Yayasan AI agen pada AWS

AWS Bimbingan Preskriptif

AWS Bimbingan Preskriptif: Yayasan AI agen pada AWS

Table of Contents

Yayasan AI agen pada AWS

Aaron Sempf, Amazon Web Services

Juli 2025 (sejarah dokumen)

Dalam dunia sistem yang semakin cerdas, terdistribusi, dan otonom, konsep agen — entitas yang dapat memahami lingkungannya, bernalar tentang keadaannya, dan bertindak dengan niat — telah menjadi dasar. Agen bukan hanya program yang menjalankan instruksi; mereka adalah entitas yang berorientasi pada tujuan, sadar konteks yang membuat keputusan atas nama pengguna, sistem, atau organisasi. Kemunculan mereka mencerminkan pergeseran dalam cara Anda membangun dan berpikir tentang perangkat lunak: pergeseran dari logika prosedural dan otomatisasi reaktif ke sistem yang beroperasi dengan otonomi dan tujuan.

Di persimpangan AI, sistem terdistribusi, dan rekayasa perangkat lunak terletak paradigma kuat yang dikenal sebagai AI agen. Generasi baru sistem cerdas ini terdiri dari agen perangkat lunak yang mampu melakukan perilaku adaptif, koordinasi kompleks, dan pengambilan keputusan yang didelegasikan.

Panduan ini memperkenalkan prinsip-prinsip yang mendefinisikan agen perangkat lunak modern dan menguraikan evolusi mereka menuju AI agen. Untuk menjelaskan pergeseran ini, panduan ini memberikan latar belakang konseptual dan kemudian melacak evolusi agen perangkat lunak ke AI agen:

- Pengantar agen perangkat lunak mendefinisikan agen perangkat lunak, membandingkannya dengan komponen perangkat lunak tradisional, dan memperkenalkan karakteristik penting yang membedakan perilaku agen dari otomatisasi tradisional dengan menggambar dari kerangka kerja yang sudah mapan.
- Tujuan agen perangkat lunak memeriksa mengapa agen perangkat lunak ada, peran apa yang mereka penuhi, masalah apa yang mereka pecahkan, dan bagaimana mereka memungkinkan delegasi cerdas, mengurangi beban kognitif, dan mendukung perilaku adaptif dalam lingkungan yang dinamis.
- Evolusi agen perangkat lunak melacak tonggak intelektual dan teknologi yang membentuk agen perangkat lunak, dari konsep awal otonomi dan konkurensi hingga munculnya sistem multi-agen dan arsitektur agen formal, menghasilkan konvergensi dengan AI generatif.
- Agen perangkat lunak untuk agen AI memperkenalkan AI agen sebagai puncak dari kemajuan puluhan tahun yang menggabungkan model agen terdistribusi dengan model dasar, komputasi

tanpa server, dan protokol orkestrasi. Bagian ini menjelaskan bagaimana konvergensi ini memungkinkan generasi baru agen cerdas yang menggunakan alat yang beroperasi dengan otonomi, asinkron, dan agensi sejati dalam skala besar.

Audiens yang dituju

Panduan ini dirancang untuk arsitek, pengembang, dan pemimpin teknologi yang ingin memahami sejarah, konsep utama, dan evolusi agen perangkat lunak untuk agen AI sebelum mereka mengadopsi teknologi ini untuk solusi cloud modern. AWS

Tujuan

Mengadopsi arsitektur agen membantu organisasi:

- Mempercepat waktu untuk menilai: Mengotomatiskan dan menskalakan pekerjaan pengetahuan, dan mengurangi upaya manual dan latensi.
- Tingkatkan keterlibatan pelanggan: Memberikan asisten cerdas di seluruh domain.
- Mengurangi biaya operasional: Mengotomatiskan arus keputusan yang sebelumnya membutuhkan input atau pengawasan manusia.
- Mendorong inovasi dan diferensiasi: Bangun produk cerdas yang beradaptasi, belajar, dan bersaing secara real time.
- Memodernisasi alur kerja lama: Bingkai ulang skrip dan monolit menjadi agen penalaran modular.

Tentang seri konten ini

Panduan ini adalah bagian dari serangkaian publikasi yang menyediakan cetak biru arsitektur dan panduan teknis untuk membangun agen perangkat lunak berbasis AI. AWS Seri ini meliputi:

- Mengoperasionalkan AI agen pada AWS
- Yayasan AI agen pada AWS (panduan ini)
- Pola dan alur kerja AI agen di AWS
- Kerangka kerja, protokol, dan alat AI agen di AWS
- Membangun arsitektur tanpa server untuk AI agen di AWS
- Membangun arsitektur multi-tenant untuk AI agen di AWS

Untuk informasi selengkapnya tentang seri konten ini, lihat Agentic AI.

Pengantar agen perangkat lunak

Konsep agen perangkat lunak telah berkembang secara signifikan dari fondasinya di entitas otonom pada 1960-an menjadi eksplorasi formal pada awal 1990-an. Ketika sistem digital tumbuh semakin kompleks — dari skrip deterministik hingga aplikasi adaptif dan cerdas — agen perangkat lunak telah menjadi blok bangunan penting untuk memungkinkan perilaku otonom, sadar konteks, dan berbasis tujuan dalam sistem komputasi. Dalam konteks arsitektur cloud-native dan AI yang ditingkatkan, terutama dengan munculnya AI generatif, model bahasa besar (LLMs), dan platform seperti Amazon Bedrock, agen perangkat lunak sedang didefinisikan ulang melalui lensa kemampuan dan skala baru.

Pengantar ini diambil dari karya mani Software Agents: An Overview oleh Hyacinth S. Nwana (Nwana 1996). Ini mendefinisikan agen perangkat lunak, membahas akar konseptual mereka, dan memperluas diskusi ke dalam kerangka kontemporer untuk mendefinisikan tiga prinsip menyeluruh agen perangkat lunak modern: otonomi, asinkron, dan agensi. Prinsip-prinsip ini membedakan agen perangkat lunak dari jenis layanan atau aplikasi lain, dan memungkinkan agen ini beroperasi dengan tujuan, ketahanan, dan kecerdasan dalam lingkungan real-time yang terdistribusi.

Dalam bagian ini

- Dari otonomi hingga intelijen terdistribusi
- Tipologi Nwana dan munculnya agen perangkat lunak
- Tiga pilar agen perangkat lunak modern

Dari otonomi hingga intelijen terdistribusi

Sebelum istilah agen perangkat lunak memasuki arus utama, penelitian komputasi awal mengeksplorasi gagasan entitas digital otonom, yang merupakan sistem yang mampu bertindak secara independen, bereaksi terhadap input, dan membuat keputusan berdasarkan aturan atau tujuan internal. Ide-ide awal ini meletakkan dasar konseptual untuk apa yang akan menjadi paradigma agen. (Untuk garis waktu historis, lihat bagian Evolusi agen perangkat lunak nanti dalam panduan ini.)

Konsep awal otonomi

Gagasan tentang mesin atau program yang bertindak secara independen dari operator manusia telah menggelitik perancang sistem selama beberapa dekade. Pekerjaan awal dalam sibernetika,

kecerdasan buatan, dan sistem kontrol memeriksa bagaimana perangkat lunak dapat menunjukkan perilaku yang mengatur diri sendiri, merespons perubahan secara dinamis, dan beroperasi tanpa pengawasan manusia yang berkelanjutan.

