Memilih GitOps alat yang tepat untuk cluster Amazon EKS Anda

AWS Panduan Preskriptif

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

AWS Panduan Preskriptif: Memilih GitOps alat yang tepat untuk cluster Amazon EKS Anda

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Pengantar 1
Hasil bisnis yang ditargetkan 2
Integrasi mulus dengan Amazon EKS 2
Skalabilitas dan kinerja 2
Keamanan dan kepatuhan 2
Kemudahan penggunaan dan kurva belajar 3
Dukungan komunitas dan jaringan 3
Kemampuan manajemen multi-cluster 3
Observabilitas dan pemantauan 4
Fleksibilitas dan kustomisasi 4
Pengiriman berkelanjutan dan dukungan penyebaran progresif 4
Efektivitas biaya dan pemanfaatan sumber daya 5
GitOps alat untuk kluster EKS 6
Argo CD 6
GitOps dukungan 6
Arsitektur 9
Fluks 10
GitOps dukungan 10
Arsitektur 13
Menenun GitOps 13
GitOps dukungan 14
Arsitektur 17
Jenkins X 17
GitOps dukungan 18
Arsitektur 20
GitLab CI/CD 22
GitOps dukungan 22
Pemintal 25
GitOps dukungan 25
Arsitektur 29
Armada Peternak 30
GitOps dukungan 30
Arsitektur 33
Codefresh 33

GitOps dukungan 34
Pulumi 37
GitOps dukungan 38
GitOps perbandingan alat 42
Kemudahan penggunaan 42
Integrasi Kubernetes 42
Kemampuan CI/CD 42
GitOps kemurnian 42
Dukungan multi-cloud 43
Dukungan multi-cluster 43
Integrasi 43
Komunitas dan dukungan 43
Fitur perusahaan 43
Fleksibilitas dan ekstensibilitas 44
Skalabilitas 44
Manajemen infrastruktur 44
Model pemrograman dan dukungan bahasa 44
Kasus penggunaan Argo CD dan Flux 45
Pertimbangan umum 45
Kasus penggunaan Argo CD 45
Kasus penggunaan fluks 46
Perbandingan fitur 48
Praktik terbaik untuk memilih GitOps alat 50
Pertanyaan yang Sering Diajukan 55
Sumber daya 58
Riwayat dokumen 59
..... lx

Memilih GitOps alat yang tepat untuk cluster Amazon EKS Anda

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

April 2025 (riwayat dokumen)

Dalam lanskap teknologi cloud-native yang berkembang pesat, GitOps telah muncul sebagai metodologi yang kuat untuk mengelola dan menerapkan aplikasi dan infrastruktur. Jika Anda menggunakan Amazon Elastic Kubernetes Service (Amazon EKS) GitOps , prinsip penerapan dapat secara signifikan meningkatkan proses penerapan, meningkatkan keandalan, dan merampingkan operasi. Berbagai GitOps alat tersedia, dan memilih yang tepat untuk klaster EKS Anda adalah keputusan penting yang dapat memengaruhi efisiensi tim Anda dan keberhasilan keseluruhan DevOps praktik Anda.

Pemilihan GitOps alat yang sesuai untuk lingkungan Amazon EKS Anda melibatkan pertimbangan yang cermat terhadap berbagai faktor, termasuk persyaratan spesifik Anda, keahlian tim, kebutuhan skalabilitas, dan kemampuan integrasi dengan yang ada Layanan AWS. Setiap alat dilengkapi dengan serangkaian fitur, kekuatan, dan batasan potensinya sendiri, jadi penting untuk menyelaraskan pilihan Anda dengan tujuan dan konteks operasional organisasi Anda.

Panduan ini mengeksplorasi pertimbangan utama dalam memilih GitOps alat untuk Amazon EKS, membandingkan opsi yang sering digunakan, dan memberikan wawasan untuk membantu Anda membuat keputusan yang tepat. Ini mencakup sembilan GitOps alat populer:

- Argo CD
- Fluks
- Menenun GitOps
- Jenkins X
- GitLab CI/CD
- Pemintal
- Armada Peternak
- Codefresh
- Pulumi

Hasil bisnis yang ditargetkan

Daftar berikut membahas potensi tujuan dan hasil ketika Anda memilih alat untuk menerapkan GitOps prinsip-prinsip dalam proses pengembangan dan operasi Anda.

Integrasi mulus dengan Amazon EKS

GitOps Alat Anda harus terintegrasi dengan lancar dengan Amazon EKS dan menyediakan kompatibilitas dengan fitur dan pengoptimalan khusus Amazon EKS.

- Dukungan Amazon EKS asli: Cari alat yang menawarkan dukungan bawaan untuk Amazon EKS, termasuk koneksi dan manajemen cluster yang mudah.
- Layanan AWS integrasi: Pastikan alat dapat berinteraksi dengan yang lain Layanan AWS seperti AWS Identity and Access Management (IAM), Amazon Elastic Container Registry (Amazon ECR)Registry (Amazon ECR), dan Amazon. CloudWatch
- Kompatibilitas add-on Amazon EKS: Konfirmasikan bahwa alat ini mendukung add-on Amazon EKS dan dapat mengelolanya secara efektif.

Skalabilitas dan kinerja

GitOps Alat Anda harus dapat menangani skala operasi Amazon EKS Anda, dari cluster kecil hingga lingkungan multi-cluster yang besar.

- Efisiensi sumber daya: Mengevaluasi konsumsi sumber daya alat dan dampaknya terhadap kinerja cluster.
- Operasi skala besar: Menilai kemampuan alat untuk mengelola berbagai aplikasi dan cluster secara bersamaan.
- Kinerja di bawah beban: Pertimbangkan kinerja alat selama pembaruan frekuensi tinggi dan penerapan skala besar.

Keamanan dan kepatuhan

Fitur keamanan dan kemampuan kepatuhan sangat penting, terutama di industri yang diatur atau saat Anda menangani data sensitif.

- Kontrol akses: Cari fitur kontrol akses berbasis peran (RBAC) yang kuat yang terintegrasi dengan IAM.

- Manajemen rahasia: Mengevaluasi bagaimana alat menangani informasi sensitif dan terintegrasi dengan AWS Secrets Manageratau solusi lainnya.
- Jejak audit: Pastikan alat ini menyediakan kemampuan pencatatan dan audit yang komprehensif untuk kepatuhan dan pemecahan masalah.
- Pemindaian keamanan: Pertimbangkan alat yang menawarkan pemindaian keamanan bawaan untuk kerentanan dalam penerapan.

Kemudahan penggunaan dan kurva belajar

Alat ini harus ramah pengguna dan selaras dengan keterampilan tim Anda untuk memastikan adopsi cepat dan penggunaan yang efisien.

- Antarmuka pengguna: Menilai intuisi fitur antarmuka baris perintah (CLI) dan antarmuka pengguna grafis (GUI).
- Kualitas dokumentasi: Cari komprehensif, up-to-date dokumentasi, dan tutorial.
- Sumber belajar: Pertimbangkan ketersediaan materi pelatihan, kursus, dan sumber daya masyarakat.

Dukungan komunitas dan jaringan

Komunitas dan jaringan yang kuat dapat menyediakan sumber daya, plugin, dan keberlanjutan jangka panjang yang berharga.

- Pengembangan aktif: Periksa frekuensi pembaruan dan daya tanggap pengelola.
- Ukuran komunitas: Pertimbangkan ukuran dan aktivitas komunitas pengguna untuk dukungan dan berbagi pengetahuan.
- Integrasi pihak ketiga: Evaluasi ketersediaan plugin dan integrasi dengan alat lain di tumpukan Anda.

Kemampuan manajemen multi-cluster

Jika Anda memiliki beberapa kluster EKS, kemampuan untuk mengelolanya secara efisien sangat penting.

- Manajemen terpusat: Cari fitur yang memungkinkan pengelolaan beberapa cluster dari satu bidang kontrol.

- Federasi klaster: Pertimbangkan alat yang mendukung federasi Kubernetes untuk aplikasi multi-cluster.
- Paritas lingkungan: Menilai seberapa baik alat mempertahankan konsistensi di berbagai lingkungan seperti pengembangan, pementasan, dan produksi.

Observabilitas dan pemantauan

Alat ini harus memberikan wawasan yang jelas tentang status penerapan dan kesehatan klaster Anda.

- Visibilitas penerapan: Cari fitur yang menawarkan tampilan status dan riwayat penerapan yang jelas.
- Integrasi dengan alat pemantauan: Pertimbangkan seberapa baik alat ini terintegrasi dengan solusi pemantauan populer seperti Prometheus dan Grafana.
- Kemampuan peringatan: Menilai kemampuan alat untuk mengatur dan mengelola peringatan untuk masalah penyebaran atau penyimpangan.

Fleksibilitas dan kustomisasi

Kemampuan untuk menyesuaikan alat dengan alur kerja dan persyaratan spesifik Anda penting untuk kepuasan jangka panjang.

- Ekstensibilitas: Cari arsitektur plugin atau APIs yang memungkinkan Anda memperluas fungsionalitas alat.
- Dukungan sumber daya khusus: Konfirmasikan bahwa alat ini dapat menangani sumber daya Kubernetes kustom secara efektif.
- Kustomisasi alur kerja: Menilai seberapa mudah Anda dapat menyesuaikan GitOps alur kerja dengan kebutuhan tim Anda.

Pengiriman berkelanjutan dan dukungan penyebaran progresif

Strategi penerapan lanjutan seringkali penting untuk meminimalkan risiko dan memastikan pembaruan yang lancar.

- Penerapan Canary: Cari dukungan bawaan untuk rilis kenari.
- Blue/green deployments: Assess the tool's capabilities for blue/greenstrategi penyebaran.

- Mekanisme rollback: Pastikan fitur yang kuat dan easy-to-use rollback untuk pemulihan cepat dari penerapan yang gagal.

Efektivitas biaya dan pemanfaatan sumber daya

Pertimbangkan biaya keseluruhan untuk mengadopsi dan memelihara alat, termasuk biaya langsung dan tidak langsung.

- Biaya lisensi: Bandingkan opsi open source dengan solusi komersial, dan pertimbangkan dukungan dan fitur perusahaan.
- Overhead operasional: Menilai biaya operasional tambahan dalam hal manajemen dan pemeliharaan.
- Konsumsi sumber daya: Mengevaluasi efisiensi alat dalam hal sumber daya komputasi dan penyimpanan yang akan diperlukan.

Dengan mempertimbangkan hasil ini dan aspek-aspeknya dengan cermat, Anda dapat membuat keputusan berdasarkan informasi tentang GitOps alat yang paling cocok untuk kluster EKS Anda, dan memastikan bahwa alat tersebut selaras dengan kebutuhan, kemampuan, dan strategi jangka panjang organisasi Anda.

GitOps alat untuk kluster EKS

Ada beberapa GitOps alat untuk Kubernetes yang saat ini tersedia di pasar. Berikut daftar beberapa opsi yang paling banyak digunakan:

- Argo CD
- Fluks
- Menenun GitOps
- Jenkins X
- GitLab CI/CD
- Pemintal
- Armada Peternak
- Codefresh
- Pulumi

Ikuti tautan untuk melihat informasi terperinci tentang cara alat ini menerapkan GitOps praktik. Setiap alat memiliki kekuatan dan kasus penggunaan. Pilihannya tergantung pada faktor-faktor seperti kebutuhan spesifik Anda, infrastruktur yang ada, keahlian tim, dan fitur yang diinginkan. Penting untuk mengevaluasi alat-alat ini berdasarkan kebutuhan organisasi Anda dan kompleksitas lingkungan Kubernetes Anda.

Argo CD

Argo CD adalah alat GitOps continuous delivery (CD) yang banyak digunakan untuk Kubernetes yang sesuai dengan beberapa prinsip utama. GitOps

GitOps dukungan

Bidang	Kemampuan alat
Konfigurasi deklaratif	Argo CD menggunakan konfigurasi deklaratif yang disimpan dalam repositori Git. Keadaan aplikasi dan infrastruktur yang diinginkan didefinisikan dalam file YAMAL. Konfigurasi ini

Bidang	Kemampuan alat
	menjelaskan apa yang harus diterapkan, bukan bagaimana menerapkannya.
Sistem kontrol versi sebagai satu-satunya sumber kebenaran	Repositori Git berfungsi sebagai sumber kebenaran tunggal untuk seluruh sistem. Semua perubahan pada aplikasi dan infrastru ktur dilakukan melalui Git. Ini memastikan jejak audit yang lengkap dan kemampuan untuk memutar kembali ke keadaan sebelumnya.
Sinkronisasi otomatis	Argo CD terus memantau repositori Git untuk perubahan. Ketika perubahan terdeteksi, secara otomatis menyinkronkan keadaan sebenarnya dari cluster dengan status yang diinginkan yang didefinisikan dalam Git. Ini memastikan bahwa cluster selalu mencermin kan status yang dijelaskan dalam repositori.
Kubernetes-native	Argo CD dirancang khusus untuk lingkungan Kubernetes. Ini memanfaatkan sifat deklaratif dan sumber daya kustom di Kubernetes untuk mengelola aplikasi.
Self-healing dan deteksi drift	Argo CD secara teratur membandingkan status langsung cluster dengan status yang diinginkan di Git. Jika mendeteksi penyimpan gan (perbedaan antara status aktual dan yang diinginkan), itu dapat secara otomatis memperbaiki perbedaan ini.
Multi-cluster dan dukungan multi-tenancy	Argo CD dapat mengelola beberapa cluster Kubernetes dari satu instance. Ini mendukung multi-tenancy, sehingga tim yang berbeda dapat mengelola aplikasi mereka secara mandiri.

