Memilih GitOps alat yang tepat untuk cluster Amazon EKS Anda

AWS Bimbingan Preskriptif

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

Table of Contents

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-asli	Argo CD dirancang khusus untuk lingkungan Kubernetes. Ini memanfaatkan sifat deklaratif dan sumber daya kustom di Kubernetes untuk mengelola aplikasi.
Penyembuhan diri 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.
Dukungan multi-cluster dan 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 cara asli Kubernetes untuk mendefinisikan apa yang harus diterapkan dan bagaimana caranya.
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.

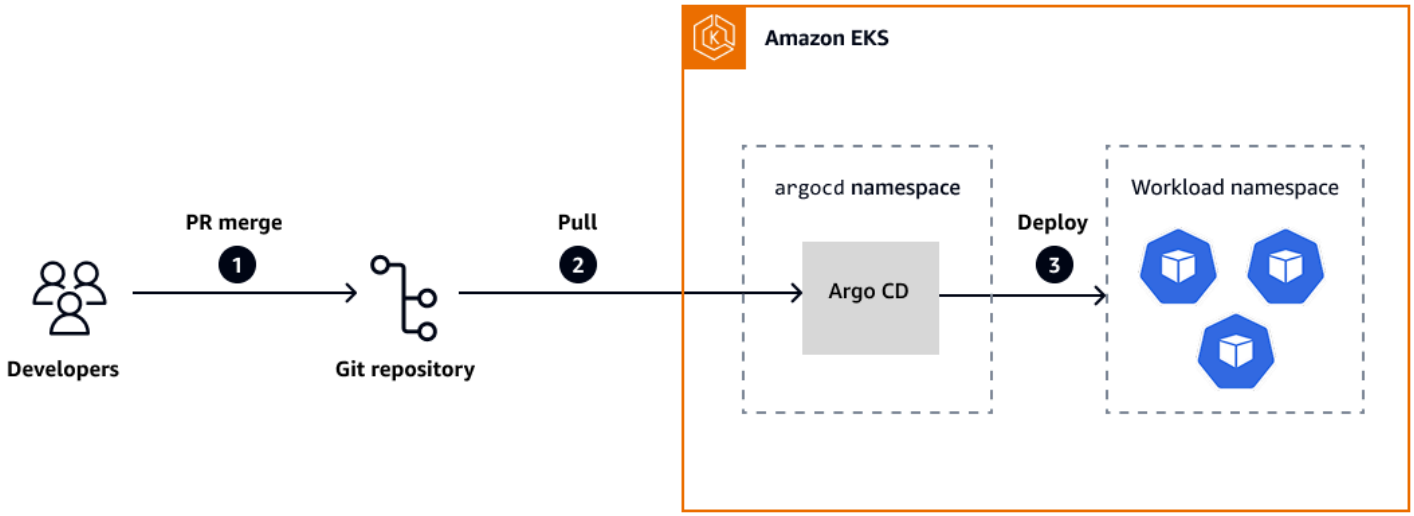
Dengan mengikuti GitOps prinsip-prinsip ini, Argo CD menyediakan cara yang kuat, terukur, dan aman untuk mengelola penerapan Kubernetes. Ini memastikan bahwa status operasional sistem Anda selalu sinkron dengan status yang diinginkan yang didefinisikan dalam repositori Git Anda, dan mempromosikan konsistensi, 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 CD GitOps -driven 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 kluster 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 dengan itu. 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-asli	Flux dibangun sebagai satu set pengendal i Kubernetes dan sumber daya kustom. Ia menggunakan mekanisme ekstensi di Kubernetes untuk menyediakan kemampuan. GitOps
Model penyebaran berbasis tarik	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 setiap penyimpangan yang terdeteksi di antara status ini.
Multi-penghunian	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.
Arsitektur berbasis peristiwa	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.
Sinkronisasi lintas-cluster	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.

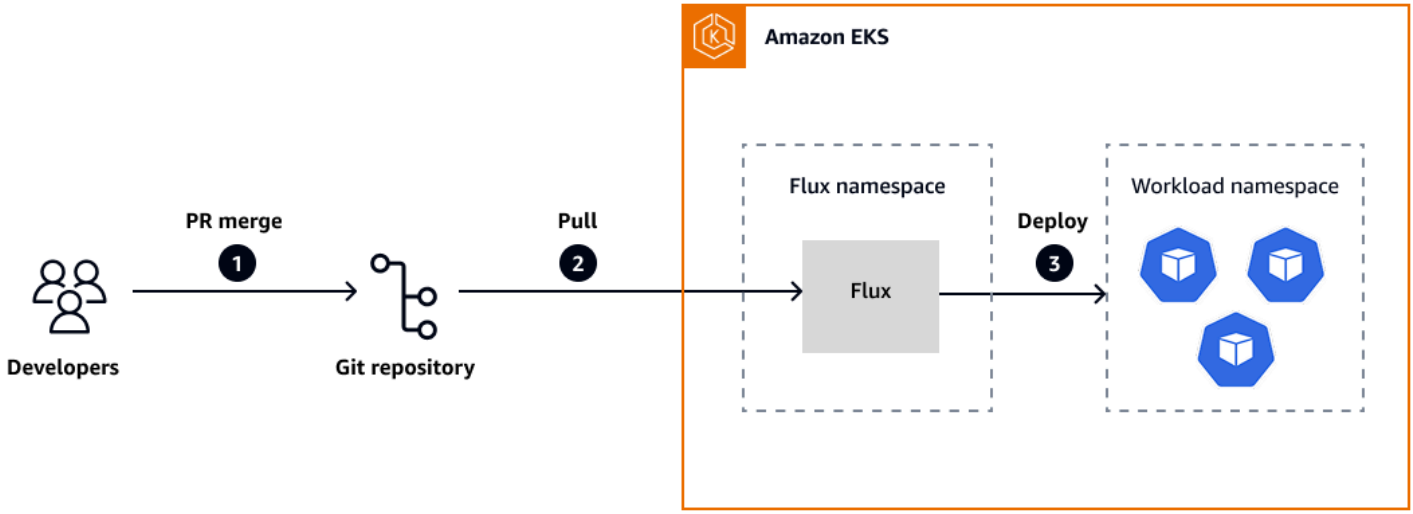
Dengan menerapkan GitOps prinsip-prinsip ini, Flux menyediakan sistem yang kuat dan fleksibel untuk mengelola klaster dan aplikasi Kubernetes. Ini memastikan bahwa infrastruktur dan aplikasi Anda selalu sinkron dengan repositori Git Anda, dan memberikan konsistensi, keandalan, dan kemudahan pengelolaan di lingkungan Kubernetes yang kompleks. Pendekatan Kubernetes-native tool 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 CD GitOps -driven 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 dengan itu.

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.
Arsitektur asli Kubernetes	Weave GitOps dibangun sebagai satu set pengendali 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.
Manajemen multi-cluster	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.
Platform pengembang swalayan	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.

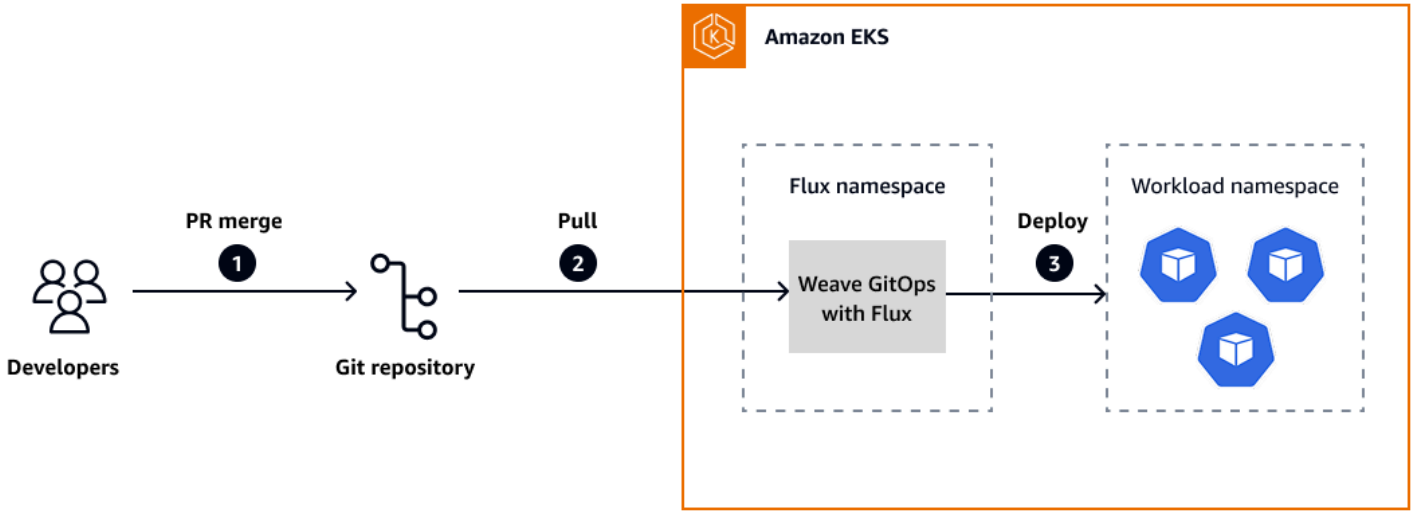
Bidang	Kemampuan alat
Pipa pengiriman berkelanjutan	Ini terintegrasi dengan CI/CD sistem untuk membuat jaringan pipa end-to-end pengiriman.
Audit dan kepatuhan	Weave DevOps menyediakan jejak audit lengkap dari semua perubahan dan tindakan. Ini membantu Anda memenuhi persyaratan kepatuhan melalui kontrol 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 alur kerja berbasis GIT.
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 CD GitOps -driven yang menggunakan Weave GitOps dalam cluster EKS. Untuk informasi rinci, 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
Alur kerja Git-sentris	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-asli	Jenkins X dibuat khusus untuk lingkungan Kubernetes. Ini menggunakan sumber daya Kubernetes dan definisi sumber daya kustom (). CRDs
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,, CloudFormation 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 observabilitas.
Dukungan multi-cloud	Jenkins X dirancang untuk bekerja di berbagai penyedia cloud dan lingkungan lokal.

Bidang	Kemampuan alat
Kolaborasi tim	Alat ini mendorong kolaborasi melalui alur kerja berbasis Git 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.

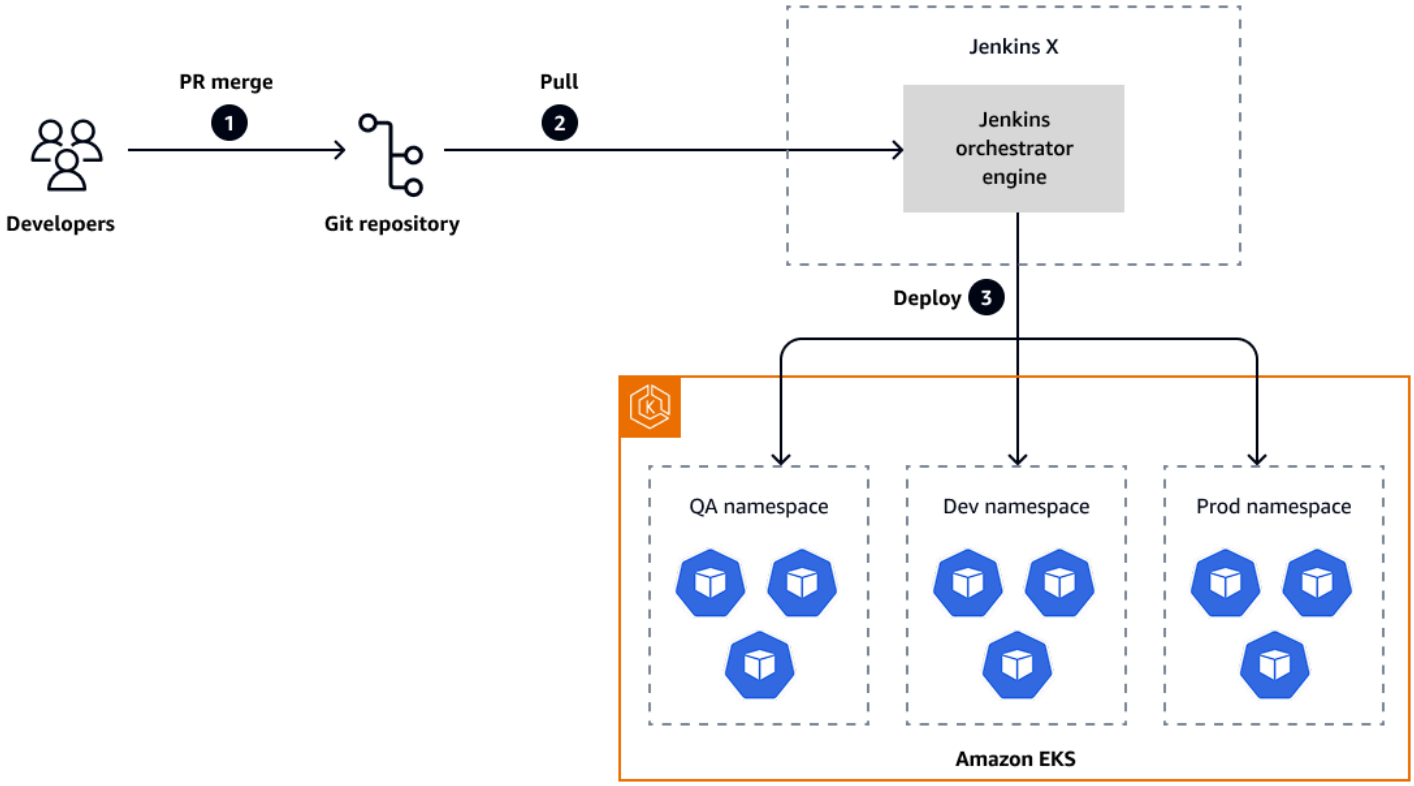
Jenkins X menerapkan GitOps prinsip-prinsip ini untuk menciptakan solusi CI/CD yang komprehensif untuk Kubernetes. Ini bertujuan untuk mengotomatiskan dan merampingkan seluruh proses pengiriman perangkat lunak, dari komit kode hingga penyebaran produksi, sambil mengikuti praktik. GitOps Dengan demikian, 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 all-in-one solusi 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 CD GitOps -driven 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 is an integrated part of the GitLab platform that provides continuous integration, delivery, and deployment capabilities. Although GitLab CI/CDbukan hanya GitOps alat, Anda dapat mengonfigurasinya untuk mengimplementasikan GitOps 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 Pipeline CI/CD 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 pipeline ketika perubahan didorong ke repositori. Pipa ini dapat mencakup tahapan untuk membangun , menguji, dan menyebarkan aplikasi.
Integrasi Kubernetes	GitLab CI/CD menyediakan integrasi Kubernete s asli dan mendukung penerapan GitOps - style ke klaster Kubernetes. Ini dapat secara otomatis membuat dan mengelola sumber daya Kubernetes berdasarkan konfigurasi di Git.