Ide-ide ini memperkenalkan otonomi sebagai atribut inti dari sistem cerdas dan mengatur panggung untuk munculnya perangkat lunak yang dapat memutuskan dan bertindak, bukan hanya bereaksi atau mengeksekusi.

Model aktor dan eksekusi asinkron

Pada 1970-an, model aktor, yang diperkenalkan dalam paper A Universal Modular ACTOR Formalism for Artificial Intelligence (Hewitt et al. 1973), menyediakan kerangka kerja formal untuk berpikir tentang komputasi yang digerakkan oleh pesan yang terdesentralisasi. Dalam model ini, aktor adalah entitas independen yang berkomunikasi secara eksklusif dengan mengirimkan pesan asinkron, dan memungkinkan sistem yang dapat diskalakan, bersamaan, dan toleran terhadap kesalahan.

Model aktor menekankan tiga atribut utama yang terus mempengaruhi desain agen modern:

- Isolasi keadaan dan perilaku
- Interaksi asinkron antar entitas
- Pembuatan dan pendelegasian tugas yang dinamis

Atribut ini selaras dengan kebutuhan sistem terdistribusi dan menggambarkan karakteristik operasional agen perangkat lunak di lingkungan cloud-native.

Intelijen terdistribusi dan sistem multi-agen

Ketika sistem komputasi menjadi lebih saling berhubungan setelah 1960-an, para peneliti mengeksplorasi kecerdasan buatan terdistribusi (DAI). Bidang ini berfokus pada bagaimana beberapa entitas otonom dapat bekerja secara kolaboratif atau kompetitif di seluruh sistem. DAI mengarah pada pengembangan sistem multi-agen, di mana setiap agen memiliki tujuan, persepsi, dan penalaran lokal tetapi juga beroperasi dalam lingkungan yang lebih luas dan saling berhubungan.

Visi kecerdasan terdistribusi ini, di mana pengambilan keputusan terdesentralisasi dan perilaku yang muncul muncul dari interaksi agen, tetap menjadi pusat bagaimana sistem berbasis agen modern dipahami dan dibangun.

Tipologi Nwana dan munculnya agen perangkat lunak

Formalisasi konsep agen perangkat lunak pada pertengahan 1990-an menandai titik balik dalam evolusi sistem cerdas. Di antara kontribusi yang paling berpengaruh untuk formalisasi ini adalah mani paper Hyacinth S. Nwana, Software Agents: An Overview (Nwana 1996), yang menyediakan salah satu kerangka komprehensif pertama untuk mengkategorikan dan memahami agen perangkat lunak di berbagai dimensi.

Dalam paper ini, Nwana mensurvei keadaan penelitian agen perangkat lunak dan mengidentifikasi perbedaan yang berkembang dalam bagaimana agen didefinisikan dan diimplementasikan. Paper ini menyoroti perlunya kerangka kerja konseptual umum dan mengusulkan tipologi yang mengklasifikasikan agen menurut kemampuan kunci mereka. Ini meninjau sistem agen perwakilan dari akademisi dan industri, membedakan agen dari program dan objek tradisional, dan menguraikan tantangan dan peluang dalam komputasi berbasis agen.

Nwana menekankan bahwa agen perangkat lunak bukanlah konsep monolitik tetapi ada di sepanjang spektrum kecanggihan dan kemampuan. Tipologi berfungsi untuk memperjelas lanskap ini dan memandu desain dan penelitian masa depan.

Nwana mendefinisikan agen perangkat lunak sebagai entitas perangkat lunak yang berfungsi terus menerus dan otonom dalam lingkungan tertentu, yang sering dihuni oleh agen dan proses lain. Definisi ini menekankan dua karakteristik utama:

- Kontinuitas: Agen beroperasi terus-menerus dari waktu ke waktu, tanpa memerlukan campur tangan manusia yang konstan.
- Otonomi: Agen memiliki kemampuan untuk membuat keputusan dan menindaklanjutinya secara mandiri, berdasarkan persepsinya terhadap lingkungan.

Definisi ini, dikombinasikan dengan tipologi agen Nwana, menekankan otoritas yang didelegasikan (melalui otonomi) dan proaktif sebagai karakteristik dasar agen. Ini membedakan antara agen dan subrutin atau layanan dengan menyoroti kemampuan agen untuk bertindak secara independen atas nama entitas lain dan untuk memulai perilaku dalam mengejar tujuan, bukan hanya menanggapi perintah langsung.

Tipologi agen Nwana

Untuk lebih membedakan antara berbagai jenis agen, Nwana memperkenalkan sistem klasifikasi berdasarkan enam atribut utama:

- Otonomi: Agen beroperasi tanpa intervensi langsung dari manusia atau orang lain.
- Kemampuan sosial: Agen berinteraksi dengan agen atau manusia lain dengan menggunakan mekanisme komunikasi.
- Reaktivitas: Agen merasakan lingkungannya dan merespons secara tepat waktu.
- Proaktif: Agen menunjukkan perilaku yang diarahkan pada tujuan dengan mengambil inisiatif.
- Kemampuan beradaptasi dan belajar: Agen meningkatkan kinerjanya dari waktu ke waktu melalui pengalaman.
- Mobilitas: Agen dapat bergerak melintasi lingkungan sistem atau jaringan yang berbeda.

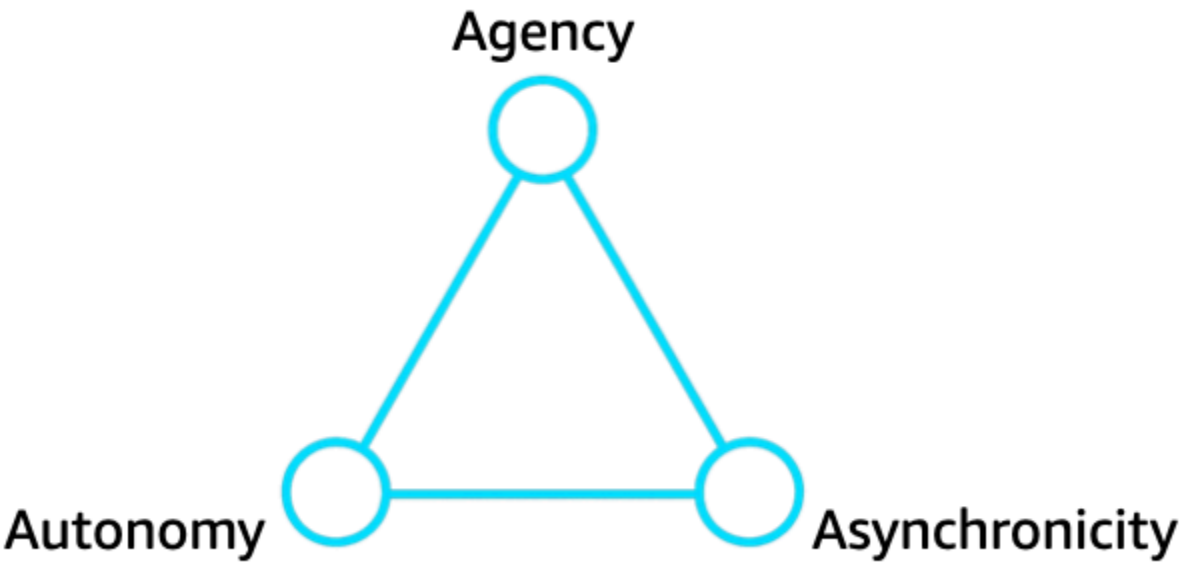
Dari tipologi hingga prinsip agen modern

Karya Nwana berfungsi sebagai taksonomi dan lensa dasar di mana komunitas komputasi dapat mengevaluasi bentuk-bentuk agensi yang berkembang dalam perangkat lunak. Penekanannya pada otonomi, proaktif, dan konsep bertindak atas nama pengguna atau sistem meletakkan dasar bagi apa yang sekarang kita anggap sebagai perilaku agen.

