Kubernetes, das GitOps Prinzipien auf einzigartige Weise implementiert.

GitOps Unterstützung

Flächen	Funktionen des Tools
Git als zentrale Informationsquelle	Flux verwendet Git-Repositorys als endgültig e Quelle für die Definition des gewünschten Systemstatus. Die gesamte Konfiguration für Anwendungen und Infrastruktur wird in Git gespeichert.
Deklarative Konfiguration	Flux arbeitet mit deklarativen Beschreibungen des gewünschten Zustands Ihres Clusters. Bei diesen Beschreibungen handelt es sich in der Regel um Kubernetes-Manifeste, Helm-Diag ramme oder Kustomize-Overlays.
Automatisierte Synchronisation	Flux überwacht das Git-Repository kontinuie rlich auf Änderungen. Wenn es Änderungen

Flächen	Funktionen des Tools
	erkennt, wendet es sie automatisch auf den Cluster an.
Kubernetes-native	Flux besteht aus einer Reihe von Kubernetes-Controllern und benutzerdefinierten Ressourcen. Es nutzt die Erweiterungsmechanismen in Kubernetes, um Funktionen bereitzustellen. GitOps
Pull-based Bereitstellungsmodell	Im Gegensatz zu herkömmlichen Push-basierten CI/CD Systemen verwendet Flux ein Pull-basiertes Modell. Der Cluster ruft den gewünschten Status aus Git ab, anstatt ein externes System zu verwenden, um Änderungen zu übertragen.
Kontinuierlicher Abgleich	Flux vergleicht ständig den aktuellen Zustand des Clusters mit dem gewünschten Zustand in Git. Es korrigiert automatisch jede Abweichung, die zwischen diesen Zuständen erkannt wird.
Multi-tenancy	Flux unterstützt Mehrmandantenfähigkeit durch seine Konzepte für Anpassungen und. HelmReleases Verschiedene Teams können ihre eigenen Teile der Konfiguration unabhängig voneinander verwalten.
Progressive Lieferung	Flux unterstützt mit seiner Flagger-Komponente fortschrittliche Bereitstellungsstrategien wie Veröffentlichungen und A/B Tests auf Canary.
Integration von Helmen	Flux bietet native Unterstützung für Helm, sodass Sie Helm-Versionen einfach verwalten können GitOps.

Flächen	Funktionen des Tools
Automatisierung der Image-Aktualisierung	Flux kann Container-Images in Git automatis ch aktualisieren, wenn neue Versionen in der Container-Registry verfügbar sind.
Unterstützung anpassen	Sie können die native Unterstützung von Flux für Kustomize verwenden, um Kubernetes-Manifeste anzupassen und zu patchen.
Sicherheit und RBAC	Flux lässt sich zur Zugriffskontrolle in Kubernetes RBAC integrieren. Es unterstüt zt die Verwaltung von Geheimnissen über verschiedene Backends.
Beobachtbarkeit	Flux bietet Statusinformationen und Kennzahle n zu Abstimmungen und Vorgängen. Es lässt sich in Überwachungstools integrieren, um die Beobachtbarkeit zu verbessern.
Event-driven Architektur	Flux verwendet einen ereignisgesteuerten Ansatz zur Implementierung von Abstimmun gen und Aktualisierungen.
Erweiterbarkeit	Das Tool ist so konzipiert, dass es erweiterb ar ist, sodass Sie benutzerdefinierte Controller und Ressourcen hinzufügen können.
Cross-cluster Synchronisation	Flux unterstützt die Verwaltung mehrerer Cluster aus einem einzigen Satz von Repositor ys.
Abhängigkeitsmanagement	Es ermöglicht die Definition von Abhängigk eiten zwischen verschiedenen Teilen Ihres Systems und stellt die korrekte Reihenfolge der Operationen sicher.

Flächen	Funktionen des Tools
Webhook-Empfänger	Sie können Flux so konfigurieren, dass es Webhooks von Git-Anbietern oder anderen Systemen empfängt, um den sofortigen Abgleich zu starten.

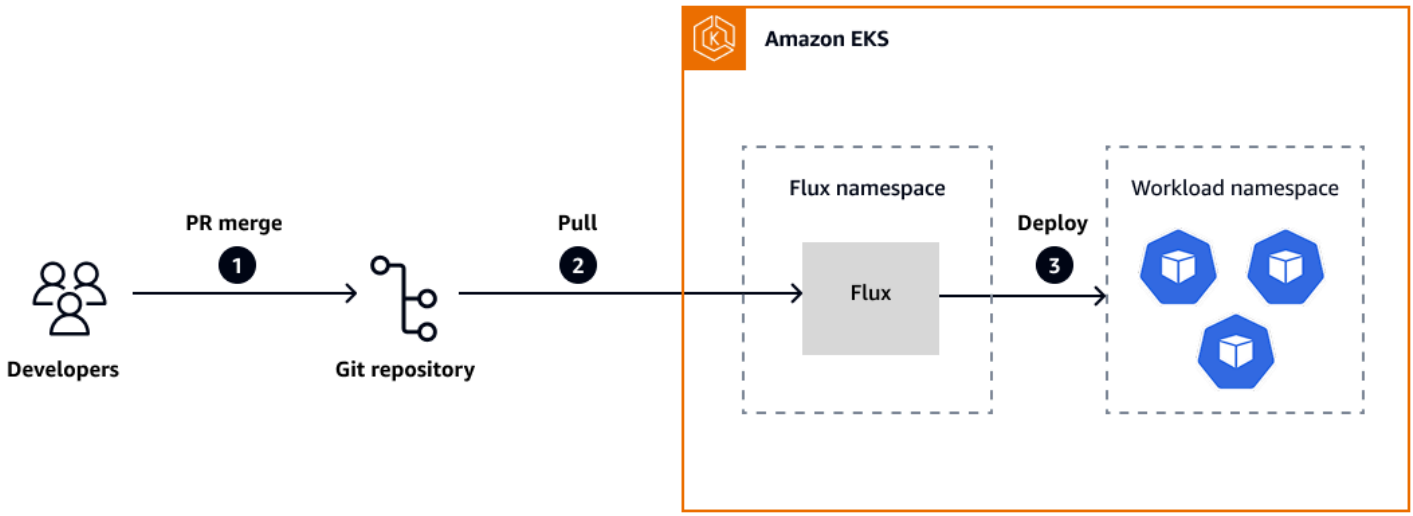
Durch die Implementierung dieser GitOps Prinzipien bietet Flux ein robustes und flexibles System für die Verwaltung von Kubernetes-Clustern und -Anwendungen. Es stellt sicher, dass Ihre Infrastruktur und Anwendungen immer mit Ihren Git-Repositorys synchronisiert sind, und sorgt für Konsistenz, Zuverlässigkeit und einfache Verwaltung in komplexen Kubernetes-Umgebungen. Durch den Kubernetes-native Ansatz und den Fokus auf Automatisierung eignet sich das Tool besonders gut für Cloud-native Umgebungen.

Szenarien und Anforderungen, die Flux erfüllen kann, finden Sie unter Flux-Anwendungsfälle weiter unten in diesem Handbuch. Einen Vergleich zwischen Argo CD und Flux finden Sie weiter unten in diesem Handbuch unter Funktionsvergleich.

Weitere Informationen finden Sie in der Flux-Dokumentation.

Architektur

Das folgende Diagramm zeigt einen GitOps-driven CD-Workflow, der Flux innerhalb eines EKS-Clusters verwendet. Detaillierte Informationen finden Sie in der Flux-Dokumentation.



Wobei:

- Schritt 1: Zusammenführung von Pull-Requests (PR). Ein Entwickler überträgt Änderungen an Kubernetes-Manifesten oder Helm-Diagrammen, die in einem Git-Repository gespeichert sind. Wenn der PR überprüft und mit dem Hauptzweig zusammengeführt wurde, wird der gewünschte Status der Anwendung in der Quellcodeverwaltung aktualisiert.
- Schritt 2: Repository-Synchronisierung. Flux läuft in einem dedizierten Namespace im EKS-Cluster und überwacht kontinuierlich das konfigurierte Git-Repository. Wenn es Änderungen erkennt, ruft es die neuesten Updates ab, um den deklarierten Status abzugleichen.
- Schritt 3: Bereitstellung im Ziel-Namespace. Flux vergleicht den gewünschten Status von Git mit dem Live-Status im Cluster. Anschließend werden die erforderlichen Änderungen auf den Ziel-Workload-Namespace angewendet, sodass die Anwendung entsprechend bereitgestellt oder aktualisiert wird.

Weben GitOps

Weave GitOps wurde von Weaveworks entwickelt, dem Unternehmen, das den Begriff eingeführt hat. GitOps Dieses Tool bietet eine umfassende GitOps Lösung, die auf den Kernprinzipien aufbaut. GitOps

GitOps Unterstützung

Flächen	Funktionen des Tools
Git als zentrale Informationsquelle	Weave GitOps verwendet Git-Repositorys als maßgebliche Quelle für die Definition des gewünschten Systemstatus. Alle Konfigura tionen, einschließlich Anwendungsmanifeste, Infrastrukturdefinitionen und Richtlinien, werden in Git gespeichert.
Deklarative Konfiguration	Das System stützt sich auf deklarative Beschreibungen des gesamten Systemstatus. Bei diesen Beschreibungen handelt es sich in der Regel um Kubernetes-Manifeste, Helm-Diagramme oder andere deklarative Formate.
Automatisierte Synchronisation	Weave überwacht Git-Repositorys GitOps kontinuierlich auf Änderungen. Wenn es

Flächen	Funktionen des Tools
	Änderungen erkennt, wendet es sie automatis ch auf die Zielumgebung an.
Kubernetes-native Architektur	Weave GitOps besteht aus einer Reihe von Kubernetes-Controllern und benutzerdefinierten Ressourcen. Es nutzt die Erweiterungsmechan ismen in Kubernetes, um Funktionen bereitzus tellen. GitOps
Kontinuierliche Versöhnung	Dieses Tool vergleicht ständig den aktuellen Status des Clusters mit dem gewünschten Status, der in Git definiert ist. Es korrigiert automatisch jede Abweichung, die zwischen diesen Zuständen festgestellt wird.
Multi-cluster Verwaltung	Weave GitOps unterstützt die Verwaltun g mehrerer Kubernetes-Cluster von einer einzigen Steuerungsebene aus. Es ermöglicht eine konsistente Anwendungsbereitstellung in verschiedenen Umgebungen.
Richtlinie als Code	Weave GitOps beinhaltet das Konzept der Richtlinie als Code zur Durchsetzung von Sicherheits- und Compliance-Regeln. Richtlini en werden zusammen mit Anwendungscode und Infrastrukturdefinitionen versionsgesteuert.
Progressive Bereitstellung	Dieses Tool unterstützt fortgeschrittene Bereitstellungsstrategien wie Canary-Releases und blue/green Deployments. Es ist in Flagger integriert und ermöglicht so eine automatisierte, progressive Bereitstellung.

Flächen	Funktionen des Tools
Beobachtbarkeit und Dashboards	Weave GitOps bietet integrierte Dashboards zur Überwachung des Status von Anwendungen und Clustern. Es bietet Einblicke in Abstimmungsprozesse und den Zustand von Clustern.
Von Haus aus sicher	Das Tool implementiert bewährte Sicherheitsverfahren, einschließlich RBAC-Integration und Verwaltung von Geheimnissen. Es unterstützt verschiedene Authentifizierungsmethoden und lässt sich in Identitätsanbieter für Unternehmen integrieren.
Erweiterbarkeit und Integration	Das Tool ist so konzipiert, dass es mit einer Vielzahl von Cloud-nativen Tools funktioniert. Es unterstützt beliebte Tools wie Flux, Helm und Kustomize.
Self-service Entwicklerplattformen	Weave GitOps ermöglicht die Schaffung von Self-Service-Plattformen für Entwickler. Es bietet Vorlagen und Leitplanken für die Anwendungsbereitstellung.
GitOps -Automatisierung	Das Tool automatisiert viele Aspekte des GitOps Workflows, einschließlich der Generierung von Pull-Requests für Updates.
Pipelines für die kontinuierliche Lieferung	Es lässt sich in CI/CD Systeme integrieren, um durchgängige Lieferleitungen zu erstellen.
Prüfung und Einhaltung der Vorschriften	Weave DevOps bietet einen vollständigen Prüfpfad aller Änderungen und Aktionen. Es hilft Ihnen, Compliance-Anforderungen durch Versionskontrolle und automatisierte Prozesse zu erfüllen.