Bidang	Kemampuan alat
Definisi aplikasi	Aplikasi dalam Argo CD didefinisikan dengan menggunakan Application CRD (definisi sumber daya khusus). Ini memungkinkan Kubernetes-native cara untuk mendefinisikan apa yang harus digunakan dan bagaimana.
Pemisahan penyebaran dan pelepasan	Argo CD memisahkan penyebaran kode dari rilisnya ke pengguna. Ini dicapai melalui berbagai strategi penyebaran seperti blue/gree n atau penyebaran kenari.
Observabilitas dan auditabilitas	Argo CD menyediakan UI web dan CLI untuk mengamati keadaan aplikasi dan cluster. Semua tindakan dicatat untuk memberikan jejak audit yang jelas tentang perubahan dan penerapan.
Keamanan dan RBAC	Argo CD terintegrasi dengan Kubernetes role-based access control (RBAC). Ini mendukung integrasi masuk tunggal untuk otentikasi dan otorisasi.
Arsitektur pluggable	Argo CD mendukung berbagai sistem manajemen kontrol sumber, bagan Helm, Kustomize, dan format manifes Kubernetes lainnya. Fleksibilitas ini memungkinkannya untuk masuk ke dalam lingkungan dan alur kerja yang beragam.
Pengiriman berkelanjutan (CD)	Meskipun Argo CD berfokus pada pengiriman berkelanjutan, ia dapat diintegrasikan dengan alat integrasi berkelanjutan (CI) untuk membuat CI/CD pipeline yang lengkap.

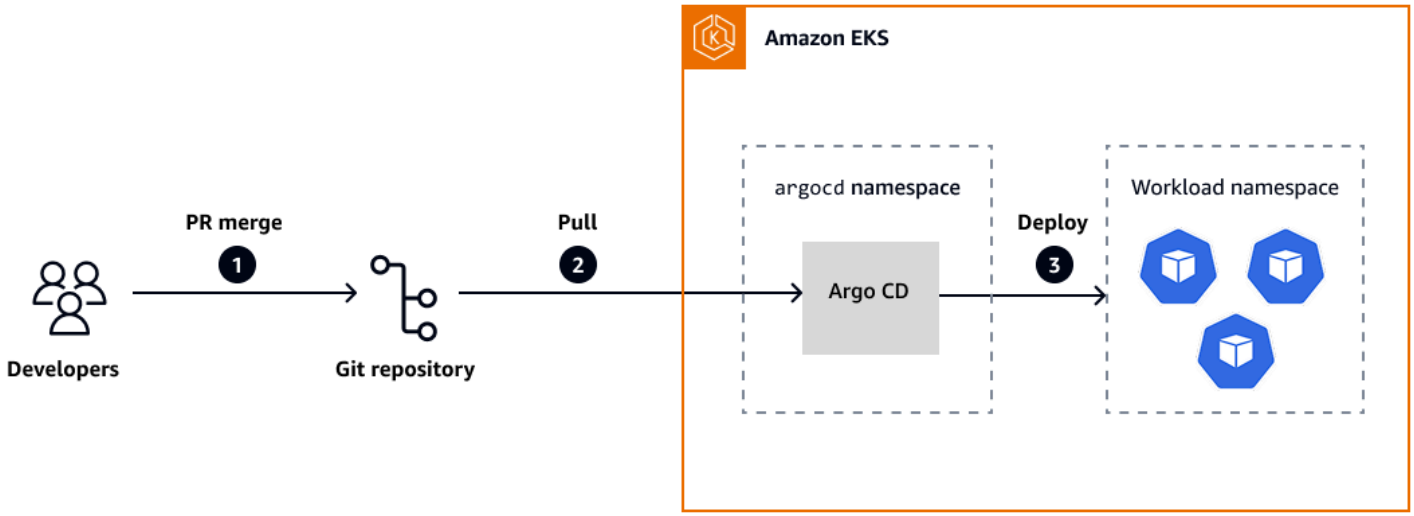
Dengan mengikuti GitOps prinsip-prinsip ini, Argo CD menyediakan cara yang kuat, terukur, dan aman untuk mengelola penerapan Kubernetes. Ini memastikan bahwa status operasional sistem Anda selalu sinkron dengan status yang diinginkan yang didefinisikan dalam repositori Git Anda, dan mempromosikan konsistensi, keandalan, dan kemudahan pengelolaan di lingkungan Kubernetes yang kompleks.

Untuk skenario dan persyaratan yang dapat ditangani oleh Argo CD, lihat Kasus penggunaan CD Argo nanti dalam panduan ini. Untuk perbandingan antara Argo CD dan Flux, lihat Perbandingan fitur nanti dalam panduan ini.

Untuk informasi tambahan, lihat dokumentasi Argo CD.

Arsitektur

Diagram berikut menggambarkan alur kerja GitOps-driven CD yang menggunakan Argo CD dalam cluster EKS. Untuk informasi lebih lanjut, lihat dokumentasi Argo CD.



di mana:

- Langkah 1: Tarik permintaan (PR) bergabung. Pengembang melakukan perubahan pada manifes Kubernetes atau bagan Helm yang disimpan dalam repositori Git. Ketika PR telah ditinjau dan digabungkan ke cabang utama, status aplikasi yang diinginkan diperbarui dalam kontrol sumber.
- Langkah 2: Sinkronisasi repositori. Argo CD berjalan dalam namespace khusus (`argocd`) di cluster EKS dan terus memantau repositori Git yang dikonfigurasi. Ketika mendeteksi perubahan, ia menarik pembaruan terbaru untuk merekonsiliasi status yang dideklarasikan.
- Langkah 3: Deployment untuk menargetkan namespace. Argo CD membandingkan status yang diinginkan dari Git dengan status langsung di cluster. Ini kemudian menerapkan perubahan yang

diperlukan pada namespace beban kerja target sehingga aplikasi diterapkan atau diperbarui sesuai. Ini termasuk mengelola sumber daya Kubernetes seperti penerapan, layanan ConfigMaps, dan rahasia untuk menjaga konsistensi klaster dengan sumber kebenaran Git.

Fluks

Flux adalah alat lain untuk Kubernetes yang mengimplementasikan GitOps prinsip dengan cara yang unik.

GitOps dukungan

Bidang	Kemampuan alat
Git sebagai satu-satunya sumber kebenaran	Flux menggunakan repositori Git sebagai sumber definitif untuk mendefinisikan keadaan sistem yang diinginkan. Semua konfigurasi untuk aplikasi dan infrastruktur disimpan dalam Git.
Konfigurasi deklaratif	Flux bekerja dengan deskripsi deklaratif dari keadaan cluster yang diinginkan. Deskripsi ini biasanya berupa manifes Kubernetes, bagan Helm, atau overlay Kustomize.
Aynchronisasi otomatis	Flux terus memantau repositori Git untuk perubahan. Ketika mendeteksi perubahan, secara otomatis menerapkannya ke cluster.
Kubernetes-native	Flux dibangun sebagai satu set pengendal i Kubernetes dan sumber daya kustom. Ia menggunakan mekanisme ekstensi di Kubernetes untuk menyediakan kemampuan. GitOps
Pull-based model penyebaran	Tidak seperti CI/CD sistem berbasis push tradisional, Flux menggunakan model berbasis tarik. Cluster menarik status yang diinginkan

Bidang	Kemampuan alat
	dari Git alih-alih menggunakan sistem eksternal untuk mendorong perubahan.
Rekonsiliasi berkelanjutan	Flux secara konstan membandingkan keadaan sebenarnya dari cluster dengan status yang diinginkan di Git. Ini secara otomatis mengoreksi penyimpangan yang terdeteksi di antara status ini.
Multi-tenancy	Flux mendukung multi-tenancy melalui konsep Kustomisasi dan. HelmReleases Tim yang berbeda dapat mengelola bagian konfigurasi mereka sendiri secara independen.
Pengiriman progresif	Flux mendukung strategi penerapan lanjutan, seperti rilis dan A/B pengujian kenari, melalui komponen Flagger-nya.
Integrasi helm	Flux menyertakan dukungan asli untuk Helm, sehingga Anda dapat dengan mudah mengelola rilis Helm. GitOps
Otomatisasi pembaruan gambar	Flux dapat secara otomatis memperbarui gambar kontainer di Git ketika versi baru tersedia di registri kontainer.
Kustomisasi dukungan	Anda dapat menggunakan dukungan asli yang disediakan oleh Flux for Kustomize untuk menyesuaikan dan menambal manifes Kubernetes.
Keamanan dan RBAC	Flux terintegrasi dengan Kubernetes RBAC untuk kontrol akses. Ini mendukung manajemen rahasia melalui berbagai backend.

Bidang	Kemampuan alat
Observabilitas	Flux menyediakan informasi status dan metrik tentang rekonsiliasi dan operasi. Ini terintegrasi dengan alat pemantauan untuk meningkatkan observabilitas.
Event-driven arsitektur	Flux menggunakan pendekatan berbasis peristiwa untuk mengimplementasikan rekonsili asi dan pembaruan.
Ekstensibilitas	Alat ini dirancang agar dapat diperluas, sehingga Anda dapat menambahkan pengontro l dan sumber daya khusus.
Cross-cluster sinkronisasi	Flux mendukung pengelolaan beberapa cluster dari satu set repositori.
Manajemen dependensi	Ini memungkinkan untuk mendefinisikan dependensi antara berbagai bagian sistem Anda dan memastikan urutan operasi yang benar.
Penerima Webhook	Anda dapat mengonfigurasi Flux untuk menerima webhook dari penyedia Git atau sistem lain untuk memulai rekonsiliasi segera.

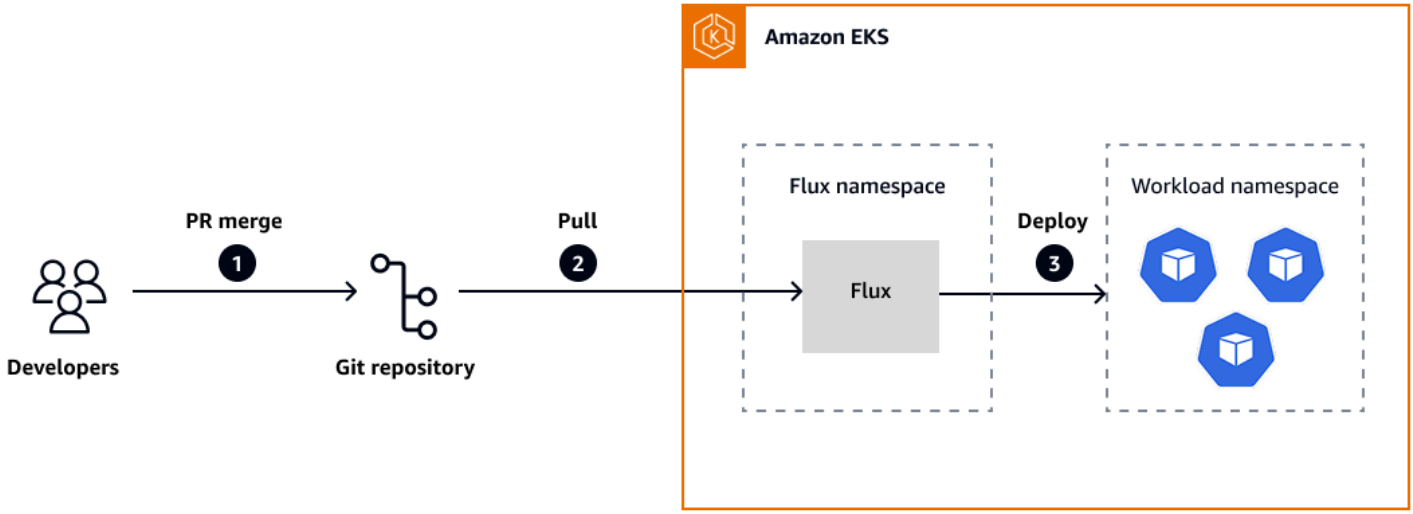
Dengan menerapkan GitOps prinsip-prinsip ini, Flux menyediakan sistem yang kuat dan fleksibel untuk mengelola klaster dan aplikasi Kubernetes. Ini memastikan bahwa infrastruktur dan aplikasi Anda selalu sinkron dengan repositori Git Anda, dan memberikan konsistensi, keandalan, dan kemudahan pengelolaan di lingkungan Kubernetes yang kompleks. Kubernetes-native Pendekatan alat dan fokus pada otomatisasi membuatnya sangat cocok untuk lingkungan cloud-native.

Untuk skenario dan persyaratan yang dapat ditangani oleh Flux, lihat Kasus penggunaan Flux nanti dalam panduan ini. Untuk perbandingan antara Argo CD dan Flux, lihat Perbandingan fitur nanti dalam panduan ini.

Untuk informasi tambahan, lihat dokumentasi Flux.