Meskipun teknologi dan lingkungan telah berubah, terutama dengan munculnya AI generatif, infrastruktur tanpa server, dan kerangka kerja orkestrasi multi-agen, wawasan dasar dari pekerjaan Nwana tetap relevan. Mereka menyediakan jembatan kritis antara teori agen awal dan tiga pilar modern agen perangkat lunak.

Tiga pilar agen perangkat lunak modern

Dalam konteks platform bertenaga AI saat ini, arsitektur layanan mikro, dan sistem berbasis peristiwa, agen perangkat lunak dapat didefinisikan oleh tiga prinsip yang saling bergantung yang membedakannya dari layanan standar atau skrip otomatisasi: otonomi, asinkron, dan agensi. Dalam ilustrasi berikut dan diagram berikutnya, segitiga mewakili tiga pilar agen perangkat lunak modern ini.



Otonomi

Agen modern beroperasi secara independen. Mereka membuat keputusan berdasarkan keadaan internal dan konteks lingkungan tanpa memerlukan dorongan manusia. Hal ini memungkinkan mereka untuk bereaksi terhadap data secara real-time, mengelola siklus hidup mereka sendiri, dan menyesuaikan perilaku mereka berdasarkan sasaran dan masukan situasional.

Otonomi adalah dasar dari perilaku agen. Hal ini memungkinkan agen untuk berfungsi tanpa pengawasan terus menerus atau aliran kontrol hardcode.

Asinkronisitas

Agen pada dasarnya asinkron. Ini berarti bahwa mereka merespons peristiwa, sinyal, dan rangsangan saat terjadi, tanpa bergantung pada pemblokiran panggilan atau alur kerja linier. Karakteristik ini memungkinkan komunikasi yang dapat diskalakan dan tidak memblokir, responsif di lingkungan terdistribusi, dan kopling longgar antar komponen.

Melalui asinkron, agen dapat berpartisipasi dalam sistem real-time dan berkoordinasi dengan layanan atau agen lain secara lancar dan efisien.

Agensi sebagai prinsip yang menentukan

Otonomi dan asinkronisitas diperlukan, tetapi fitur-fitur ini saja tidak cukup untuk menjadikan sistem agen perangkat lunak sejati. Pembeda kritis adalah agensi, yang memperkenalkan:

- Perilaku yang diarahkan pada tujuan: Agen mengejar tujuan dan mengevaluasi kemajuan ke arah mereka.
- Pengambilan keputusan: Agen menilai opsi dan memilih tindakan berdasarkan aturan, model, atau kebijakan yang dipelajari.
- Maksud yang didelegasikan: Agen bertindak atas nama seseorang, sistem, atau organisasi dan memiliki tujuan yang tertanam.
- Penalaran kontekstual: Agen menggabungkan memori atau model lingkungan mereka untuk memandu perilaku secara cerdas.

Sistem yang otonom dan asinkron mungkin masih merupakan layanan reaktif. Apa yang membuatnya menjadi agen perangkat lunak adalah kemampuannya untuk bertindak dengan niat dan tujuan, untuk menjadi agen.

Agensi dengan tujuan

Prinsip-prinsip otonomi, asinkron, dan agensi memungkinkan sistem beroperasi secara cerdas, adaptif, dan independen di seluruh lingkungan terdistribusi. Prinsip-prinsip ini berakar pada dekade evolusi konseptual dan arsitektur, dan sekarang mendukung banyak sistem AI paling canggih yang sedang dibangun saat ini.

Di era baru AI generatif, orkestrasi berorientasi pada tujuan, dan kolaborasi multi-agen, penting untuk memahami apa yang membuat agen perangkat lunak benar-benar agen. Mengakui agensi sebagai karakteristik yang menentukan membantu kita bergerak melampaui otomatisasi dan memasuki ranah kecerdasan otonom dengan tujuan.

Tujuan agen perangkat lunak

Karena sistem modern telah menjadi semakin kompleks, terdistribusi, dan cerdas, peran agen perangkat lunak telah menjadi terkenal di seluruh domain yang berkisar dari operasi otonom hingga teknologi bantuan pengguna. Tapi apa tujuan yang mendasari agen perangkat lunak? Mengapa kita merancang sistem yang melampaui skrip, layanan, atau model statis, dan sebagai gantinya mendelegasikan tugas ke entitas yang mampu memahami, bernalar, dan bertindak?

Bagian ini mengeksplorasi tujuan mendasar agen perangkat lunak: untuk memungkinkan pendelegasian tugas yang cerdas dalam lingkungan yang dinamis, dengan fokus pada otonomi, kemampuan beradaptasi, dan tindakan yang bertujuan. Ini memperkenalkan dasar konseptual agen perangkat lunak, melacak struktur kognitif mereka, dan menguraikan masalah dunia nyata yang secara unik mereka dilengkapi untuk memecahkan.

Dalam bagian ini

Dari model aktor hingga kognisi agen

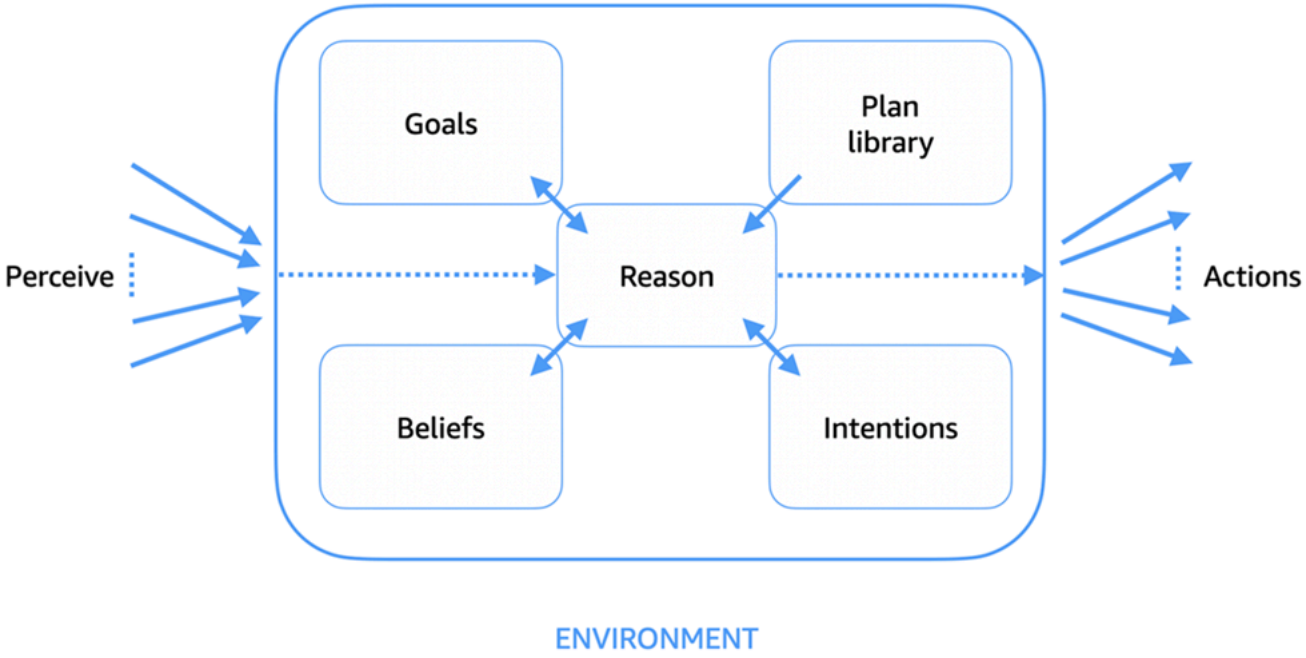
Tujuan dan struktur agen perangkat lunak didasarkan pada ide-ide yang muncul dari model komputasi awal, khususnya model aktor yang diperkenalkan oleh Carl Hewitt pada 1970-an (Hewitt et al. 1973).