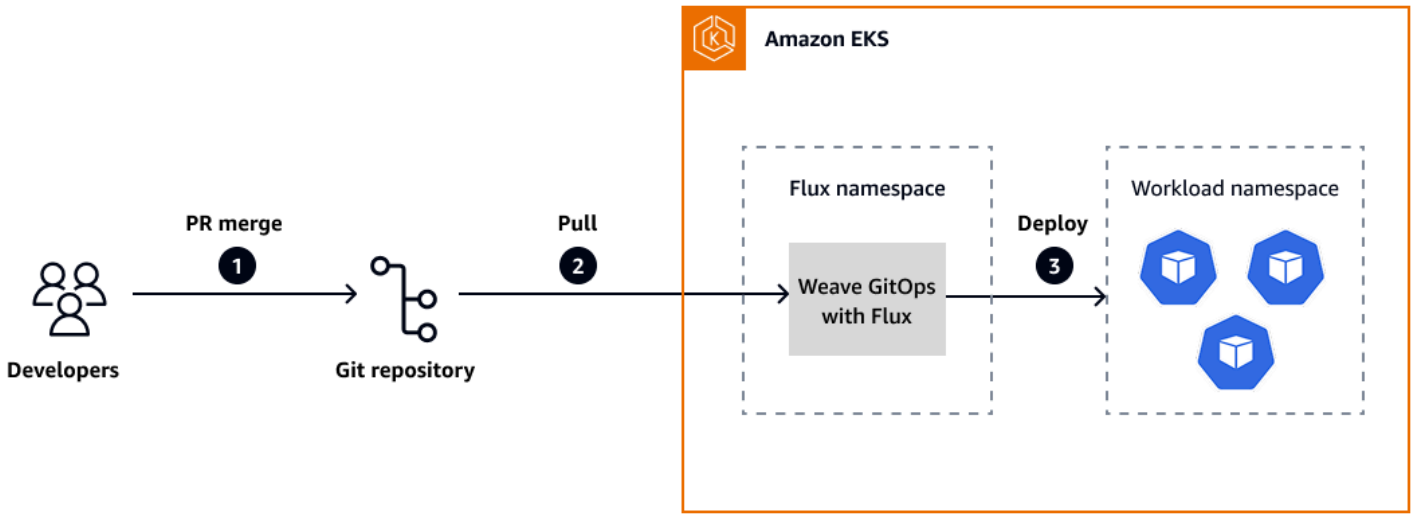
Flächen	Funktionen des Tools
Skalierbarkeit	Das Tool ist so konzipiert, dass es von kleinen Projekten bis hin zu großen Implementierungen auf Unternehmensebene skaliert werden kann.
Zusammenarbeit im Team	Weave GitOps erleichtert die Zusammenarbeit zwischen Entwicklungs- und Betriebsteams durch Workflows. Git-based
GitOps als Service	Dieses Tool wird GitOps als verwalteter Service angeboten, was die Einführung und Verwaltung vereinfacht.
Hybrid- und Multi-Cloud-Unterstützung	Weave GitOps ermöglicht ein konsistentes Management für verschiedene Cloud-Anbieter und lokale Umgebungen.
Kontinuierliche Sicherheit	Das Tool integriert Sicherheitsscans und die Durchsetzung von Richtlinien während des gesamten Bereitstellungsprozesses.

Weave GitOps implementiert diese Prinzipien, um eine umfassende GitOps Lösung bereitzustellen, die über die grundlegende Automatisierung der Bereitstellung hinausgeht. Ziel ist es, ein vollständiges Betriebsmodell für Cloud-native Anwendungen zu entwickeln, das sich auf Sicherheit, Skalierbarkeit und Benutzerfreundlichkeit konzentriert. Durch die Einhaltung dieser GitOps Prinzipien GitOps hilft Weave Unternehmen dabei, eine konsistente, überprüfbare und effiziente Verwaltung ihrer Kubernetes-Umgebungen über mehrere Cluster und Cloud-Anbieter hinweg zu erreichen.

Weitere Informationen finden Sie in der Weave-Dokumentation. GitOps

Architektur

Das folgende Diagramm zeigt einen GitOps-driven CD-Workflow, der Weave GitOps innerhalb eines EKS-Clusters verwendet. Detaillierte Informationen finden Sie im Weave-Repository GitOps.



Wobei:

- Schritt 1: Zusammenführung von Pull-Requests (PR). Ein Entwickler überträgt Änderungen an Kubernetes-Manifesten oder Helm-Diagrammen, die in einem Git-Repository gespeichert sind. Wenn der PR überprüft und mit dem Hauptzweig zusammengeführt wurde, wird der gewünschte Status der Anwendung in der Quellcodeverwaltung aktualisiert.
- Schritt 2: Repository-Synchronisierung. Weave GitOps läuft im Flux-Namespace im EKS-Cluster und überwacht kontinuierlich das konfigurierte Git-Repository. Wenn es Änderungen erkennt, ruft es die neuesten Updates ab, um den deklarierten Status abzugleichen.
- Schritt 3: Bereitstellung im Ziel-Namespace. Weave GitOps vergleicht den gewünschten Status von Git mit dem Live-Status im Cluster. Anschließend werden die erforderlichen Änderungen auf den Ziel-Workload-Namespace angewendet, sodass die Anwendung entsprechend bereitgestellt oder aktualisiert wird.

Jenkins X

Jenkins X ist eine Cloud-native CI/CD Open-Source-Plattform, die Prinzipien für Kubernetes-Umgebungen implementiert. GitOps Jenkins X ist zwar nicht ausschließlich ein GitOps Tool wie Argo CD oder Flux, integriert aber Praktiken in seine Workflows. GitOps

GitOps Unterstützung

Flächen	Funktionen des Tools
Git-centric Arbeitsablauf	Jenkins X verwendet Git-Repositorys als primäre Informationsquelle sowohl für den Anwendungscode als auch für die Konfigura tion. Alle Änderungen an Anwendungen und Infrastruktur werden über Git vorgenommen.
Umgebung als Code (EaC)	Umgebungen (wie Staging und Produktion) werden in Git-Repositorys als Code definiert . Dies ermöglicht die Versionskontrolle und Überprüfung von Umgebungskonfigurationen.
Automatisierte CI/CD Pipelines	Jenkins X richtet automatisch CI/CD Pipelines für Projekte ein. Diese Pipelines sind als Code (Pipeline als Code) definiert und in Git gespeichert.
Kubernetes-native	Jenkins X wurde speziell für Kubernetes-Umgebungen entwickelt. Es verwendet Kubernetes-Ressourcen und benutzerdefinierte Ressourcendefinitionen (CRDs).
Umgebungen in der Vorschau anzeigen	Jenkins X erstellt automatisch temporäre Umgebungen für Pull-Requests. Es ermöglich t die einfache Überprüfung und das Testen von Änderungen vor der Zusammenführung.
Förderung zwischen Umgebungen	Jenkins X verwendet einen GitOps Ansatz zur Förderung von Anwendungen zwischen Umgebungen (z. B. vom Staging bis zur Produktion). Werbeaktionen werden mithilfe von Pull-Requests abgewickelt, um einen ordnungsgemäßen Überprüfungs- und Genehmigungsprozess sicherzustellen.

Flächen	Funktionen des Tools
Verwaltung von Helm-Diagrammen	Jenkins X verwendet Helm-Diagramme, um Anwendungen zu verpacken und bereitzus tellen. Diagramme werden in Git-Repositorys versionsgesteuert.
Automatisierte Versionierung	Jenkins X verwaltet automatisch die Versionie rung von Anwendungen und Releases. Es verwendet semantische Versionierung und generiert Versionshinweise.
ChatOps Integration	Jenkins X unterstützt gängige ChatOps Operationen. Dies entspricht den GitOps Prinzipien der Automatisierung und Zusammenarbeit.
Erweiterbarkeit	Dieses Tool bietet ein Plugin-System zur Erweiterung der Funktionalität. Es ermöglicht die Integration mit verschiedenen Cloud-nativen Tools.
Infrastructure as Code (IaC)	Jenkins X unterstützt Terraform, CloudForm ation AWS Cloud Development Kit (AWS CDK), und andere IaC-Tools zur Definition und Verwaltung der Infrastruktur. Infrastrukturdefin itionen werden zusammen mit dem Anwendung scode versionsgesteuert.
Automatisierte Rollbacks	Jenkins X unterstützt automatisierte Rollbacks , wenn nach der Bereitstellung Probleme festgestellt werden.
Verwaltung von Secrets	Das Tool lässt sich in externe Secrets-M anagement-Lösungen integrieren, um vertrauli che Informationen sicher zu handhaben.

Flächen	Funktionen des Tools
Beobachtbarkeit	Jenkins X bietet die Integration mit Überwachungs- und Protokollierungstools für die Beobachtbarkeit.
Multi-cloud Unterstützung	Jenkins X ist so konzipiert, dass es mit verschiedenen Cloud-Anbietern und lokalen Umgebungen funktioniert.
Zusammenarbeit im Team	Dieses Tool fördert die Zusammenarbeit durch Git-based Workflows und Pull-Requests.
Kontinuierliches Feedback	Das Tool bietet schnelles Feedback zu Änderungen durch automatisierte Test- und Vorschauumgebungen.
DevOps bewährte Verfahren	Jenkins X implementiert standardmäßig DevOps bewährte Verfahren, einschließlich GitOps Prinzipien.
Deklarative Konfiguration	Das Tool verwendet deklarative Konfigurationen zur Definition von Anwendungen und Umgebungen.
Automatisierte Upgrades	Jenkins X bietet Tools zur Automatisierung von Upgrades der Jenkins X-Plattform selbst.

Jenkins X implementiert diese GitOps Prinzipien, um eine umfassende Lösung für Kubernetes zu erstellen. CI/CD Es zielt darauf ab, den gesamten Softwarebereitstellungsprozess, vom Code-Commit bis zur Produktionsbereitstellung, zu automatisieren und zu rationalisieren und dabei die Praktiken einzuhalten. GitOps Auf diese Weise unterstützt es Teams dabei, schnellere, zuverlässigere und konsistentere Implementierungen in cloudnativen Umgebungen zu erreichen.