Bidang	Kemampuan alat
Pengelolaan 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 berkelanjutan	GitLab CI/CD dapat dikonfigurasi untuk secara otomatis menerapkan perubahan pada klaster Kubernetes ketika perubahan digabungkan ke cabang tertentu.
IAc	GitLab CI/CD mendukung integrasi dengan alat-alat seperti Terraform dan CloudForm ation 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 includes built-in security scanning tools that can be integrated into the CI/CDpipe line untuk menegakkan keamanan sebagai bagian dari GitOps alur kerja.

Bidang	Kemampuan alat
Registri kontainer	GitLab CI/CD menyertakan registri kontainer bawaan untuk integrasi manajemen gambar kontainer yang mulus dalam alur kerja. GitOps
Otomatis DevOps	DevOps Fitur Auto di GitLab CI/CD can automatically configure CI/CD pipeline yang mengikuti GitOps prinsip penerapan Kubernete s.
Alur kerja persetujuan	GitLab CI/CD mendukung proses persetujuan untuk penerapan, yang menyediakan promosi terkontrol antar lingkungan.
Manajemen rahasia	GitLab CI/CD provides features to securely manage and use secrets within CI/CDjaringan pipa.
Pembuatan versi dan rilis	GitLab CI/CD supports automatic versioning and release management as part of the CI/ CDproses.
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
Jaringan pipa multi-proyek	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 GitLab CI/CD is not exclusively designed for GitOps, it can be used effectively to implement GitOps practices, especially for teams that already use GitLab as their primary development platform. Its integrated approach, which combines source control, CI/CD, dan manajemen Kubernetes, menjadikannya alat yang ampuh untuk mengimplementasikan alur kerja. GitOps

Perbedaan utama antara GitLab CI/CD and dedicated GitOps tools such as Argo CD or Flux is that GitLab provides a more comprehensive platform that includes source control management, issue tracking, and other development tools along with its CI/CD kemampuan. Ini membuatnya sangat cocok untuk tim yang membutuhkan all-in-one solusi yang dapat menerapkan GitOps praktik dalam sistem pengembangan yang lebih luas.

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

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 YAMG. Definisi pipeline ini dapat dikontrol versi di repositori Git, selaras dengan praktik. GitOps
IAc	Spinnaker mendukung definisi konfigurasi infrastruktur dan penerapan sebagai kode.

Bidang	Kemampuan alat
	Definisi ini dapat disimpan dalam repositor i Git, dan dapat berfungsi sebagai sumber kebenaran tunggal.
Penerapan multi-cloud	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 manajemen gaya 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
Kontrol akses berbasis peran (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.

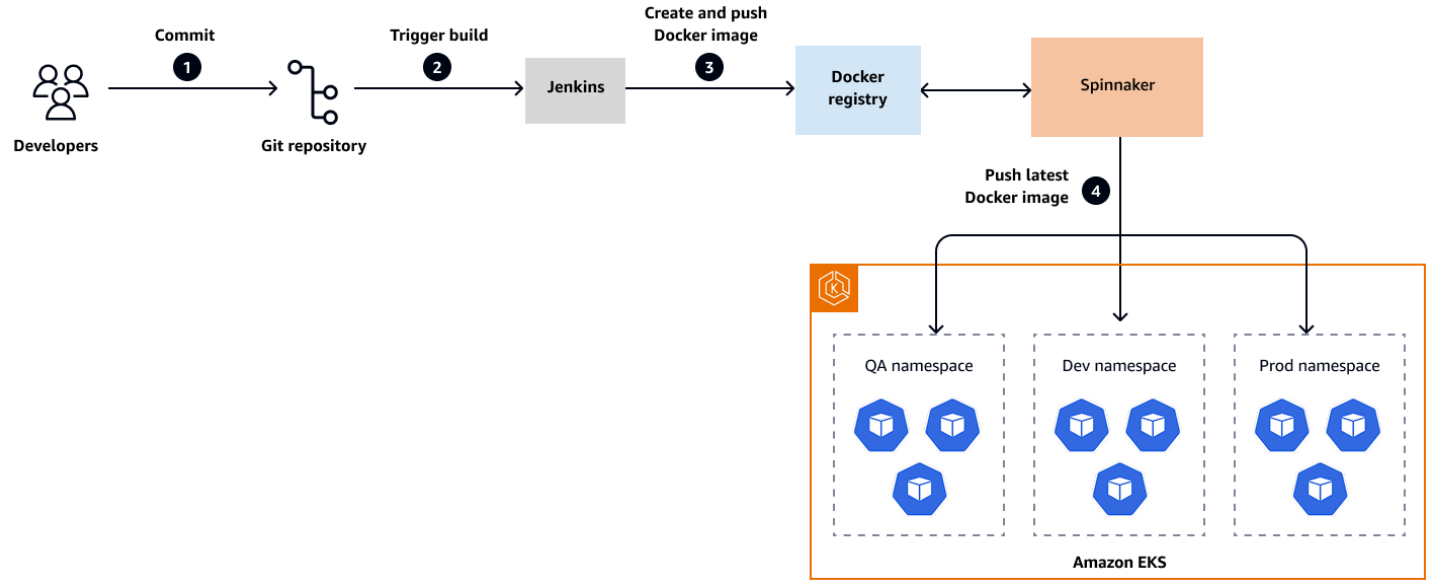
Meskipun Spinnaker tidak terutama dipasarkan sebagai GitOps alat, fleksibilitas dan set fitur yang kuat membuatnya mampu mengimplementasikan GitOps alur kerja, terutama di lingkungan multi-cloud yang kompleks. Perbedaan utama antara Spinnaker dan GitOps alat khusus seperti Argo CD atau Flux adalah bahwa 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 CD GitOps -driven 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 khusus Fleet.
Sinkronisasi otomatis	Armada terus memantau repositori Git untuk perubahan. Secara otomatis menerapka n perubahan pada kluster target ketika mendeteksi perbedaan antara status Git dan status cluster.
Manajemen multi-cluster	Armada dirancang khusus untuk mengelola penyebaran di beberapa klaster Kubernetes. Ini dapat menangani ribuan cluster dari satu bidang kontrol.
Arsitektur asli Kubernetes	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 (CRDs)	Armada menggunakan sumber daya khusus seperti GitRepo dan Bundle untuk menentukan penerapan. Ini CRDs menyediakan cara asli Kubernetes untuk mendefinisikan alur kerja. GitOps
Keamanan dan RBAC	Armada terintegrasi dengan Kubernetes RBAC untuk kontrol akses. Ini mendukung manajemen yang aman dari informasi sensitif dan kredensil.

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 berbasis GIS yang dikendalikan versi.

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 penerapan GitOps -style di beberapa cluster. Ini mendukung berbagai sumber daya Kubernetes dan definisi sumber daya kustom (). CRDs

Bidang	Kemampuan alat
Pengelolaan lingkungan	Anda dapat mendefinisikan dan mengelola beberapa lingkungan (seperti pengembangan, pementasan, dan produksi) sebagai kode. Codefresh mendukung promosi antar lingkunga n dengan menggunakan GitOps praktik.
Integrasi Argo CD	Codefresh terintegrasi dengan Argo CD untuk kemampuan yang ditingkatkan. 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).
Penerapan multi-cloud dan hybrid	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 terkenal GitOps karena integrasi kemampuan CI/CD dengan praktik. GitOps 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 all-in-one solusi 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 mengimplementasikan 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 step-by-step proses 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.
Dukungan multi-cloud dan 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 menggunak an CrossGuard Pulumi. Ini memungkin kan manajemen kepatuhan dan kebijakan keamanan GitOps yang dikontrol versi dan bergaya.
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.

Bidang	Kemampuan alat
Pratinjau dan rencanakan	Pulumi menawarkan kemampuan untuk melihat pratinjau perubahan sebelum menerapka nnya. Ini mendukung GitOps prinsip perubahan infrastruktur yang aman dan dapat diprediksi.
Rollback dan sejarah	Pulumi mempertahankan riwayat penyebara n 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 infrastru ktur.
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 observabi litas GitOps praktik.
Integrasi dengan alat lain	Pulumi dapat berintegrasi dengan berbagai alat di cloud. Fleksibilitas ini memungkinkan GitOps alur kerja yang komprehensif.

Bidang	Kemampuan alat
Pengelolaan 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.
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 layanan berbasis API 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 CI/CD tradisional ke GitOps karena UI intuitifnya.
- Flux sering disukai di lingkungan cloud-native di mana alur kerja berbasis CLI 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.

Pendekatan aplikasi-sentris:

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

Manajemen multi-cluster:

- 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-penyewaan:

- 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 penerapan berbasis HELM yang 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 Kubernetes CRDs dan pengontrol 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
Dukungan multi-tenancy dan 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

AWS Glosarium Panduan Preskriptif

Berikut ini adalah istilah yang umum digunakan dalam strategi, panduan, dan pola yang disediakan oleh Panduan AWS Preskriptif. Untuk menyarankan entri, silakan gunakan tautan Berikan umpan balik di akhir glosarium.

Nomor

7 Rs

Tujuh strategi migrasi umum untuk memindahkan aplikasi ke cloud. Strategi ini dibangun di atas 5 Rs yang diidentifikasi Gartner pada tahun 2011 dan terdiri dari yang berikut:

- Refactor/re-architect — Pindahkan aplikasi dan modifikasi arsitekturnya dengan memanfaatkan sepenuhnya fitur cloud-native untuk meningkatkan kelincahan, kinerja, dan skalabilitas. Ini biasanya melibatkan porting sistem operasi dan database. Contoh: Migrasikan database Oracle lokal Anda ke Amazon Aurora Edition. PostgreSQL-Compatible
- Replatform (angkat dan bentuk ulang) — Pindahkan aplikasi ke cloud, dan perkenalkan beberapa tingkat pengoptimalan untuk memanfaatkan kemampuan cloud. Contoh: Memigrasikan database Oracle lokal Anda ke Amazon Relational Database Service (Amazon RDS) untuk Oracle di. AWS Cloud
- Pembelian kembali (drop and shop) - Beralih ke produk yang berbeda, biasanya dengan beralih dari lisensi tradisional ke model SaaS. Contoh: Migrasikan sistem manajemen hubungan pelanggan (CRM) Anda ke. Salesforce.com
- Rehost (lift dan shift) — Pindahkan aplikasi ke cloud tanpa membuat perubahan apa pun untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan database Oracle lokal Anda ke Oracle pada instans EC2 di. AWS Cloud
- Relokasi (hypervisor-level lift and shift) — Pindahkan infrastruktur ke cloud tanpa membeli perangkat keras baru, menulis ulang aplikasi, atau memodifikasi operasi yang ada. Anda memigrasikan server dari platform lokal ke layanan cloud untuk platform yang sama. Contoh: Migrasikan Microsoft Hyper-V aplikasi ke AWS.
- Pertahankan (kunjungi kembali) - Simpan aplikasi di lingkungan sumber Anda. Ini mungkin termasuk aplikasi yang memerlukan refactoring besar, dan Anda ingin menunda pekerjaan itu sampai nanti, dan aplikasi lama yang ingin Anda pertahankan, karena tidak ada pembenaran bisnis untuk memigrasikannya.

- Pensiun — Menonaktifkan atau menghapus aplikasi yang tidak lagi diperlukan di lingkungan sumber Anda.

A

A2A () Agent-to-Agent

Protokol stateful untuk kolaborasi agen-ke-agen yang mendukung delegasi tugas dan transfer negara.

ABAC

Lihat kontrol akses berbasis atribut.

layanan abstrak

Lihat layanan terkelola.

ASAM

Lihat atomisitas, konsistensi, isolasi, daya tahan.

migrasi aktif-aktif

Metode migrasi database di mana basis data sumber dan target tetap sinkron (dengan menggunakan alat replikasi dua arah atau operasi penulisan ganda), dan kedua database menangani transaksi dari menghubungkan aplikasi selama migrasi. Metode ini mendukung migrasi dalam batch kecil yang terkontrol alih-alih memerlukan pemotongan satu kali. Ini lebih fleksibel tetapi membutuhkan lebih banyak pekerjaan daripada migrasi aktif-pasif.

migrasi aktif-pasif

Metode migrasi database di mana database sumber dan target disimpan dalam sinkron, tetapi hanya database sumber yang menangani transaksi dari menghubungkan aplikasi sementara data direplikasi ke database target. Basis data target tidak menerima transaksi apa pun selama migrasi.