Model aktor memperlakukan komputasi sebagai kumpulan entitas independen yang mengeksekusi secara bersamaan yang disebut aktor. Setiap aktor merangkum keadaannya sendiri, berinteraksi semata-mata melalui pesan asinkron, dan dapat membuat aktor baru dan mendelegasikan tugas.

Model ini memberikan landasan konseptual untuk penalaran, reaktivitas, dan isolasi yang terdesentralisasi—yang semuanya mendukung arsitektur perilaku agen perangkat lunak modern.

Fungsi agen: memahami, bernalar, bertindak

Inti dari setiap agen perangkat lunak adalah siklus kognitif yang sering digambarkan sebagai loop persepsi, alasan, tindakan. Proses ini diilustrasikan dalam diagram berikut. Ini mendefinisikan bagaimana agen beroperasi secara otonom dalam lingkungan yang dinamis.



- Persepsi: Agen mengumpulkan informasi (misalnya, peristiwa, input sensor, atau sinyal API) dari lingkungan dan memperbarui keadaan internal atau keyakinan mereka.
- Alasan: Agen menganalisis keyakinan, tujuan, dan pengetahuan kontekstual saat ini dengan menggunakan perpustakaan rencana atau sistem logika. Proses ini mungkin melibatkan prioritas tujuan, resolusi konflik, atau pemilihan niat.
- Tindakan: Agen memilih dan melaksanakan tindakan yang membuat mereka lebih dekat untuk mencapai tujuan yang didelegasikan.

Arsitektur ini mendukung kemampuan agen untuk berfungsi di luar pemrograman yang kaku dan memungkinkan perilaku yang fleksibel, peka konteks, dan diarahkan pada tujuan. Ini membentuk kerangka mental yang memandu tujuan yang lebih luas dari agen perangkat lunak.

Kolaborasi dan intensionalitas otonom

Tujuan agen perangkat lunak adalah untuk membawa otonomi, kesadaran konteks, dan delegasi cerdas ke komputasi modern. Karena agen dibangun di atas prinsip-prinsip model aktor dan diwujudkan dalam siklus persepsi, alasan, tindakan, mereka memungkinkan sistem yang tidak hanya reaktif, tetapi proaktif dan terarah.

Agen memberdayakan perangkat lunak untuk memutuskan, beradaptasi, dan bertindak dalam lingkungan yang kompleks. Mereka mewakili pengguna, menafsirkan tujuan, dan

mengimplementasikan tugas dengan kecepatan mesin. Saat kita bergerak lebih dalam ke era AI agen, agen perangkat lunak menjadi antarmuka operasional antara niat manusia dan tindakan digital cerdas.

Mendelegasikan niat

Tidak seperti komponen perangkat lunak tradisional, agen perangkat lunak ada untuk bertindak atas nama sesuatu yang lain: pengguna, sistem lain, atau layanan tingkat yang lebih tinggi. Mereka membawa niat yang didelegasikan, yang berarti bahwa mereka:

- Beroperasi secara independen setelah inisiasi.
- Buat pilihan yang selaras dengan tujuan delegator.
- Arahkan ketidakpastian dan trade-off dalam eksekusi.

Agen menjembatani kesenjangan antara instruksi dan hasil, yang memungkinkan pengguna untuk mengekspresikan niat pada tingkat abstraksi yang lebih tinggi daripada memerlukan instruksi eksplisit.

Beroperasi di lingkungan yang dinamis dan tidak dapat diprediksi

Agen perangkat lunak dirancang untuk lingkungan di mana kondisi berubah secara konstan, data tiba secara real time, dan kontrol dan konteks didistribusikan.

Tidak seperti program statis yang memerlukan input yang tepat atau eksekusi sinkron, agen beradaptasi dengan lingkungan mereka dan merespons secara dinamis. Ini adalah kemampuan vital dalam infrastruktur cloud-native, komputasi tepi, jaringan Internet of Things (IoT), dan sistem pengambilan keputusan real-time.

Mengurangi beban kognitif manusia

Salah satu tujuan utama agen perangkat lunak adalah untuk mengurangi beban kognitif dan operasional pada manusia. Agen dapat:

- Terus memantau sistem dan alur kerja.
- Mendeteksi dan merespons kondisi yang telah ditentukan atau muncul.
- Otomatiskan keputusan berulang dan volume tinggi.
- Bereaksi terhadap perubahan lingkungan dengan latensi minimal.

Ketika pengambilan keputusan bergeser dari pengguna ke agen, sistem menjadi lebih responsif, tangguh, dan manusia-sentris, dan dapat beradaptasi secara real time dengan informasi atau gangguan baru. Hal ini memungkinkan perputaran reaksi yang lebih cepat serta kontinuitas operasional yang lebih besar di lingkungan dengan kompleksitas tinggi atau skala tinggi. Hasilnya adalah pergeseran fokus manusia, dari pengambilan keputusan tingkat mikro ke pengawasan strategis dan pemecahan masalah yang kreatif.

Mengaktifkan intelijen terdistribusi

Kemampuan agen perangkat lunak untuk beroperasi secara individu atau kolektif memungkinkan desain sistem multi-agen (MAS) yang berkoordinasi lintas lingkungan atau organisasi. Sistem ini dapat mendistribusikan tugas secara cerdas dan bernegosiasi, bekerja sama, atau bersaing menuju tujuan gabungan.

Misalnya, dalam sistem rantai pasokan global, agen individu mengelola pabrik, pengiriman, gudang, dan pengiriman jarak jauh. Setiap agen beroperasi dengan otonomi lokal: Agen pabrik mengoptimalkan produksi berdasarkan kendala sumber daya, agen gudang menyesuaikan arus inventaris secara real time, dan agen pengiriman mengalihkan pengiriman berdasarkan lalu lintas dan ketersediaan pelanggan.

Agen ini berkomunikasi dan berkoordinasi secara dinamis, dan beradaptasi dengan gangguan seperti penundaan pelabuhan atau kegagalan truk tanpa kontrol terpusat. Kecerdasan keseluruhan sistem muncul dari interaksi ini dan memungkinkan logistik yang tangguh dan dioptimalkan yang berada di luar kemampuan satu komponen.