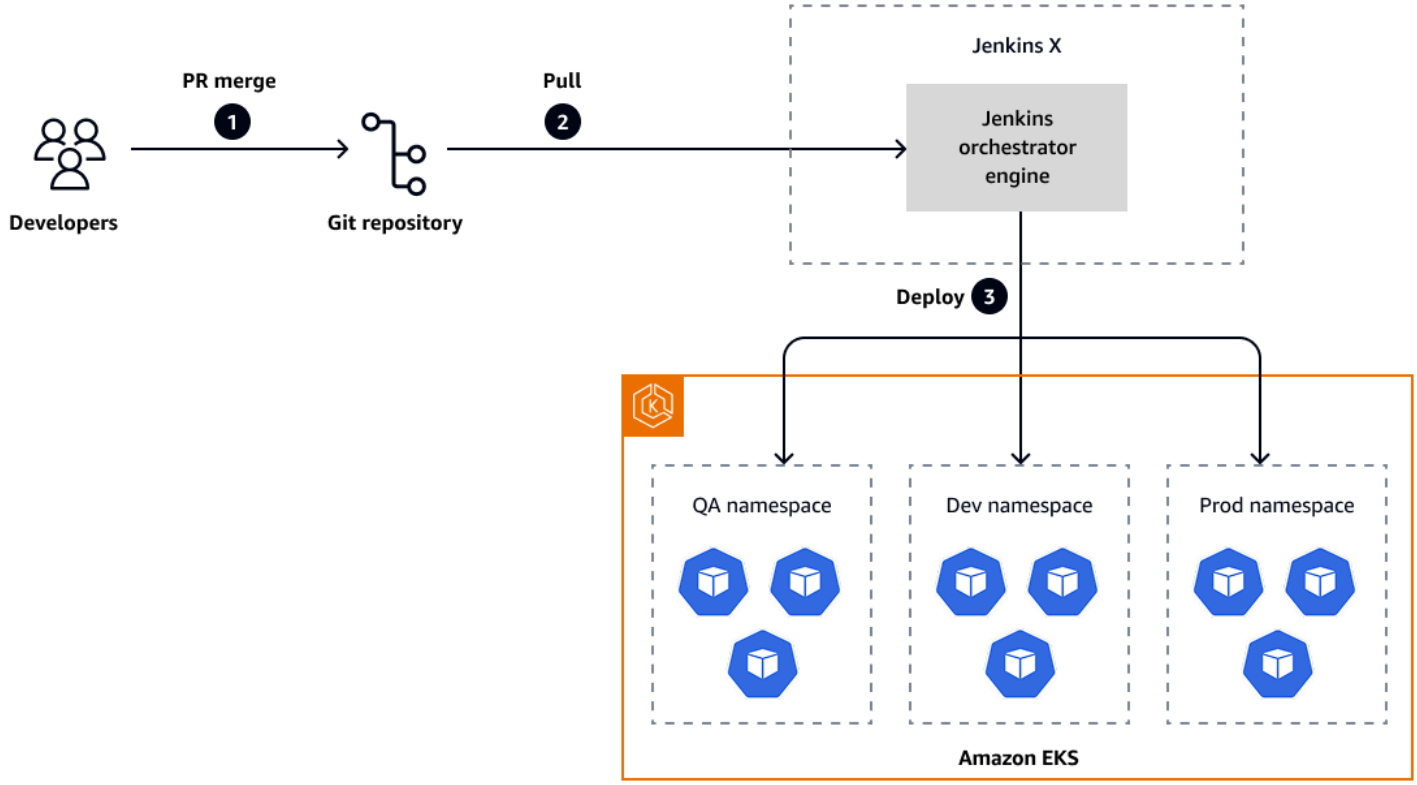
Der Hauptunterschied zwischen Jenkins X und Tools wie Argo CD oder Flux besteht darin, dass Jenkins X eine umfassendere CI/CD Lösung bietet, einschließlich Build-Automatisierung und Pipeline-Management, während gleichzeitig Prinzipien für die Bereitstellung und das Umgebungsmanagement berücksichtigt GitOps werden. Dadurch eignet es sich besonders für

Teams, die eine Komplettlösung benötigen, die sowohl CI- als auch CD-Aspekte in einem einzigen Framework abdeckt. GitOps

Weitere Informationen finden Sie in der Jenkins X-Dokumentation.

Architektur

Das folgende Diagramm zeigt einen GitOps-driven CD-Workflow, der Jenkins X verwendet. Ausführliche Informationen finden Sie in der Jenkins X-Dokumentation.



Wobei:

- Schritt 1: Zusammenführung von Pull-Requests (PR). Ein Entwickler erstellt eine Pull-Anfrage, die Änderungen an Kubernetes-Manifesten, Helm-Diagrammen oder Anwendungscode enthält, der in einem Git-Repository gespeichert ist. Nach der Überprüfung und Genehmigung wird der PR mit dem Hauptzweig zusammengeführt und aktualisiert den gewünschten Status in der Quellcodeverwaltung.
- Schritt 2: Repository-Synchronisierung. Jenkins X löst automatisch eine CI/CD Pipeline aus, wenn es die Änderung erkennt. Die Pipeline erstellt, testet und verbreitet die Anwendung mithilfe von Prinzipien in verschiedenen Umgebungen (z. B. Staging und Produktion). GitOps

- Schritt 3: Bereitstellung in Ziel-Namespaces. Jenkins X aktualisiert die Umgebungs-Repositorys (Staging und Produktion) mit den neuen Anwendungsversionen. Der Cluster gleicht die Änderungen automatisch ab, indem er die neuesten Manifeste aus Git abruft und die Anwendung in den entsprechenden Namespaces bereitstellt.

GitLab CI/CD

GitLab CI/CD ist ein integrierter Bestandteil der GitLab Plattform, der kontinuierliche Integrations-, Bereitstellungs- und Bereitstellungsfunktionen bietet. Obwohl es sich GitLab CI/CD nicht ausschließlich um ein GitOps Tool handelt, können Sie es so konfigurieren, dass es GitOps Prinzipien implementiert, insbesondere wenn Sie es für Kubernetes-Bereitstellungen verwenden.

GitOps Unterstützung

Flächen	Funktionen des Tools
Git als zentrale Informationsquelle	GitLab CI/CD verwendet Git-Repositorys, um sowohl Anwendungscode als auch Infrastrukturkonfigurationen zu speichern. Alle Änderungen am System werden über Git vorgenommen, wodurch eine vollständige Historie und ein Audit-Trail gewährleistet sind.
Deklarative Konfiguration	GitLab CI/CD Pipelines werden in einer .gitlab-ci.yml-Datei definiert, einer deklarativen Konfiguration, die im Git-Repository gespeicherte ist. Kubernetes-Manifeste, Helm-Diagramme oder andere Infrastructure-as-Code-Dateien (IaC) können im selben Repository gespeicherte werden, um den gewünschten Status der Infrastruktur zu definieren.
Automatisierte Pipelines	GitLab CI/CD löst automatisch Pipelines aus, wenn Änderungen in das Repository übertragen werden. Diese Pipelines können Phasen für die Erstellung, das Testen und die Bereitstellung von Anwendungen beinhalten.

Flächen	Funktionen des Tools
Kubernetes-Integration	GitLab CI/CD bietet native Kubernetes-Integration und unterstützt GitOps-style Bereitstellungen in Kubernetes-Clustern. Es kann automatisch Kubernetes-Ressourcen basierend auf der Konfiguration in Git erstellen und verwalten.
Verwaltung der Umgebung	GitLab CI/CD unterstützt die Definition mehrerer Umgebungen (wie Staging und Produktion) als Code. Bereitstellungen in diesen Umgebungen können automatisiert werden oder erfordern möglicherweise eine manuelle Genehmigung, sofern die geltenden Verfahren eingehalten GitOps werden.
Bewerbungen überprüfen	GitLab kann automatisch temporäre Umgebungen für Merge-Anfragen erstellen, ähnlich wie Vorschauumgebungen in anderen GitOps Tools. Dies ermöglicht das einfache Überprüfen und Testen von Änderungen vor Zusammenführungen.
Fortlaufende Bereitstellung	GitLab CI/CD kann so konfiguriert werden, dass Änderungen automatisch an Kubernetes-Clustern bereitgestellt werden, wenn Änderungen in bestimmten Branches zusammengeführt werden.
IaC	GitLab CI/CD unterstützt die Integration mit Tools wie Terraform und die Verwaltung der Infrastruktur als CloudFormation Code. Infrastrukturdefinitionen können zusammen mit dem Anwendungscode versionsgesteuert werden.

Flächen	Funktionen des Tools
Beobachtbarkeit und Überwachung	GitLab CI/CD bietet integrierte Überwachungs- und Beobachtbarkeitsfunktionen, einschließlich der Integration mit Prometheus und Grafana.
Sicherheitsscans	GitLab CI/CD umfasst integrierte Tools für Sicherheitsscans, die in die CI/CD Pipeline integriert werden können, um die Sicherheit als Teil des GitOps Workflows durchzusetzen.
Container-Registrierung	GitLab CI/CD beinhaltet eine integrierte Container-Registry für die nahtlose Integration der Container-Imageverwaltung in den GitOps Workflow.
Automatisch DevOps	Mit der automatischen DevOps Funktion GitLab CI/CD können automatisch CI/CD Pipelines konfiguriert werden, die den GitOps Prinzipien für Kubernetes-Bereitstellungen folgen.
Workflows für Genehmigungen	GitLab CI/CD unterstützt Genehmigungsverfah ren für Bereitstellungen, die kontrollierte Beförderungen zwischen Umgebungen ermöglichen.
Verwaltung von Secrets	GitLab CI/CD bietet Funktionen zur sicheren Verwaltung und Verwendung von Geheimnis sen innerhalb von CI/CD Pipelines.
Versionierung und Veröffentlichungen	GitLab CI/CD unterstützt die automatische Versionierung und das Versionsmanagement als Teil des CI/CD Prozesses.
Rollbacks	GitLab CI/CD ermöglicht einfache Rollbacks auf frühere Versionen, falls nach der Bereitstellung Probleme festgestellt werden.

Flächen	Funktionen des Tools
Prüfungsprotokolle	GitLab CI/CD bietet umfassende Auditprot okolle für alle Aktionen, um den Aspekt der Rückverfolgbarkeit von zu unterstützen. GitOps
Multi-project Rohrleitungen	GitLab CI/CD unterstützt komplexe GitOps Workflows, die sich über mehrere Projekte oder Repositorien erstrecken.
ChatOps	GitLab CI/CD unterstützt ChatOps Integrati onen, die Zusammenarbeit und Abläufe über Chat-Schnittstellen ermöglichen.
Verwaltung von Kubernetes-Clustern	GitLab CI/CD bietet Funktionen für die Verwaltung von Kubernetes-Clustern direkt über die Benutzeroberfläche. GitLab

Es GitLab CI/CD ist zwar nicht ausschließlich dafür konzipiert GitOps, kann aber effektiv zur Implementierung von GitOps Praktiken eingesetzt werden, insbesondere für Teams, die dies bereits GitLab als primäre Entwicklungsplattform verwenden. Sein integrierter Ansatz, der Quellcodeverwaltung und Kubernetes-Management kombiniert, macht es zu einem leistungsstarken Tool für die Implementierung GitOps von Workflows. CI/CD

Der Hauptunterschied zwischen GitLab CI/CD speziellen GitOps Tools wie Argo CD oder Flux besteht darin, dass sie eine umfassendere Plattform GitLab bieten, die neben ihren Funktionen auch Quellcodeverwaltungsmanagement, Problemverfolgung und andere Entwicklungstools umfasst. CI/CD Dadurch eignet es sich besonders für Teams, die eine Komplettlösung benötigen, mit der GitOps Verfahren innerhalb eines umfassenderen Entwicklungssystems implementiert werden können.

Weitere Informationen zu GitLab CI/CD und ihrer Architektur finden Sie in der GitLab CI/CD Dokumentation.

Spinnaker

Spinnaker ist zwar nicht ausschließlich als GitOps Tool konzipiert, aber Sie können es so konfigurieren, dass es GitOps Prinzipien implementiert, insbesondere wenn Sie es für Cloud-native und Kubernetes-Bereitstellungen verwenden.