Agen

Sistem AI yang dapat secara mandiri bernalar, merencanakan, dan mengambil tindakan menggunakan alat untuk mencapai tujuan.

Agen Ops

Praktik operasional untuk membangun, menguji, menyebarkan, dan menjalankan agen AI dalam produksi dalam skala besar.

fungsi agregat

Fungsi SQL yang beroperasi pada sekelompok baris dan menghitung nilai pengembalian tunggal untuk grup. Contoh fungsi agregat meliputi `SUM` dan`MAX`.

AI

Lihat kecerdasan buatan.

AIOps

Lihat operasi kecerdasan buatan.

anonimisasi

Proses menghapus informasi pribadi secara permanen dalam kumpulan data. Anonimisasi dapat membantu melindungi privasi pribadi. Data anonim tidak lagi dianggap sebagai data pribadi.

anti-pola

Solusi yang sering digunakan untuk masalah berulang di mana solusinya kontra-produktif, tidak efektif, atau kurang efektif daripada alternatif.

kontrol aplikasi

Pendekatan keamanan yang memungkinkan penggunaan hanya aplikasi yang disetujui untuk membantu melindungi sistem dari malware.

portofolio aplikasi

Kumpulan informasi rinci tentang setiap aplikasi yang digunakan oleh organisasi, termasuk biaya untuk membangun dan memelihara aplikasi, dan nilai bisnisnya. Informasi ini adalah kunci untuk penemuan portofolio dan proses analisis dan membantu mengidentifikasi dan memprioritaskan aplikasi yang akan dimigrasi, dimodernisasi, dan dioptimalkan.

kecerdasan buatan (AI)

Bidang ilmu komputer yang didedikasikan untuk menggunakan teknologi komputasi untuk melakukan fungsi kognitif yang biasanya terkait dengan manusia, seperti belajar, memecahkan masalah, dan mengenali pola. Untuk informasi lebih lanjut, lihat Apa itu Kecerdasan Buatan?

operasi kecerdasan buatan (AIOps)

Proses menggunakan teknik pembelajaran mesin untuk memecahkan masalah operasional, mengurangi insiden operasional dan intervensi manusia, dan meningkatkan kualitas layanan. Untuk informasi selengkapnya tentang cara AIOps digunakan dalam strategi AWS migrasi, lihat panduan integrasi operasi.

enkripsi asimetris

Algoritma enkripsi yang menggunakan sepasang kunci, kunci publik untuk enkripsi dan kunci pribadi untuk dekripsi. Anda dapat berbagi kunci publik karena tidak digunakan untuk dekripsi, tetapi akses ke kunci pribadi harus sangat dibatasi.

atomisitas, konsistensi, isolasi, daya tahan (ACID)

Satu set properti perangkat lunak yang menjamin validitas data dan keandalan operasional database, bahkan dalam kasus kesalahan, kegagalan daya, atau masalah lainnya.

kontrol akses berbasis atribut (ABAC)

Praktik membuat izin berbutir halus berdasarkan atribut pengguna, seperti departemen, peran pekerjaan, dan nama tim. Untuk informasi selengkapnya, lihat ABAC untuk AWS dokumentasi AWS Identity and Access Management (IAM).

sumber data otoritatif

Lokasi di mana Anda menyimpan versi utama data, yang dianggap sebagai sumber informasi yang paling dapat diandalkan. Anda dapat menyalin data dari sumber data otoritatif ke lokasi lain untuk tujuan memproses atau memodifikasi data, seperti menganonimkan, menyunting, atau membuat nama samaran.

Zona Ketersediaan

Lokasi berbeda di dalam AWS Region yang terisolasi dari kegagalan di Availability Zone lainnya dan menyediakan konektivitas jaringan latensi rendah yang murah ke Availability Zone lainnya di Wilayah yang sama.

AWS Kerangka Adopsi Cloud (AWS CAF)

Kerangka pedoman dan praktik terbaik AWS untuk membantu organisasi mengembangkan rencana yang efisien dan efektif untuk bergerak dengan sukses ke cloud. AWS CAF mengatur panduan ke dalam enam area fokus yang disebut perspektif: bisnis, orang, tata kelola, platform, keamanan, dan operasi. Perspektif bisnis, orang, dan tata kelola fokus pada keterampilan dan proses bisnis; perspektif platform, keamanan, dan operasi fokus pada keterampilan dan proses teknis. Misalnya, perspektif masyarakat menargetkan pemangku kepentingan yang menangani

sumber daya manusia (SDM), fungsi kepegawaian, dan manajemen orang. Untuk perspektif ini, AWS CAF memberikan panduan untuk pengembangan, pelatihan, dan komunikasi orang untuk membantu mempersiapkan organisasi untuk adopsi cloud yang sukses. Untuk informasi lebih lanjut, lihat situs web AWS CAF dan whitepaper AWS CAF.

AWS Kerangka Kualifikasi Beban Kerja (AWS WQF)

Alat yang mengevaluasi beban kerja migrasi database, merekomendasikan strategi migrasi, dan memberikan perkiraan kerja. AWS WQF disertakan dengan AWS Schema Conversion Tool ()AWS SCT. Ini menganalisis skema database dan objek kode, kode aplikasi, dependensi, dan karakteristik kinerja, dan memberikan laporan penilaian.

B

bot buruk

Bot yang dimaksudkan untuk mengganggu atau menyebabkan kerugian bagi individu atau organisasi.

BCP

Lihat perencanaan kontinuitas bisnis.

grafik perilaku

Pandangan interaktif yang terpadu tentang perilaku dan interaksi sumber daya dari waktu ke waktu. Anda dapat menggunakan grafik perilaku dengan Amazon Detective untuk memeriksa upaya logon yang gagal, panggilan API yang mencurigakan, dan tindakan serupa. Untuk informasi selengkapnya, lihat Data dalam grafik perilaku di dokumentasi Detektif.

sistem big-endian

Sistem yang menyimpan byte paling signifikan terlebih dahulu. Lihat juga endianness.

klasifikasi biner

Sebuah proses yang memprediksi hasil biner (salah satu dari dua kelas yang mungkin). Misalnya, model ML Anda mungkin perlu memprediksi masalah seperti “Apakah email ini spam atau bukan spam?” atau “Apakah produk ini buku atau mobil?”

filter mekar

Struktur data probabilistik dan efisien memori yang digunakan untuk menguji apakah suatu elemen adalah anggota dari suatu himpunan.

blue/green penyebaran

Strategi penyebaran tempat Anda membuat dua lingkungan yang terpisah namun identik. Anda menjalankan versi aplikasi saat ini di satu lingkungan (biru) dan versi aplikasi baru di lingkungan lain (hijau). Strategi ini membantu Anda dengan cepat memutar kembali dengan dampak minimal.

bot

Aplikasi perangkat lunak yang menjalankan tugas otomatis melalui internet dan mensimulasikan aktivitas atau interaksi manusia. Beberapa bot berguna atau bermanfaat, seperti perayap web yang mengindeks informasi di internet. Beberapa bot lain, yang dikenal sebagai bot buruk, dimaksudkan untuk mengganggu atau membahayakan individu atau organisasi.

botnet

Jaringan bot yang terinfeksi oleh malware dan berada di bawah kendali satu pihak, yang dikenal sebagai bot herder atau operator bot. Botnet adalah mekanisme paling terkenal untuk skala bot dan dampaknya.

cabang

Area berisi repositori kode. Cabang pertama yang dibuat dalam repositori adalah cabang utama. Anda dapat membuat cabang baru dari cabang yang ada, dan Anda kemudian dapat mengembangkan fitur atau memperbaiki bug di cabang baru. Cabang yang Anda buat untuk membangun fitur biasanya disebut sebagai cabang fitur. Saat fitur siap dirilis, Anda menggabungkan cabang fitur kembali ke cabang utama. Untuk informasi selengkapnya, lihat Tentang cabang (GitHub dokumentasi).

akses break-glass

Dalam keadaan luar biasa dan melalui proses yang disetujui, cara cepat bagi pengguna untuk mendapatkan akses ke Akun AWS yang biasanya tidak memiliki izin untuk mengaksesnya. Untuk informasi lebih lanjut, lihat indikator Implementasikan prosedur break-glass dalam panduan. AWS Well-Architected

strategi brownfield

Infrastruktur yang ada di lingkungan Anda. Saat mengadopsi strategi brownfield untuk arsitektur sistem, Anda merancang arsitektur di sekitar kendala sistem dan infrastruktur saat ini. Jika Anda memperluas infrastruktur yang ada, Anda dapat memadukan strategi brownfield dan greenfield.

cache penyangga

Area memori tempat data yang paling sering diakses disimpan.

kemampuan bisnis

Apa yang dilakukan bisnis untuk menghasilkan nilai (misalnya, penjualan, layanan pelanggan, atau pemasaran). Arsitektur layanan mikro dan keputusan pengembangan dapat didorong oleh kemampuan bisnis. Untuk informasi selengkapnya, lihat bagian Terorganisir di sekitar kemampuan bisnis dari Menjalankan layanan mikro kontainer di whitepaper. AWS

perencanaan kelangsungan bisnis (BCP)

Rencana yang membahas dampak potensial dari peristiwa yang mengganggu, seperti migrasi skala besar, pada operasi dan memungkinkan bisnis untuk melanjutkan operasi dengan cepat.

C

KAFE

Lihat Kerangka Adopsi AWS Cloud.

penyebaran kenari

Rilis versi yang lambat dan bertahap untuk pengguna akhir. Ketika Anda yakin, Anda menyebarkan versi baru dan mengganti versi saat ini secara keseluruhan.

CCoE

Lihat Cloud Center of Excellence.

CDC

Lihat mengubah pengambilan data.

ubah pengambilan data (CDC)

Proses melacak perubahan ke sumber data, seperti tabel database, dan merekam metadata tentang perubahan tersebut. Anda dapat menggunakan CDC untuk berbagai tujuan, seperti mengaudit atau mereplikasi perubahan dalam sistem target untuk mempertahankan sinkronisasi.

rekayasa kekacauan

Sengaja memperkenalkan kegagalan atau peristiwa yang mengganggu untuk menguji ketahanan sistem. Anda dapat menggunakan AWS Fault Injection Service (AWS FIS) untuk melakukan eksperimen yang menekankan AWS beban kerja Anda dan mengevaluasi responsnya.

CI/CD

Lihat integrasi berkelanjutan dan pengiriman berkelanjutan.

klasifikasi

Proses kategorisasi yang membantu menghasilkan prediksi. Model ML untuk masalah klasifikasi memprediksi nilai diskrit. Nilai diskrit selalu berbeda satu sama lain. Misalnya, model mungkin perlu mengevaluasi apakah ada mobil dalam gambar atau tidak.

Pengembang Warga

Pengguna bisnis yang membuat aplikasi AI menggunakan platform tanpa code/low kode tanpa keterampilan teknis khusus.

Enkripsi sisi klien

Enkripsi data secara lokal, sebelum target Layanan AWS menerimanya.

Cloud Center of Excellence (CCoE)

Tim multi-disiplin yang mendorong upaya adopsi cloud di seluruh organisasi, termasuk mengembangkan praktik terbaik cloud, memobilisasi sumber daya, menetapkan jadwal migrasi, dan memimpin organisasi melalui transformasi skala besar. Untuk informasi selengkapnya, lihat posting CCoE di Blog Strategi AWS Cloud Perusahaan.

komputasi cloud

Teknologi cloud yang biasanya digunakan untuk penyimpanan data jarak jauh dan manajemen perangkat IoT. Cloud computing umumnya terhubung ke teknologi edge computing.

model operasi cloud

Dalam organisasi TI, model operasi yang digunakan untuk membangun, mematangkan, dan mengoptimalkan satu atau lebih lingkungan cloud. Untuk informasi selengkapnya, lihat Membangun Model Operasi Cloud Anda.

tahap adopsi cloud

Empat fase yang biasanya dilalui organisasi ketika mereka bermigrasi ke AWS Cloud:

- Proyek — Menjalankan beberapa proyek terkait cloud untuk bukti konsep dan tujuan pembelajaran
- Foundation — Melakukan investasi dasar untuk meningkatkan adopsi cloud Anda (misalnya, membuat landing zone, mendefinisikan CCoE, membuat model operasi)
- Migrasi — Migrasi aplikasi individual
- Re-invention — Mengoptimalkan produk dan layanan, dan berinovasi di cloud

Tahapan ini didefinisikan oleh Stephen Orban dalam posting blog The Journey Toward Cloud-First & the Stages of Adoption di blog Strategi AWS Cloud Perusahaan. Untuk informasi tentang bagaimana kaitannya dengan strategi AWS migrasi, lihat panduan kesiapan migrasi.