Dalam model ini, agen bertindak sebagai simpul dalam struktur intelijen yang lebih luas. Mereka membentuk sistem yang muncul yang mampu memecahkan masalah yang tidak dapat ditangani oleh komponen tunggal sendiri.

Bertindak dengan tujuan, bukan hanya reaksi

Otomatisasi saja tidak cukup dalam sistem yang kompleks. Tujuan yang menentukan dari agen perangkat lunak adalah untuk bertindak dengan tujuan dan untuk mengevaluasi tujuan, menimbang konteks, dan membuat pilihan berdasarkan informasi. Ini berarti bahwa agen perangkat lunak mengejar tujuan alih-alih hanya menanggapi pemicu. Mereka dapat merevisi keyakinan dan niat berdasarkan pengalaman atau umpan balik. Dalam konteks ini, keyakinan mengacu pada representasi internal agen terhadap lingkungan (misalnya, “paket X ada di gudang A”), berdasarkan persepsinya (input dan sensor). Niat mengacu pada rencana yang dipilih agen untuk mencapai tujuan

(misalnya, “gunakan rute pengiriman B dan beri tahu penerima”). Agen juga dapat meningkatkan, menunda, atau menyesuaikan tindakan yang diperlukan.

Kesengajaan inilah yang membuat agen perangkat lunak tidak hanya pelaksana reaktif, tetapi kolaborator otonom dalam sistem cerdas.

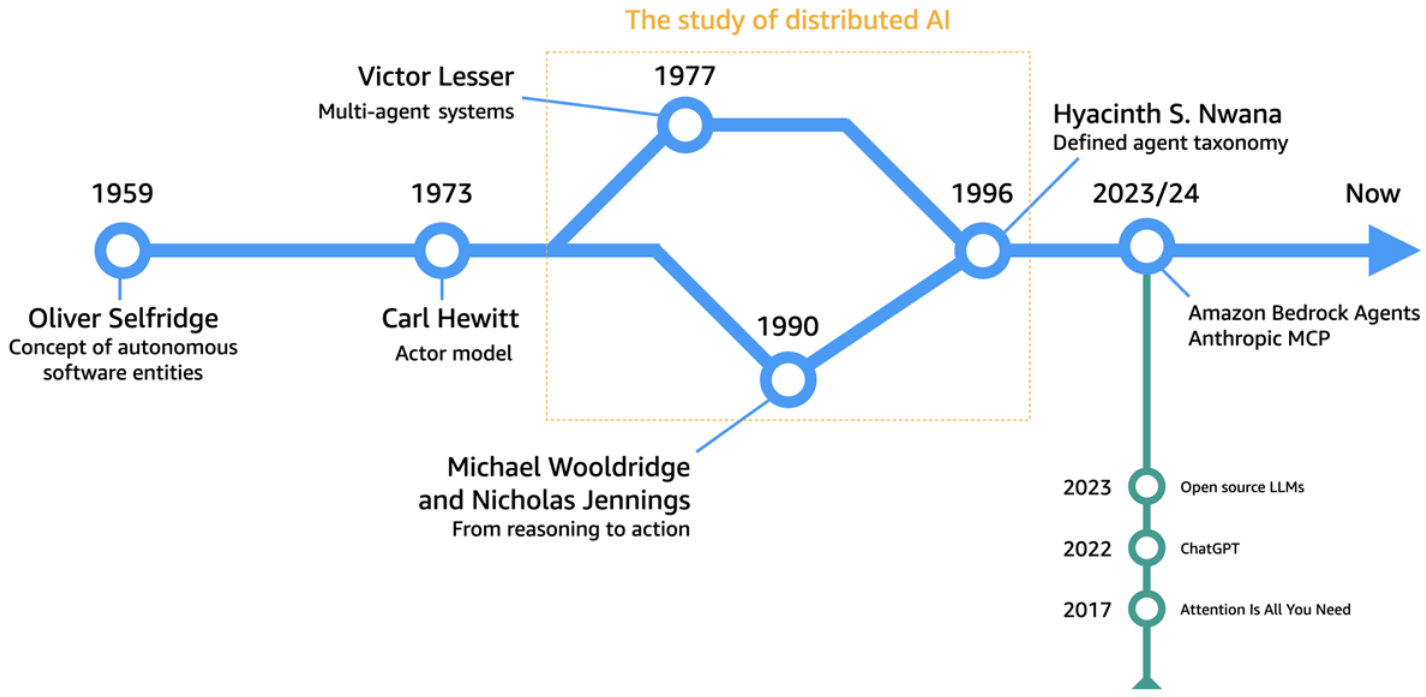
Evolusi agen perangkat lunak

Perjalanan dari sistem otomatis sederhana ke agen perangkat lunak yang cerdas, otonom, dan diarahkan pada tujuan mencerminkan dekade evolusi dalam ilmu komputer, kecerdasan buatan, dan sistem terdistribusi.

Evolusi ini diikuti oleh munculnya pembelajaran mesin, yang menggeser paradigma dari aturan buatan tangan ke pengenalan pola statistik. Sistem ini dapat belajar dari data dan memungkinkan kemajuan dalam persepsi, klasifikasi, dan pengambilan keputusan.

Model bahasa besar (LLMs) mewakili konvergensi skala, arsitektur, dan pembelajaran tanpa pengawasan. LLMs dapat bernalar, menghasilkan, dan menyesuaikan tugas dengan sedikit atau tanpa pelatihan khusus tugas. LLMs Dengan menggabungkan infrastruktur cloud-native yang dapat diskalakan dan arsitektur yang dapat dikomposisi, kami sekarang mencapai visi penuh AI agen: agen perangkat lunak cerdas yang dapat beroperasi dengan otonomi, kesadaran konteks, dan kemampuan beradaptasi pada skala perusahaan.

Bagian ini mengeksplorasi sejarah agen perangkat lunak dari teori dasar hingga praktik modern, seperti yang diilustrasikan dalam diagram berikut. Ini menyoroti konvergensi kecerdasan buatan terdistribusi (DAI) dan AI generatif berbasis transformer, dan mengidentifikasi tonggak kunci yang telah membentuk munculnya AI agen.



Dalam bagian ini

Yayasan agen perangkat lunak

1959 - Oliver Selfridge: lahirnya otonomi dalam perangkat lunak

Akar agen perangkat lunak ditelusuri kembali ke Oliver Selfridge, yang memperkenalkan konsep entitas perangkat lunak otonom (setan) —program yang mampu memahami lingkungan mereka dan bertindak secara independen (Selfridge 1959). Karya awalnya dalam persepsi dan pembelajaran mesin meletakkan dasar filosofis untuk gagasan agen masa depan sebagai sistem yang independen dan cerdas.

1973 - Carl Hewitt: model aktor

Kemajuan penting datang dengan model aktor Carl Hewitt (Hewitt et al. 1973), yang merupakan model komputasi formal yang menggambarkan agen sebagai entitas independen dan bersamaan. Dalam model ini, agen dapat merangkum keadaan dan perilaku mereka sendiri, berkomunikasi dengan menggunakan pengiriman pesan asinkron, dan secara dinamis membuat aktor lain dan mendelegasikan tugas kepada mereka.