GitOps Unterstützung

Flächen	Funktionen des Tools
Deklarative Konfiguration	Spinnaker verwendet deklarative Pipeline-Definitionen, die normalerweise als JSON- oder YAML-Dateien gespeichert werden. Diese Pipeline-Definitionen können in Git-Repositorys versionsgesteuert werden, was den Gepflogen heiten entspricht. GitOps
IaC	Spinnaker unterstützt die Definition von Infrastruktur- und Bereitstellungskonfiguratio nen als Code. Diese Definitionen können in Git-Repositorys gespeichert werden und können als zentrale Informationsquelle dienen.
Multi-cloud Bereitstellungen	Spinnaker ist so konzipiert, dass es mit mehreren Cloud-Anbietern und Kubernetes-Clustern funktioniert. Es ermöglicht konsistente GitOps Praktiken in verschiedenen Umgebunge n.
Pipeline als Code	Spinnaker-Pipelines können als Code definiert und in Git-Repositorys gespeichert werden. Dies ermöglicht die Versionskontrolle und die Überprüfung der Bereitstellungsprozesse.
Automatisierte Bereitstellungen	Sie können Spinnaker so konfigurieren, dass Bereitstellungen auf der Grundlage von Änderungen in Git-Repositorys automatis ch gestartet werden. Das Tool unterstützt kontinuierliche Bereitstellungspraktiken, die für von zentraler Bedeutung sind. GitOps
Unveränderliche Infrastruktur	Spinnaker fördert die Verwendung einer unveränderlichen Infrastruktur, was ein Schlüsselkonzept von ist. GitOps Es fördert

Flächen	Funktionen des Tools
	die Bereitstellung neuer Instanzen, anstatt bestehende Instanzen zu modifizieren.
Rollbacks und Versionierung	Spinnaker bietet robuste Rollback-Funktione n und ermöglicht eine schnelle Rückkehr zu früheren, als funktionierend bekannten Status. Es unterstützt die Versionierung von Bereitste llungen im Einklang mit den Prinzipien der Rückverfolgbarkeit. GitOps
Workflows für Genehmigungen	Spinnaker integriert manuelle Beurteilu ngsphasen in Pipelines, um kontrollierte Werbeaktionen zwischen verschiedenen Umgebungen zu unterstützen. Dies unterstüt zt GitOps Praktiken der Trennung zwischen Bereitstellungen und Releases.
Canary und Bereitstellungen blue/green	Spinnaker unterstützt fortschrittliche Einsatzst rategien, die sich an den GitOps Praktiken für sichere und kontrollierte Freisetzungen orientier en.
Integration mit Versionskontrollsystemen	Spinnaker kann in verschiedene Git-Anbie ter integriert werden, um Pipelines auf der Grundlage von Repository-Ereignissen zu starten.
Kubernetes-Integration	Spinnaker bietet native Unterstützung für Kubernetes und unterstützt die Verwaltung von Kubernetes-Ressourcen. GitOps-style
Verwaltung von Artefakten	Spinnaker unterstützt Artefaktmanagement und Versionierung, die für die Aufrechte rhaltung eines Workflows von entscheidender Bedeutung sind. GitOps

Flächen	Funktionen des Tools
Beobachtbarkeit und Überwachung	Spinnaker bietet die Integration mit Überwachungstools, um den Aspekt der Beobachtbarkeit von zu unterstützen. GitOps
Prüfpfad	Spinnaker bietet detaillierte Bereitstellungsprotokolle und -verläufe, die das Prinzip der Überprüfbarkeit von unterstützen. GitOps
Role-based Zugriffskontrolle (RBAC)	Dieses Tool implementiert RBAC für eine genaue Kontrolle darüber, wer welche Aktionen ausführen kann, und zwar in Übereinstimmung mit den Sicherheitspraktiken. GitOps
Templating und Parametrisierung	Spinnaker unterstützt Templates in Pipeline-Definitionen, um wiederverwendbare und parametrisierte Bereitstellungen zu ermöglichen.
Förderung der Umwelt	Spinnaker ermöglicht die kontrollierte Förderung von Anwendungen zwischen Umgebungen (z. B. von der Bereitstellung bis zur Produktion).
Integration mit CI-Tools	Spinnaker kann in verschiedene CI-Tools (Continuous Integration) integriert werden, um eine vollständige CI/CD Pipeline bereitzustellen, die den Prinzipien entspricht. GitOps
Benutzerdefinierte Stufen und Erweiterungen	Dieses Tool unterstützt benutzerdefinierte Phasen und Erweiterungen, sodass Teams GitOps Workflows implementieren können, die auf ihre Bedürfnisse zugeschnitten sind.
Zentralisierte Verwaltung	Spinnaker bietet eine zentralisierte Plattform für die Verwaltung von Bereitstellungen in mehreren Umgebungen und Cloud-Anbietern.

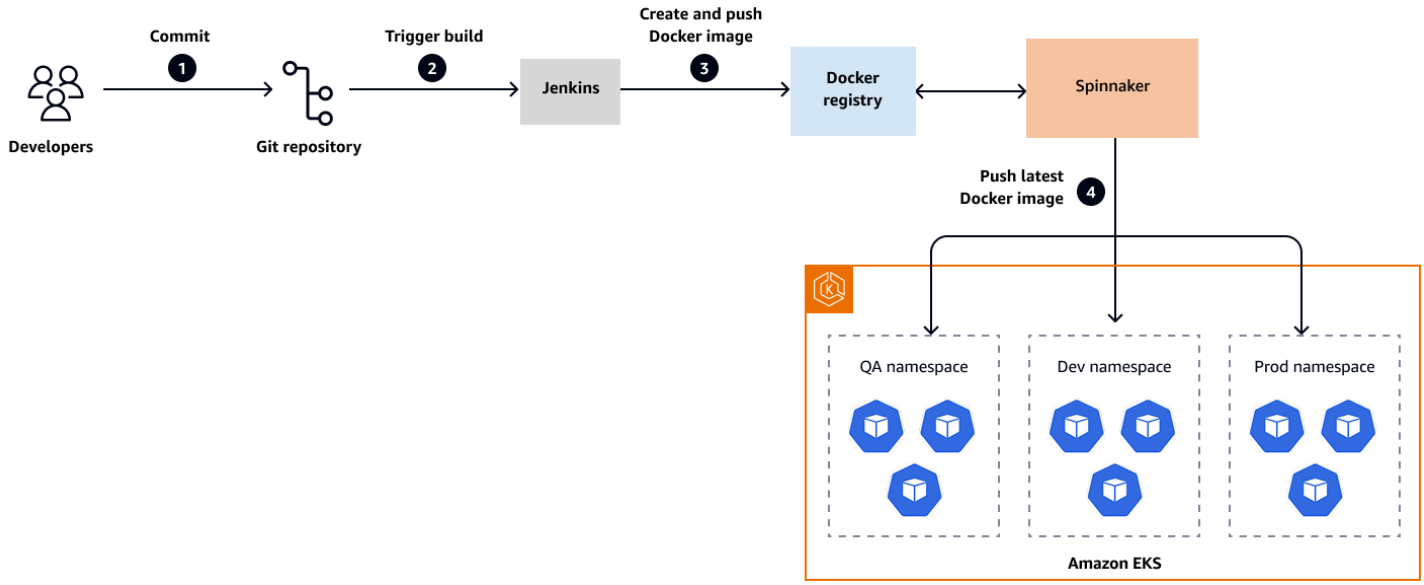
Obwohl Spinnaker nicht in erster Linie als GitOps Tool vermarktet wird, ist es aufgrund seiner Flexibilität und seines robusten Funktionsumfangs in der Lage, GitOps Workflows zu implementieren, insbesondere in komplexen Multi-Cloud-Umgebungen. Der Hauptunterschied zwischen Spinnaker und speziellen GitOps Tools wie Argo CD oder Flux besteht darin, dass Spinnaker eine umfassendere Continuous-Delivery-Plattform mit fortschrittlichen Bereitstellungsstrategien und Multi-Cloud-Unterstützung anbietet.

Die Stärke von Spinnaker liegt in seiner Fähigkeit, komplexe Bereitstellungsszenarien bei verschiedenen Cloud-Anbietern zu bewältigen, und in der Unterstützung fortschrittlicher Bereitstellungsstrategien. Wenn Spinnaker richtig konfiguriert ist, kann es Prinzipien effektiv umsetzen. GitOps Dies macht es zu einem leistungsstarken Tool für Unternehmen, die GitOps Praktiken in unterschiedlichen und komplexen Umgebungen anwenden möchten.

Weitere Informationen finden Sie in der Spinnaker-Dokumentation.

Architektur

Das folgende Diagramm zeigt einen GitOps-driven CD-Workflow, der Spinnaker und Jenkins X verwendet. Ausführliche Informationen finden Sie in der Spinnaker-Dokumentation.



Wobei:

- Schritt 1: Code-Commit. Entwickler übertragen Änderungen am Anwendungscode in ein Git-Repository. Diese Änderungen könnten Aktualisierungen der Anwendung selbst, Dockerfiles oder Kubernetes-Manifeste beinhalten.

- Schritt 2: Jenkins-Build und Image-Erstellung. Jenkins wird automatisch vom Git-Repository über einen Webhook oder eine Umfrage ausgelöst. Jenkins erstellt die Anwendung, erstellt ein Docker-Image und überträgt das erstellte Image in eine konfigurierte Docker-Registry (wie Amazon ECR oder Docker Hub).
- Schritt 3: Spinnaker-Image-Überwachung und Pipeline-Trigger. Spinnaker überwacht die Docker-Registry kontinuierlich auf neue Images. Wenn eine neue Image-Version erkannt wird, löst Spinnaker automatisch eine Pipeline aus, um den Bereitstellungsprozess zu starten.
- Schritt 4: Bereitstellung in Ziel-Namespaces. Spinnaker stellt das neue Docker-Image auf Amazon EKS bereit. Basierend auf den Pipeline-Konfigurationen wird das Image in Ziel-Namespaces im Cluster bereitgestellt. Spinnaker stellt sicher, dass die neueste Anwendungsversion bereitgestellt wird und dabei definierte Bereitstellungsstrategien wie Canary-Deployments eingehalten werden. blue/green

Rancher-Flotte

Rancher Fleet ist eine GitOps-at-scale Lösung, die speziell für die Verwaltung mehrerer Kubernetes-Cluster entwickelt wurde. Sie hält sich strikt an die GitOps Prinzipien und konzentriert sich gleichzeitig auf Skalierbarkeit und Multi-Cluster-Management.

GitOps Unterstützung

Flächen	Funktionen des Tools
Git als zentrale Informationsquelle	Fleet verwendet Git-Repositorys als maßgeblic he Quelle für die Definition des gewünschten Status von Anwendungen und Ressourcen in mehreren Clustern. Alle Konfigurationen, einschließlich Kubernetes-Manifeste, Helm-Charts und benutzerdefinierte Ressourcen, werden in Git gespeichert.
Deklarative Konfiguration	Fleet arbeitet mit deklarativen Beschreibungen des gewünschten Zustands für Anwendung en und Ressourcen. Dabei kann es sich um rohe Kubernetes-YAML, Helm-Diagramme,

Flächen	Funktionen des Tools
	Kustomize-Dateien oder benutzerdefinierte Ressourcen handeln. Fleet-specific
Automatisierte Synchronisation	Fleet überwacht die Git-Repositorys kontinuie rlich auf Änderungen. Es wendet automatisch Änderungen auf die Zielcluster an, wenn es Unterschiede zwischen dem Git-Status und dem Cluster-Status erkennt.
Multi-cluster Verwaltung	Fleet wurde speziell für die Verwaltung von Bereitstellungen in mehreren Kubernetes-Clustern entwickelt. Es kann Tausende von Clustern von einer einzigen Steuerungsebene aus verwalten.
Kubernetes-native Architektur	Fleet besteht aus einer Reihe von benutzerd efinierten Kubernetes-Ressourcen und Controllern. Es verwendet die Erweiteru ngsmechanismen in Kubernetes für den Betrieb. GitOps
Kontinuierlicher Abgleich	Fleet vergleicht ständig den aktuellen Status von Clustern mit dem gewünschten Status, der in Git definiert ist. Es korrigiert automatisch jede Abweichung, die zwischen diesen Zuständen festgestellt wird.
Gruppierung und Targeting von Clustern	Mit Fleet können Sie Cluster gruppieren und Bereitstellungen auf bestimmte Gruppen oder einzelne Cluster ausrichten. Es unterstützt die konsistente Anwendungsbereitstellung in verschiedenen Umgebungen und Clustertypen.