CMDB

Lihat database manajemen konfigurasi.

repositori kode

Lokasi di mana kode sumber dan aset lainnya, seperti dokumentasi, sampel, dan skrip, disimpan dan diperbarui melalui proses kontrol versi. Repositori cloud umum termasuk GitHub atau. Bitbucket Cloud Setiap versi kode disebut cabang. Dalam struktur layanan mikro, setiap repositori dikhususkan untuk satu bagian fungsionalitas. Satu CI/CD pipa dapat menggunakan beberapa repositori.

cache dingin

Cache buffer yang kosong, tidak terisi dengan baik, atau berisi data basi atau tidak relevan. Ini mempengaruhi kinerja karena instance database harus membaca dari memori utama atau disk, yang lebih lambat daripada membaca dari cache buffer.

data dingin

Data yang jarang diakses dan biasanya historis. Saat menanyakan jenis data ini, kueri lambat biasanya dapat diterima. Memindahkan data ini ke tingkat atau kelas penyimpanan yang berkinerja lebih rendah dan lebih murah dapat mengurangi biaya.

visi komputer (CV)

Bidang AI yang menggunakan pembelajaran mesin untuk menganalisis dan mengekstrak informasi dari format visual seperti gambar dan video digital. Misalnya, Amazon SageMaker AI menyediakan algoritma pemrosesan gambar untuk CV.

konfigurasi drift

Untuk beban kerja, konfigurasi berubah dari status yang diharapkan. Ini dapat menyebabkan beban kerja menjadi tidak patuh, dan biasanya bertahap dan tidak disengaja.

database manajemen konfigurasi (CMDB)

Repositori yang menyimpan dan mengelola informasi tentang database dan lingkungan TI, termasuk komponen perangkat keras dan perangkat lunak dan konfigurasinya. Anda biasanya menggunakan data dari CMDB dalam penemuan portofolio dan tahap analisis migrasi.

paket kesesuaian

Kumpulan AWS Config aturan dan tindakan remediasi yang dapat Anda kumpulkan untuk menyesuaikan kepatuhan dan pemeriksaan keamanan Anda. Anda dapat menerapkan paket kesesuaian sebagai entitas tunggal di Akun AWS dan Region, atau di seluruh organisasi, dengan menggunakan templat YAMM. Untuk informasi selengkapnya, lihat Paket kesesuaian dalam dokumentasi. AWS Config

integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD)

Proses mengotomatiskan sumber, membangun, menguji, pementasan, dan tahap produksi dari proses rilis perangkat lunak. CI/CD biasanya digambarkan sebagai pipa. CI/CD dapat membantu Anda mengotomatiskan proses, meningkatkan produktivitas, meningkatkan kualitas kode, dan memberikan lebih cepat. Untuk informasi lebih lanjut, lihat Manfaat pengiriman berkelanjutan. CD juga dapat berarti penerapan berkelanjutan. Untuk informasi selengkapnya, lihat Continuous Delivery vs Continuous Deployment.

CV

Lihat visi komputer.

D

data saat istirahat

Data yang stasioner di jaringan Anda, seperti data yang ada di penyimpanan.

klasifikasi data

Proses untuk mengidentifikasi dan mengkategorikan data dalam jaringan Anda berdasarkan kekritisan dan sensitivitasnya. Ini adalah komponen penting dari setiap strategi manajemen risiko keamanan siber karena membantu Anda menentukan perlindungan dan kontrol retensi yang tepat untuk data. Klasifikasi data adalah komponen pilar keamanan dalam AWS Well-Architected Framework. Untuk informasi selengkapnya, lihat Klasifikasi data.

penyimpangan data

Variasi yang berarti antara data produksi dan data yang digunakan untuk melatih model ML, atau perubahan yang berarti dalam data input dari waktu ke waktu. Penyimpangan data dapat mengurangi kualitas, akurasi, dan keadilan keseluruhan dalam prediksi model ML.

data dalam transit

Data yang aktif bergerak melalui jaringan Anda, seperti antara sumber daya jaringan.

jala data

Kerangka arsitektur yang menyediakan kepemilikan data terdistribusi dan terdesentralisasi dengan manajemen dan tata kelola terpusat.

minimalisasi data

Prinsip pengumpulan dan pemrosesan hanya data yang sangat diperlukan. Mempraktikkan minimalisasi data di dalamnya AWS Cloud dapat mengurangi risiko privasi, biaya, dan jejak karbon analitik Anda.

perimeter data

Satu set pagar pembatas pencegahan di AWS lingkungan Anda yang membantu memastikan bahwa hanya identitas tepercaya yang mengakses sumber daya tepercaya dari jaringan yang diharapkan. Untuk informasi selengkapnya, lihat Membangun perimeter data pada AWS.

prapemrosesan data

Untuk mengubah data mentah menjadi format yang mudah diuraikan oleh model ML Anda. Preprocessing data dapat berarti menghapus kolom atau baris tertentu dan menangani nilai yang hilang, tidak konsisten, atau duplikat.

asal data

Proses melacak asal dan riwayat data sepanjang siklus hidupnya, seperti bagaimana data dihasilkan, ditransmisikan, dan disimpan.

subjek data

Individu yang datanya dikumpulkan dan diproses.

gudang data

Sistem manajemen data yang mendukung intelijen bisnis, seperti analitik. Gudang data biasanya berisi sejumlah besar data historis, dan biasanya digunakan untuk kueri dan analisis.

bahasa definisi database (DDL)

Pernyataan atau perintah untuk membuat atau memodifikasi struktur tabel dan objek dalam database.

bahasa manipulasi basis data (DHTML)

Pernyataan atau perintah untuk memodifikasi (memasukkan, memperbarui, dan menghapus) informasi dalam database.

DDL

Lihat bahasa definisi database.

ansambel yang dalam

Untuk menggabungkan beberapa model pembelajaran mendalam untuk prediksi. Anda dapat menggunakan ansambel dalam untuk mendapatkan prediksi yang lebih akurat atau untuk memperkirakan ketidakpastian dalam prediksi.

pembelajaran mendalam

Subbidang ML yang menggunakan beberapa lapisan jaringan saraf tiruan untuk mengidentifikasi pemetaan antara data input dan variabel target yang diinginkan.

pertahanan-mendalam

Pendekatan keamanan informasi di mana serangkaian mekanisme dan kontrol keamanan dilapisi dengan cermat di seluruh jaringan komputer untuk melindungi kerahasiaan, integritas, dan ketersediaan jaringan dan data di dalamnya. Saat Anda mengadopsi strategi ini AWS, Anda menambahkan beberapa kontrol pada lapisan AWS Organizations struktur yang berbeda untuk membantu mengamankan sumber daya. Misalnya, pendekatan defense-in-depth mungkin menggabungkan otentikasi multi-faktor, segmentasi jaringan, dan enkripsi.

administrator yang didelegasikan

Di AWS Organizations, layanan yang kompatibel dapat mendaftarkan akun AWS anggota untuk mengelola akun organisasi dan mengelola izin untuk layanan tersebut. Akun ini disebut administrator yang didelegasikan untuk layanan itu. Untuk informasi selengkapnya dan daftar layanan yang kompatibel, lihat Layanan yang berfungsi dengan AWS Organizations AWS Organizations dokumentasi.

deployment

Proses pembuatan aplikasi, fitur baru, atau perbaikan kode tersedia di lingkungan target. Deployment melibatkan penerapan perubahan dalam basis kode dan kemudian membangun dan menjalankan basis kode itu di lingkungan aplikasi.

lingkungan pengembangan

Lihat lingkungan.

kontrol detektif

Kontrol keamanan yang dirancang untuk mendeteksi, mencatat, dan memperingatkan setelah suatu peristiwa terjadi. Kontrol ini adalah garis pertahanan kedua, memperingatkan Anda tentang peristiwa keamanan yang melewati kontrol pencegahan yang ada. Untuk informasi selengkapnya, lihat Kontrol Detektif dalam Menerapkan kontrol keamanan pada. AWS

pemetaan aliran nilai pengembangan (DVSM)

Sebuah proses yang digunakan untuk mengidentifikasi dan memprioritaskan kendala yang mempengaruhi kecepatan dan kualitas dalam siklus hidup pengembangan perangkat lunak. DVSM memperluas proses pemetaan aliran nilai yang awalnya dirancang untuk praktik manufaktur ramping. Ini berfokus pada langkah-langkah dan tim yang diperlukan untuk menciptakan dan memindahkan nilai melalui proses pengembangan perangkat lunak.

kembar digital

Representasi virtual dari sistem dunia nyata, seperti bangunan, pabrik, peralatan industri, atau jalur produksi. Kembar digital mendukung pemeliharaan prediktif, pemantauan jarak jauh, dan optimalisasi produksi.

tabel dimensi

Dalam skema bintang, tabel yang lebih kecil yang berisi atribut data tentang data kuantitatif dalam tabel fakta. Atribut tabel dimensi biasanya bidang teks atau angka diskrit yang berperilaku seperti teks. Atribut ini biasanya digunakan untuk pembatasan kueri, pemfilteran, dan pelabelan set hasil.

musibah

Peristiwa yang mencegah beban kerja atau sistem memenuhi tujuan bisnisnya di lokasi utama yang digunakan. Peristiwa ini dapat berupa bencana alam, kegagalan teknis, atau akibat dari tindakan manusia, seperti kesalahan konfigurasi yang tidak disengaja atau serangan malware.

pemulihan bencana (DR)

Strategi dan proses yang Anda gunakan untuk meminimalkan downtime dan kehilangan data yang disebabkan oleh bencana. Untuk informasi selengkapnya, lihat Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML~

Lihat bahasa manipulasi database.

desain berbasis domain

Pendekatan untuk mengembangkan sistem perangkat lunak yang kompleks dengan menghubungkan komponennya ke domain yang berkembang, atau tujuan bisnis inti, yang dilayani setiap komponen. Konsep ini diperkenalkan oleh Eric Evans dalam bukunya, Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Untuk informasi tentang cara menggunakan desain berbasis domain dengan pola gambar pencekik, lihat Memodernisasi layanan web Microsoft ASP.NET (ASMX) lama secara bertahap menggunakan container dan Amazon API Gateway.

DR

Lihat pemulihan bencana.

deteksi drift

Melacak penyimpangan dari konfigurasi dasar. Misalnya, Anda dapat menggunakan AWS CloudFormation untuk mendeteksi penyimpangan dalam sumber daya sistem, atau Anda dapat menggunakannya AWS Control Tower untuk mendeteksi perubahan di landing zone yang mungkin memengaruhi kepatuhan terhadap persyaratan tata kelola.

DVSM

Lihat pemetaan aliran nilai pengembangan.

E

EDA

Lihat analisis data eksplorasi.

EDI

Lihat pertukaran data elektronik.

komputasi tepi

Teknologi yang meningkatkan daya komputasi untuk perangkat pintar di tepi jaringan IoT. Jika dibandingkan dengan komputasi awan, komputasi tepi dapat mengurangi latensi komunikasi dan meningkatkan waktu respons.

pertukaran data elektronik (EDI)

Pertukaran otomatis dokumen bisnis antar organisasi. Untuk informasi selengkapnya, lihat Apa itu Pertukaran Data Elektronik.

enkripsi

Proses komputasi yang mengubah data plaintext, yang dapat dibaca manusia, menjadi ciphertext.

kunci enkripsi

String kriptografi dari bit acak yang dihasilkan oleh algoritma enkripsi. Panjang kunci dapat bervariasi, dan setiap kunci dirancang agar tidak dapat diprediksi dan unik.

endianness

Urutan byte disimpan dalam memori komputer. Big-endian sistem menyimpan byte paling signifikan terlebih dahulu. Little-endian sistem menyimpan byte paling tidak signifikan terlebih dahulu.

titik akhir

Lihat titik akhir layanan.

layanan endpoint

Layanan yang dapat Anda host di cloud pribadi virtual (VPC) untuk dibagikan dengan pengguna lain. Anda dapat membuat layanan endpoint dengan AWS PrivateLink dan memberikan izin kepada prinsipal lain Akun AWS atau ke AWS Identity and Access Management (IAM). Akun atau prinsipal ini dapat terhubung ke layanan endpoint Anda secara pribadi dengan membuat titik akhir VPC antarmuka. Untuk informasi selengkapnya, lihat Membuat layanan titik akhir di dokumentasi Amazon Virtual Private Cloud (Amazon VPC).

perencanaan sumber daya perusahaan (ERP)

Sistem yang mengotomatiskan dan mengelola proses bisnis utama (seperti akuntansi, MES, dan manajemen proyek) untuk suatu perusahaan.

enkripsi amplop

Proses mengenkripsi kunci enkripsi dengan kunci enkripsi lain. Untuk informasi selengkapnya, lihat Enkripsi amplop dalam dokumentasi AWS Key Management Service (AWS KMS).

lingkungan

Sebuah contoh dari aplikasi yang sedang berjalan. Berikut ini adalah jenis lingkungan yang umum dalam komputasi awan:

- Development Environment — Sebuah contoh dari aplikasi yang berjalan yang hanya tersedia untuk tim inti yang bertanggung jawab untuk memelihara aplikasi. Lingkungan pengembangan digunakan untuk menguji perubahan sebelum mempromosikannya ke lingkungan atas. Jenis lingkungan ini kadang-kadang disebut sebagai lingkungan pengujian.

- lingkungan yang lebih rendah — Semua lingkungan pengembangan untuk aplikasi, seperti yang digunakan untuk build awal dan pengujian.
- lingkungan produksi — Sebuah contoh dari aplikasi yang berjalan yang dapat diakses oleh pengguna akhir. Dalam sebuah CI/CD pipeline, lingkungan produksi adalah lingkungan penyebaran terakhir.
- lingkungan atas — Semua lingkungan yang dapat diakses oleh pengguna selain tim pengembangan inti. Ini dapat mencakup lingkungan produksi, lingkungan praproduksi, dan lingkungan untuk pengujian penerimaan pengguna.

epik

Dalam metodologi tangkas, kategori fungsional yang membantu mengatur dan memprioritaskan pekerjaan Anda. Epik memberikan deskripsi tingkat tinggi tentang persyaratan dan tugas implementasi. Misalnya, epos keamanan AWS CAF mencakup manajemen identitas dan akses, kontrol detektif, keamanan infrastruktur, perlindungan data, dan respons insiden. Untuk informasi selengkapnya tentang epos dalam strategi AWS migrasi, lihat panduan implementasi program.

ERP

Lihat perencanaan sumber daya perusahaan.

analisis data eksplorasi (EDA)

Proses menganalisis dataset untuk memahami karakteristik utamanya. Anda mengumpulkan atau mengumpulkan data dan kemudian melakukan penyelidikan awal untuk menemukan pola, mendeteksi anomali, dan memeriksa asumsi. EDA dilakukan dengan menghitung statistik ringkasan dan membuat visualisasi data.