Arsitektur

Diagram berikut menggambarkan alur kerja GitOps-driven CD yang menggunakan Flux dalam cluster EKS. Untuk informasi lebih lanjut, lihat dokumentasi Flux.



di mana:

- Langkah 1: Tarik permintaan (PR) bergabung. Pengembang melakukan perubahan pada manifes Kubernetes atau bagan Helm yang disimpan dalam repositori Git. Ketika PR telah ditinjau dan digabungkan ke cabang utama, status aplikasi yang diinginkan diperbarui dalam kontrol sumber.
- Langkah 2: Sinkronisasi repositori. Flux berjalan dalam namespace khusus di cluster EKS dan terus memantau repositori Git yang dikonfigurasi. Ketika mendeteksi perubahan, ia menarik pembaruan terbaru untuk merekonsiliasi status yang dideklarasikan.
- Langkah 3: Deployment untuk menargetkan namespace. Flux membandingkan status yang diinginkan dari Git dengan status langsung di cluster. Ini kemudian menerapkan perubahan yang diperlukan pada namespace beban kerja target sehingga aplikasi diterapkan atau diperbarui sesuai.

Menenun GitOps

Weave GitOps dikembangkan oleh Weaveworks, yang merupakan perusahaan yang memperkenalkan istilah tersebut. GitOps Alat ini memberikan GitOps solusi komprehensif yang dibangun di atas GitOps prinsip-prinsip inti.

GitOps dukungan

Bidang	Kemampuan alat
Git sebagai satu-satunya sumber kebenaran	Weave GitOps menggunakan repositori Git sebagai sumber otoritatif untuk mendefini sikan keadaan sistem yang diinginkan. Semua konfigurasi, termasuk manifes aplikasi, definisi infrastruktur, dan kebijakan, disimpan dalam Git.
Konfigurasi deklaratif	Sistem ini bergantung pada deskripsi deklarati f dari seluruh status sistem. Deskripsi ini biasanya berupa manifes Kubernetes, bagan Helm, atau format deklaratif lainnya.
Sinkronisasi otomatis	Weave GitOps terus memantau repositor i Git untuk perubahan. Ketika mendeteksi perubahan, secara otomatis menerapkannya ke lingkungan target.
Kubernetes-native arsitektur	Weave GitOps dibangun sebagai satu set kontroler Kubernetes dan sumber daya kustom. Ia menggunakan mekanisme ekstensi di Kubernetes untuk menyediakan kemampuan. GitOps
Rekonsiliasi berkelanjutan	Alat ini terus-menerus membandingkan keadaan sebenarnya dari cluster dengan status yang diinginkan yang didefinisikan dalam Git. Ini secara otomatis mengoreksi penyimpangan yang terdeteksi di antara negara-negara ini.
Multi-cluster manajemen	Weave GitOps mendukung pengelolaan beberapa cluster Kubernetes dari satu bidang kontrol. Ini memungkinkan penerapan aplikasi yang konsisten di berbagai lingkungan.

Bidang	Kemampuan alat
Kebijakan sebagai kode	Weave GitOps menggabungkan konsep kebijakan sebagai kode untuk menegakkan aturan keamanan dan kepatuhan. Kebijakan dikendalikan versi bersama kode aplikasi dan definisi infrastruktur.
Pengiriman progresif	Alat ini mendukung strategi penerapan lanjutan seperti rilis dan blue/green penerapan kenari. Ini terintegrasi dengan Flagger untuk pengirima n otomatis dan progresif.
Observabilitas dan dasbor	Weave GitOps menyediakan dasbor bawaan untuk memantau keadaan aplikasi dan cluster. Ini menawarkan wawasan tentang proses rekonsiliasi dan kesehatan cluster.
Aman dengan desain	Alat ini menerapkan praktik terbaik keamanan, termasuk integrasi RBAC dan manajemen rahasia. Ini mendukung berbagai metode otentikasi dan terintegrasi dengan penyedia identitas perusahaan.
Ekstensibilitas dan integrasi	Alat ini dirancang untuk bekerja dengan berbagai alat cloud-native. Ini mendukung alat populer seperti Flux, Helm, dan Kustomize.
Self-service platform pengembang	Weave GitOps memungkinkan pembuatan platform swalayan untuk pengembang. Ini menyediakan template dan pagar pembatas untuk penerapan aplikasi.
GitOps otomatisasi	Alat ini mengotomatiskan banyak aspek GitOps alur kerja, termasuk pembuatan permintaan tarik untuk pembaruan.

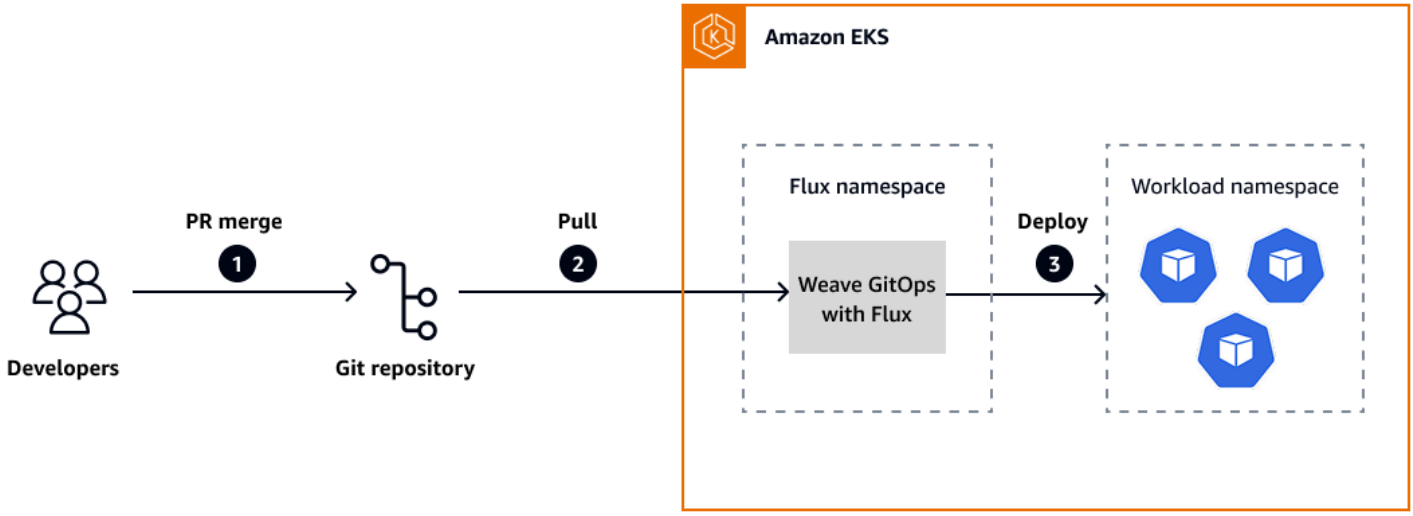
Bidang	Kemampuan alat
Pipa pengiriman berkelanjutan	Ini terintegrasi dengan CI/CD sistem untuk membuat jaringan pipa pengiriman ujung ke ujung.
Audit dan kepatuhan	Weave DevOps menyediakan jejak audit lengkap dari semua perubahan dan tindakan. Ini membantu Anda memenuhi persyaratan kepatuhan melalui kontrol versi dan proses otomatis.
Skalabilitas	Alat ini dirancang untuk skala dari proyek kecil hingga penerapan kelas perusahaan besar.
Kolaborasi tim	Weave GitOps memfasilitasi kolaborasi antara tim pengembangan dan operasi melalui Git-based alur kerja.
GitOps sebagai layanan	Alat ini menawarkan GitOps sebagai layanan terkelola, yang menyederhanakan adopsi dan manajemen.
Dukungan hybrid dan multi-cloud	Weave GitOps memungkinkan pengelolaan yang konsisten di berbagai penyedia cloud dan lingkungan lokal.
Keamanan berkelanjutan	Alat ini mengintegrasikan pemindaian keamanan dan penegakan kebijakan selama proses penyebaran.

Weave GitOps menerapkan prinsip-prinsip ini untuk memberikan GitOps solusi komprehensif yang melampaui otomatisasi penyebaran dasar. Ini bertujuan untuk menciptakan model operasional lengkap untuk aplikasi cloud-native yang berfokus pada keamanan, skalabilitas, dan kemudahan penggunaan. Dengan mematuhi GitOps prinsip-prinsip ini, Weave GitOps membantu organisasi mencapai pengelolaan lingkungan Kubernetes yang konsisten, dapat diaudit, dan efisien di berbagai klaster dan penyedia cloud.

Untuk informasi selengkapnya, lihat GitOpsdokumentasi Weave.

Arsitektur

Diagram berikut menggambarkan alur kerja GitOps-driven CD yang menggunakan Weave GitOps dalam cluster EKS. Untuk informasi lebih lanjut, lihat GitOpsrepositori Weave.



di mana:

- Langkah 1: Tarik permintaan (PR) bergabung. Pengembang melakukan perubahan pada manifes Kubernetes atau bagan Helm yang disimpan dalam repositori Git. Ketika PR telah ditinjau dan digabungkan ke cabang utama, status aplikasi yang diinginkan diperbarui dalam kontrol sumber.
- Langkah 2: Sinkronisasi repositori. Weave GitOps berjalan di dalam namespace Flux di cluster EKS dan terus memantau repositori Git yang dikonfigurasi. Ketika mendeteksi perubahan, ia menarik pembaruan terbaru untuk merekonsiliasi status yang dideklarasikan.
- Langkah 3: Deployment untuk menargetkan namespace. Weave GitOps membandingkan status yang diinginkan dari Git dengan status langsung di cluster. Ini kemudian menerapkan perubahan yang diperlukan pada namespace beban kerja target sehingga aplikasi diterapkan atau diperbarui sesuai.

Jenkins X

Jenkins X adalah CI/CD platform cloud-native, open-source yang mengimplementasikan GitOps prinsip-prinsip untuk lingkungan Kubernetes. Meskipun Jenkins X tidak secara eksklusif GitOps alat seperti Argo CD atau Flux, Jenkins X menggabungkan GitOps praktik ke dalam alur kerjanya.

GitOps dukungan

Bidang	Kemampuan alat
Git-centric alur kerja	Jenkins X menggunakan repositori Git sebagai sumber utama kebenaran untuk kode aplikasi dan konfigurasi. Semua perubahan pada aplikasi dan infrastruktur dilakukan melalui Git.
Lingkungan sebagai kode (eAC)	Lingkungan (seperti pementasan dan produksi) didefinisikan sebagai kode dalam repositori Git. Ini memungkinkan kontrol versi dan peninjauan konfigurasi lingkungan.
CI/CD Pipa otomatis	Jenkins X secara otomatis menyiapkan CI/CD saluran pipa untuk proyek. Pipeline ini didefinis ikan sebagai kode (pipeline sebagai kode) dan disimpan dalam Git.
Kubernetes-native	Jenkins X dibuat khusus untuk lingkungan Kubernetes. Ini menggunakan sumber daya Kubernetes dan definisi sumber daya kustom (CRD).
Lingkungan pratinjau	Jenkins X secara otomatis membuat lingkunga n sementara untuk permintaan tarik. Ini memungkinkan peninjauan dan pengujian perubahan yang mudah sebelum penggabun gan.
Promosi antar lingkungan	Jenkins X menggunakan GitOps pendekatan untuk mempromosikan aplikasi antar lingkunga n (misalnya, dari pementasan hingga produksi) . Promosi ditangani dengan menggunakan permintaan tarik untuk memastikan proses peninjauan dan persetujuan yang tepat.

Bidang	Kemampuan alat
Manajemen bagan helm	Jenkins X menggunakan bagan Helm untuk mengemas dan menyebarkan aplikasi. Bagan dikendalikan versi di repositori Git.
Pembuatan versi otomatis	Jenkins X secara otomatis mengelola versi aplikasi dan rilis. Ini menggunakan versi semantik dan menghasilkan catatan rilis.
ChatOps integrasi	Jenkins X mendukung ChatOps operasi umum. Ini sejalan dengan GitOps prinsip-prinsip otomatisasi dan kolaborasi.
Ekstensibilitas	Alat ini menyediakan sistem plugin untuk memperluas fungsionalitas. Ini memungkinkan integrasi dengan berbagai alat cloud-native.
Infrastruktur sebagai kode (IAc)	Jenkins X mendukung Terraform,, CloudForm ation AWS Cloud Development Kit (AWS CDK), dan alat IAc lainnya untuk mendefinisikan dan mengelola infrastruktur. Definisi infrastruktur dikendalikan versi bersama kode aplikasi.
Rollback otomatis	Jenkins X mendukung rollback otomatis jika masalah terdeteksi setelah penerapan.
Manajemen rahasia	Alat ini terintegrasi dengan solusi manajemen rahasia eksternal untuk menangani informasi sensitif dengan aman.
Observabilitas	Jenkins X menyediakan integrasi dengan alat pemantauan dan pencatatan untuk observabi litas.
Multi-cloud dukungan	Jenkins X dirancang untuk bekerja di berbagai penyedia cloud dan lingkungan lokal.

Bidang	Kemampuan alat
Kolaborasi tim	Alat ini mendorong kolaborasi melalui Git-based alur kerja dan permintaan tarik.
Umpan balik berkelanjutan	Alat ini memberikan umpan balik cepat tentang perubahan melalui pengujian otomatis dan lingkungan pratinjau.
DevOps praktik terbaik	Jenkins X menerapkan praktik DevOps terbaik secara default, termasuk GitOps prinsip.
Konfigurasi deklaratif	Alat ini menggunakan konfigurasi deklaratif untuk mendefinisikan aplikasi dan lingkungan.
Pemutakhiran otomatis	Jenkins X menyediakan alat untuk mengotoma tiskan peningkatan platform Jenkins X itu sendiri.