Flächen	Funktionen des Tools
Mehrschichtige Konfigurationen	Fleet unterstützt mehrschichtige Konfigurationen, die Basiskonfigurationen mit umgebungsspezifischen Overlays bieten. Dies entspricht den GitOps Praktiken zur effizienten Verwaltung mehrerer Umgebungen.
Helmintegration	Fleet bietet native Unterstützung für Helm-Diagramme und ermöglicht die einfache Verwaltung komplexer Anwendungen. Es kann Helm-Releases über GitOps Workflows versionieren und verwalten.
Benutzerdefinierte Ressourcendefinitionen (CRDs)	Fleet verwendet benutzerdefinierte Ressourcen wie GitRepo Bundle, um Bereitstellungen zu definieren. Diese CRDs bieten eine Kubernetes-native Möglichkeit, Workflows zu definieren GitOps .
Sicherheit und RBAC	Fleet lässt sich zur Zugriffskontrolle in Kubernetes RBAC integrieren. Es unterstützt die sichere Verwaltung sensibler Informationen und Anmeldeinformationen.
Beobachtbarkeit	Fleet bietet Statusinformationen über den Synchronisierungsstatus von Clustern und Anwendungen. Es bietet Einblicke in die GitOps Prozesse in der gesamten Clusterflotte.
Skalierbarkeit	Fleet ist so konzipiert, dass es Tausende von Clustern effizient verwalten kann. Es unterstützt umfangreiche GitOps Operationen in Unternehmensumgebungen.

Flächen	Funktionen des Tools
Abhängigkeitsmanagement	Sie können Abhängigkeiten zwischen verschied enen Ressourcen und Anwendungen definiere n. Fleet stellt sicher, dass bei komplexen Bereitstellungen die richtige Reihenfolge der Abläufe eingehalten wird.
Anpassung und Erweiterbarkeit	Fleet unterstützt benutzerdefinierte Skripts und Lifecycle-Hooks für eine erweiterte Anpassung von Bereitstellungen. Es ermöglicht die Integrati on in bestehende Tools und Workflows.
Offline- und Air-Gap-Support	Fleet kann in Umgebungen mit eingeschränkter oder keiner Internetverbindung eingesetzt werden. Es unterstützt GitOps Arbeitsabläufe in hochsicheren oder regulierten Umgebungen.
Schrittweise Rollouts	Fleet unterstützt stufenweise Rollouts in mehreren Clustern, was kontrollierte und schrittweise Implementierungsstrategien ermöglicht.
Einheitliche Verwaltungsoberfläche	Fleet bietet eine einzige Oberfläche für die Verwaltung von GitOps Workflows in allen Clustern. Es vereinfacht den Betrieb in komplexen Umgebungen mit mehreren Clustern.
Integration mit anderen Rancher-Tools	Fleet lässt sich in andere Rancher-Tools integrieren, um eine umfassende Kubernetes-Managementlösung bereitzustellen.
Prüfpfad und Einhaltung von Vorschriften	Fleet führt einen klaren Prüfpfad für alle Änderungen und Implementierungen. Es hilft Ihnen, Compliance-Anforderungen durch versionskontrollierte Abläufe zu erfüllen. Git-based

Rancher Fleet implementiert diese GitOps Prinzipien mit einem starken Fokus auf Skalierbarkeit und Multi-Cluster-Management. Das Design eignet sich besonders für Unternehmen, die eine große Anzahl von Kubernetes-Clustern in verschiedenen Umgebungen, Rechenzentren oder Cloud-Anbietern verwalten.

Das wichtigste Unterscheidungsmerkmal von Fleet ist die Fähigkeit, das System skalierbar zu handhaben GitOps . Diese Funktion macht es besonders wertvoll für große Unternehmen oder Anbieter von Managed Services, die zahlreiche Cluster verwalten. Tools wie Argo CD oder Flux werden häufig für die Verwaltung einzelner Cluster verwendet, wohingegen Fleet für die Verwaltung GitOps einer großen Clusterflotte konzipiert ist.

Durch die Einhaltung dieser GitOps Prinzipien bietet Rancher Fleet eine Lösung für Unternehmen, die ein konsistentes, skalierbares und automatisiertes Management von Anwendungen und Ressourcen in einer vielfältigen und großen Kubernetes-Umgebung implementieren möchten.

Weitere Informationen finden Sie in der Fleet-Dokumentation.

Architektur

Informationen zur Architektur und zum Arbeitsablauf finden Sie im Fleet-Repository.

Codefresh

Codefresh ist eine moderne CI/CD Plattform, die GitOps Prinzipien unterstützt, insbesondere für Kubernetes-Implementierungen. Codefresh bietet einen umfassenden CI/CD Funktionsumfang, und seine Fähigkeiten sind bemerkenswert. GitOps

GitOps Unterstützung

Flächen	Funktionen des Tools
Git als zentrale Informationsquelle	Codefresh verwendet Git-Repositorys als maßgebliche Quelle für Anwendungscode, Infrastrukturdefinitionen und Pipeline-Konfigurationen. Alle Änderungen am System werden über Git vorgenommen, wodurch eine vollständige Historie und ein Audit-Trail gewährleistet sind.

Flächen	Funktionen des Tools
Deklarative Konfiguration	Codefresh unterstützt deklarative Pipeline-Definitionen mithilfe von YAML-Dateien, die in Git gespeichert sind. Kubernetes-Manifeste, Helm-Diagramme, CloudFormation Vorlagen und andere IaC-Dateien können in denselben Repositorys versionsgesteuert werden.
GitOps Dashboard	Codefresh bietet ein spezielles GitOps Dashboard zur Visualisierung und Verwaltung GitOps von Workflows. Es bietet einen klaren Überblick über den Synchronisationsstatus zwischen Git- und Cluster-Status.
Automatisierte Synchronisation	Codefresh überwacht die Git-Repositorys kontinuierlich auf Änderungen. Es startet automatisch Pipelines, um Änderungen an den Zielumgebungen vorzunehmen, wenn es Unterschiede feststellt.
Kubernetes-Integration	Codefresh bietet eine umfassende Integrati on mit Kubernetes, um Bereitstellungen in mehreren Clustern zu unterstützen GitOps-st yle . Es unterstützt verschiedene Kubernetes-Ressourcen und benutzerdefinierte Ressource ndefinitionen (CRDs).
Verwaltung der Umgebung	Sie können mehrere Umgebungen (z. B. Entwicklung, Staging und Produktion) als Code definieren und verwalten. Codefresh unterstüt zt die Werbung zwischen Umgebungen mithilfe GitOps von Methoden.

Flächen	Funktionen des Tools
Integration mit Argo CD	Codefresh lässt sich für erweiterte Funktione n in Argo CD integrieren. GitOps Es kombinier t seine CI-Funktionen mit den CD-Stärken von Argo CD, um eine Komplettlösung bereitzus tellen. GitOps
Helm-Unterstützung	Codefresh unterstützt Helm-Charts und ermöglicht so eine einfache Verwaltung komplexer Anwendungen. GitOps Es bietet auch Versionierung und Werbung für Helm-Charts.
Progressive Lieferung	Codefresh unterstützt fortschrittliche Bereitste llungsstrategien wie Canary und blue/green Deployments. Sie können diese Strategien mithilfe GitOps von Workflows implementieren und verwalten.
Rollbacks und Versionierung	Codefresh ermöglicht einfache Rollbacks zu früheren Versionen, wenn nach der Bereitste llung Probleme festgestellt werden. Es behält die Versionierung der Bereitstellung zur Rückverfolgbarkeit bei.
Workflows für Genehmigungen	Codefresh unterstützt manuelle und automatis ierte Genehmigungsprozesse für Bereitste llungen. Es ermöglicht kontrollierte Werbeakti onen zwischen verschiedenen Umgebunge n, die den geltenden Praktiken entsprechen GitOps.
IaC	Codefresh unterstützt die Integration mit IaC-Tools wie Terraform CloudFormation . Es ermöglicht die Versionskontrolle von Infrastrukturdefinitionen zusammen mit dem Anwendungscode.

Flächen	Funktionen des Tools
Beobachtbarkeit und Überwachung	Codefresh bietet integrierte Überwachungs- und Beobachtbarkeitsfunktionen. Es bietet auch Integrationen mit externen Überwachungstools für eine verbesserte Sichtbarkeit.
Sicherheitsscannen	Codefresh bietet Funktionen für Sicherhei tsscans, die in GitOps Arbeitsabläufe integrier t werden können. Sicherheitsüberprüfungen sind Teil des automatisierten Bereitstellungspro zesses.
Audit-Trails	Codefresh führt umfassende Auditprotokolle für alle Aktionen und Änderungen. Es unterstüt zt die Rückverfolgbarkeits- und Compliance-Aspekte von. GitOps
RBAC und Zugriffskontrolle	Codefresh implementiert eine rollenbasierte Zugriffskontrolle (RBAC) für eine differenz ierte Rechteverwaltung. Dies trägt dazu bei, sichere Abläufe in Teams und Umgebungen zu gewährleisten. GitOps
GitOps -Automatisierung	Codefresh bietet Funktionen zur Automatis ierung verschiedener Aspekte von GitOps Workflows, einschließlich der Erstellung und Zusammenführung von Pull-Requests (PR).
Multi-cloud und hybride Bereitstellungen	Codefresh unterstützt GitOps Workflows über mehrere Cloud-Anbieter und lokale Umgebungen hinweg.
Templating und Parametrisierung	Codefresh unterstützt Vorlagen in Pipeline- und Bereitstellungskonfigurationen. Dies ermöglich t wiederverwendbare und parametrisierte Workflows GitOps.

Flächen	Funktionen des Tools
Integriertes Bildmanagement	Codefresh bietet integrierte Funktionen zur Verwaltung von Container-Images. Es integrier t Image-Builds und Bereitstellungen in GitOps Workflows.
GitOps für die Verwaltung von Geheimnissen	Codefresh bietet sichere Möglichkeiten zur Verwaltung von Geheimnissen innerhalb von GitOps Workflows. Es lässt sich in externe Lösungen zur Verwaltung von Geheimnissen integrieren.
Funktionen für die Zusammenarbeit	Codefresh bietet Funktionen für die Teamzusammenarbeit innerhalb von GitOps Prozessen. Zu diesen Funktionen gehören Kommentare, Benachrichtigungen und gemeinsame Dashboards.

Der Codefresh-Ansatz GitOps zeichnet sich durch die Integration von CI/CD Funktionen und Praktiken aus. GitOps Es zielt darauf ab, eine umfassende Plattform bereitzustellen, die den gesamten Lebenszyklus der Softwarebereitstellung abdeckt und gleichzeitig die Prinzipien GitOps einhält.

Das Hauptunterscheidungsmerkmal von Codefresh in diesem GitOps Bereich ist sein einheitlicher Plattformansatz, der CI-Funktionen mit CD und Funktionen kombiniert. GitOps Dies macht es besonders für Teams geeignet, die eine Komplettlösung suchen, die komplexe CI/CD Szenarien bewältigen und gleichzeitig Praktiken implementieren kann. GitOps

Codefresh bietet eine Plattform für Unternehmen, die GitOps Methoden in einem breiteren CI/CD Kontext anwenden möchten, insbesondere bei der Arbeit mit Kubernetes und Cloud-nativen Technologien.

Weitere Informationen finden Sie in der Codefresh-Dokumentation.

Pulumi

Pulumi ist eine IaC-Plattform, die nicht ausschließlich dafür konzipiert ist. GitOps Sie kann jedoch effektiv zur Implementierung von GitOps Prinzipien eingesetzt werden, insbesondere für Cloud-Infrastruktur- und Kubernetes-Implementierungen.