F

tabel fakta

Tabel tengah dalam skema bintang. Ini menyimpan data kuantitatif tentang operasi bisnis. Biasanya, tabel fakta berisi dua jenis kolom: kolom yang berisi ukuran dan yang berisi kunci asing ke tabel dimensi.

gagal cepat

Filosofi yang menggunakan pengujian yang sering dan bertahap untuk mengurangi siklus hidup pengembangan. Ini adalah bagian penting dari pendekatan tangkas.

batas isolasi kesalahan

Dalam AWS Cloud, batas seperti Availability Zone, AWS Region, control plane, atau data plane yang membatasi efek kegagalan dan membantu meningkatkan ketahanan beban kerja. Untuk informasi selengkapnya, lihat Batas Isolasi AWS Kesalahan.

cabang fitur

Lihat cabang.

fitur

Data input yang Anda gunakan untuk membuat prediksi. Misalnya, dalam konteks manufaktur, fitur bisa berupa gambar yang diambil secara berkala dari lini manufaktur.

pentingnya fitur

Seberapa signifikan fitur untuk prediksi model. Ini biasanya dinyatakan sebagai skor numerik yang dapat dihitung melalui berbagai teknik, seperti Shapley Additive Explanations (SHAP) dan gradien terintegrasi. Untuk informasi lebih lanjut, lihat Interpretabilitas model pembelajaran mesin dengan. AWS

transformasi fitur

Untuk mengoptimalkan data untuk proses ML, termasuk memperkaya data dengan sumber tambahan, menskalakan nilai, atau mengekstrak beberapa set informasi dari satu bidang data. Hal ini memungkinkan model ML untuk mendapatkan keuntungan dari data. Misalnya, jika Anda memecah tanggal “2021-05-27 00:15:37" menjadi “2021", “Mei”, “Kamis”, dan “15", Anda dapat membantu algoritme pembelajaran mempelajari pola bernuansa yang terkait dengan komponen data yang berbeda.

beberapa tembakan mendorong

Menyediakan LLM dengan sejumlah kecil contoh yang menunjukkan tugas dan output yang diinginkan sebelum memintanya untuk melakukan tugas serupa. Teknik ini adalah aplikasi pembelajaran dalam konteks, di mana model belajar dari contoh (bidikan) yang tertanam dalam petunjuk. Few-shot prompt bisa efektif untuk tugas-tugas yang membutuhkan pemformatan, penalaran, atau pengetahuan domain tertentu. Lihat juga bidikan nol.

FGAC

Lihat kontrol akses berbutir halus.

kontrol akses berbutir halus (FGAC)

Penggunaan beberapa kondisi untuk mengizinkan atau menolak permintaan akses.

migrasi flash-cut

Metode migrasi database yang menggunakan replikasi data berkelanjutan melalui pengambilan data perubahan untuk memigrasikan data dalam waktu sesingkat mungkin, alih-alih menggunakan pendekatan bertahap. Tujuannya adalah untuk menjaga downtime seminimal mungkin.

FM

Lihat model pondasi.

model pondasi (FM)

Jaringan saraf pembelajaran mendalam yang besar yang telah melatih kumpulan data besar-besaran data umum dan tidak berlabel. FM mampu melakukan berbagai tugas umum, seperti memahami bahasa, menghasilkan teks dan gambar, dan berbicara dalam bahasa alami. Untuk informasi selengkapnya, lihat Apa itu Model Foundation.

Gerbang FM

Perantara terpusat yang mengontrol dan menormalkan akses ke model pondasi. Juga dikenal sebagai gateway LLM.

G

AI generatif

Subset model AI yang telah dilatih pada sejumlah besar data dan yang dapat menggunakan prompt teks sederhana untuk membuat konten dan artefak baru, seperti gambar, video, teks, dan audio. Untuk informasi lebih lanjut, lihat Apa itu AI Generatif.

pemblokiran geografis

Lihat pembatasan geografis.

pembatasan geografis (pemblokiran geografis)

Di Amazon CloudFront, opsi untuk mencegah pengguna di negara tertentu mengakses distribusi konten. Anda dapat menggunakan daftar izinkan atau daftar blokir untuk menentukan negara yang disetujui dan dilarang. Untuk informasi selengkapnya, lihat Membatasi distribusi geografis konten Anda dalam dokumentasi. CloudFront

Alur kerja Gitflow

Pendekatan di mana lingkungan bawah dan atas menggunakan cabang yang berbeda dalam repositori kode sumber. Alur kerja Gitflow dianggap warisan, dan alur kerja berbasis batang adalah pendekatan modern yang lebih disukai.

gambar emas

Sebuah snapshot dari sistem atau perangkat lunak yang digunakan sebagai template untuk menyebarkan instance baru dari sistem atau perangkat lunak itu. Misalnya, di bidang manufaktur, gambar emas dapat digunakan untuk menyediakan perangkat lunak pada beberapa perangkat dan membantu meningkatkan kecepatan, skalabilitas, dan produktivitas dalam operasi manufaktur perangkat.

strategi greenfield

Tidak adanya infrastruktur yang ada di lingkungan baru. Saat mengadopsi strategi greenfield untuk arsitektur sistem, Anda dapat memilih semua teknologi baru tanpa batasan kompatibilitas dengan infrastruktur yang ada, juga dikenal sebagai brownfield. Jika Anda memperluas infrastruktur yang ada, Anda dapat memadukan strategi brownfield dan greenfield.

pagar pembatas

Aturan tingkat tinggi yang membantu mengatur sumber daya, kebijakan, dan kepatuhan di seluruh unit organisasi (OU). Pagar pembatas preventif menegakkan kebijakan untuk memastikan keselarasan dengan standar kepatuhan. Mereka diimplementasikan dengan menggunakan kebijakan kontrol layanan dan batas izin IAM. Detective guardrails mendeteksi pelanggaran kebijakan dan masalah kepatuhan, dan menghasilkan peringatan untuk remediasi. Mereka diimplementasikan dengan menggunakan AWS Config, AWS Security Hub CSPM, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector, dan pemeriksaan khusus AWS Lambda .

pagar pembatas (AI)

Mekanisme keamanan yang menyaring, memvalidasi, dan membatasi input dan output agen untuk membantu memastikan perilaku AI yang bertanggung jawab dan aman.

H

HA

Lihat ketersediaan tinggi.

migrasi database heterogen

Memigrasi database sumber Anda ke database target yang menggunakan mesin database yang berbeda (misalnya, Oracle ke Amazon Aurora). Migrasi heterogen biasanya merupakan bagian dari upaya arsitektur ulang, dan mengubah skema dapat menjadi tugas yang kompleks. AWS menyediakan AWS SCT yang membantu dengan konversi skema.

ketersediaan tinggi (HA)

Kemampuan beban kerja untuk beroperasi terus menerus, tanpa intervensi, jika terjadi tantangan atau bencana. Sistem HA dirancang untuk gagal secara otomatis, secara konsisten memberikan kinerja berkualitas tinggi, dan menangani beban dan kegagalan yang berbeda dengan dampak kinerja minimal.

modernisasi sejarawan

Pendekatan yang digunakan untuk memodernisasi dan meningkatkan sistem teknologi operasional (OT) untuk melayani kebutuhan industri manufaktur dengan lebih baik. Sejarawan adalah jenis database yang digunakan untuk mengumpulkan dan menyimpan data dari berbagai sumber di pabrik.

data penahanan

Sebagian dari data historis berlabel yang ditahan dari kumpulan data yang digunakan untuk melatih model pembelajaran mesin. Anda dapat menggunakan data penahanan untuk mengevaluasi kinerja model dengan membandingkan prediksi model dengan data penahanan.

manusia-dalam-lingkaran (HiTL)

Pola alur kerja di mana eksekusi agen berhenti untuk peninjauan dan persetujuan manusia pada titik keputusan kritis.

migrasi database homogen

Memigrasi database sumber Anda ke database target yang berbagi mesin database yang sama (misalnya, Microsoft SQL Server ke Amazon RDS for SQL Server). Migrasi homogen biasanya merupakan bagian dari upaya rehosting atau replatforming. Anda dapat menggunakan utilitas database asli untuk memigrasi skema.

data panas

Data yang sering diakses, seperti data real-time atau data translasi terbaru. Data ini biasanya memerlukan tingkat atau kelas penyimpanan berkinerja tinggi untuk memberikan respons kueri yang cepat.

perbaikan terbaru

Perbaikan mendesak untuk masalah kritis dalam lingkungan produksi. Karena urgensinya, perbaikan terbaru biasanya dibuat di luar alur kerja DevOps rilis biasa.

periode hypercare

Segera setelah cutover, periode waktu ketika tim migrasi mengelola dan memantau aplikasi yang dimigrasi di cloud untuk mengatasi masalah apa pun. Biasanya, periode ini panjangnya 1-4 hari. Pada akhir periode hypercare, tim migrasi biasanya mentransfer tanggung jawab untuk aplikasi ke tim operasi cloud.

I

IAc

Lihat infrastruktur sebagai kode.

kebijakan berbasis identitas

Kebijakan yang dilampirkan pada satu atau beberapa prinsip IAM yang mendefinisikan izin mereka dalam lingkungan. AWS Cloud

aplikasi idle

Aplikasi yang memiliki penggunaan CPU dan memori rata-rata antara 5 dan 20 persen selama periode 90 hari. Dalam proyek migrasi, adalah umum untuk menghentikan aplikasi ini atau mempertahankannya di tempat.

IIoT

Lihat Internet of Things industri.

infrastruktur yang tidak dapat diubah

Model yang menyebarkan infrastruktur baru untuk beban kerja produksi alih-alih memperbarui, menambal, atau memodifikasi infrastruktur yang ada. Infrastruktur yang tidak dapat diubah secara inheren lebih konsisten, andal, dan dapat diprediksi daripada infrastruktur yang dapat berubah. Untuk informasi selengkapnya, lihat praktik terbaik Deploy using immutable infrastructure in the Framework. AWS Well-Architected

masuk (masuknya) VPC

Dalam arsitektur AWS multi-akun, VPC yang menerima, memeriksa, dan merutekan koneksi jaringan dari luar aplikasi. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan

akun Jaringan Anda dengan VPC masuk, keluar, dan inspeksi untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

migrasi inkremental

Strategi cutover di mana Anda memigrasikan aplikasi Anda dalam bagian-bagian kecil alih-alih melakukan satu cutover penuh. Misalnya, Anda mungkin hanya memindahkan beberapa layanan mikro atau pengguna ke sistem baru pada awalnya. Setelah Anda memverifikasi bahwa semuanya berfungsi dengan baik, Anda dapat secara bertahap memindahkan layanan mikro atau pengguna tambahan hingga Anda dapat menonaktifkan sistem lama Anda. Strategi ini mengurangi risiko yang terkait dengan migrasi besar.

Industri 4.0

Sebuah istilah yang diperkenalkan oleh Klaus Schwab pada tahun 2016 untuk merujuk pada modernisasi proses manufaktur melalui kemajuan dalam konektivitas, data real-time, otomatisasi, analitik, dan. AI/ML

infrastruktur

Semua sumber daya dan aset yang terkandung dalam lingkungan aplikasi.

infrastruktur sebagai kode (IAc)

Proses penyediaan dan pengelolaan infrastruktur aplikasi melalui satu set file konfigurasi. IAc dirancang untuk membantu Anda memusatkan manajemen infrastruktur, menstandarisasi sumber daya, dan menskalakan dengan cepat sehingga lingkungan baru dapat diulang, andal, dan konsisten.

Internet of Things industri (IIoT)

Penggunaan sensor dan perangkat yang terhubung ke internet di sektor industri, seperti manufaktur, energi, otomotif, perawatan kesehatan, ilmu kehidupan, dan pertanian. Untuk informasi selengkapnya, lihat Membangun strategi transformasi digital Internet of Things (IIoT) industri.

inspeksi VPC

Dalam arsitektur AWS multi-akun, VPC terpusat yang mengelola inspeksi lalu lintas jaringan antara VPC (dalam hal yang sama atau berbeda Region AWS), internet, dan jaringan lokal. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan VPC masuk, keluar, dan inspeksi untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

Internet of Things (IoT)

Jaringan objek fisik yang terhubung dengan sensor atau prosesor tertanam yang berkomunikasi dengan perangkat dan sistem lain melalui internet atau melalui jaringan komunikasi lokal. Untuk informasi selengkapnya, lihat Apa itu IoT?

interpretabilitas

Karakteristik model pembelajaran mesin yang menggambarkan sejauh mana manusia dapat memahami bagaimana prediksi model bergantung pada inputnya. Untuk informasi lebih lanjut, lihat Interpretabilitas model pembelajaran mesin dengan. AWS

IoT

Lihat Internet of Things.

Perpustakaan informasi TI (ITIL)

Serangkaian praktik terbaik untuk memberikan layanan TI dan menyelaraskan layanan ini dengan persyaratan bisnis. ITIL menyediakan dasar untuk ITSM.

Manajemen layanan TI (ITSM)

Kegiatan yang terkait dengan merancang, menerapkan, mengelola, dan mendukung layanan TI untuk suatu organisasi. Untuk informasi tentang mengintegrasikan operasi cloud dengan alat ITSM, lihat panduan integrasi operasi.

ITIL

Lihat perpustakaan informasi TI.

ITSM

Lihat manajemen layanan TI.