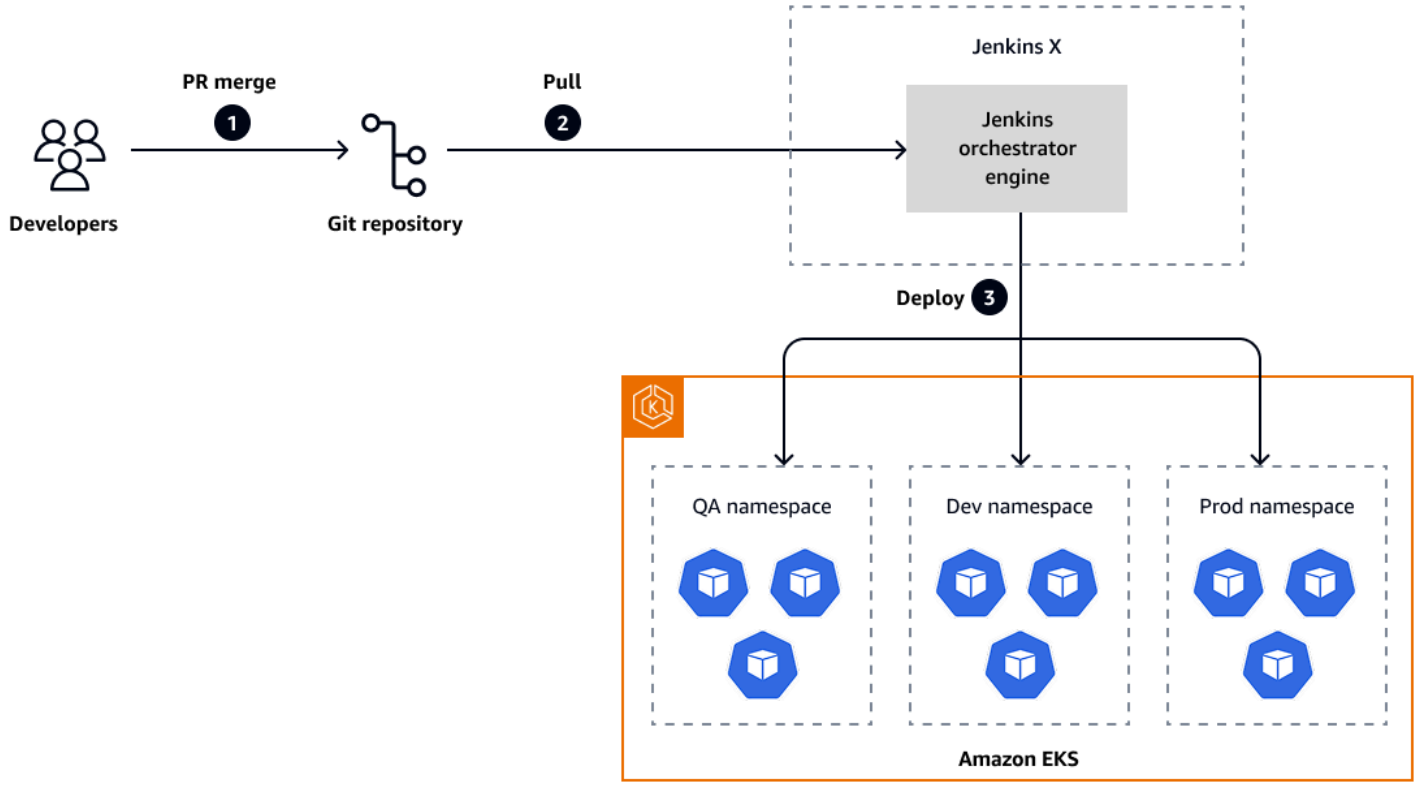
Jenkins X menerapkan GitOps prinsip-prinsip ini untuk menciptakan CI/CD solusi komprehensif bagi Kubernetes. Ini bertujuan untuk mengotomatiskan dan merampingkan seluruh proses pengiriman perangkat lunak, dari komit kode hingga penyebaran produksi, sambil mengikuti praktik. GitOps Dengan demikian, ini membantu tim mencapai penerapan yang lebih cepat, lebih andal, dan lebih konsisten di lingkungan cloud-native.

Perbedaan utama antara Jenkins X dan alat-alat seperti Argo CD atau Flux adalah bahwa Jenkins X memberikan CI/CD solusi yang lebih komprehensif, termasuk otomatisasi build dan manajemen pipa, sambil tetap menggabungkan GitOps prinsip-prinsip untuk penerapan dan manajemen lingkungan. Ini membuatnya sangat cocok untuk tim yang membutuhkan solusi all-in-one yang mencakup aspek CI dan CD dalam satu GitOps kerangka kerja.

Untuk informasi lebih lanjut, lihat dokumentasi Jenkins X.

Arsitektur

Diagram berikut menggambarkan alur kerja GitOps-driven CD yang menggunakan Jenkins X. Untuk informasi rinci, lihat dokumentasi Jenkins X.



di mana:

- Langkah 1: Tarik permintaan (PR) bergabung. Pengembang membuat permintaan tarik yang mencakup perubahan pada manifes Kubernetes, bagan Helm, atau kode aplikasi yang disimpan dalam repositori Git. Setelah peninjauan dan persetujuan, PR digabungkan ke cabang utama dan memperbarui status yang diinginkan dalam kontrol sumber.
- Langkah 2: Sinkronisasi repositori. Jenkins X secara otomatis memicu CI/CD pipeline ketika mendeteksi perubahan. Pipeline membangun, menguji, dan mempromosikan aplikasi melalui lingkungan yang berbeda (misalnya, pementasan dan produksi) dengan menggunakan GitOps prinsip.
- Langkah 3: Penerapan untuk menargetkan ruang nama. Jenkins X memperbarui repositori lingkungan (pementasan dan produksi) dengan versi aplikasi baru. Cluster secara otomatis merekonsiliasi perubahan dengan menarik manifes terbaru dari Git dan menerapkan aplikasi ke ruang nama yang sesuai.

GitLab CI/CD

GitLab CI/CD adalah bagian terintegrasi dari GitLab platform yang menyediakan kemampuan integrasi, pengiriman, dan penyebaran berkelanjutan. Meskipun GitLab CI/CD bukan hanya sebuah GitOps alat, Anda dapat mengonfigurasinya untuk mengimplementasikan GitOps prinsip-prinsip, terutama ketika Anda menggunakannya untuk penerapan Kubernetes.

GitOps dukungan

Bidang	Kemampuan alat
Git sebagai satu-satunya sumber kebenaran	GitLab CI/CD menggunakan repositori Git untuk menyimpan kode aplikasi dan konfigura si infrastruktur. Semua perubahan pada sistem dilakukan melalui Git, yang memastikan riwayat lengkap dan jejak audit.
Konfigurasi deklaratif	GitLab CI/CD pipelines didefinisikan dalam file.gitlab-ci.yl, yang merupakan konfigurasi deklaratif yang disimpan dalam repositori Git. File manifes Kubernetes, bagan Helm, atau file infrastruktur lainnya sebagai kode (IAc) dapat disimpan dalam repositori yang sama untuk menentukan status infrastruktur yang diinginka n.
Pipa otomatis	GitLab CI/CD secara otomatis memicu saluran pipa ketika perubahan didorong ke repositor i. Pipa ini dapat mencakup tahapan untuk membangun, menguji, dan menyebarkan aplikasi.
Integrasi Kubernetes	GitLab CI/CD menyediakan integrasi Kubernete s asli dan mendukung GitOps-style penerapan ke klaster Kubernetes. Ini dapat secara otomatis membuat dan mengelola sumber daya Kubernetes berdasarkan konfigurasi di Git.

Bidang	Kemampuan alat
Manajemen lingkungan	GitLab CI/CD mendukung definisi beberapa lingkungan (seperti pementasan dan produksi) sebagai kode. Penerapan ke lingkungan ini dapat diotomatisasi atau mungkin memerluka n persetujuan manual, sesuai dengan GitOps praktik.
Tinjau aplikasi	GitLab dapat secara otomatis membuat lingkungan sementara untuk permintaan gabungan, mirip dengan lingkungan pratinjau di GitOps alat lain. Ini mendukung peninjauan dan pengujian perubahan yang mudah sebelum penggabungan.
Deployment kontinu	GitLab CI/CD dapat dikonfigurasi untuk secara otomatis menerapkan perubahan ke klaster Kubernetes ketika perubahan digabungkan ke cabang tertentu.
IAc	GitLab CI/CD mendukung integrasi dengan alat seperti Terraform dan CloudFormation untuk mengelola infrastruktur sebagai kode. Definisi infrastruktur dapat dikontrol versi bersama kode aplikasi.
Observabilitas dan pemantauan	GitLab CI/CD menyediakan fitur pemantauan dan observabilitas bawaan, termasuk integrasi dengan Prometheus dan Grafana.
Pemindaian keamanan	GitLab CI/CD termasuk alat pemindaian keamanan bawaan yang dapat diintegrasikan ke dalam CI/CD pipa untuk menegakkan keamanan sebagai bagian dari GitOps alur kerja.

Bidang	Kemampuan alat
Registri kontainer	GitLab CI/CD termasuk registri kontainer bawaan untuk integrasi manajemen gambar kontainer yang mulus dalam GitOps alur kerja.
Otomatis DevOps	DevOps Fitur Auto in GitLab CI/CD dapat secara otomatis mengonfigurasi CI/CD pipeline yang mengikuti GitOps prinsip penerapan Kubernetes.
Alur kerja persetujuan	GitLab CI/CD mendukung proses persetujuan untuk penerapan, yang menyediakan promosi terkontrol antar lingkungan.
Manajemen rahasia	GitLab CI/CD menyediakan fitur untuk mengelola dan menggunakan rahasia dengan aman di dalam CI/CD jaringan pipa.
Versi dan rilis	GitLab CI/CD mendukung pembuatan versi otomatis dan manajemen rilis sebagai bagian dari proses. CI/CD
Rollback	GitLab CI/CD memungkinkan rollback mudah ke versi sebelumnya jika masalah terdeteksi setelah penerapan.
Log Audit	GitLab CI/CD menyediakan log audit komprehensif untuk semua tindakan untuk mendukung aspek keterlacakan. GitOps
Multi-project jaringan pipa	GitLab CI/CD mendukung GitOps alur kerja kompleks yang mencakup beberapa proyek atau repositori.
ChatOps	GitLab CI/CD mendukung ChatOps integrasi , yang menyediakan kolaborasi dan operasi melalui antarmuka obrolan.

Bidang	Kemampuan alat
Manajemen klaster Kubernetes	GitLab CI/CD menyediakan fitur untuk mengelola cluster Kubernetes langsung dari antarmuka. GitLab

Meskipun tidak GitLab CI/CD dirancang khusus untuk GitOps, ini dapat digunakan secara efektif untuk menerapkan GitOps praktik, terutama untuk tim yang sudah menggunakan GitLab sebagai platform pengembangan utama mereka. Pendekatan terintegrasinya, yang menggabungkan kontrol sumber CI/CD, dan manajemen Kubernetes, menjadikannya alat yang ampuh untuk mengimplementasikan alur kerja. GitOps

Perbedaan utama antara GitLab CI/CD dan GitOps alat khusus seperti Argo CD atau Flux adalah yang GitLab menyediakan platform yang lebih komprehensif yang mencakup manajemen kontrol sumber, pelacakan masalah, dan alat pengembangan lainnya beserta kemampuannya CI/CD. Ini membuatnya sangat cocok untuk tim yang membutuhkan solusi all-in-one yang dapat menerapkan GitOps praktik dalam sistem pengembangan yang lebih luas.

Untuk informasi lebih lanjut tentang GitLab CI/CD dan arsitekturnya, lihat GitLab CI/CD dokumentasi.

Pemintal

Meskipun Spinnaker tidak dirancang secara eksklusif sebagai GitOps alat, Anda dapat mengonfigurasinya untuk mengimplementasikan GitOps prinsip, terutama saat Anda menggunakannya untuk penerapan cloud-native dan Kubernetes.