GitOps Unterstützung

Flächen	Funktionen des Tools
IaC	Mit Pulumi können Sie Ihre Infrastruktur mithilfe von Allzweck-Programmiersprachen wie Python und Go definieren. TypeScript Dieser codebasierte Ansatz entspricht der GitOps Betonung versionierter, deklarativer Konfigura tionen.
Git als zentrale Informationsquelle	Infrastrukturcode in Pulumi kann in Git-Repos itorys gespeichert und versionsgesteuert werden. Dadurch wird sichergestellt, dass Git als zentrale Informationsquelle für Infrastru kturdefinitionen dient.
Deklarativer gewünschter Zustand	Pulumi verwendet zwar Programmiersprache n, beschreibt aber dennoch den gewünschten Zustand der Infrastruktur deklarativ. Der Code definiert, wie die Infrastruktur aussehen soll, und nicht, wie sie Schritt für Schritt erstellt wird.
Automatisierte Synchronisation	Pulumi kann in CI/CD Pipelines integrier t werden, um Änderungen automatisch anzuwenden, wenn Code in Git aktualisi ert wird. Dies ermöglicht die kontinuierliche Implementierung von Infrastrukturänderungen, was ein GitOps Schlüsselprinzip ist.
Multi-cloud und Kubernetes-Unterstützung	Pulumi unterstützt eine Vielzahl von Cloud-Anbietern und Kubernetes, sodass Sie den

Flächen	Funktionen des Tools
	Praktiken in verschiedenen Umgebungen folgen GitOps können. Das Tool ermöglicht eine konsistente Verwaltung von Ressourcen auf verschiedenen Plattformen.
Statusverwaltung	Pulumi verwaltet den Zustand der Infrastru ktur, die remote und sicher gespeichert werden kann. Dieses Zustandsmanagement ist für die GitOps Praxis von entscheidender Bedeutung und gewährleistet die Konsistenz zwischen dem definierten Zustand und dem tatsächlichen Zustand Ihrer Infrastruktur.
Erkennung und Abgleich von Abweichungen	Pulumi kann Unterschiede zwischen dem gewünschten Zustand (im Code) und dem tatsächlichen Zustand der Infrastruktur erkennen. Sie gleicht diese Unterschiede im Einklang mit dem GitOps Grundsatz der kontinuierlichen Aussöhnung aus.
Politik als Code	Mit Pulumi CrossGuard können Sie Richtlini en als Code definieren und durchsetzen. Dies ermöglicht eine versionskontrollierte GitOps-st yle Verwaltung von Compliance- und Sicherhei tsrichtlinien.
Verwaltung von Secrets	Pulumi bietet sichere Methoden zur Verwaltun g sensibler Informationen innerhalb des Infrastrukturcodes. Es unterstützt die Integrati on mit externen Systemen zur Verwaltung von Geheimnissen, was für GitOps Sicherhei tspraktiken von entscheidender Bedeutung ist.

Flächen	Funktionen des Tools
Modulare und wiederverwendbare Komponenten	Pulumi unterstützt die Erstellung wiederverwendbarer Komponenten und Module. Diese Modularität entspricht den GitOps Praktiken zur Verwaltung komplexer Bereitstellungen in mehreren Umgebungen.
Vorschau anzeigen und planen	Pulumi bietet die Möglichkeit, eine Vorschau von Änderungen anzuzeigen, bevor sie übernommen werden. Dies unterstützt den GitOps Grundsatz sicherer, vorhersehbarer Änderungen an der Infrastruktur.
Rollbacks und Geschichte	Pulumi führt eine Historie der Bereitstellungen und unterstützt Rollbacks zu früheren Status. Dies entspricht den GitOps Grundsätzen der Rückverfolgbarkeit und Reversibilität.
Kontinuierliche Bereitstellung für die Infrastruktur	Pulumi kann in CI/CD Pipelines integriert werden, um die kontinuierliche Umsetzung von Infrastrukturänderungen zu gewährleisten. Es unterstützt automatisiertes Testen und Validieren von Infrastrukturcode.
RBAC und Zugriffskontrolle	Pulumi bietet eine rollenbasierte Zugriffskontrolle für die Verwaltung, wer Änderungen an der Infrastruktur vornehmen kann. Dies unterstützt GitOps Sicherheits- und Governance-Praktiken.
Beobachtbarkeit und Protokollierung	Pulumi bietet Funktionen zur Protokollierung und Überwachung von Infrastrukturänderungen. Diese Funktionen unterstützen den Aspekt der Beobachtbarkeit von Praktiken. GitOps

Flächen	Funktionen des Tools
Integration mit anderen Tools	Pulumi kann in verschiedene Tools in der Cloud integriert werden. Diese Flexibilität ermöglicht umfassende GitOps Workflows.
Umweltmanagement	Pulumi unterstützt die Verwaltung mehrerer Umgebungen (Entwicklung, Staging, Produktio n), indem dieselbe Codebasis mit unterschi edlichen Konfigurationen verwendet wird. Dies entspricht den GitOps Praktiken für ein konsistentes Management mehrerer Umgebungen.
Abhängigkeitsmanagement	Pulumi kümmert sich um Abhängigk eiten zwischen Ressourcen und stellt die korrekte Reihenfolge der Abläufe sicher. Dies ist entscheidend für komplexe GitOps Bereitstellungen, die voneinander abhängige Komponenten beinhalten.
Anbieter benutzerdefinierter Ressourcen	Mit Pulumi können Sie benutzerdefinierte Anbieter erstellen, um jeden API-driven Dienst zu verwalten. Dadurch werden die GitOps Praktiken auf eine Vielzahl von Ressourcen ausgedehnt, die über die Standard-Cloud-Ang ebote hinausgehen.
Funktionen für die Zusammenarbeit	Pulumi unterstützt die Zusammenarbeit im Team durch gemeinsame Status- und Zugriffsk ontrollen. Dies erleichtert GitOps Arbeitsabläufe in Teamumgebungen.

Mithilfe dieser Pulumi-Funktionen können Unternehmen GitOps Verfahren für ihre Infrastruktur implementieren, insbesondere in Szenarien, in denen sie eine detaillierte Steuerung oder komplexe Logik benötigen oder eine Vielzahl von Cloud- und lokalen Ressourcen innerhalb eines einzigen, konsistenten Frameworks verwalten möchten.

Pulumis Ansatz GitOps ist einzigartig, da er die Leistungsfähigkeit und Flexibilität von Allzweck-Programmiersprachen für das Infrastrukturmanagement nutzt und gleichzeitig die Prinzipien einhält. GitOps Dies kann besonders für Teams von Vorteil sein, die es vorziehen, mit vertrauten Programmiersprachen zu arbeiten und bewährte Methoden der Softwareentwicklung auf das Infrastrukturmanagement anwenden möchten.

Das Hauptunterscheidungsmerkmal von Pulumi in GitOps ist die Verwendung von Standard-Programmiersprachen zur Definition der Infrastruktur. Herkömmliche GitOps Tools verwenden häufig YAML oder domänenspezifische Sprachen, wohingegen Pulumi eine komplexere Logik, eine bessere Wiederverwendung von Code und eine einfachere Integration in bestehende Entwicklungsworkflows ermöglicht.

Weitere Informationen finden Sie in der Pulumi-Dokumentation.

GitOps Vergleich der Tools

Hier finden Sie einen Vergleich der neun GitOps Tools, die in den vorherigen Abschnitten besprochen wurden. Berücksichtigen Sie bei der Auswahl eines Tools Ihre spezifischen Anforderungen, die vorhandene Infrastruktur, das Fachwissen des Teams und das gewünschte Maß an Kontrolle und Anpassung.

Benutzerfreundlichkeit

- Argo CD, Flux und Rancher Fleet sind im Allgemeinen einfacher einzurichten.
- Spinnaker und Jenkins X haben steilere Lernkurven.
- Weave erfordert GitOps möglicherweise mehr Einstellungen für erweiterte Funktionen.
- GitLab CI/CD und Codefresh bieten integrierte Erlebnisse.

Kubernetes-Integration

- Argo CD, Flux und Rancher Fleet sind sehr auf Kubernetes ausgerichtet.
- Jenkins X und Weave bieten breitere Funktionen. GitOps DevOps
- Die anderen Tools unterstützen Kubernetes, ohne sich ausschließlich darauf zu konzentrieren.

CI/CD-Funktionen

- Jenkins X, Lösungen. GitLab CI/CD, and Codefresh offer complete CI/CD
- Argo CD, Flux und Weave GitOps konzentrieren sich mehr auf den CD-Aspekt des Workflows und erfordern häufig die Integration mit separaten CI-Tools.

GitOps Reinheit

- Argo CD und Flux sind Tools, die sich speziell auf GitOps Folgendes konzentrieren:
- Die anderen Tools beinhalten GitOps Prinzipien in unterschiedlichem Maße.

Multi-Cloud-Unterstützung

- Spinnaker und Pulumi zeichnen sich in Multi-Cloud-Szenarien aus.
- Die anderen Tools können cloudübergreifend eingesetzt werden, erfordern jedoch möglicherweise eine zusätzliche Einrichtung.

Multi-Cluster-Support

- Alle Tools unterstützen Bereitstellungen mit mehreren Clustern.
- Argo CD und Weave GitOps verfügen über erweiterte Multi-Cluster-Managementfunktionen.

Integration

- Flux wird stark von der Cloud Native Computing Foundation (CNCF) unterstützt.
- Argo CD hat eine große und aktive Community.
- Argo CD und Flux verfügen über eine starke Kubernetes-Integration.
- Jenkins X verwendet das umfassendere Jenkins-System.
- Weave GitOps ist neuer, wächst aber mit starker kommerzieller Unterstützung.
- GitLab CI/CD lässt sich eng in integrieren. GitLab
- Rancher Fleet funktioniert gut innerhalb des Rancher-Systems.

Gemeinschaft und Unterstützung

- Flux hat eine starke CNCF-Unterstützung.
- Argo CD GitLab, und Spinnaker haben große Gemeinschaften.
- Kommerzieller Support ist für die meisten Tools verfügbar.

Funktionen für Unternehmen

- Weave GitOps und Jenkins X bieten standardmäßig mehr unternehmensorientierte Funktionen.
- Argo CD und Flux sind für Unternehmen erhältlich oder können für den Einsatz in Unternehmen erweitert werden.

Flexibilität und Erweiterbarkeit

- Flux ist hochmodular und erweiterbar.
- Argo CD bietet gute Anpassungsmöglichkeiten.
- Jenkins X ist sehr erweiterbar, erfordert jedoch möglicherweise mehr Aufwand.
- Weave GitOps zielt darauf ab, eine Komplettlösung mit weniger Erweiterbarkeit anzubieten.

Skalierbarkeit

- Spinnaker und GitLab CI/CD sind bekannt für ihre Skalierbarkeit auf Unternehmensebene.
- Argo CD und Flux eignen sich gut für groß angelegte Kubernetes-Implementierungen.

Verwaltung der Infrastruktur

- Pulumi konzentriert sich auf das Infrastrukturmanagement.
- Weave GitOps und Flux bieten gute IaC-Funktionen.

Programmiermodell und Sprachunterstützung

- In Pulumi können Sie die Infrastruktur mithilfe von Allzweck-Programmiersprachen wie Python, Go TypeScript, C# und Java definieren. Pulumis Verwendung von Standardsprachen ermöglicht die Integration von Infrastrukturcode in vertraute Entwicklungsabläufe, Testpraktiken und komplexe Logik.
- Terraform verwendet die HashiCorp Konfigurationssprache (HCL).
- CloudFormation verwendet JSON- und YAML-Vorlagen.
- Argo CD, Flux, Rancher Fleet, Weave GitOps, Spinnaker und GitLab CI/CD verwalten hauptsächlich YAML- oder deklarative Konfigurationsdateien.
- Jenkins X verwaltet YAML- und skriptbasierte Pipelines, bietet aber keine systemeigene Allzweckprogrammierung für IaC.

Anwendungsfälle für Argo CD und Flux

Dieser Abschnitt konzentriert sich auf zwei Tools, Argo CD und Flux, die reine GitOps Funktionalität bieten. In diesem Zusammenhang GitOps bezieht sich Pure auf ein Modell, bei dem ein Git-Repository als zentrale Informationsquelle für den gewünschten Status von Anwendungen und Infrastruktur dient. Alle Änderungen werden über Git-Commits vorgenommen, und das System synchronisiert die Live-Umgebung automatisch so, dass sie dem im Repository definierten Status entspricht. Außerhalb der Git-Operationen sind keine manuellen Eingriffe erforderlich.