L

kontrol akses berbasis label (LBAC)

Implementasi kontrol akses wajib (MAC) di mana pengguna dan data itu sendiri masing-masing secara eksplisit diberi nilai label keamanan. Persimpangan antara label keamanan pengguna dan label keamanan data menentukan baris dan kolom mana yang dapat dilihat oleh pengguna.

landing zone

Landing zone adalah AWS lingkungan multi-akun yang dirancang dengan baik yang dapat diskalakan dan aman. Ini adalah titik awal dari mana organisasi Anda dapat dengan cepat meluncurkan dan menyebarkan beban kerja dan aplikasi dengan percaya diri dalam lingkungan keamanan dan infrastruktur mereka. Untuk informasi selengkapnya tentang zona pendaratan, lihat Menyiapkan lingkungan multi-akun AWS yang aman dan dapat diskalakan.

model bahasa besar (LLM)

Model AI pembelajaran mendalam yang dilatih sebelumnya pada sejumlah besar data. LLM dapat melakukan beberapa tugas, seperti menjawab pertanyaan, meringkas dokumen, menerjemahkan teks ke bahasa lain, dan menyelesaikan kalimat. Untuk informasi lebih lanjut, lihat Apa itu LLM.

migrasi besar

Migrasi 300 atau lebih server.

LBAC

Lihat kontrol akses berbasis label.

hak istimewa paling sedikit

Praktik keamanan terbaik untuk memberikan izin minimum yang diperlukan untuk melakukan tugas. Untuk informasi selengkapnya, lihat Menerapkan izin hak istimewa terkecil dalam dokumentasi IAM.

angkat dan geser

Lihat 7 Rs.

sistem endian kecil

Sebuah sistem yang menyimpan byte paling tidak signifikan terlebih dahulu. Lihat juga endianness.

LLM

Lihat model bahasa besar.

lingkungan yang lebih rendah

Lihat lingkungan.

M

pembelajaran mesin (ML)

Jenis kecerdasan buatan yang menggunakan algoritma dan teknik untuk pengenalan pola dan pembelajaran. ML menganalisis dan belajar dari data yang direkam, seperti data Internet of Things (IoT), untuk menghasilkan model statistik berdasarkan pola. Untuk informasi selengkapnya, lihat Machine Learning.

cabang utama

Lihat cabang.

malware

Perangkat lunak yang dirancang untuk membahayakan keamanan atau privasi komputer. Malware dapat mengganggu sistem komputer, membocorkan informasi sensitif, atau mendapatkan akses yang tidak sah. Contoh malware termasuk virus, worm, ransomware, Trojan horse, spyware, dan keyloggers.

layanan terkelola

Layanan AWS yang AWS mengoperasikan lapisan infrastruktur, sistem operasi, dan platform, dan Anda mengakses titik akhir untuk menyimpan dan mengambil data. Amazon Simple Storage Service (Amazon S3) dan Amazon DynamoDB adalah contoh layanan terkelola. Ini juga dikenal sebagai layanan abstrak.

sistem eksekusi manufaktur (MES)

Sistem perangkat lunak untuk melacak, memantau, mendokumentasikan, dan mengendalikan proses produksi yang mengubah bahan baku menjadi produk jadi di lantai toko.

PETA

Lihat Program Percepatan Migrasi.

MCP

Lihat Protokol Konteks Model.

Protokol Konteks Model (MCP)

Protokol stateless untuk komunikasi agen -to- alat.

Server MCP

Layanan yang mengekspos satu atau lebih alat melalui Protokol Konteks Model.

mekanisme

Proses lengkap di mana Anda membuat alat, mendorong adopsi alat, dan kemudian memeriksa hasilnya untuk melakukan penyesuaian. Mekanisme adalah siklus yang memperkuat dan meningkatkan dirinya sendiri saat beroperasi. Untuk informasi selengkapnya, lihat Membangun mekanisme dalam AWS Well-Architected Kerangka Kerja.

akun anggota

Semua Akun AWS selain akun manajemen yang merupakan bagian dari organisasi di AWS Organizations. Akun dapat menjadi anggota dari hanya satu organisasi pada suatu waktu.

MES

Lihat sistem eksekusi manufaktur.

Transportasi Telemetri Antrian Pesan (MQTT)

Protokol komunikasi mesin-ke-mesin (M2M) yang ringan, berdasarkan pola publish/subscribe, untuk perangkat IoT yang dibatasi sumber daya.

layanan mikro

Layanan kecil dan independen yang berkomunikasi melalui API yang terdefinisi dengan baik dan biasanya dimiliki oleh tim kecil yang mandiri. Misalnya, sistem asuransi mungkin mencakup layanan mikro yang memetakan kemampuan bisnis, seperti penjualan atau pemasaran, atau subdomain, seperti pembelian, klaim, atau analitik. Manfaat layanan mikro termasuk kelincahan, penskalaan yang fleksibel, penyebaran yang mudah, kode yang dapat digunakan kembali, dan ketahanan. Untuk informasi selengkapnya, lihat Mengintegrasikan layanan mikro dengan menggunakan layanan tanpa AWS server.

arsitektur microservices

Pendekatan untuk membangun aplikasi dengan komponen independen yang menjalankan setiap proses aplikasi sebagai layanan mikro. Layanan mikro ini berkomunikasi melalui antarmuka yang terdefinisi dengan baik dengan menggunakan API ringan. Setiap layanan mikro dalam arsitektur ini dapat diperbarui, digunakan, dan diskalakan untuk memenuhi permintaan fungsi tertentu dari suatu aplikasi. Untuk informasi selengkapnya, lihat Menerapkan layanan mikro di AWS.

Program Percepatan Migrasi (MAP)

AWS Program yang menyediakan dukungan konsultasi, pelatihan, dan layanan untuk membantu organisasi membangun fondasi operasional yang kuat untuk pindah ke cloud, dan untuk membantu mengimbangi biaya awal migrasi. MAP mencakup metodologi migrasi untuk

mengeksekusi migrasi lama dengan cara metodis dan seperangkat alat untuk mengotomatisasi dan mempercepat skenario migrasi umum.

migrasi dalam skala

Proses memindahkan sebagian besar portofolio aplikasi ke cloud dalam gelombang, dengan lebih banyak aplikasi bergerak pada tingkat yang lebih cepat di setiap gelombang. Fase ini menggunakan praktik terbaik dan pelajaran yang dipetik dari fase sebelumnya untuk mengimplementasikan pabrik migrasi tim, alat, dan proses untuk merampingkan migrasi beban kerja melalui otomatisasi dan pengiriman tangkas. Ini adalah fase ketiga dari strategi AWS migrasi.

pabrik migrasi

Cross-functional tim yang merampingkan migrasi beban kerja melalui pendekatan otomatis dan gesit. Tim pabrik migrasi biasanya mencakup operasi, analis dan pemilik bisnis, insinyur migrasi, pengembang, dan DevOps profesional yang bekerja di sprint. Antara 20 dan 50 persen portofolio aplikasi perusahaan terdiri dari pola berulang yang dapat dioptimalkan dengan pendekatan pabrik. Untuk informasi selengkapnya, lihat diskusi tentang pabrik migrasi dan panduan Pabrik Migrasi Cloud di kumpulan konten ini.

metadata migrasi

Informasi tentang aplikasi dan server yang diperlukan untuk menyelesaikan migrasi. Setiap pola migrasi memerlukan satu set metadata migrasi yang berbeda. Contoh metadata migrasi termasuk subnet target, grup keamanan, dan akun. AWS

pola migrasi

Tugas migrasi berulang yang merinci strategi migrasi, tujuan migrasi, dan aplikasi atau layanan migrasi yang digunakan. Contoh: Rehost migrasi ke Amazon EC2 AWS dengan Layanan Migrasi Aplikasi.

Penilaian Portofolio Migrasi (MPA)

Alat online yang menyediakan informasi untuk memvalidasi kasus bisnis untuk bermigrasi ke. AWS Cloud MPA menyediakan penilaian portofolio terperinci (ukuran kanan server, harga, perbandingan TCO, analisis biaya migrasi) serta perencanaan migrasi (analisis data aplikasi dan pengumpulan data, pengelompokan aplikasi, prioritas migrasi, dan perencanaan gelombang). Alat MPA (memerlukan login) tersedia gratis untuk semua AWS konsultan dan konsultan APN Partner.

Penilaian Kesiapan Migrasi (MRA)

Proses mendapatkan wawasan tentang status kesiapan cloud organisasi, mengidentifikasi kekuatan dan kelemahan, dan membangun rencana aksi untuk menutup kesenjangan yang diidentifikasi, menggunakan CAF. AWS Untuk informasi selengkapnya, lihat panduan kesiapan migrasi. MRA adalah tahap pertama dari strategi AWS migrasi.

strategi migrasi

Pendekatan yang digunakan untuk memigrasikan beban kerja ke. AWS Cloud Untuk informasi lebih lanjut, lihat entri 7 Rs di glosarium ini dan lihat Memobilisasi organisasi Anda untuk mempercepat migrasi skala besar.

ML

Lihat pembelajaran mesin.

modernisasi

Mengubah aplikasi usang (warisan atau monolitik) dan infrastrukturnya menjadi sistem yang gesit, elastis, dan sangat tersedia di cloud untuk mengurangi biaya, mendapatkan efisiensi, dan memanfaatkan inovasi. Untuk informasi selengkapnya, lihat Strategi untuk memodernisasi aplikasi di. AWS Cloud

penilaian kesiapan modernisasi

Evaluasi yang membantu menentukan kesiapan modernisasi aplikasi organisasi; mengidentifikasi manfaat, risiko, dan dependensi; dan menentukan seberapa baik organisasi dapat mendukung keadaan masa depan aplikasi tersebut. Hasil penilaian adalah cetak biru arsitektur target, peta jalan yang merinci fase pengembangan dan tonggak untuk proses modernisasi, dan rencana aksi untuk mengatasi kesenjangan yang diidentifikasi. Untuk informasi lebih lanjut, lihat Mengevaluasi kesiapan modernisasi untuk aplikasi di. AWS Cloud

aplikasi monolitik (monolit)

Aplikasi yang berjalan sebagai layanan tunggal dengan proses yang digabungkan secara ketat. Aplikasi monolitik memiliki beberapa kelemahan. Jika satu fitur aplikasi mengalami lonjakan permintaan, seluruh arsitektur harus diskalakan. Menambahkan atau meningkatkan fitur aplikasi monolitik juga menjadi lebih kompleks ketika basis kode tumbuh. Untuk mengatasi masalah ini, Anda dapat menggunakan arsitektur microservices. Untuk informasi lebih lanjut, lihat Mengurai monolit menjadi layanan mikro.

MPA

Lihat Penilaian Portofolio Migrasi.

MQTT

Lihat Transportasi Telemetri Antrian Pesan.

klasifikasi multiclass

Sebuah proses yang membantu menghasilkan prediksi untuk beberapa kelas (memprediksi satu dari lebih dari dua hasil). Misalnya, model ML mungkin bertanya "Apakah produk ini buku, mobil, atau telepon?" atau "Kategori produk mana yang paling menarik bagi pelanggan ini?"

infrastruktur yang bisa berubah

Model yang memperbarui dan memodifikasi infrastruktur yang ada untuk beban kerja produksi. Untuk meningkatkan konsistensi, keandalan, dan prediktabilitas, AWS Well-Architected Framework merekomendasikan penggunaan infrastruktur yang tidak dapat diubah sebagai praktik terbaik.

O

OAC

Lihat kontrol akses asal.

OAI

Lihat identitas akses asal.

OCM

Lihat manajemen perubahan organisasi.

migrasi offline

Metode migrasi di mana beban kerja sumber diturunkan selama proses migrasi. Metode ini melibatkan waktu henti yang diperpanjang dan biasanya digunakan untuk beban kerja kecil dan tidak kritis.

OI

Lihat integrasi operasi.

OLA

Lihat perjanjian tingkat operasional.

migrasi online

Metode migrasi di mana beban kerja sumber disalin ke sistem target tanpa diambil offline. Aplikasi yang terhubung ke beban kerja dapat terus berfungsi selama migrasi. Metode ini melibatkan waktu henti nol hingga minimal dan biasanya digunakan untuk beban kerja produksi yang kritis.

OPC-UA

Lihat Komunikasi Proses Terbuka - Arsitektur Terpadu.

Komunikasi Proses Terbuka - Arsitektur Terpadu () OPC-UA

Protokol komunikasi mesin-ke-mesin (M2M) untuk otomasi industri. OPC-UA menyediakan standar interoperabilitas dengan enkripsi data, otentikasi, dan skema otorisasi.

perjanjian tingkat operasional (OLA)

Perjanjian yang menjelaskan apa yang dijanjikan kelompok TI fungsional untuk diberikan satu sama lain, untuk mendukung perjanjian tingkat layanan (SLA).