GitOps dukungan

Bidang	Kemampuan alat
Konfigurasi deklaratif	Spinnaker menggunakan definisi pipeline deklaratif, yang biasanya disimpan sebagai file JSON atau YAMM. Definisi pipeline ini dapat dikontrol versi di repositori Git, selaras dengan praktik. GitOps

Bidang	Kemampuan alat
IAc	Spinnaker mendukung definisi konfigurasi infrastruktur dan penerapan sebagai kode. Definisi ini dapat disimpan dalam repositor i Git, dan dapat berfungsi sebagai sumber kebenaran tunggal.
Multi-cloud penyebaran	Spinnaker dirancang untuk bekerja di beberapa penyedia cloud dan cluster Kubernetes. Ini memungkinkan GitOps praktik yang konsisten di berbagai lingkungan.
Pipeline sebagai kode	Pipeline spinnaker dapat didefinisikan sebagai kode dan disimpan dalam repositori Git. Hal ini memungkinkan untuk kontrol versi dan peninjauan proses penyebaran.
Penerapan otomatis	Anda dapat mengonfigurasi Spinnaker untuk memulai penerapan secara otomatis berdasarkan perubahan di repositori Git. Alat ini mendukung praktik penerapan berkelanjutan yang merupakan pusat. GitOps
Infrastruktur yang tidak dapat diubah	Spinnaker mempromosikan penggunaan infrastruktur yang tidak dapat diubah, yang merupakan konsep kunci dalam. GitOps Ini mendorong penerapan instance baru alih-alih memodifikasi instance yang ada.
Rollback dan pembuatan versi	Spinnaker menyediakan kemampuan rollback yang kuat dan pengembalian cepat ke keadaan baik yang diketahui sebelumnya. Ini mendukung pembuatan versi penerapan, selaras dengan prinsip-prinsip keterlacakan. GitOps

Bidang	Kemampuan alat
Alur kerja persetujuan	Spinnaker mencakup tahapan penilaian manual dalam jaringan pipa untuk mendukung promosi terkontrol antar lingkungan. Ini mendukung GitOps praktik pemisahan antara penerapan dan rilis.
Canary dan penyebaran blue/green	Spinnaker mendukung strategi penerapan lanjutan yang selaras dengan GitOps praktik untuk rilis yang aman dan terkontrol.
Integrasi dengan sistem kontrol versi	Spinnaker dapat berintegrasi dengan berbagai penyedia Git untuk memulai pipeline berdasark an peristiwa repositori.
Integrasi Kubernetes	Spinnaker menyediakan dukungan asli untuk Kubernetes dan mendukung GitOps-style pengelolaan sumber daya Kubernetes.
Manajemen Artifak	Spinnaker mendukung manajemen artefak dan pembuatan versi, yang sangat penting untuk mempertahankan alur kerja. GitOps
Observabilitas dan pemantauan	Spinnaker menawarkan integrasi dengan alat pemantauan untuk mendukung aspek observabilitas. GitOps
Jejak audit	Spinnaker menyediakan log dan riwayat penyebaran terperinci, yang mendukung prinsip auditabilitas. GitOps
Role-based kontrol akses (RBAC)	Alat ini mengimplementasikan RBAC untuk kontrol halus atas siapa yang dapat melakukan tindakan mana, sesuai dengan praktik keamanan. GitOps

Bidang	Kemampuan alat
Templating dan parameterisasi	Spinnaker mendukung templating dalam definisi pipeline untuk mengaktifkan penerapan yang dapat digunakan kembali dan diparamet erisasi.
Promosi lingkungan	Spinnaker memfasilitasi promosi aplikasi antar lingkungan (misalnya, dari pementasan ke produksi) secara terkontrol.
Integrasi dengan alat CI	Spinnaker dapat berintegrasi dengan berbagai alat integrasi berkelanjutan (CI) untuk menyediakan CI/CD pipa lengkap yang mematuhi prinsip. GitOps
Tahapan dan ekstensi khusus	Alat ini mendukung tahapan dan ekstensi khusus, sehingga tim dapat menerapkan GitOps alur kerja yang disesuaikan dengan kebutuhan mereka.
Manajemen terpusat	Spinnaker menyediakan platform terpusat untuk mengelola penyebaran di berbagai lingkungan dan penyedia cloud.

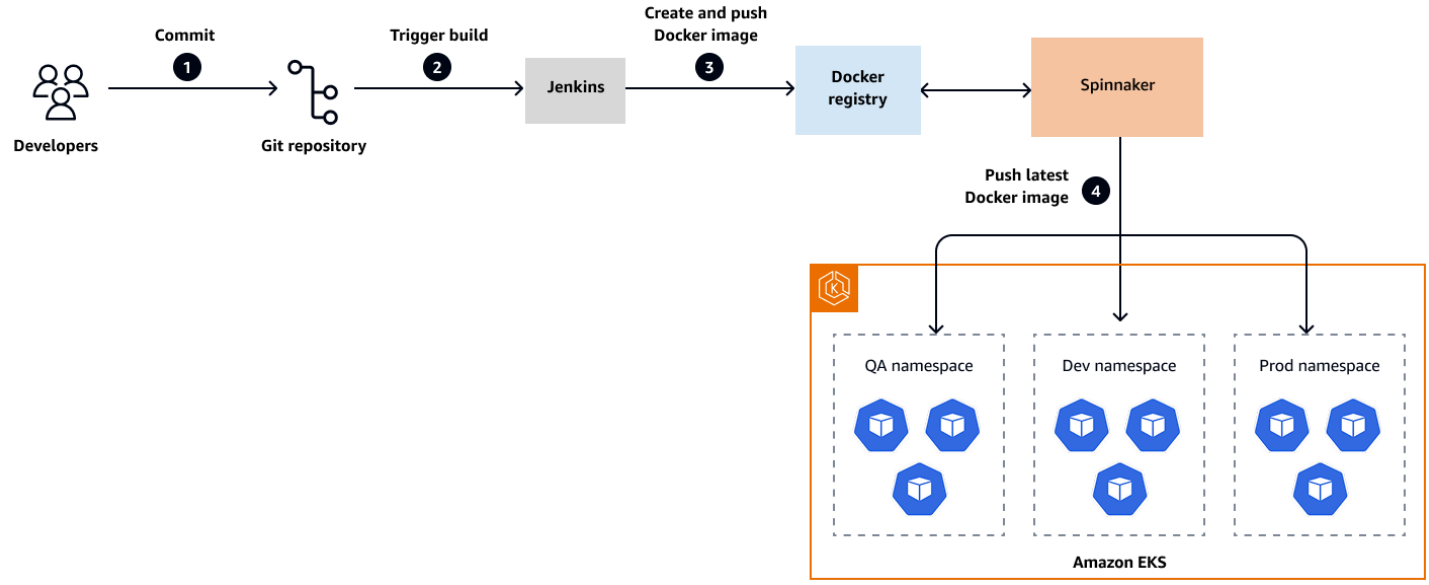
Meskipun Spinnaker tidak terutama dipasarkan sebagai GitOps alat, fleksibilitas dan set fitur yang kuat membuatnya mampu mengimplementasikan GitOps alur kerja, terutama di lingkungan multi-cloud yang kompleks. Perbedaan utama antara Spinnaker dan GitOps alat khusus seperti Argo CD atau Flux adalah bahwa Spinnaker menawarkan platform pengiriman berkelanjutan yang lebih komprehensif dengan strategi penyebaran lanjutan dan dukungan multi-cloud.

Kekuatan Spinnaker terletak pada kemampuannya untuk menangani skenario penyebaran yang kompleks di berbagai penyedia cloud dan dukungannya untuk strategi penerapan lanjutan. Ketika Spinnaker dikonfigurasi dengan benar, ia dapat menerapkan GitOps prinsip-prinsip secara efektif. Ini menjadikannya alat yang ampuh bagi organisasi yang ingin mengadopsi GitOps praktik di lingkungan yang beragam dan kompleks.

Untuk informasi lebih lanjut, lihat dokumentasi Spinnaker.

Arsitektur

Diagram berikut menggambarkan alur kerja GitOps-driven CD yang menggunakan Spinnaker dan Jenkins X. Untuk informasi rinci, lihat dokumentasi Spinnaker.



di mana:

- Langkah 1: Kode komit. Pengembang melakukan perubahan kode aplikasi ke repositori Git. Perubahan ini dapat mencakup pembaruan pada aplikasi itu sendiri, Dockerfiles, atau manifes Kubernetes.
- Langkah 2: Jenkins membangun dan membuat gambar. Jenkins secara otomatis dipicu oleh repositori Git melalui webhook atau polling. Jenkins membangun aplikasi, membuat gambar Docker, dan mendorong gambar yang dibangun ke registri Docker yang dikonfigurasi (seperti Amazon ECR atau Docker Hub).
- Langkah 3: Pemantauan gambar Spinnaker dan pemicu pipa. Spinnaker terus memantau registri Docker untuk gambar baru. Ketika versi gambar baru terdeteksi, Spinnaker secara otomatis memicu pipeline untuk memulai proses penerapan.
- Langkah 4: Penerapan untuk menargetkan ruang nama. Spinnaker menyebarkan gambar Docker baru ke Amazon EKS. Berdasarkan konfigurasi pipeline, gambar digunakan untuk menargetkan ruang nama di cluster. Spinnaker memastikan bahwa versi aplikasi terbaru diterapkan sambil mengikuti strategi penerapan yang ditentukan seperti atau penerapan canary. blue/green

Armada Peternak

Rancher Fleet adalah GitOps-at-scale solusi yang dirancang khusus untuk mengelola beberapa cluster Kubernetes. Ini mematuhi GitOps prinsip-prinsip sambil berfokus pada skalabilitas dan manajemen multi-cluster.

GitOps dukungan

Bidang	Kemampuan alat
Git sebagai satu-satunya sumber kebenaran	Fleet menggunakan repositori Git sebagai sumber otoritatif untuk mendefinisikan status aplikasi dan sumber daya yang diinginkan di beberapa cluster. Semua konfigurasi, termasuk manifes Kubernetes, bagan Helm, dan sumber daya kustom, disimpan di Git.
Konfigurasi deklaratif	Armada bekerja dengan deskripsi deklaratif dari keadaan yang diinginkan untuk aplikasi dan sumber daya. Ini bisa berupa YAMM Kubernete s mentah, bagan Helm, file Kustomisasi, atau sumber daya khusus. Fleet-specific
Sinkronisasi otomatis	Armada terus memantau repositori Git untuk perubahan. Secara otomatis menerapka n perubahan pada cluster target ketika mendeteksi perbedaan antara status Git dan status cluster.
Multi-cluster manajemen	Armada dirancang khusus untuk mengelola penyebaran di beberapa klaster Kubernetes. Ini dapat menangani ribuan cluster dari satu bidang kontrol.
Kubernetes-native arsitektur	Armada dibangun sebagai satu set sumber daya dan pengontrol kustom Kubernetes. Ini menggunakan mekanisme ekstensi di Kubernetes untuk operasi. GitOps

Bidang	Kemampuan alat
Rekonsiliasi berkelanjutan	Armada terus-menerus membandingkan keadaan cluster yang sebenarnya dengan status yang diinginkan yang didefinisikan dalam Git. Ini secara otomatis mengoreksi penyimpangan yang terdeteksi di antara negara-negara ini.
Pengelompokan dan penargetan cluster	Armada memungkinkan Anda mengelompokkan kluster dan menargetkan penyebaran ke grup atau kluster individu tertentu. Ini mendukung penerapan aplikasi yang konsisten di berbagai lingkungan dan jenis cluster.
konfigurasi berlapis	Armada mendukung konfigurasi berlapis, yang menyediakan konfigurasi dasar dengan hamparan khusus lingkungan. Ini sejalan dengan GitOps praktik mengelola beberapa lingkungan secara efisien.
Integrasi helm	Fleet menyediakan dukungan asli untuk bagan Helm dan menyediakan pengelolaan aplikasi yang kompleks dengan mudah. Itu dapat membuat versi dan mengelola rilis Helm melalui GitOps alur kerja.
Definisi sumber daya khusus (CRD)	Armada menggunakan sumber daya khusus seperti GitRepo dan Bundle untuk menentukan penerapan. CRD ini menyediakan Kubernetes-native cara untuk menentukan GitOps alur kerja.
Keamanan dan RBAC	Armada terintegrasi dengan Kubernetes RBAC untuk kontrol akses. Ini mendukung pengelolaan informasi sensitif dan kredensil yang aman.

Bidang	Kemampuan alat
Observabilitas	Armada menyediakan informasi status tentang keadaan sinkronisasi cluster dan aplikasi. Ini menawarkan wawasan tentang GitOps proses di seluruh armada cluster.
Skalabilitas	Armada dirancang untuk skala untuk mengelola ribuan cluster secara efisien. Ini mendukung GitOps operasi skala besar di lingkungan perusahaan.
Manajemen dependensi	Anda dapat menentukan dependensi antara sumber daya dan aplikasi yang berbeda. Armada memastikan bahwa urutan operasi yang benar diikuti dalam penyebaran yang kompleks.
Kustomisasi dan ekstensibilitas	Armada mendukung skrip khusus dan kait siklus hidup untuk penyesuaian penerapan lanjutan. Ini memungkinkan integrasi dengan alat dan alur kerja yang ada.
Dukungan offline dan celah udara	Armada dapat beroperasi di lingkungan yang memiliki konektivitas internet terbatas atau tidak ada. Ini mendukung GitOps alur kerja di lingkungan dengan keamanan tinggi atau diatur.
Peluncuran progresif	Armada mendukung peluncuran bertahap di seluruh cluster, yang memungkinkan strategi penyebaran yang terkontrol dan bertahap.
Antarmuka manajemen terpadu	Fleet menyediakan antarmuka tunggal untuk mengelola GitOps alur kerja di semua cluster. Ini menyederhanakan operasi di lingkungan multi-cluster yang kompleks.

Bidang	Kemampuan alat
Integrasi dengan alat Rancher lainnya	Fleet terintegrasi dengan alat Rancher lainnya untuk memberikan solusi manajemen Kubernetes yang komprehensif.
Jejak audit dan kepatuhan	Armada mempertahankan jejak audit yang jelas dari semua perubahan dan penyebara n. Ini membantu Anda memenuhi persyaratan kepatuhan melalui operasi yang dikendalikan versi. Git-based

Rancher Fleet menerapkan GitOps prinsip-prinsip ini dengan fokus yang kuat pada skalabilitas dan manajemen multi-cluster. Desainnya sangat cocok untuk organisasi yang mengelola sejumlah besar cluster Kubernetes di berbagai lingkungan, pusat data, atau penyedia cloud.