Allgemeine Überlegungen

- Möglicherweise ziehen Sie es vor, Argo CD in Umgebungen zu verwenden, in denen visuelles Management und anwendungsorientierte Workflows wichtig sind.
- Sie könnten sich für Flux entscheiden, wenn Sie leichtere Lösungen, eine starke Mehrmandantenfähigkeit oder eine tiefe Integration in das breitere Netzwerk der Cloud Native Computing Foundations (CNCF) benötigen.
- Argo CD spricht aufgrund seiner intuitiven Benutzeroberfläche häufig Teams an, die von der traditionellen CI/CD Version zur Umstellung übergehen. GitOps
- Flux wird häufig in Cloud-nativen Umgebungen bevorzugt, in denen CLI-based Workflows und IaC-Praktiken bereits etabliert sind.

Letztlich hängt die Wahl zwischen Argo CD und Flux oft von Ihren spezifischen organisatorischen Anforderungen, den vorhandenen Tools und den Präferenzen Ihres Teams ab. Beide Tools sind in der Lage, die meisten GitOps Szenarien zu bewältigen. Wir empfehlen Ihnen daher, sie auf der Grundlage Ihrer spezifischen Anwendungsfälle und Anforderungen zu bewerten.

Anwendungsfälle für Argo CD

Visuelles Management:

- Wenn Sie eine benutzerfreundliche Benutzeroberfläche benötigen, um Bereitstellungen zu verwalten und Anwendungsstatus zu visualisieren.
- Für Teams, die eine grafische Oberfläche für die Überwachung und Fehlerbehebung bevorzugen.

Application-centric Ansatz:

- Wenn Sie Bereitstellungen auf Anwendungsebene verwalten möchten, anstatt einzelne Ressourcen zu verwalten.
- Für Organisationen, die ihre Bereitstellungen nach Anwendungskonzepten strukturieren.

Multi-cluster Verwaltung:

- Wenn die Verwaltung von Bereitstellungen über mehrere Cluster hinweg eine Hauptanforderung ist.
- Für komplexe, verteilte Umgebungen mit vielen Clustern.

Rollback- und Synchronisierungswellen:

- Wenn Sie eine genaue Kontrolle über den Bereitstellungsprozess benötigen, einschließlich Synchronisierungswellen und manueller Eingriffe.
- Für Szenarien, die komplexe Rollback-Strategien erfordern.

Integration mit vorhandenen Tools:

- Wenn Sie im Argo-Projekt bereits andere Tools wie Argo Workflows und Argo Events verwenden.

Unternehmensumgebungen:

- Für große Unternehmen, die standardmäßig eine robuste RBAC- und Single Sign-On-Integration benötigen.

Flux-Anwendungsfälle

Einfache Bereitstellungen:

- Wenn Sie eine leichtere, weniger GitOps ressourcenintensive Lösung benötigen.
- Für Edge-Computing- oder IoT-Szenarien, in denen Ressourcen eingeschränkt sein könnten.

Automatisierte Image-Updates:

- Wenn die automatische Erkennung und Bereitstellung neuer Container-Images eine zentrale Anforderung ist.

- Für Teams, die sich auf die kontinuierliche Bereitstellung mit häufigen Image-Updates konzentrieren.

Multi-tenancy:

- Wenn eine starke Unterstützung für mehrere Mandanten erforderlich ist, insbesondere in gemeinsam genutzten Cluster-Umgebungen.
- Für Dienstanbieter oder große Organisationen, in denen Teams oder Projekte strikt voneinander getrennt sind.

IaC:

- Wenn es wichtig ist, sowohl Anwendungen als auch Infrastruktur über denselben GitOps Workflow zu verwalten.
- Für Teams, die stark in das IaC-Paradigma investiert haben.

Helmintegration:

- Wenn die umfangreiche Verwendung von Helm-Diagrammen Teil Ihrer Bereitstellungsstrategie ist.
- Für Umgebungen mit komplexen Helm-based Bereitstellungen.

CNCF-Projektintegration:

- Wenn eine enge Integration mit anderen CNCF-Projekten wichtig ist.
- Für Organisationen, die sich an den Technologien und Prinzipien von CNCF orientieren.

Modulare Architektur:

- Wenn Sie Flexibilität benötigen, um nur bestimmte Komponenten des GitOps Toolkits zu verwenden.
- Für Teams, die mithilfe modularer Komponenten benutzerdefinierte GitOps Workflows erstellen möchten.

Progressive Bereitstellung:

- Wenn fortschrittliche Bereitstellungsstrategien wie Canary-Releases oder A/B Tests Kernanforderungen sind.

Funktionsvergleich

Flächen	Argo CD	Flux
Support der GitOps Kernprinz ipien	✅Ja	✅Ja
Architektur	End-to-end Anwendung zur Implementierung von Kubernetes-Workflows GitOps	Bietet Kubernetes-CRDs und Controller für GitOps
Einrichtung	Einfach	Komplex
Unterstützung für Helme	✅Ja	✅Ja
Passen Sie den Support an	✅Ja	✅Ja
Integrierte GUI	CLI und Web-UI mit vollem Funktionsumfang	CLI und optionales, leichtes Webinterface
RBAC-Unterstützung	Granulare Steuerung	Natives RBAC von Kubernetes
Multi-tenancy und Multi-Clu ster-Unterstützung	Hervorragende Unterstützung für Multi-Cluster	Hervorragende Unterstützung für Mehrmandantenfähigkeit
Authentifizierung mit einmalige r Anmeldung	✅Ja	✅Ja
Automatisierung synchroni sieren	Fähigkeit, Fenster zu synchronisieren	Möglichkeit, Abstimmun gsintervalle festzulegen
Teilweise Synchronisierung	✅Ja	❌Nein
Versöhnungsprozess	Unterstützt manuelle und automatische Synchroni sierungen. Es stehen	Unterstützt manuelle und automatische Synchroni sierungen.

Flächen	Argo CD	Flux
	verschiedene Strategien zur Verfügung.	
Erweiterbarkeit	Unterstützt benutzerdefinierte Plugins. Eingeschränkte Anpassungsmöglichkeiten.	Unterstützt benutzerdefinierte Controller. Gute Erweiterbarkeit und Integrationen von Drittanbietern.
Community-Unterstützung	Große und aktive Community.	Kleinere, aber wachsende Gemeinschaft.
Skalierbarkeit	Gute Skalierbarkeit, aber begrenzt durch die Datenabrufrate der Weboberfläche. Community-Analysen deuten darauf hin, dass Zehntausende von Anwendungen unterstützt werden.	Klare Anleitungen für die horizontale und vertikale Skalierbarkeit von bis zu Zehntausenden von Anwendungen.

Bewährte Methoden für die Auswahl eines GitOps Tools

Dieser Abschnitt enthält Überlegungen, Tipps und bewährte Methoden zur Auswahl eines GitOps Tools für Ihren EKS-Cluster. Die richtige Wahl hängt von Ihrem spezifischen Kontext, Ihren Anforderungen und Ihrer langfristigen Strategie ab. Oft ist es von Vorteil, einen Machbarkeitsnachweis mit Ihren wichtigsten Entscheidungen durchzuführen, bevor Sie eine endgültige Entscheidung treffen.

Beurteilen Sie die Bedürfnisse und Fähigkeiten Ihres Unternehmens:

- Berücksichtigen Sie die aktuellen Fähigkeiten und die Bereitschaft Ihres Teams, neue Tools zu erlernen.
- Bewerten Sie die Komplexität Ihrer Amazon EKS-Umgebung. (Verwenden Sie beispielsweise einen einzelnen Cluster oder mehrere Cluster?)
- Ermitteln Sie Ihre spezifischen Anforderungen an Compliance, Sicherheit und Skalierbarkeit.

> **Bewährte Methode**
>
> Erstellen Sie ein detailliertes Anforderungsdokument, in dem die erforderlichen Funktionen und nützliche, aber nicht erforderliche Funktionen beschrieben werden.

Bewerten Sie den Reifegrad und die Akzeptanz des Tools:

- Untersuchen Sie den Reifegrad potenzieller GitOps Tools und deren Akzeptanz in der Branche.
- Suchen Sie nach Tools, die sich in Amazon EKS-Umgebungen bewährt haben.

> **Bewährte Methode**
>
> Priorisieren Sie Tools, die weit verbreitet sind und eine starke Präsenz im Netzwerk der Cloud Native Computing Foundation (CNCF) haben.

Erwägen Sie die Integration mit Ihrer bestehenden Toolchain:

- Beurteilen Sie, wie gut sich das GitOps Tool in Ihre aktuelle CI/CD Pipeline, Überwachungslösungen und andere betriebliche Tools integrieren lässt.

- Suchen Sie nach systemeigenen Integrationen AWS-Services wie IAM, Amazon ECR und. CloudWatch

> **Bewährte Methode**
>
> Erstellen Sie einen Machbarkeitsnachweis, um die Integrationsfunktionen zu testen, bevor Sie eine endgültige Entscheidung treffen.

Bewerten Sie die Sicherheitsfunktionen:

- Priorisieren Sie Tools, die über robuste Funktionen für die rollenbasierte Zugriffskontrolle (RBAC) verfügen und sich gut in IAM integrieren lassen.
- Suchen Sie nach Funktionen, die die sichere Verwaltung von Geheimnissen und die Durchsetzung von Richtlinien unterstützen.

> **Bewährte Methode**
>
> Wählen Sie ein Tool, das fundierte GitOps Sicherheitspraktiken unterstützt, darunter Policy-as-Code und automatisierte Konformitätsprüfungen.

Beurteilen Sie Skalierbarkeit und Leistung:

- Überlegen Sie, wie das Tool bei einer großen Anzahl von Anwendungen und Clustern abschneidet.
- Bewerten Sie die Auswirkungen auf die Clusterleistung und den Ressourcenverbrauch.

> **Bewährte Methode**
>
> Führen Sie Leistungstests mit Workloads durch, die Ihrer Produktionsumgebung ähneln, um sicherzustellen, dass das Tool Ihrer Größenordnung gewachsen ist.

Erwägen Sie die Unterstützung mehrerer Cluster und mehrerer Umgebungen:

- Wenn Sie über mehrere EKS-Cluster verfügen oder dies planen, sollten Sie Tools priorisieren, die über leistungsstarke Verwaltungsfunktionen für mehrere Cluster verfügen.

- Suchen Sie nach Funktionen, die konsistente Bereitstellungen in verschiedenen Umgebungen (z. B. Entwicklung, Staging und Produktion) unterstützen.

> **Bewährte Methode**
>
> Wählen Sie ein Tool, das die zentrale Verwaltung mehrerer Cluster unter Beibehaltung der umgebungsspezifischen Konfigurationen ermöglicht.

Evaluieren Sie die Beobachtbarkeits- und Überwachungsfunktionen:

- Suchen Sie nach Tools, die einen klaren Überblick über den Status Ihrer Bereitstellungen und den Zustand Ihres Clusters bieten.
- Überlegen Sie, wie gut sich das Tool in Ihre bestehenden Überwachungs- und Protokollierungslösungen integrieren lässt.

> **Bewährte Methode**
>
> Priorisieren Sie Tools, die anpassbare Dashboards und Warnmechanismen für die proaktive Problemerkennung bieten.

Beurteilen Sie die Lernkurve und die Dokumentation:

- Beurteilen Sie die Qualität und Vollständigkeit der Dokumentation des Tools.
- Berücksichtigen Sie die Verfügbarkeit von Schulungsressourcen und die Unterstützung durch die Community.

> **Bewährte Methode**
>
> Wählen Sie ein Tool mit gut gepflegter Dokumentation, aktiven Community-Foren und offiziellen Schulungsprogrammen oder Zertifizierungen.