Tinjauan Kesiapan Operasional (ORR)

Daftar pertanyaan dan praktik terbaik terkait yang membantu Anda memahami, mengevaluasi, mencegah, atau mengurangi ruang lingkup insiden dan kemungkinan kegagalan. Untuk informasi selengkapnya, lihat Ulasan Kesiapan Operasional (ORR) dalam Kerangka Kerja. AWS Well-Architected

teknologi operasional (OT)

Sistem perangkat keras dan perangkat lunak yang bekerja dengan lingkungan fisik untuk mengendalikan operasi industri, peralatan, dan infrastruktur. Di bidang manufaktur, integrasi sistem OT dan teknologi informasi (TI) adalah fokus utama untuk transformasi Industri 4.0.

integrasi operasi (OI)

Proses modernisasi operasi di cloud, yang melibatkan perencanaan kesiapan, otomatisasi, dan integrasi. Untuk informasi selengkapnya, lihat panduan integrasi operasi.

jejak organisasi

Jejak yang dibuat oleh AWS CloudTrail itu mencatat semua peristiwa untuk semua Akun AWS dalam organisasi di AWS Organizations. Jejak ini dibuat di setiap Akun AWS bagian organisasi dan melacak aktivitas di setiap akun. Untuk informasi selengkapnya, lihat Membuat jejak untuk organisasi dalam CloudTrail dokumentasi.

manajemen perubahan organisasi (OCM)

Kerangka kerja untuk mengelola transformasi bisnis utama yang mengganggu dari perspektif orang, budaya, dan kepemimpinan. OCM membantu organisasi mempersiapkan, dan transisi ke, sistem dan strategi baru dengan mempercepat adopsi perubahan, mengatasi masalah transisi, dan mendorong perubahan budaya dan organisasi. Dalam strategi AWS migrasi, kerangka kerja ini disebut percepatan orang, karena kecepatan perubahan yang diperlukan dalam proyek adopsi cloud. Untuk informasi lebih lanjut, lihat panduan OCM.

kontrol akses asal (OAC)

Di CloudFront, opsi yang disempurnakan untuk membatasi akses untuk mengamankan konten Amazon Simple Storage Service (Amazon S3) Anda. OAC mendukung semua bucket S3 di semua Region AWS, enkripsi sisi server dengan AWS KMS (SSE-KMS), dan dinamis `PUT` dan `DELETE` permintaan ke bucket S3.

identitas akses asal (OAI)

Di CloudFront, opsi untuk membatasi akses untuk mengamankan konten Amazon S3 Anda. Saat Anda menggunakan OAI, CloudFront buat prinsipal yang dapat diautentikasi oleh Amazon S3. Prinsipal yang diautentikasi dapat mengakses konten dalam bucket S3 hanya melalui distribusi tertentu. CloudFront Lihat juga OAC, yang menyediakan kontrol akses yang lebih terperinci dan ditingkatkan.

ORR

Lihat tinjauan kesiapan operasional.

OT

Lihat teknologi operasional.

keluar (jalan keluar) VPC

Dalam arsitektur AWS multi-akun, VPC yang menangani koneksi jaringan yang dimulai dari dalam aplikasi. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan VPC masuk, keluar, dan inspeksi untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

P

batas izin

Kebijakan manajemen IAM yang dilampirkan pada prinsipal IAM untuk menetapkan izin maksimum yang dapat dimiliki pengguna atau peran. Untuk informasi selengkapnya, lihat Batas izin dalam dokumentasi IAM.

Informasi Identifikasi Pribadi (PII)

Informasi yang, jika dilihat secara langsung atau dipasangkan dengan data terkait lainnya, dapat digunakan untuk menyimpulkan identitas individu secara wajar. Contoh PII termasuk nama, alamat, dan informasi kontak.

PII

Lihat informasi yang dapat diidentifikasi secara pribadi.

buku pedoman

Serangkaian langkah yang telah ditentukan sebelumnya yang menangkap pekerjaan yang terkait dengan migrasi, seperti mengirimkan fungsi operasi inti di cloud. Buku pedoman dapat berupa skrip, runbook otomatis, atau ringkasan proses atau langkah-langkah yang diperlukan untuk mengoperasikan lingkungan modern Anda.

PLC

Lihat pengontrol logika yang dapat diprogram.

PLM

Lihat manajemen siklus hidup produk.

kebijakan

Objek yang dapat menentukan izin (lihat kebijakan berbasis identitas), menentukan kondisi akses (lihat kebijakanberbasis sumber daya), atau menentukan izin maksimum untuk semua akun dalam organisasi di (lihat kebijakan kontrol layanan). AWS Organizations

persistensi poliglot

Secara independen memilih teknologi penyimpanan data microservice berdasarkan pola akses data dan persyaratan lainnya. Jika layanan mikro Anda memiliki teknologi penyimpanan data yang sama, mereka dapat menghadapi tantangan implementasi atau mengalami kinerja yang buruk. Layanan mikro lebih mudah diimplementasikan dan mencapai kinerja dan skalabilitas yang lebih baik jika mereka menggunakan penyimpanan data yang paling sesuai dengan kebutuhan mereka.

penilaian portofolio

Proses menemukan, menganalisis, dan memprioritaskan portofolio aplikasi untuk merencanakan migrasi. Untuk informasi selengkapnya, lihat Mengevaluasi kesiapan migrasi.

predikat

Kondisi kueri yang mengembalikan `true` atau`false`, biasanya terletak di `WHERE` klausa.

predikat pushdown

Teknik pengoptimalan kueri database yang menyaring data dalam kueri sebelum transfer. Ini mengurangi jumlah data yang harus diambil dan diproses dari database relasional, dan meningkatkan kinerja kueri.

kontrol preventif

Kontrol keamanan yang dirancang untuk mencegah suatu peristiwa terjadi. Kontrol ini adalah garis pertahanan pertama untuk membantu mencegah akses tidak sah atau perubahan yang tidak diinginkan ke jaringan Anda. Untuk informasi selengkapnya, lihat Kontrol pencegahan dalam Menerapkan kontrol keamanan pada. AWS

principal

Entitas AWS yang dapat melakukan tindakan dan mengakses sumber daya. Entitas ini biasanya merupakan pengguna root untuk Akun AWS, peran IAM, atau pengguna. Untuk informasi selengkapnya, lihat Prinsip dalam istilah dan konsep Peran dalam dokumentasi IAM.

privasi berdasarkan desain

Pendekatan rekayasa sistem yang memperhitungkan privasi melalui seluruh proses pengembangan.

zona host pribadi

Container yang menyimpan informasi tentang bagaimana Anda ingin Amazon Route 53 merespons kueri DNS untuk domain dan subdomainnya dalam satu atau beberapa VPC. Untuk informasi selengkapnya, lihat Bekerja dengan zona yang dihosting pribadi di dokumentasi Route 53.

kontrol proaktif

Kontrol keamanan yang dirancang untuk mencegah penyebaran sumber daya yang tidak sesuai. Kontrol ini memindai sumber daya sebelum disediakan. Jika sumber daya tidak sesuai dengan kontrol, maka itu tidak disediakan. Untuk informasi selengkapnya, lihat panduan referensi Kontrol

dalam AWS Control Tower dokumentasi dan lihat Kontrol proaktif dalam Menerapkan kontrol keamanan pada AWS.

manajemen siklus hidup produk (PLM)

Manajemen data dan proses untuk suatu produk di seluruh siklus hidupnya, mulai dari desain, pengembangan, dan peluncuran, melalui pertumbuhan dan kematangan, hingga penurunan dan penghapusan.

lingkungan produksi

Lihat lingkungan.

pengontrol logika yang dapat diprogram (PLC)

Di bidang manufaktur, komputer yang sangat andal dan mudah beradaptasi yang memantau mesin dan mengotomatiskan proses manufaktur.

rantai cepat

Menggunakan output dari satu prompt LLM sebagai input untuk prompt berikutnya untuk menghasilkan respons yang lebih baik. Teknik ini digunakan untuk memecah tugas yang kompleks menjadi subtugas, atau untuk secara iteratif memperbaiki atau memperluas respons awal. Ini membantu meningkatkan akurasi dan relevansi respons model dan memungkinkan hasil yang lebih terperinci dan dipersonalisasi.

pseudonimisasi

Proses penggantian pengenal pribadi dalam kumpulan data dengan nilai placeholder. Pseudonimisasi dapat membantu melindungi privasi pribadi. Data pseudonim masih dianggap sebagai data pribadi.

publish/subscribe (pub/sub)

Pola yang memungkinkan komunikasi asinkron antara layanan mikro untuk meningkatkan skalabilitas dan daya tanggap. Misalnya, dalam MES berbasis layanan mikro, layanan mikro dapat mempublikasikan pesan peristiwa ke saluran yang dapat berlangganan layanan mikro lainnya. Sistem dapat menambahkan layanan mikro baru tanpa mengubah layanan penerbitan.

Q

rencana kueri

Serangkaian langkah, seperti instruksi, yang digunakan untuk mengakses data dalam sistem database relasional SQL.

regresi rencana kueri

Ketika pengoptimal layanan database memilih rencana yang kurang optimal daripada sebelum perubahan yang diberikan ke lingkungan database. Hal ini dapat disebabkan oleh perubahan statistik, kendala, pengaturan lingkungan, pengikatan parameter kueri, dan pembaruan ke mesin database.

R

Matriks RACI

Lihat bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI).

LAP

Lihat Retrieval Augmented Generation.

ransomware

Perangkat lunak berbahaya yang dirancang untuk memblokir akses ke sistem komputer atau data sampai pembayaran dilakukan.

Matriks RASCI

Lihat bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI).

RCAC

Lihat kontrol akses baris dan kolom.

replika baca

Salinan database yang digunakan untuk tujuan read-only. Anda dapat merutekan kueri ke replika baca untuk mengurangi beban pada database utama Anda.

arsitek ulang

Lihat 7 Rs.

tujuan titik pemulihan (RPO)

Jumlah waktu maksimum yang dapat diterima sejak titik pemulihan data terakhir. Ini menentukan apa yang dianggap sebagai kehilangan data yang dapat diterima antara titik pemulihan terakhir dan gangguan layanan.

tujuan waktu pemulihan (RTO)

Penundaan maksimum yang dapat diterima antara gangguan layanan dan pemulihan layanan.

refactor

Lihat 7 Rs.

Region

Kumpulan AWS sumber daya di wilayah geografis. Masing-masing AWS Region terisolasi dan independen dari yang lain untuk memberikan toleransi kesalahan, stabilitas, dan ketahanan. Untuk informasi selengkapnya, lihat Menentukan Region AWS akun yang dapat digunakan.

regresi

Teknik ML yang memprediksi nilai numerik. Misalnya, untuk memecahkan masalah “Berapa harga rumah ini akan dijual?” Model ML dapat menggunakan model regresi linier untuk memprediksi harga jual rumah berdasarkan fakta yang diketahui tentang rumah (misalnya, luas persegi).

rehost

Lihat 7 Rs.

melepaskan

Dalam proses penyebaran, tindakan mempromosikan perubahan pada lingkungan produksi.

memindahkan

Lihat 7 Rs.

memplatform ulang

Lihat 7 Rs.

pembelian kembali

Lihat 7 Rs.

ketahanan

Kemampuan aplikasi untuk melawan atau pulih dari gangguan. Ketersediaan tinggi dan pemulihan bencana adalah pertimbangan umum ketika merencanakan ketahanan di. AWS Cloud Untuk informasi lebih lanjut, lihat AWS Cloud Ketahanan.

kebijakan berbasis sumber daya

Kebijakan yang dilampirkan ke sumber daya, seperti bucket Amazon S3, titik akhir, atau kunci enkripsi. Jenis kebijakan ini menentukan prinsipal mana yang diizinkan mengakses, tindakan yang didukung, dan kondisi lain yang harus dipenuhi.

matriks yang bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI)

Matriks yang mendefinisikan peran dan tanggung jawab untuk semua pihak yang terlibat dalam kegiatan migrasi dan operasi cloud. Nama matriks berasal dari jenis tanggung jawab yang didefinisikan dalam matriks: bertanggung jawab (R), akuntabel (A), dikonsultasikan (C), dan diinformasikan (I). Jenis dukungan (S) adalah opsional. Jika Anda menyertakan dukungan, matriks disebut matriks RASCI, dan jika Anda mengecualikannya, itu disebut matriks RACI.

kontrol responsif

Kontrol keamanan yang dirancang untuk mendorong remediasi efek samping atau penyimpangan dari garis dasar keamanan Anda. Untuk informasi selengkapnya, lihat Kontrol responsif dalam Menerapkan kontrol keamanan pada AWS.

melestarikan

Lihat 7 Rs.

pensiun

Lihat 7 Rs.

Retrieval Augmented Generation (RAG)

Teknologi AI generatif di mana LLM mereferensikan sumber data otoritatif yang berada di luar sumber data pelatihannya sebelum menghasilkan respons. Misalnya, model RAG mungkin melakukan pencarian semantik dari basis pengetahuan organisasi atau data kustom. Untuk informasi lebih lanjut, lihat Apa itu RAG.

rotasi

Proses memperbarui rahasia secara berkala untuk membuatnya lebih sulit bagi penyerang untuk mengakses kredensil.

kontrol akses baris dan kolom (RCAC)

Penggunaan ekspresi SQL dasar dan fleksibel yang telah menetapkan aturan akses. RCAC terdiri dari izin baris dan topeng kolom.

RPO

Lihat tujuan titik pemulihan.

RTO

Lihat tujuan waktu pemulihan.

buku runbook

Satu set prosedur manual atau otomatis yang diperlukan untuk melakukan tugas tertentu. Ini biasanya dibangun untuk merampingkan operasi berulang atau prosedur dengan tingkat kesalahan yang tinggi.

D

SAML 2.0

Standar terbuka yang digunakan oleh banyak penyedia identitas (IdPs). Fitur ini memungkinkan sistem masuk tunggal gabungan (SSO), sehingga pengguna dapat masuk ke Konsol Manajemen AWS atau memanggil operasi AWS API tanpa Anda harus membuat pengguna di IAM untuk semua orang di organisasi Anda. Untuk informasi lebih lanjut tentang federasi berbasis SAMP 2.0, lihat Tentang federasi berbasis SAMP 2.0 dalam dokumentasi IAM.

SCADA

Lihat kontrol pengawasan dan akuisisi data.

SCP

Lihat kebijakan kontrol layanan.