Pembeda utama Armada adalah kemampuannya untuk menangani dalam GitOps skala besar. Fitur ini membuatnya sangat berharga bagi perusahaan besar atau penyedia layanan terkelola yang mengelola banyak cluster. Alat seperti Argo CD atau Flux sering digunakan untuk manajemen cluster individu, sedangkan Armada dirancang untuk mengelola GitOps seluruh armada besar cluster.

Dengan mematuhi GitOps prinsip-prinsip ini, Rancher Fleet memberikan solusi bagi organisasi yang ingin menerapkan manajemen aplikasi dan sumber daya yang konsisten, terukur, dan otomatis di seluruh lingkungan Kubernetes yang beragam dan berskala besar.

Untuk informasi lebih lanjut, lihat dokumentasi Armada.

Arsitektur

Untuk informasi arsitektur dan alur kerja, lihat repositori Armada.

Codefresh

Codefresh adalah CI/CD platform modern yang mendukung GitOps prinsip, terutama untuk penerapan Kubernetes. Codefresh menawarkan serangkaian CI/CD fitur yang komprehensif, dan GitOps kemampuannya terkenal.

GitOps dukungan

Bidang	Kemampuan alat
Git sebagai satu-satunya sumber kebenaran	Codefresh menggunakan repositori Git sebagai sumber otoritatif untuk kode aplikasi, definisi infrastruktur, dan konfigurasi pipeline. Semua perubahan pada sistem dilakukan melalui Git, yang memastikan riwayat lengkap dan jejak audit.
Konfigurasi deklaratif	Codefresh mendukung definisi pipeline deklaratif dengan menggunakan file YAMM yang disimpan di Git. Manifes Kubernetes, bagan Helm, CloudFormation template, dan file IAc lainnya dapat dikontrol versi dalam repositori yang sama.
GitOps dasbor	Codefresh menyediakan GitOps dasbor khusus untuk memvisualisasikan dan mengelola GitOps alur kerja. Ini menawarkan pandangan yang jelas tentang status sinkronisasi antara Git dan status cluster.
Sinkronisasi otomatis	Codefresh terus memantau repositori Git untuk perubahan. Ini secara otomatis memulai saluran pipa untuk menerapkan perubahan pada lingkungan target ketika mendeteksi perbedaan.
Integrasi Kubernetes	Codefresh menawarkan integrasi mendalam dengan Kubernetes untuk mendukung GitOps-style penerapan di beberapa cluster. Ini mendukung berbagai sumber daya Kubernetes dan definisi sumber daya kustom (CRD).
Manajemen lingkungan	Anda dapat mendefinisikan dan mengelola beberapa lingkungan (seperti pengembangan,

Bidang	Kemampuan alat
	pementasan, dan produksi) sebagai kode. Codefresh mendukung promosi antar lingkunga n dengan menggunakan GitOps praktik.
Integrasi Argo CD	Codefresh terintegrasi dengan Argo CD untuk meningkatkan kemampuan. GitOps Ini menggabungkan kemampuan CI dengan kekuatan CD Argo CD untuk memberikan GitOps solusi lengkap.
Dukungan helm	Codefresh mendukung bagan Helm, dan menyediakan pengelolaan aplikasi kompleks yang mudah melalui. GitOps Ini juga menawarkan versi dan promosi bagan Helm.
Pengiriman progresif	Codefresh mendukung strategi penerapan lanjutan seperti canary dan deployment. blue/ green Anda dapat menerapkan dan mengelola strategi ini melalui GitOps alur kerja.
Rollback dan pembuatan versi	Codefresh memungkinkan rollback mudah ke versi sebelumnya jika masalah terdeteksi setelah penerapan. Ini mempertahankan versi penerapan untuk keterlacakan.
Alur kerja persetujuan	Codefresh mendukung proses persetuju an manual dan otomatis untuk penerapan. Ini memungkinkan promosi terkontrol antar lingkungan, sesuai dengan GitOps praktik.
IAc	Codefresh mendukung integrasi dengan alat IAc seperti CloudFormation dan Terraform. Ini memungkinkan kontrol versi definisi infrastru ktur bersama kode aplikasi.

Bidang	Kemampuan alat
Observabilitas dan pemantauan	Codefresh menyediakan fitur pemantauan dan observabilitas bawaan. Ini juga menawarkan integrasi dengan alat pemantauan eksternal untuk meningkatkan visibilitas.
Pemindaian keamanan	Codefresh mencakup kemampuan pemindaian keamanan yang dapat diintegrasikan ke dalam GitOps alur kerja. Pemeriksaan keamanan adalah bagian dari proses penyebaran otomatis.
Jejak audit	Codefresh memelihara log audit komprehen sif untuk semua tindakan dan perubahan . Ini mendukung aspek ketertelusuran dan kepatuhan. GitOps
RBAC dan kontrol akses	Codefresh mengimplementasikan kontrol akses berbasis peran (RBAC) untuk manajemen izin berbutir halus. Ini membantu memastikan GitOps operasi yang aman di seluruh tim dan lingkungan.
GitOps otomatisasi	Codefresh menawarkan fitur untuk mengotoma tiskan berbagai aspek GitOps alur kerja, termasuk pembuatan dan penggabungan pull request (PR).
Multi-cloud dan penyebaran hibrida	Codefresh mendukung GitOps alur kerja di beberapa penyedia cloud dan lingkungan lokal.
Templating dan parameterisasi	Codefresh mendukung template dalam konfigurasi pipeline dan deployment. Ini memungkinkan alur kerja yang dapat digunakan kembali dan GitOps diparamet erisasi.

Bidang	Kemampuan alat
Manajemen gambar terintegrasi	Codefresh menyediakan kemampuan manajemen gambar kontainer bawaan. Ini mengintegrasikan build dan penerapan gambar ke dalam alur kerja. GitOps
GitOps untuk manajemen rahasia	Codefresh menawarkan cara aman untuk mengelola rahasia dalam GitOps alur kerja. Ini terintegrasi dengan solusi manajemen rahasia eksternal.
Fitur kolaborasi	Codefresh menyediakan fitur untuk kolaboras i tim dalam GitOps proses. Fitur-fitur ini termasuk komentar, pemberitahuan, dan dasbor bersama.

Pendekatan Codefresh GitOps terkenal karena integrasi CI/CD kemampuannya dengan GitOps praktik. Ini bertujuan untuk menyediakan platform komprehensif yang mencakup seluruh siklus hidup pengiriman perangkat lunak sambil mematuhi prinsip. GitOps

Pembeda utama Codefresh di GitOps area ini adalah pendekatan platform terpadu, yang menggabungkan kemampuan CI dengan CD dan fitur. GitOps Ini membuatnya sangat cocok untuk tim yang menginginkan solusi all-in-one yang dapat menangani CI/CD skenario kompleks saat menerapkan GitOps praktik.

Codefresh menawarkan platform untuk organisasi yang ingin mengadopsi GitOps metodologi dalam CI/CD konteks yang lebih luas, terutama ketika bekerja dengan Kubernetes dan teknologi cloud-native.

Untuk informasi selengkapnya, lihat dokumentasi Codefresh.

Pulumi

Pulumi adalah platform IAc yang tidak dirancang khusus untuk. GitOps Namun, ini dapat digunakan secara efektif untuk menerapkan GitOps prinsip-prinsip, terutama untuk infrastruktur cloud dan penerapan Kubernetes.

GitOps dukungan

Bidang	Kemampuan alat
IAc	Pulumi memungkinkan Anda menentukan infrastruktur Anda dengan menggunakan bahasa pemrograman tujuan umum seperti Python,, dan Go. TypeScript Pendekatan berbasis kode ini sejalan dengan GitOps penekanan pada konfigurasi deklaratif berversi.
Git sebagai satu-satunya sumber kebenaran	Kode infrastruktur di Pulumi dapat disimpan dan dikontrol versi di repositori Git. Ini memastikan bahwa Git berfungsi sebagai sumber kebenaran tunggal untuk definisi infrastruktur.
Status yang diinginkan deklaratif	Meskipun Pulumi menggunakan bahasa pemrograman, ia masih menggambarkan keadaan infrastruktur yang diinginkan secara deklaratif. Kode mendefinisikan seperti apa infrastruktur seharusnya, bukan proses langkah demi langkah untuk membuatnya.
Sinkronisasi otomatis	Pulumi dapat diintegrasikan dengan CI/CD pipeline untuk secara otomatis menerapkan perubahan ketika kode diperbarui di Git. Hal ini memungkinkan penyebaran terus menerus dari perubahan infrastruktur, yang merupakan GitOps prinsip utama.
Multi-cloud dan dukungan Kubernetes	Pulumi mendukung berbagai penyedia cloud dan Kubernetes, sehingga Anda dapat mengikuti GitOps praktik di berbagai lingkunga n. Alat ini memungkinkan pengelolaan sumber daya yang konsisten di berbagai platform.

Bidang	Kemampuan alat
Manajemen negara	Pulumi mengelola keadaan infrastruktur, yang dapat disimpan dari jarak jauh dan aman. Manajemen negara ini sangat penting untuk GitOps praktik, dan memastikan konsistensi antara keadaan yang ditentukan dan keadaan sebenarnya dari infrastruktur Anda.
Deteksi dan rekonsiliasi drift	Pulumi dapat mendeteksi perbedaan antara keadaan yang diinginkan (dalam kode) dan keadaan infrastruktur yang sebenarnya. Ini merekonsiliasi perbedaan-perbedaan ini dalam keselarasan dengan GitOps prinsip rekonsiliasi berkelanjutan.
Kebijakan sebagai kode	Anda dapat mendefinisikan dan menegakkan kebijakan sebagai kode dengan menggunakan CrossGuard Pulumi. Ini memungkinkan GitOps-style manajemen kepatuhan dan kebijakan keamanan yang dikendalikan versi.
Manajemen rahasia	Pulumi menyediakan cara aman untuk mengelola informasi sensitif dalam kode infrastruktur. Ini mendukung integrasi dengan sistem manajemen rahasia eksternal, yang sangat penting untuk praktik GitOps keamanan.
Komponen modular dan dapat digunakan kembali	Pulumi mendukung pembuatan komponen dan modul yang dapat digunakan kembali. Modularitas ini sejalan dengan GitOps praktik untuk mengelola penyebaran multi-lingkungan yang kompleks.
Pratinjau dan rencanakan	Pulumi menawarkan kemampuan untuk melihat perubahan sebelum menerapkannya. Ini mendukung GitOps prinsip perubahan infrastru ktur yang aman dan dapat diprediksi.

Bidang	Kemampuan alat
Rollback dan sejarah	Pulumi mempertahankan riwayat penyebaran dan mendukung rollback ke negara bagian sebelumnya. Ini sejalan dengan GitOps prinsip ketertelusuran dan reversibilitas.
Pengiriman berkelanjutan untuk infrastruktur	Pulumi dapat diintegrasikan ke dalam CI/CD jaringan pipa untuk pengiriman perubahan infrastruktur yang berkelanjutan. Ini mendukung pengujian otomatis dan validasi kode infrastruktur.
RBAC dan kontrol akses	Pulumi menyediakan kontrol akses berbasis peran untuk mengelola siapa yang dapat melakukan perubahan infrastruktur. Ini mendukung praktik GitOps keamanan dan tata kelola.
Observabilitas dan pencatatan	Pulumi menawarkan kemampuan pencatatan dan pemantauan untuk perubahan infrastruktur. Kemampuan ini mendukung aspek observabilitas GitOps praktik.
Integrasi dengan alat lain	Pulumi dapat berintegrasi dengan berbagai alat di cloud. Fleksibilitas ini memungkinkan GitOps alur kerja yang komprehensif.
Manajemen lingkungan	Pulumi mendukung pengelolaan beberapa lingkungan (pengembangan, pementasan, produksi) dengan menggunakan basis kode yang sama dengan konfigurasi yang berbeda. Ini sejalan dengan GitOps praktik untuk manajemen multi-lingkungan yang konsisten.

Bidang	Kemampuan alat
Manajemen dependensi	Pulumi menangani dependensi antar sumber daya, dan memastikan urutan operasi yang benar. Ini sangat penting untuk GitOps penerapan kompleks yang melibatkan komponen yang saling bergantung.
Penyedia sumber daya khusus	Pulumi memungkinkan Anda membuat penyedia kustom untuk mengelola API-drive n layanan apa pun. Ini memperluas GitOps praktik ke berbagai sumber daya di luar penawaran cloud standar.
Fitur kolaborasi	Pulumi mendukung kolaborasi tim melalui kontrol status dan akses bersama. Ini memfasili tasi GitOps alur kerja di lingkungan tim.

Dengan menggunakan fitur Pulumi ini, organisasi dapat menerapkan GitOps praktik untuk infrastruktur mereka, terutama dalam skenario di mana mereka membutuhkan kontrol halus atau logika kompleks, atau ingin mengelola beragam sumber daya cloud dan lokal dalam satu kerangka kerja yang konsisten.

Pendekatan Pulumi GitOps adalah unik karena membawa kekuatan dan fleksibilitas bahasa pemrograman tujuan umum untuk manajemen infrastruktur sambil berpegang pada prinsip-prinsip. GitOps Ini bisa sangat menguntungkan bagi tim yang lebih suka bekerja dengan bahasa pemrograman yang sudah dikenal dan ingin menerapkan praktik terbaik rekayasa perangkat lunak untuk manajemen infrastruktur.