Model aktor memberikan landasan teoretis dan paradigma arsitektur untuk sistem berbasis agen terdistribusi. Model ini menggambarkan implementasi konkurensi modern seperti bahasa pemrograman Erlang dan kerangka kerja Akka.

Mendewasakan bidang: dari penalaran ke tindakan

1977 — Victor Lesser: sistem multi-agen

Pada akhir 1970-an, kecerdasan buatan terdistribusi (DAI) muncul. Itu diperjuangkan oleh Victor Lesser, yang secara luas diakui untuk merintis sistem multi-agen (MAS). Karyanya berfokus pada bagaimana entitas perangkat lunak independen dapat bekerja sama, berkoordinasi, dan bernegosiasi (lihat bagian Sumber Daya). Perkembangan ini mengarah pada sistem yang mampu memecahkan masalah kompleks secara kolektif—lompatan penting dalam membangun intelijen terdistribusi.

1990-an - Michael Wooldridge dan Nicholas Jennings: spektrum agen

Pada 1990-an, bidang intelijen terdistribusi telah matang dengan kontribusi dari para peneliti seperti Michael Wooldridge dan Nicholas Jennings. Para sarjana ini mengkategorikan agen di sepanjang spektrum, dari reaktif hingga deliberatif, dari sistem non-kognitif hingga agen penalaran yang didorong oleh tujuan (Wooldridge dan Jennings 1995). Pekerjaan mereka menekankan bahwa agen bukan lagi ide abstrak tetapi diterapkan di berbagai domain praktis, dari robotika hingga perangkat lunak perusahaan.

Para peneliti ini juga memperkenalkan pergeseran fokus: dari penalaran terpusat ke tindakan terdistribusi. Agen tidak lagi hanya pemikir—mereka adalah pelaku yang beroperasi dalam lingkungan waktu nyata dengan otonomi dan tujuan.

1996 - Hyacinth S. Nwana: meresmikan konsep agen

Pada tahun 1996, Hyacinth S. Nwana menerbitkan paper berpengaruh Software Agents: An Overview, yang memberikan klasifikasi agen paling komprehensif hingga saat ini. Tipologinya mencakup atribut seperti otonomi, kemampuan sosial, reaktivitas, proaktif, pembelajaran, dan mobilitas, dan dibedakan antara agen perangkat lunak dan konstruksi perangkat lunak tradisional.

Nwana juga menawarkan definisi yang sekarang diterima secara luas, diparafrasekan: Agen perangkat lunak adalah program komputer berbasis perangkat lunak yang bertindak untuk pengguna atau program lain dalam hubungan agensi, yang berasal dari gagasan delegasi.

Formalisasi ini berperan penting dalam transisi agen perangkat lunak dari konstruksi teoritis ke aplikasi dunia nyata. Ini memunculkan generasi sistem berbasis agen di berbagai bidang seperti telekomunikasi, otomatisasi alur kerja, dan asisten cerdas.

Pekerjaan Nwana berada pada titik konvergensi penelitian AI terdistribusi awal dan arsitektur operasional agen modern. Ini adalah jembatan penting antara teori kognitif agen dan penyebaran praktis mereka dalam sistem saat ini.

Garis waktu paralel: munculnya model bahasa besar

Sementara kerangka kerja agen berkembang, revolusi paralel dan konvergen terjadi dalam pemrosesan bahasa alami dan pembelajaran mesin:

- 2017 - transformer: The paper Attention Is All You Need (Vaswani et al. 2017) memperkenalkan arsitektur transformator, yang secara dramatis meningkatkan cara mesin memproses dan menghasilkan bahasa.

- 2022 - ChatGPT: OpenAI merilis antarmuka berbasis obrolan ke GPT-3.5 yang disebut ChatGPT, yang memungkinkan percakapan interaktif alami dengan sistem AI tujuan umum.
- 2023 - open source LLMs: Rilis Llama, Falcon, dan Mistral membuat model yang kuat dapat diakses secara luas dan mempercepat pengembangan kerangka kerja agen di lingkungan open source dan perusahaan.

Inovasi ini mengubah model bahasa menjadi mesin penalaran yang mampu mengurai konteks, merencanakan tindakan, dan merantai respons, dan LLMs menjadi pendukung utama agen perangkat lunak cerdas.

Garis waktu bertemu: munculnya AI agen

2023-2024 - platform agen tingkat perusahaan

Konvergensi arsitektur agen perangkat lunak terdistribusi dan berbasis transformer LLMs memuncak pada munculnya AI agen.

- Amazon Bedrock Agents memperkenalkan cara yang dikelola sepenuhnya untuk membangun agen perangkat lunak yang digerakkan oleh tujuan dan menggunakan alat dengan menggunakan model fondasi dari Amazon Bedrock.
- Model Context Protocol (MCP) dari Anthropic mendefinisikan metode untuk model bahasa besar untuk mengakses, dan berinteraksi dengan, alat eksternal, lingkungan, dan memori. Ini adalah kunci untuk perilaku kontekstual, persisten, dan otonom.

Kedua tonggak ini mewakili sintesis agensi dan intelijen. Agen tidak lagi terbatas pada alur kerja statis atau otomatisasi kaku. Mereka sekarang dapat bernalar di beberapa langkah, berkoordinasi dengan alat dan APIs, mempertahankan keadaan kontekstual, dan belajar dan beradaptasi dari waktu ke waktu.

Januari-Juni 2025 — kemampuan perusahaan yang diperluas

Pada paruh pertama tahun 2025, lanskap AI agen berkembang secara signifikan dengan kemampuan perusahaan baru. Pada Februari 2025, Anthropic merilis Claude 3.7 Sonnet, yang merupakan model penalaran hibrida pertama di pasar, dan spesifikasi MCP memperoleh adopsi luas.

Asisten pengkodean AI seperti Amazon Q Developer, Cursor, dan MCP WindSurf terintegrasi untuk membakukan pembuatan kode, analisis repositori, dan alur kerja pengembangan. Rilis MCP Maret

2025 memperkenalkan fitur siap perusahaan yang signifikan, termasuk integrasi keamanan OAuth 2.1, jenis sumber daya yang diperluas untuk akses data yang beragam, dan opsi konektivitas yang ditingkatkan melalui Streamable HTTP. Membangun yayasan ini, AWS diumumkan pada Mei 2025 bahwa mereka bergabung dengan komite pengarah MCP dan berkontribusi pada kemampuan agent-to-agent komunikasi baru. Ini semakin memperkuat posisi protokol sebagai standar industri untuk interoperabilitas AI agen.

Pada Mei 2025, AWS memperkuat opsi pelanggan untuk membangun alur kerja AI agen dengan membuka sumber kerangka kerja Strands Agents. Kerangka kerja penyedia independen dan model-agnostik ini memungkinkan pengembang untuk menggunakan model fondasi di seluruh platform sambil mempertahankan integrasi layanan yang mendalam. AWS Seperti yang disorot dalam Blog AWS Open Source, Strands Agents mengikuti filosofi desain model-first yang menempatkan model fondasi pada inti intelijen agen. Ini memudahkan pelanggan untuk membangun dan menyebarkan agen AI canggih untuk kasus penggunaan spesifik mereka.