Berücksichtigen Sie die Kosten und die Ressourcennutzung:

- Bewerten Sie sowohl die direkten Kosten (wie Lizenzierung und Support) als auch die indirekten Kosten (wie Betriebskosten und Schulungskosten), die mit der Einführung des Tools verbunden sind.
- Beurteilen Sie die Effizienz des Tools in Bezug auf den Verbrauch von Rechen- und Speicherressourcen.

> **Bewährte Methode**
>
> Führen Sie eine Analyse der Gesamtbetriebskosten (TCO) durch, die sowohl kurzfristige als auch langfristige Kosten umfasst.

Evaluieren Sie Flexibilität und Anpassungsoptionen:

- Suchen Sie nach Tools, mit denen Sie Workflows an Ihre spezifischen Bedürfnisse anpassen können.
- Berücksichtigen Sie die Erweiterbarkeit des Tools durch Plugins oder APIs.

> **Bewährte Methode**
>
> Wählen Sie ein Tool, das Standardfunktionen mit der Möglichkeit zur Anpassung an Ihre individuellen Anforderungen in Einklang bringt.

Beurteilen Sie die Funktionen für kontinuierliche Bereitstellung und progressive Bereitstellung:

- Suchen Sie nach Tools, die fortschrittliche Bereitstellungsstrategien wie Canary-Versionen und blue/green Bereitstellungen unterstützen.
- Prüfen Sie, wie einfach die Implementierung und Verwaltung dieser Strategien ist.

> **Bewährte Methode**
>
> Priorisieren Sie Tools, die integrierte Unterstützung für progressive Bereitstellungsmuster bieten, um das Risiko bei Ihren Implementierungen zu minimieren.

Ziehen Sie Herstellerbindung und Portabilität in Betracht:

- Beurteilen Sie die Abhängigkeiten des Tools von bestimmten Cloud-Anbietern oder -Technologien.
- Denken Sie darüber nach, wie einfach es ist, in future bei Bedarf zu einem anderen Tool zu migrieren.

> **Bewährte Methode**
>
> Bevorzugen Sie Tools, die offene Standards verwenden und Exportfunktionen für Ihre GitOps Konfigurationen bieten.

Evaluieren Sie den Community-Support und die Erweiterungen:

- Schauen Sie sich die Größe und Aktivität der Benutzergemeinschaft an.
- Beurteilen Sie die Verfügbarkeit von Integrationen und Plugins von Drittanbietern.

> **Bewährte Methode**
>
> Treten Sie Community-Foren oder Benutzergruppen bei, um Erfahrungen aus erster Hand von anderen Benutzern zu erhalten, bevor Sie eine Entscheidung treffen.

Beachten Sie die Compliance- und Prüfanforderungen:

- Prüfen Sie, wie gut das Tool Ihre Compliance-Anforderungen unterstützt, einschließlich Prüfprotokollen und Berichten.
- Suchen Sie nach Funktionen, die Ihnen helfen, die Einhaltung der Vorschriften aufrechtzuerhalten und nachzuweisen.

> **Bewährte Methode**
>
> Wählen Sie ein Tool, das umfassende Auditprotokolle bereitstellt und die Erstellung von Compliance-Berichten unterstützt.

Beurteilen Sie die Rollback- und Disaster-Recovery-Funktionen:

- Beurteilen Sie die Benutzerfreundlichkeit und Zuverlässigkeit von Rollback-Mechanismen.
- Überlegen Sie, wie das Tool Notfallwiederherstellungsszenarien unterstützt.

Bewährte Methode

Testen Sie die Rollback- und Wiederherstellungsprozesse im Rahmen Ihrer Bewertung gründlich.

Häufig gestellte Fragen

F: Was sind die beliebtesten GitOps Tools für Amazon EKS?

A: Zu den beliebtesten GitOps Tools für Amazon EKS gehören Argo CD, Flux, Jenkins X und CI/CD. GitLab Jedes Tool hat Stärken, aber Argo CD und Flux sind besonders bekannt für ihren Kubernetes-nativen Ansatz und die starke Unterstützung durch die Community.

F: Wie verbessert sich das EKS-Clustermanagement? GitOps

A: GitOps Verbessert die EKS-Clusterverwaltung durch Versionskontrolle für die Infrastruktur, automatisierte Bereitstellungen, verbesserte Sicherheit durch deklarative Konfigurationen, einfachere Rollbacks und bessere Überprüfbarkeit. Es verbessert auch die Zusammenarbeit und reduziert menschliche Fehler bei Bereitstellungen.

F: Auf welche Hauptfunktionen sollte ich bei einem GitOps Tool für Amazon EKS achten?

A: Zu den wichtigsten Funktionen, auf die Sie achten sollten, gehören: nahtlose Amazon EKS-Integration, robustes RBAC, Multi-Cluster-Unterstützung, Observability-Funktionen, Unterstützung für progressive Bereitstellungsstrategien, Skalierbarkeit und Integration mit AWS-Services IAM und Amazon ECR.

F: Wie stelle ich die Sicherheit bei der Implementierung GitOps in Amazon EKS sicher?

A: Um die Sicherheit zu gewährleisten, sollten Sie ein Tool wählen, das über eine starke RBAC-Integration mit IAM, eine sichere Verwaltung von Geheimnissen, Unterstützung für verschlüsselte Git-Repositorys und die Möglichkeit verfügt, Sicherheitsrichtlinien als Code zu implementieren. Stellen Sie außerdem sicher, dass das Tool umfassende Auditprotokolle bereitstellt.

F: Können GitOps Tools Amazon EKS-Umgebungen mit mehreren Clustern verarbeiten?

A: Ja, GitOps Tools wie Argo CD und Flux verfügen über robuste Managementfunktionen für mehrere Cluster. Sie ermöglichen es Ihnen, mehrere EKS-Cluster von einer einzigen Steuerungsebene aus zu verwalten, wodurch die Konsistenz zwischen den Umgebungen gewährleistet wird.

F: Wie lassen sich GitOps Tools in bestehende CI/CD-Pipelines integrieren?

A: GitOps Tools lassen sich in der Regel in bestehende CI/CD-Pipelines integrieren, indem sie als Bereitstellungsphase der Pipeline fungieren. Sie können durch CI-Tools ausgelöst

werden, wenn Änderungen in das Git-Repository übertragen werden, und sie automatisieren den Bereitstellungsprozess für EKS-Cluster.

F: Was sind die Herausforderungen bei der Implementierung GitOps in Amazon EKS?

A: Zu den häufigsten Herausforderungen gehören die sichere Verwaltung von Geheimnissen, die Sicherstellung angemessener Zugriffskontrollen, der Umgang mit zustandsbehafteten Anwendungen, der Umgang mit Abweichungen zwischen Git und Cluster-Status und die Anpassung der Team-Workflows an das GitOps Modell.

F: Wie gehen GitOps Tools mit Rollbacks in Amazon EKS um?

A: GitOps Tools behandeln Rollbacks in der Regel, indem sie zu einem früheren Commit im Git-Repository zurückkehren. Dadurch wird automatisch ein Deployment des vorherigen als funktionierend bekannten Status ausgelöst, was zu schnellen und zuverlässigen Rollbacks führt.

F: Können GitOps Tools Amazon EKS-Add-Ons und andere AWS Ressourcen verwalten?

A: Viele GitOps Tools können Amazon EKS-Add-Ons und einige AWS Ressourcen verwalten, insbesondere wenn sie mit IaC-Tools wie Terraform oder kombiniert werden. CloudFormation Der Umfang dieser Funktion kann jedoch variieren. Spezifische Informationen zu den einzelnen GitOpsTools finden Sie im Abschnitt Tools.

F: Wie unterstützen GitOps Tools die Compliance-Anforderungen in Amazon EKS?

A: GitOps Tools unterstützen die Einhaltung der Vorschriften, indem sie einen klaren Prüfpfad für alle Änderungen bereitstellen, Genehmigungsprozesse durchsetzen, Richtlinien als Code für automatisierte Konformitätsprüfungen implementieren und detaillierte Protokollierungs- und Berichtsfunktionen bieten.

F: Was ist die Lernkurve für die Implementierung GitOps in Amazon EKS?

A: Die Lernkurve kann je nach Tool und vorhandenem Wissen Ihres Teams variieren. Im Allgemeinen passen sich Teams, die mit Git, Kubernetes und Amazon EKS vertraut sind, schneller an als andere. Die meisten gängigen Tools bieten umfangreiche Dokumentations- und Schulungsressourcen, um die Einführung zu erleichtern.

F: Wie handhaben GitOps Tools die Verwaltung von Geheimnissen in Amazon EKS?

A: GitOps Tools lassen sich in der Regel in externe Secrets-Management-Lösungen wie AWS Secrets Manager HashiCorp Vault integrieren. Einige Tools bieten auch eine integrierte Verschlüsselung für Geheimnisse, die in Git-Repositorys gespeichert sind.

F: Können GitOps Tools sowohl mit statusfreien als auch mit statusbehafteten Anwendungen in Amazon EKS funktionieren?

A: Ja, GitOps Tools können sowohl mit statusfreien als auch mit statusbehafteten Anwendungen verwendet werden. Die Verwaltung von statusbehafteten Anwendungen erfordert jedoch häufig zusätzliche Überlegungen, wie z. B. den Umgang mit persistenten Volumes und die Sicherstellung der Datenkonsistenz bei Aktualisierungen.

F: Wie unterstützen GitOps Tools Canary oder blue/green Bereitstellungen in Amazon EKS?

A: Viele GitOps Tools bieten integrierten Support für erweiterte Bereitstellungsstrategien. Sie können die schrittweise Einführung neuer Versionen verwalten, nach Problemen suchen und bei Erkennung von Problemen automatisch ein Rollback durchführen. Alle diese Operationen sind als Code im Git-Repository definiert.

F: Was ist der Unterschied zwischen der Verwendung eines GitOps Tools und der Verwendung **`kubectl apply`** mit einer CI/CD Pipeline?

A: GitOps Tools bieten Vorteile gegenüber einfachen `kubectl apply` Befehlen, darunter automatische Erkennung und Abstimmung von Abweichungen, erhöhte Sicherheit durch Pull-basierte Bereitstellungen, bessere Überprüfbarkeit und ausgefeiltere Bereitstellungsstrategien. Sie bieten auch einen umfassenderen Ansatz zur Verwaltung des gesamten Clusterstatus.

Ressourcen

Die folgenden Ressourcen bieten offizielle Dokumentation, praktische Leitfäden, Fallstudien und eingehende Analysen, die Ihnen helfen können, bei der Auswahl eines GitOps Tools für Ihren EKS-Cluster eine fundierte Entscheidung zu treffen. Sie behandeln verschiedene Aspekte von GitOps, darunter Implementierungsstrategien, bewährte Verfahren, Vergleiche zwischen verschiedenen Tools und praktische Erfahrungen.

AWS Ressourcen:

- Amazon EKS-Dokumentation
- Automatisieren von Amazon EKS mit GitOps (AWS Blogbeitrag)
- Einführung in EKS mit Weaveworks (Workshop) GitOps AWS
- Flux-Labor (Amazon EKS-Workshop)
- Argo CD-Labor (Amazon EKS-Werkstatt)

GitOps und Dokumentation zum Tool:

- GitOps Bewährte Methoden für kontinuierliche Bereitstellung und fortschrittliche Sicherheit (On-Demand-Webinar DevOps auf Abruf)
- Kubernetes-Dokumentation
- Argo-CD-Dokumentation
- Flux-Dokumentation
- Dokumentation weben GitOps
- Jenkins X-Dokumentation
- GitLab CI/CD Dokumentation
- Spinnaker-Dokumentation
- Dokumentation zur Rancher Fleet
- Codefresh-Dokumentation

Dokumentverlauf

In der folgenden Tabelle werden wichtige Änderungen in diesem Leitfaden beschrieben. Um Benachrichtigungen über zukünftige Aktualisierungen zu erhalten, können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Erste Veröffentlichung	—	30. April 2025

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.