Pembeda utama Pulumi in GitOps adalah penggunaan bahasa pemrograman standar untuk mendefinisikan infrastruktur. GitOps Alat tradisional sering menggunakan YAMAL atau bahasa khusus domain, sedangkan Pulumi memungkinkan logika yang lebih kompleks, penggunaan kembali kode yang lebih baik, dan integrasi yang lebih mudah dengan alur kerja pengembangan yang ada.

Untuk informasi lebih lanjut, lihat dokumentasi Pulumi.

GitOps perbandingan alat

Berikut adalah perbandingan dari sembilan GitOps alat yang dibahas di bagian sebelumnya. Saat Anda memilih alat, pertimbangkan persyaratan spesifik Anda, infrastruktur yang ada, keahlian tim, dan tingkat kontrol dan penyesuaian yang diinginkan.

Kemudahan penggunaan

- Argo CD, Flux, dan Rancher Fleet umumnya lebih mudah diatur.
- Spinnaker dan Jenkins X memiliki kurva belajar yang lebih curam.
- Weave GitOps mungkin memerlukan lebih banyak pengaturan untuk fitur-fitur canggih.
- GitLab CI/CD dan Codefresh menawarkan pengalaman terintegrasi.

Integrasi Kubernetes

- Argo CD, Flux, dan Rancher Fleet sangat berpusat pada Kubernetes.
- Jenkins X dan Weave GitOps menawarkan kemampuan yang lebih luas DevOps .
- Alat-alat lain mendukung Kubernetes tanpa fokus eksklusif padanya.

Kemampuan CI/CD

- Jenkins X, GitLab CI/CD, and Codefresh offer complete CI/CD solusi.
- Argo CD, Flux, dan Weave lebih GitOps fokus pada aspek CD dari alur kerja, dan seringkali memerlukan integrasi dengan alat CI terpisah.

GitOps kemurnian

- Argo CD dan Flux adalah alat yang fokus secara khusus. GitOps
- Alat-alat lain menggabungkan GitOps prinsip-prinsip ke berbagai tingkat.

Dukungan multi-cloud

- Spinnaker dan Pulumi unggul dalam skenario multi-cloud.
- Alat lain dapat bekerja di cloud tetapi mungkin memerlukan pengaturan tambahan.

Dukungan multi-cluster

- Semua alat mendukung penerapan multi-cluster.
- Argo CD dan Weave GitOps memiliki fitur manajemen multi-cluster yang lebih canggih.

Integrasi

- Flux memiliki dukungan Cloud Native Computing Foundation (CNCF) yang kuat.
- Argo CD memiliki komunitas yang besar dan aktif.
- Argo CD dan Flux memiliki integrasi Kubernetes yang kuat.
- Jenkins X menggunakan sistem Jenkins yang lebih luas.
- Weave GitOps lebih baru tetapi tumbuh dengan dukungan komersial yang kuat.
- GitLab CI/CD terintegrasi erat dengan. GitLab
- Armada Peternak bekerja dengan baik dalam sistem Rancher.

Komunitas dan dukungan

- Flux memiliki dukungan CNCF yang kuat.
- Argo CD, GitLab, dan Spinnaker memiliki komunitas besar.
- Dukungan komersial tersedia untuk sebagian besar alat.

Fitur perusahaan

- Weave GitOps dan Jenkins X menawarkan lebih banyak fitur yang berfokus pada perusahaan secara default.
- Argo CD dan Flux memiliki penawaran perusahaan atau dapat diperpanjang untuk penggunaan perusahaan.

Fleksibilitas dan ekstensibilitas

- Fluks sangat modular dan dapat diperluas.
- Argo CD menawarkan opsi penyesuaian yang baik.
- Jenkins X sangat dapat diperluas tetapi mungkin membutuhkan lebih banyak usaha.
- Weave GitOps bertujuan untuk memberikan solusi lengkap dengan lebih sedikit kebutuhan untuk ekstensibilitas.

Skalabilitas

- Spinnaker dan GitLab CI/CD dikenal dengan skalabilitas perusahaan.
- Argo CD dan Flux menangani penerapan Kubernetes skala besar dengan baik.

Manajemen infrastruktur

- Pulumi berfokus pada manajemen infrastruktur.
- Weave GitOps dan Flux menawarkan kemampuan IAc yang baik.

Model pemrograman dan dukungan bahasa

- Di Pulumi, Anda dapat menentukan infrastruktur dengan menggunakan bahasa pemrograman tujuan umum seperti Python, Go,, C #, dan Java TypeScript. Penggunaan bahasa standar oleh Pulumi memungkinkan integrasi kode infrastruktur dengan alur kerja pengembangan yang sudah dikenal, praktik pengujian, dan logika kompleks.
- Terraform menggunakan HashiCorp Configuration Language (HCL).
- CloudFormation menggunakan template JSON dan YAMG.
- Argo CD, Flux, Rancher Fleet, Weave, Spinnaker GitOps, dan GitLab CI/CD terutama mengelola YAMAL atau file konfigurasi deklaratif.
- Jenkins X mengelola YAMAL dan pipeline berbasis skrip tetapi tidak secara native menawarkan pemrograman tujuan umum untuk IAc.

Kasus penggunaan Argo CD dan Flux

Bagian ini berfokus pada dua alat, Argo CD dan Flux, yang menyediakan fungsionalitas murni GitOps. Dalam konteks ini, pure GitOps mengacu pada model di mana repositori Git berfungsi sebagai sumber kebenaran tunggal untuk keadaan aplikasi dan infrastruktur yang diinginkan. Semua perubahan dilakukan melalui komit Git, dan sistem secara otomatis menyinkronkan lingkungan hidup agar sesuai dengan status yang ditentukan dalam repositori. Tidak diperlukan intervensi manual di luar operasi Git.

Pertimbangan umum

- Anda mungkin lebih suka menggunakan Argo CD di lingkungan di mana manajemen visual dan alur kerja yang berpusat pada aplikasi penting.
- Anda dapat memilih Flux jika Anda memerlukan solusi yang lebih ringan, multi-tenancy yang kuat, atau integrasi mendalam dengan jaringan Cloud Native Computing Foundations (CNCF) yang lebih luas.
- Argo CD sering menarik bagi tim yang beralih dari tradisional CI/CD ke GitOps karena UI intuitifnya.
- Flux sering disukai di lingkungan cloud-native di mana CLI-based alur kerja dan praktik IAc sudah ditetapkan.

Pada akhirnya, pilihan antara Argo CD dan Flux sering tergantung pada kebutuhan organisasi spesifik Anda, perkakas yang ada, dan preferensi tim. Kedua alat ini mampu menangani sebagian besar GitOps skenario, jadi kami sarankan Anda mengevaluasinya berdasarkan kasus dan persyaratan penggunaan spesifik Anda.

Kasus penggunaan Argo CD

Manajemen visual:

- Saat Anda membutuhkan UI yang ramah pengguna untuk mengelola penerapan dan memvisualisasikan status aplikasi.
- Untuk tim yang lebih memilih antarmuka grafis untuk pemantauan dan pemecahan masalah.

Application-centric pendekatan:

- Bila Anda ingin mengelola penerapan di tingkat aplikasi alih-alih mengelola sumber daya individu.
- Untuk organisasi yang menyusun penyebaran mereka di sekitar konsep aplikasi.

Multi-cluster manajemen:

- Saat mengelola penerapan di beberapa cluster adalah persyaratan utama.
- Untuk lingkungan yang kompleks dan terdistribusi dengan banyak cluster.

Gelombang rollback dan sinkronisasi:

- Saat Anda memerlukan kontrol halus atas proses penerapan, termasuk gelombang sinkronisasi dan intervensi manual.
- Untuk skenario yang membutuhkan strategi rollback yang kompleks.

Integrasi dengan alat yang ada:

- Saat Anda sudah menggunakan alat lain dalam proyek Argo seperti Argo Workflows dan Argo Events.

Lingkungan perusahaan:

- Untuk perusahaan besar yang membutuhkan RBAC yang kuat dan integrasi masuk tunggal secara default.

Kasus penggunaan fluks

Penerapan ringan:

- Bila Anda membutuhkan solusi yang lebih ringan dan kurang intensif sumber daya GitOps.
- Untuk komputasi tepi atau skenario IoT di mana sumber daya mungkin dibatasi.

Pembaruan gambar otomatis:

- Ketika deteksi otomatis dan penyebaran gambar kontainer baru adalah persyaratan utama.
- Untuk tim yang fokus pada penerapan berkelanjutan dengan pembaruan gambar yang sering.

Multi-tenancy:

- Ketika dukungan multi-tenancy yang kuat diperlukan, terutama di lingkungan cluster bersama.
- Untuk penyedia layanan atau organisasi besar yang memiliki pemisahan ketat antara tim atau proyek.

IAc:

- Ketika mengelola aplikasi dan infrastruktur melalui GitOps alur kerja yang sama adalah penting.
- Untuk tim yang banyak berinvestasi dalam paradigma IAc.

Integrasi helm:

- Ketika penggunaan grafik Helm secara ekstensif adalah bagian dari strategi penyebaran Anda.
- Untuk lingkungan yang memiliki Helm-based penerapan kompleks.

Integrasi proyek CNCF:

- Ketika integrasi yang ketat dengan proyek CNCF lainnya penting.
- Untuk organisasi yang selaras dengan teknologi dan prinsip CNCF.

Arsitektur modular:

- Bila Anda membutuhkan fleksibilitas untuk hanya menggunakan komponen tertentu dari GitOps toolkit.
- Untuk tim yang ingin membangun GitOps alur kerja kustom dengan menggunakan komponen modular.

Pengiriman progresif:

- Ketika strategi penerapan lanjutan seperti rilis kenari atau A/B pengujian adalah persyaratan inti.

Perbandingan fitur

Bidang	Argo CD	Fluks
Support untuk GitOps prinsip-prinsip inti	✓ Ya	✓ Ya
Arsitektur	End-to-end aplikasi untuk mengimplementasikan alur kerja Kubernetes GitOps	Menyediakan CRD dan pengontrol Kubernetes untuk GitOps
Pengaturan	Sederhana	Kompleks
Dukungan helm	✓ Ya	✓ Ya
Kustomisasi dukungan	✓ Ya	✓ Ya
GUI terintegrasi	CLI dan UI web berfitur lengkap	CLI dan antarmuka web ringan opsional
Dukungan RBAC	Kontrol granular	RBAC asli Kubernetes
Multi-tenancy dan dukungan multi-cluster	Dukungan yang sangat baik untuk multi-cluster	Dukungan yang sangat baik untuk multi-tenancy
Otentikasi masuk tunggal	✓ Ya	✓ Ya
Otomatisasi sinkronisasi	Kemampuan untuk menyinkronkan jendela	Kemampuan untuk mengatur interval rekonsiliasi
Sinkronisasi sebagian	✓ Ya	✗ Tidak
Proses rekonsiliasi	Mendukung sinkronisasi manual dan otomatis. Beberapa strategi berbeda tersedia.	Mendukung sinkronisasi manual dan otomatis.
Ekstensibilitas	Mendukung lugins kustom. Opsi kustomisasi terbatas.	Mendukung pengontrol kustom. Ekstensibilitas yang

Bidang	Argo CD	Fluks
		baik dan integrasi pihak ketiga.
Dukungan cummunity	Komunitas yang besar dan aktif.	Komunitas yang lebih kecil tetapi berkembang.
Skalabilitas	Skalabilitas yang baik, tetapi dibatasi oleh tingkat pengambilan data UI web. Analisis komunitas menyarankan dukungan untuk puluhan ribu aplikasi.	Panduan yang jelas untuk skalabilitas horizontal dan vertikal, hingga puluhan ribu aplikasi.

Praktik terbaik untuk memilih GitOps alat

Bagian ini memberikan pertimbangan, tip, dan praktik terbaik untuk memilih GitOps alat untuk kluster EKS Anda. Pilihan yang tepat tergantung pada konteks spesifik Anda, persyaratan, dan strategi jangka panjang. Seringkali bermanfaat untuk melakukan bukti konsep dengan pilihan utama Anda sebelum Anda membuat keputusan akhir.

Menilai kebutuhan dan kemampuan organisasi Anda:

- Pertimbangkan keahlian tim Anda saat ini dan kemauan untuk mempelajari alat baru.
- Evaluasi kompleksitas lingkungan Amazon EKS Anda. (Misalnya, apakah Anda menggunakan satu cluster atau beberapa cluster?)
- Tentukan persyaratan spesifik Anda untuk kepatuhan, keamanan, dan skalabilitas.

> **Praktik terbaik**
>
> Buat dokumen persyaratan terperinci yang menguraikan fitur yang diperlukan dan kemampuan yang berguna, tetapi tidak diperlukan.

Evaluasi kematangan dan adopsi alat:

- Teliti kematangan GitOps alat potensial dan tingkat adopsi mereka di industri.
- Cari alat yang memiliki rekam jejak yang terbukti di lingkungan Amazon EKS.

> **Praktik terbaik**
>
> Prioritaskan alat yang telah diadopsi secara luas dan memiliki kehadiran yang kuat di jaringan Cloud Native Computing Foundation (CNCF).