Munculnya — AI agen

Evolusi agen perangkat lunak, dari gagasan awal otonomi hingga orkestrasi modern yang mendukung LLM, telah lama dan berlapis. Apa yang dimulai dengan visi Oliver Selfridge untuk memahami program telah tumbuh menjadi ekosistem yang kuat dari agen perangkat lunak yang cerdas, sadar konteks, dan didorong oleh tujuan yang dapat berkolaborasi, beradaptasi, dan bernalar.

Konvergensi kecerdasan buatan terdistribusi (DAI) dan AI generatif berbasis transformer menandai awal era baru di mana agen perangkat lunak tidak lagi hanya alat, tetapi aktor otonom dalam sistem cerdas.

Agentic AI mewakili evolusi berikutnya dalam sistem perangkat lunak. Ini menyediakan kelas agen cerdas yang otonom, asinkron, dan agen, dan dapat bertindak dengan niat yang didelegasikan dan beroperasi dengan sengaja dalam lingkungan yang dinamis dan terdistribusi. Agentic AI menyatukan hal-hal berikut:

- Garis keturunan arsitektur sistem multi-agen dan model aktor
- Model kognitif persepsi, alasan, tindakan
- Kekuatan generatif LLMs dan transformator
- Fleksibilitas operasional komputasi cloud-native dan tanpa server

Agen perangkat lunak untuk agen AI

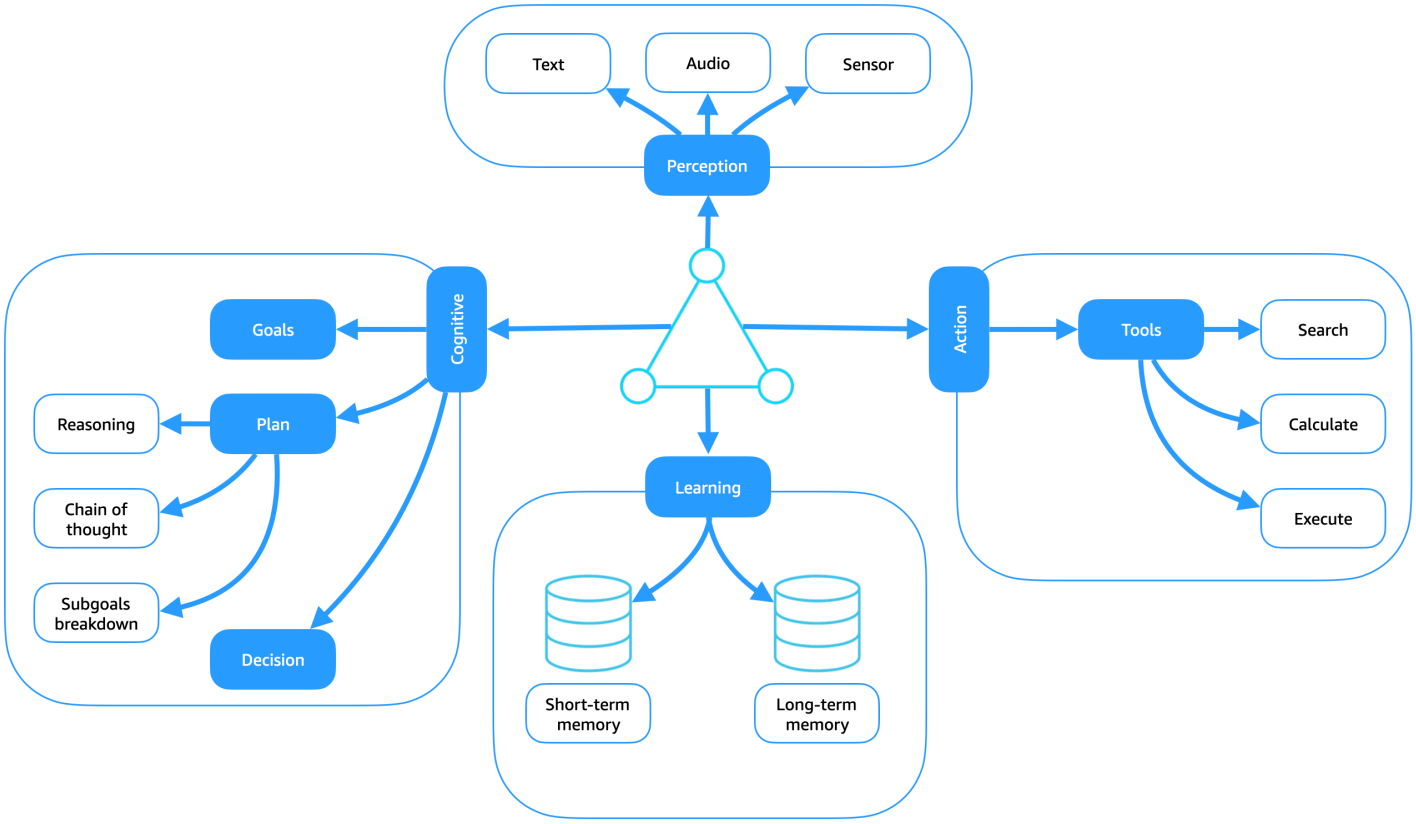
Agen perangkat lunak adalah entitas digital otonom yang dirancang untuk memahami lingkungan mereka, alasan tentang tujuan mereka, dan bertindak sesuai dengan itu. Tidak seperti program perangkat lunak tradisional yang mengikuti logika tetap, agen menyesuaikan perilaku mereka berdasarkan input kontekstual dan kerangka keputusan. Ini membuatnya ideal untuk lingkungan yang dinamis dan terdistribusi seperti sistem cloud-native, robotika, otomatisasi cerdas, dan sekarang, orkestrasi AI generatif.

Bagian ini memperkenalkan blok bangunan inti agen perangkat lunak dan menjelaskan bagaimana komponen ini berinteraksi dalam arsitektur tradisional berdasarkan model persepsi, alasan, tindakan. Ini membahas bagaimana AI generatif, terutama model bahasa besar (LLMs), telah mengubah cara agen perangkat lunak bernalar dan merencanakan. Ini menandai pergeseran mendasar dari sistem berbasis aturan ke kecerdasan AI agen berbasis data yang dipelajari.

Dalam bagian ini

Blok bangunan inti agen perangkat lunak

Diagram berikut menyajikan modul fungsional utama yang ditemukan di sebagian besar agen cerdas. Setiap komponen berkontribusi pada kemampuan agen untuk beroperasi secara mandiri di lingkungan yang kompleks.



Dalam konteks loop persepsi, alasan, tindakan, kemampuan penalaran agen didistribusikan di modul kognitif dan pembelajarannya. Melalui integrasi memori dan pembelajaran, agen mengembangkan penalaran adaptif yang didasarkan pada pengalaman masa lalu. Ketika agen bertindak dalam lingkungannya, ia menciptakan loop umpan balik yang muncul: Setiap tindakan memengaruhi persepsi masa depan, dan pengalaman yang dihasilkan dimasukkan ke dalam memori dan model internal melalui modul pembelajaran. Lingkaran persepsi, penalaran, dan tindakan yang berkelanjutan ini memungkinkan agen untuk meningkat dari waktu ke waktu dan menyelesaikan siklus persepsi, alasan, tindakan penuh.