Rahasia

Dalam AWS Secrets Manager, informasi rahasia atau terbatas, seperti kata sandi atau kredensil pengguna, yang Anda simpan dalam bentuk terenkripsi. Ini terdiri dari nilai rahasia dan metadata-nya. Nilai rahasia dapat berupa biner, string tunggal, atau beberapa string. Untuk informasi selengkapnya, lihat Apa yang ada di rahasia Secrets Manager? dalam dokumentasi Secrets Manager.

keamanan dengan desain

Pendekatan rekayasa sistem yang memperhitungkan keamanan melalui seluruh proses pengembangan.

kontrol keamanan

Pagar pembatas teknis atau administratif yang mencegah, mendeteksi, atau mengurangi kemampuan pelaku ancaman untuk mengeksploitasi kerentanan keamanan. Ada empat jenis kontrol keamanan utama: preventif, detektif, responsif, dan proaktif.

pengerasan keamanan

Proses mengurangi permukaan serangan untuk membuatnya lebih tahan terhadap serangan. Ini dapat mencakup tindakan seperti menghapus sumber daya yang tidak lagi diperlukan, menerapkan praktik keamanan terbaik untuk memberikan hak istimewa paling sedikit, atau menonaktifkan fitur yang tidak perlu dalam file konfigurasi.

sistem informasi keamanan dan manajemen acara (SIEM)

Alat dan layanan yang menggabungkan sistem manajemen informasi keamanan (SIM) dan manajemen acara keamanan (SEM). Sistem SIEM mengumpulkan, memantau, dan menganalisis data dari server, jaringan, perangkat, dan sumber lain untuk mendeteksi ancaman dan pelanggaran keamanan, dan untuk menghasilkan peringatan.

otomatisasi respons keamanan

Tindakan yang telah ditentukan dan diprogram yang dirancang untuk secara otomatis merespons atau memulihkan peristiwa keamanan. Otomatisasi ini berfungsi sebagai kontrol keamanan detektif atau responsif yang membantu Anda menerapkan praktik terbaik AWS keamanan. Contoh tindakan respons otomatis termasuk memodifikasi grup keamanan VPC, menambal instans Amazon EC2, atau memutar kredensil.

enkripsi sisi server

Enkripsi data di tujuannya, oleh Layanan AWS yang menerimanya.

kebijakan kontrol layanan (SCP)

Kebijakan yang menyediakan kontrol terpusat atas izin untuk semua akun di organisasi. AWS Organizations SCP menentukan pagar pembatas atau menetapkan batasan pada tindakan yang dapat didelegasikan oleh administrator kepada pengguna atau peran. Anda dapat menggunakan SCP sebagai daftar izin atau daftar penolakan, untuk menentukan layanan atau tindakan mana

yang diizinkan atau dilarang. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam AWS Organizations dokumentasi.

titik akhir layanan

URL titik masuk untuk file Layanan AWS. Anda dapat menggunakan endpoint untuk terhubung secara terprogram ke layanan target. Untuk informasi selengkapnya, lihat Layanan AWS titik akhir di Referensi Umum AWS.

perjanjian tingkat layanan (SLA)

Perjanjian yang menjelaskan apa yang dijanjikan oleh tim TI untuk diberikan kepada pelanggan mereka, seperti uptime dan kinerja layanan.

indikator tingkat layanan (SLI)

Pengukuran aspek kinerja layanan, seperti tingkat kesalahan, ketersediaan, atau throughputnya.

tujuan tingkat layanan (SLO)

Metrik target yang mewakili kesehatan layanan, yang diukur dengan indikator tingkat layanan.

model tanggung jawab bersama

Model yang menjelaskan tanggung jawab yang Anda bagikan AWS untuk keamanan dan kepatuhan cloud. AWS bertanggung jawab atas keamanan cloud, sedangkan Anda bertanggung jawab atas keamanan di cloud. Untuk informasi selengkapnya, lihat Model tanggung jawab bersama.

Bayangan AI

Aplikasi AI yang tidak sah dibuat atau digunakan di luar saluran yang diatur dalam suatu organisasi.

SIEM

Lihat informasi keamanan dan sistem manajemen acara.

titik kegagalan tunggal (SPOF)

Kegagalan dalam satu komponen penting dari aplikasi yang dapat mengganggu sistem.

SLA

Lihat perjanjian tingkat layanan.

SLI

Lihat indikator tingkat layanan.

SLO

Lihat tujuan tingkat layanan.

model split-and-seed

Pola untuk menskalakan dan mempercepat proyek modernisasi. Ketika fitur baru dan rilis produk didefinisikan, tim inti berpisah untuk membuat tim produk baru. Ini membantu meningkatkan kemampuan dan layanan organisasi Anda, meningkatkan produktivitas pengembang, dan mendukung inovasi yang cepat. Untuk informasi lebih lanjut, lihat Pendekatan bertahap untuk memodernisasi aplikasi di. AWS Cloud

SPOF

Lihat satu titik kegagalan.

skema bintang

Struktur organisasi database yang menggunakan satu tabel fakta besar untuk menyimpan data transaksional atau terukur dan menggunakan satu atau lebih tabel dimensi yang lebih kecil untuk menyimpan atribut data. Struktur ini dirancang untuk digunakan dalam gudang data atau untuk tujuan intelijen bisnis.

pola ara pencekik

Pendekatan untuk memodernisasi sistem monolitik dengan menulis ulang secara bertahap dan mengganti fungsionalitas sistem sampai sistem warisan dapat dinonaktifkan. Pola ini menggunakan analogi pohon ara yang tumbuh menjadi pohon yang sudah mapan dan akhirnya mengatasi dan menggantikan inangnya. Pola ini diperkenalkan oleh Martin Fowler sebagai cara untuk mengelola risiko saat menulis ulang sistem monolitik. Untuk contoh cara menerapkan pola ini, lihat Memodernisasi layanan web ASP.NET Microsoft (ASMX) lama secara bertahap menggunakan container dan Amazon API Gateway.

subnet

Rentang alamat IP dalam VPC Anda. Subnet harus berada di Availability Zone tunggal.

kontrol pengawasan dan akuisisi data (SCADA)

Di bidang manufaktur, sistem yang menggunakan perangkat keras dan perangkat lunak untuk memantau aset fisik dan operasi produksi.

enkripsi simetris

Algoritma enkripsi yang menggunakan kunci yang sama untuk mengenkripsi dan mendekripsi data.

pengujian sintetis

Menguji sistem dengan cara yang mensimulasikan interaksi pengguna untuk mendeteksi potensi masalah atau untuk memantau kinerja. Anda dapat menggunakan Amazon CloudWatch Synthetics untuk membuat tes ini.

sistem prompt

Teknik untuk memberikan konteks, instruksi, atau pedoman ke LLM untuk mengarahkan perilakunya. Permintaan sistem membantu mengatur konteks dan menetapkan aturan untuk interaksi dengan pengguna.

T

tag

Key-value pasangan yang bertindak sebagai metadata untuk mengatur sumber daya Anda AWS . Tag membantu Anda mengelola, mengidentifikasi, mengatur, dan memfilter sumber daya. Untuk informasi selengkapnya, lihat Menandai sumber daya AWS.

variabel target

Nilai yang Anda coba prediksi dalam ML yang diawasi. Ini juga disebut sebagai variabel hasil. Misalnya, dalam pengaturan manufaktur, variabel target bisa menjadi cacat produk.

daftar tugas

Alat yang digunakan untuk melacak kemajuan melalui runbook. Daftar tugas berisi ikhtisar runbook dan daftar tugas umum yang harus diselesaikan. Untuk setiap tugas umum, itu termasuk perkiraan jumlah waktu yang dibutuhkan, pemilik, dan kemajuan.

lingkungan uji

Lihat lingkungan.

pelatihan

Untuk menyediakan data bagi model ML Anda untuk dipelajari. Data pelatihan harus berisi jawaban yang benar. Algoritma pembelajaran menemukan pola dalam data pelatihan yang memetakan atribut data input ke target (jawaban yang ingin Anda prediksi). Ini menghasilkan model ML yang menangkap pola-pola ini. Anda kemudian dapat menggunakan model ML untuk membuat prediksi pada data baru yang Anda tidak tahu targetnya.

alat

Fungsi atau API yang dapat dipanggil agen untuk melakukan operasi di sistem eksternal.

gerbang transit

Hub transit jaringan yang dapat Anda gunakan untuk menghubungkan VPC dan jaringan lokal Anda. Untuk informasi selengkapnya, lihat Apa itu gateway transit dalam AWS Transit Gateway dokumentasi.

alur kerja berbasis batang

Pendekatan di mana pengembang membangun dan menguji fitur secara lokal di cabang fitur dan kemudian menggabungkan perubahan tersebut ke cabang utama. Cabang utama kemudian dibangun untuk pengembangan, praproduksi, dan lingkungan produksi, secara berurutan.

akses tepercaya

Memberikan izin ke layanan yang Anda tentukan untuk melakukan tugas di organisasi Anda di dalam AWS Organizations dan di akunnya atas nama Anda. Layanan tepercaya menciptakan peran terkait layanan di setiap akun, ketika peran itu diperlukan, untuk melakukan tugas manajemen untuk Anda. Untuk informasi selengkapnya, lihat Menggunakan AWS Organizations dengan AWS layanan lain dalam AWS Organizations dokumentasi.

penyetelan

Untuk mengubah aspek proses pelatihan Anda untuk meningkatkan akurasi model ML. Misalnya, Anda dapat melatih model ML dengan membuat set pelabelan, menambahkan label, dan kemudian mengulangi langkah-langkah ini beberapa kali di bawah pengaturan yang berbeda untuk mengoptimalkan model.

tim dua pizza

Sebuah DevOps tim kecil yang bisa Anda beri makan dengan dua pizza. Ukuran tim dua pizza memastikan peluang terbaik untuk berkolaborasi dalam pengembangan perangkat lunak.

U

waswas

Sebuah konsep yang mengacu pada informasi yang tidak tepat, tidak lengkap, atau tidak diketahui yang dapat merusak keandalan model ML prediktif. Ada dua jenis ketidakpastian:

ketidakpastian epistemik disebabkan oleh data yang terbatas dan tidak lengkap, sedangkan ketidakpastian aleatorik disebabkan oleh kebisingan dan keacakan yang melekat dalam data.

tugas yang tidak terdiferensiasi

Juga dikenal sebagai angkat berat, pekerjaan yang diperlukan untuk membuat dan mengoperasikan aplikasi tetapi itu tidak memberikan nilai langsung kepada pengguna akhir atau memberikan keunggulan kompetitif. Contoh tugas yang tidak terdiferensiasi termasuk pengadaan, pemeliharaan, dan perencanaan kapasitas.

lingkungan atas

Lihat lingkungan.

V

menyedot debu

Operasi pemeliharaan database yang melibatkan pembersihan setelah pembaruan tambahan untuk merebut kembali penyimpanan dan meningkatkan kinerja.

kendali versi

Proses dan alat yang melacak perubahan, seperti perubahan kode sumber dalam repositori.

Peering VPC

Koneksi antara dua VPC yang memungkinkan Anda merutekan lalu lintas dengan menggunakan alamat IP pribadi. Untuk informasi selengkapnya, lihat Apa itu peering VPC di dokumentasi VPC Amazon.

kerentanan

Kelemahan perangkat lunak atau perangkat keras yang membahayakan keamanan sistem.

W

cache hangat

Cache buffer yang berisi data terkini dan relevan yang sering diakses. Instance database dapat membaca dari cache buffer, yang lebih cepat daripada membaca dari memori utama atau disk.

data hangat

Data yang jarang diakses. Saat menanyakan jenis data ini, kueri yang cukup lambat biasanya dapat diterima.

fungsi jendela

Fungsi SQL yang melakukan perhitungan pada sekelompok baris yang berhubungan dengan catatan saat ini. Fungsi jendela berguna untuk memproses tugas, seperti menghitung rata-rata bergerak atau mengakses nilai baris berdasarkan posisi relatif dari baris saat ini.

beban kerja

Kumpulan sumber daya dan kode yang memberikan nilai bisnis, seperti aplikasi yang dihadapi pelanggan atau proses backend.

aliran kerja

Grup fungsional dalam proyek migrasi yang bertanggung jawab atas serangkaian tugas tertentu. Setiap alur kerja independen tetapi mendukung alur kerja lain dalam proyek. Misalnya, alur kerja portofolio bertanggung jawab untuk memprioritaskan aplikasi, perencanaan gelombang, dan mengumpulkan metadata migrasi. Alur kerja portofolio mengirimkan aset ini ke alur kerja migrasi, yang kemudian memigrasikan server dan aplikasi.

CACING

Lihat menulis sekali, baca banyak.

WQF

Lihat AWS Kerangka Kualifikasi Beban Kerja.

tulis sekali, baca banyak (WORM)

Model penyimpanan yang menulis data satu kali dan mencegah data dihapus atau dimodifikasi. Pengguna yang berwenang dapat membaca data sebanyak yang diperlukan, tetapi mereka tidak dapat mengubahnya. Infrastruktur penyimpanan data ini dianggap tidak dapat diubah.

Z

eksploitasi zero-day

Serangan, biasanya malware, yang memanfaatkan kerentanan zero-day.

kerentanan zero-day

Cacat atau kerentanan yang tak tanggung-tanggung dalam sistem produksi. Aktor ancaman dapat menggunakan jenis kerentanan ini untuk menyerang sistem. Pengembang sering menyadari kerentanan sebagai akibat dari serangan tersebut.

bisikan zero-shot

Memberikan LLM dengan instruksi untuk melakukan tugas tetapi tidak ada contoh (tembakan) yang dapat membantu membimbingnya. LLM harus menggunakan pengetahuan pra-terlatih untuk menangani tugas. Efektivitas bidikan nol tergantung pada kompleksitas tugas dan kualitas prompt. Lihat juga beberapa bidikan yang diminta.

aplikasi zombie

Aplikasi yang memiliki CPU rata-rata dan penggunaan memori di bawah 5 persen. Dalam proyek migrasi, adalah umum untuk menghentikan aplikasi ini.

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