Pertimbangkan integrasi dengan toolchain Anda yang ada:

- Nilai seberapa baik GitOps alat ini terintegrasi dengan CI/CD pipeline Anda saat ini, solusi pemantauan, dan alat operasional lainnya.
- Cari integrasi asli dengan Layanan AWS seperti IAM, Amazon ECR, dan. CloudWatch

> **Praktik terbaik**
>
> Buat bukti konsep untuk menguji kemampuan integrasi sebelum Anda membuat keputusan akhir.

Evaluasi fitur keamanan:

- Prioritaskan alat yang memiliki kemampuan kontrol akses berbasis peran (RBAC) yang kuat dan terintegrasi dengan baik dengan IAM.
- Cari fitur yang mendukung manajemen rahasia yang aman dan penegakan kebijakan.

> **Praktik terbaik**
>
> Pilih alat yang mendukung praktik keamanan GitOps berbasis, termasuk kebijakan sebagai kode dan pemeriksaan kepatuhan otomatis.

Menilai skalabilitas dan kinerja:

- Pertimbangkan bagaimana alat bekerja dengan sejumlah besar aplikasi dan cluster.
- Mengevaluasi dampaknya terhadap kinerja cluster dan konsumsi sumber daya.

> **Praktik terbaik**
>
> Lakukan pengujian kinerja dengan beban kerja yang mirip dengan lingkungan produksi Anda untuk memastikan alat tersebut dapat menangani timbangan Anda.

Pertimbangkan dukungan multi-cluster dan multi-lingkungan:

- Jika Anda memiliki, atau berencana untuk memiliki, beberapa kluster EKS, prioritaskan alat yang memiliki kemampuan manajemen multi-cluster yang kuat.
- Cari fitur yang mendukung penerapan yang konsisten di berbagai lingkungan (seperti pengembangan, pementasan, dan produksi).

> **Praktik terbaik**
>
> Pilih alat yang memungkinkan pengelolaan terpusat dari beberapa cluster sambil mempertahankan konfigurasi khusus lingkungan.

Mengevaluasi kemampuan observabilitas dan pemantauan:

- Cari alat yang memberikan visibilitas yang jelas ke status penerapan dan kesehatan klaster Anda.
- Pertimbangkan seberapa baik alat ini terintegrasi dengan solusi pemantauan dan pencatatan yang ada.

> **Praktik terbaik**
>
> Prioritaskan alat yang menawarkan dasbor yang dapat disesuaikan dan mekanisme peringatan untuk deteksi masalah proaktif.

Menilai kurva pembelajaran dan dokumentasi:

- Mengevaluasi kualitas dan kelengkapan dokumentasi alat.
- Pertimbangkan ketersediaan sumber daya pelatihan dan dukungan masyarakat.

> **Praktik terbaik**
>
> Pilih alat yang memiliki dokumentasi yang terpelihara dengan baik, forum komunitas aktif, dan program pelatihan atau sertifikasi resmi.

Pertimbangkan biaya dan pemanfaatan sumber daya:

- Mengevaluasi biaya langsung (seperti lisensi dan dukungan) dan biaya tidak langsung (seperti biaya overhead operasional dan pelatihan) untuk mengadopsi alat.
- Menilai efisiensi alat dalam hal konsumsi sumber daya komputasi dan penyimpanan.

> **Praktik terbaik**
>
> Lakukan analisis biaya kepemilikan total (TCO) yang mencakup biaya jangka pendek dan jangka panjang.

Mengevaluasi fleksibilitas dan opsi penyesuaian:

- Cari alat yang memungkinkan Anda menyesuaikan alur kerja agar sesuai dengan kebutuhan spesifik Anda.
- Pertimbangkan ekstensibilitas alat melalui plugin atau. APIs

> **Praktik terbaik**
>
> Pilih alat yang menyeimbangkan fungsionalitas default dengan kemampuan untuk menyesuaikan kebutuhan unik Anda.

Menilai pengiriman berkelanjutan dan kemampuan penyebaran progresif:

- Cari alat yang mendukung strategi penerapan lanjutan seperti rilis dan blue/green penerapan kenari.
- Mengevaluasi kemudahan menerapkan dan mengelola strategi ini.

> **Praktik terbaik**
>
> Prioritaskan alat yang menawarkan dukungan bawaan untuk pola pengiriman progresif untuk meminimalkan risiko dalam penerapan Anda.

Pertimbangkan penguncian vendor dan portabilitas:

- Nilai ketergantungan alat pada penyedia atau teknologi cloud tertentu.
- Pertimbangkan kemudahan migrasi ke alat yang berbeda di masa depan jika diperlukan.

> **Praktik terbaik**
>
> Mendukung alat yang menggunakan standar terbuka dan menyediakan kemampuan ekspor untuk GitOps konfigurasi Anda.

Evaluasi dukungan dan ekstensi komunitas:

- Lihatlah ukuran dan aktivitas komunitas pengguna.
- Menilai ketersediaan integrasi dan plugin pihak ketiga.

> **Praktik terbaik**
>
> Bergabunglah dengan forum komunitas atau grup pengguna untuk mendapatkan pengalaman langsung dari pengguna lain sebelum Anda membuat keputusan.

Pertimbangkan persyaratan kepatuhan dan audit:

- Evaluasi seberapa baik alat ini mendukung kebutuhan kepatuhan Anda, termasuk jejak audit dan pelaporan.
- Cari fitur yang membantu mempertahankan dan menunjukkan kepatuhan.

> **Praktik terbaik**
>
> Pilih alat yang menyediakan log audit komprehensif dan mendukung pembuatan laporan kepatuhan.

Menilai kemampuan rollback dan pemulihan bencana:

- Mengevaluasi kemudahan dan keandalan mekanisme rollback.
- Pertimbangkan bagaimana alat ini mendukung skenario pemulihan bencana.

> **Praktik terbaik**
>
> Uji proses rollback dan pemulihan secara menyeluruh sebagai bagian dari evaluasi Anda.

Pertanyaan yang Sering Diajukan

T: Apa GitOps alat paling populer untuk Amazon EKS?

A: GitOps Alat paling populer untuk Amazon EKS termasuk Argo CD, Flux, Jenkins X, dan CI/CD. GitLab Setiap alat memiliki kekuatan, tetapi Argo CD dan Flux sangat dihormati karena pendekatan asli Kubernetes dan dukungan komunitas yang kuat.

T: Bagaimana cara GitOps meningkatkan manajemen cluster EKS?

J: GitOps meningkatkan manajemen klaster EKS dengan menyediakan kontrol versi untuk infrastruktur, penerapan otomatis, peningkatan keamanan melalui konfigurasi deklaratif, rollback yang lebih mudah, dan auditabilitas yang lebih baik. Ini juga meningkatkan kolaborasi dan mengurangi kesalahan manusia dalam penerapan.

T: Fitur utama apa yang harus saya cari di GitOps alat untuk Amazon EKS?

J: Fitur utama yang harus dicari meliputi: integrasi Amazon EKS yang mulus, RBAC yang kuat, dukungan multi-cluster, fitur observabilitas, dukungan untuk strategi pengiriman progresif, skalabilitas, dan integrasi dengan Layanan AWS seperti IAM dan Amazon ECR.

T: Bagaimana cara memastikan keamanan saat menerapkan GitOps di Amazon EKS?

J: Untuk memastikan keamanan, pilih alat yang memiliki integrasi RBAC yang kuat dengan IAM, manajemen rahasia yang aman, dukungan untuk repositori Git terenkripsi, dan kemampuan untuk menerapkan kebijakan keamanan sebagai kode. Juga, verifikasi bahwa alat ini menyediakan log audit yang komprehensif.

T: Dapatkah GitOps alat menangani lingkungan Amazon EKS multi-cluster?

J: Ya, GitOps alat seperti Argo CD dan Flux memiliki kemampuan manajemen multi-cluster yang kuat. Mereka memungkinkan Anda untuk mengelola beberapa kluster EKS dari satu bidang kontrol, yang memastikan konsistensi di seluruh lingkungan.

T: Bagaimana GitOps alat terintegrasi dengan pipeline CI/CD yang ada?

A: GitOps alat biasanya terintegrasi dengan pipa CI/CD yang ada dengan bertindak sebagai tahap penyebaran pipa. Mereka dapat dipicu oleh alat CI ketika perubahan didorong ke repositori Git, dan mereka mengotomatiskan proses penerapan ke kluster EKS.

T: Apa tantangan penerapan GitOps di Amazon EKS?

J: Tantangan umum termasuk mengelola rahasia dengan aman, memastikan kontrol akses yang tepat, menangani aplikasi stateful, mengelola penyimpangan antara Git dan status klaster, dan mengadaptasi alur kerja tim ke model. GitOps

T: Bagaimana GitOps alat menangani rollback di Amazon EKS?

A: GitOps alat biasanya menangani rollback dengan mengembalikan ke komit sebelumnya di repositori Git. Ini secara otomatis memicu penyebaran status baik yang diketahui sebelumnya, yang menghasilkan kemunduran yang cepat dan andal.

T: Dapatkah GitOps alat mengelola add-on Amazon EKS dan AWS sumber daya lainnya?

J: Banyak GitOps alat dapat mengelola add-on Amazon EKS dan beberapa AWS sumber daya, terutama jika digabungkan dengan alat IAc seperti Terraform atau. CloudFormation Namun, tingkat kemampuan ini dapat bervariasi; lihat bagian GitOps alat untuk informasi spesifik tentang setiap alat.

T: Bagaimana GitOps alat mendukung persyaratan kepatuhan di Amazon EKS?

A: GitOps alat mendukung kepatuhan dengan menyediakan jejak audit yang jelas dari semua perubahan, menegakkan proses persetujuan, menerapkan kebijakan sebagai kode untuk pemeriksaan kepatuhan otomatis, dan menawarkan fitur pencatatan dan pelaporan terperinci.

T: Apa kurva pembelajaran untuk diterapkan GitOps di Amazon EKS?

J: Kurva belajar dapat bervariasi tergantung pada alat dan pengetahuan tim Anda yang ada. Umumnya, tim yang akrab dengan Git, Kubernetes, dan Amazon EKS akan beradaptasi lebih cepat daripada yang lain. Alat paling populer menawarkan dokumentasi dan sumber daya pelatihan yang ekstensif untuk memudahkan adopsi.

T: Bagaimana GitOps alat menangani manajemen rahasia di Amazon EKS?

A: GitOps alat biasanya terintegrasi dengan solusi manajemen rahasia eksternal seperti AWS Secrets Manager atau HashiCorp Vault. Beberapa alat juga menawarkan enkripsi bawaan untuk rahasia yang disimpan di repositori Git.

T: Dapatkah GitOps alat bekerja dengan aplikasi stateless dan stateful di Amazon EKS?

J: Ya, GitOps alat dapat bekerja dengan aplikasi stateless dan stateful. Namun, mengelola aplikasi stateful seringkali memerlukan pertimbangan tambahan, seperti menangani volume persisten dan memastikan konsistensi data selama pembaruan.

T: Bagaimana GitOps alat mendukung kenari atau blue/green penerapan di Amazon EKS?

J: Banyak GitOps alat menawarkan dukungan bawaan untuk strategi penerapan lanjutan. Mereka dapat mengelola peluncuran bertahap versi baru, memantau masalah, dan secara otomatis memutar kembali jika masalah terdeteksi. Semua operasi ini didefinisikan sebagai kode dalam repositori Git.

T: Apa perbedaan antara menggunakan GitOps alat dan menggunakan **`kubectl apply`** dengan CI/CD pipa?

J: GitOps alat menawarkan keunggulan dibandingkan `kubectl apply` perintah sederhana, termasuk deteksi dan rekonsiliasi drift otomatis, peningkatan keamanan melalui penerapan berbasis tarik, auditabilitas yang lebih baik, dan strategi penyebaran yang lebih canggih. Mereka juga memberikan pendekatan yang lebih komprehensif untuk mengelola seluruh status cluster.

Sumber daya

Sumber daya berikut menyediakan dokumentasi resmi, panduan praktis, studi kasus, dan analisis mendalam yang dapat membantu Anda membuat keputusan berdasarkan informasi saat memilih GitOps alat untuk kluster EKS Anda. Mereka mencakup berbagai aspek GitOps, termasuk strategi implementasi, praktik terbaik, perbandingan antara alat yang berbeda, dan pengalaman dunia nyata.

AWS sumber daya:

- Dokumentasi Amazon EKS
- Mengotomatiskan Amazon EKS dengan GitOps (posting AWS blog)
- Pengantar GitOps tentang EKS dengan Weaveworks (lokakarya)AWS
- Laboratorium fluks (bengkel Amazon EKS)
- Laboratorium CD Argo (bengkel Amazon EKS)

GitOps dan dokumentasi alat:

- GitOps Praktik Terbaik untuk Penerapan Berkelanjutan dan Keamanan Progresif (DevOpswebinar.com sesuai permintaan)
- Dokumentasi Kubernetes
- Dokumentasi Argo CD
- Dokumentasi fluks
- Dokumentasi menenun GitOps
- Dokumentasi Jenkins X
- GitLab CI/CD dokumentasi
- Dokumentasi Spinnaker
- Dokumentasi Armada Peternak
- Dokumentasi Codefresh

Riwayat dokumen

Tabel berikut menjelaskan perubahan signifikan pada panduan ini. Jika Anda ingin diberi tahu tentang pembaruan masa depan, Anda dapat berlangganan umpan RSS.

Perubahan	Deskripsi	Tanggal
Publikasi awal	—	April 30, 2025

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.