Modul persepsi

Modul persepsi memungkinkan agen untuk berinteraksi dengan lingkungannya melalui beragam modalitas input seperti teks, audio, dan sensor. Masukan ini membentuk data mentah yang didasarkan pada semua penalaran dan tindakan. Input teks mungkin termasuk prompt bahasa alami, perintah terstruktur, atau dokumen. Input audio mencakup instruksi lisan atau suara lingkungan. Input sensor termasuk data fisik seperti umpan visual, sinyal gerak, atau koordinat GPS. Fungsi inti persepsi adalah untuk mengekstrak fitur dan representasi yang bermakna dari data mentah ini. Hal ini memungkinkan agen untuk membangun pemahaman yang akurat dan dapat ditindaklanjuti dari konteksnya saat ini. Prosesnya mungkin melibatkan ekstraksi fitur, pengenalan objek atau peristiwa,

dan interpretasi semantik, dan membentuk langkah pertama yang kritis dalam loop persepsi, alasan, tindakan. Persepsi yang efektif memastikan bahwa penalaran hilir dan pengambilan keputusan didasarkan pada kesadaran situasional yang relevan. up-to-date

Modul kognitif

Modul kognitif berfungsi sebagai inti deliberatif dari agen perangkat lunak. Ini bertanggung jawab untuk menafsirkan persepsi, membentuk niat, dan membimbing perilaku yang bertujuan melalui perencanaan dan pengambilan keputusan yang didorong oleh tujuan. Modul ini mengubah input menjadi proses penalaran terstruktur, yang memungkinkan agen untuk beroperasi dengan sengaja daripada reaktif. Proses-proses ini dikelola melalui tiga submodul utama: tujuan, perencanaan, dan pengambilan keputusan.

Submodul tujuan

Submodul tujuan mendefinisikan maksud dan arah agen. Sasaran dapat eksplisit (misalnya, “navigasi ke lokasi” atau “kirimkan laporan”) atau implisit (misalnya, “memaksimalkan keterlibatan pengguna” atau “meminimalkan latensi”). Mereka adalah pusat siklus penalaran agen, dan memberikan keadaan target untuk perencanaan dan keputusannya.

Agen terus mengevaluasi kemajuan menuju tujuannya dan mungkin memprioritaskan kembali atau meregenerasi tujuan berdasarkan persepsi atau pembelajaran baru. Kesadaran tujuan ini membuat agen beradaptasi dalam lingkungan yang dinamis.

Submodul perencanaan

Submodul perencanaan membangun strategi untuk mencapai tujuan agen saat ini. Ini menghasilkan urutan tindakan, menguraikan tugas secara hierarkis, dan memilih dari rencana yang telah ditentukan atau dihasilkan secara dinamis.

Untuk beroperasi secara efektif di lingkungan non-deterministik atau berubah, perencanaan tidak statis. Agen modern dapat menghasilkan chain-of-thought urutan, memperkenalkan subtujuan sebagai langkah perantara, dan merevisi rencana secara real time ketika kondisi bergeser.

Submodul ini terhubung erat dengan memori dan pembelajaran, dan memungkinkan agen untuk menyempurnakan perencanaannya dari waktu ke waktu berdasarkan hasil masa lalu.

Submodul pengambilan keputusan

Submodul pengambilan keputusan mengevaluasi rencana dan tindakan yang tersedia untuk memilih langkah berikutnya yang paling tepat. Ini mengintegrasikan masukan dari persepsi, rencana saat ini, tujuan agen, dan konteks lingkungan.

Akun pengambilan keputusan untuk:

- Trade-off antara tujuan yang saling bertentangan
- Ambang kepercayaan (misalnya, ketidakpastian dalam persepsi)
- Konsekuensi tindakan
- Pengalaman agen yang dipelajari

Bergantung pada arsitekturnya, agen mungkin mengandalkan penalaran simbolis, heuristik, pembelajaran penguatan, atau model bahasa (LLMs) untuk membuat keputusan berdasarkan informasi. Proses ini memastikan perilaku agen tetap sadar konteks, selaras dengan tujuan, dan adaptif.

Modul aksi

Modul tindakan bertanggung jawab untuk mengeksekusi keputusan yang dipilih agen dan berinteraksi dengan dunia eksternal atau sistem internal untuk menghasilkan efek yang berarti. Ini mewakili fase Act dari loop persepsi, alasan, tindakan, di mana niat diubah menjadi perilaku.

Ketika modul kognitif memilih tindakan, modul tindakan mengoordinasikan eksekusi melalui submodul khusus, di mana setiap submodul selaras dengan lingkungan terintegrasi agen:

- Aktuasi fisik: Untuk agen yang tertanam dalam sistem robot atau perangkat IoT, submodul ini menerjemahkan keputusan ke dalam gerakan fisik dunia nyata atau instruksi tingkat perangkat keras.

  Contoh: mengemudikan robot, memicu katup, menyalakan sensor.
- Interaksi terintegrasi: Submodul ini menangani tindakan non-fisik tetapi terlihat secara eksternal seperti berinteraksi dengan sistem perangkat lunak, platform, atau. APIs

  Contoh: mengirim perintah ke layanan cloud, memperbarui database, mengirimkan laporan dengan memanggil API.

- Pemanggilan alat: Agen sering memperluas kemampuan mereka dengan menggunakan alat khusus untuk menyelesaikan sub-tugas seperti berikut:
  - Pencarian: menanyakan sumber pengetahuan terstruktur atau tidak terstruktur
  - Ringkasan: mengompresi input teks besar menjadi ikhtisar tingkat tinggi
  - Perhitungan: melakukan perhitungan logis, numerik, atau simbolik

  Pemanggilan alat memungkinkan komposisi perilaku yang kompleks melalui keterampilan modular yang dapat dipanggil.

Modul pembelajaran

Modul pembelajaran memungkinkan agen untuk beradaptasi, menggeneralisasi, dan meningkatkan dari waktu ke waktu berdasarkan pengalaman. Ini mendukung proses penalaran dengan terus menyempurnakan model internal agen, strategi, dan kebijakan keputusan dengan menggunakan umpan balik dari persepsi dan tindakan.

Modul ini beroperasi dalam koordinasi dengan memori jangka pendek dan jangka panjang:

- Memori jangka pendek: Menyimpan konteks sementara, seperti status dialog, informasi tugas saat ini, dan pengamatan terbaru. Ini membantu agen mempertahankan kontinuitas dalam interaksi dan tugas.
- Memori jangka panjang: Mengkodekan pengetahuan persisten dari pengalaman masa lalu, termasuk tujuan yang ditemui sebelumnya, hasil tindakan, dan keadaan lingkungan. Memori jangka panjang memungkinkan agen untuk mengenali pola, menggunakan kembali strategi, dan menghindari kesalahan berulang.

Mode pembelajaran

Modul pembelajaran mendukung berbagai paradigma, seperti pembelajaran yang diawasi, tanpa pengawasan, dan penguatan, yang mendukung lingkungan dan peran agen yang berbeda:

- Pembelajaran yang diawasi: Memperbarui model internal berdasarkan contoh berlabel, seringkali dari umpan balik manusia atau kumpulan data pelatihan.

  Contoh: belajar mengklasifikasikan maksud pengguna berdasarkan percakapan sebelumnya.
- Pembelajaran tanpa pengawasan: Mengidentifikasi pola atau struktur tersembunyi dalam data tanpa label eksplisit.

Contoh: mengelompokkan sinyal lingkungan untuk mendeteksi anomali.

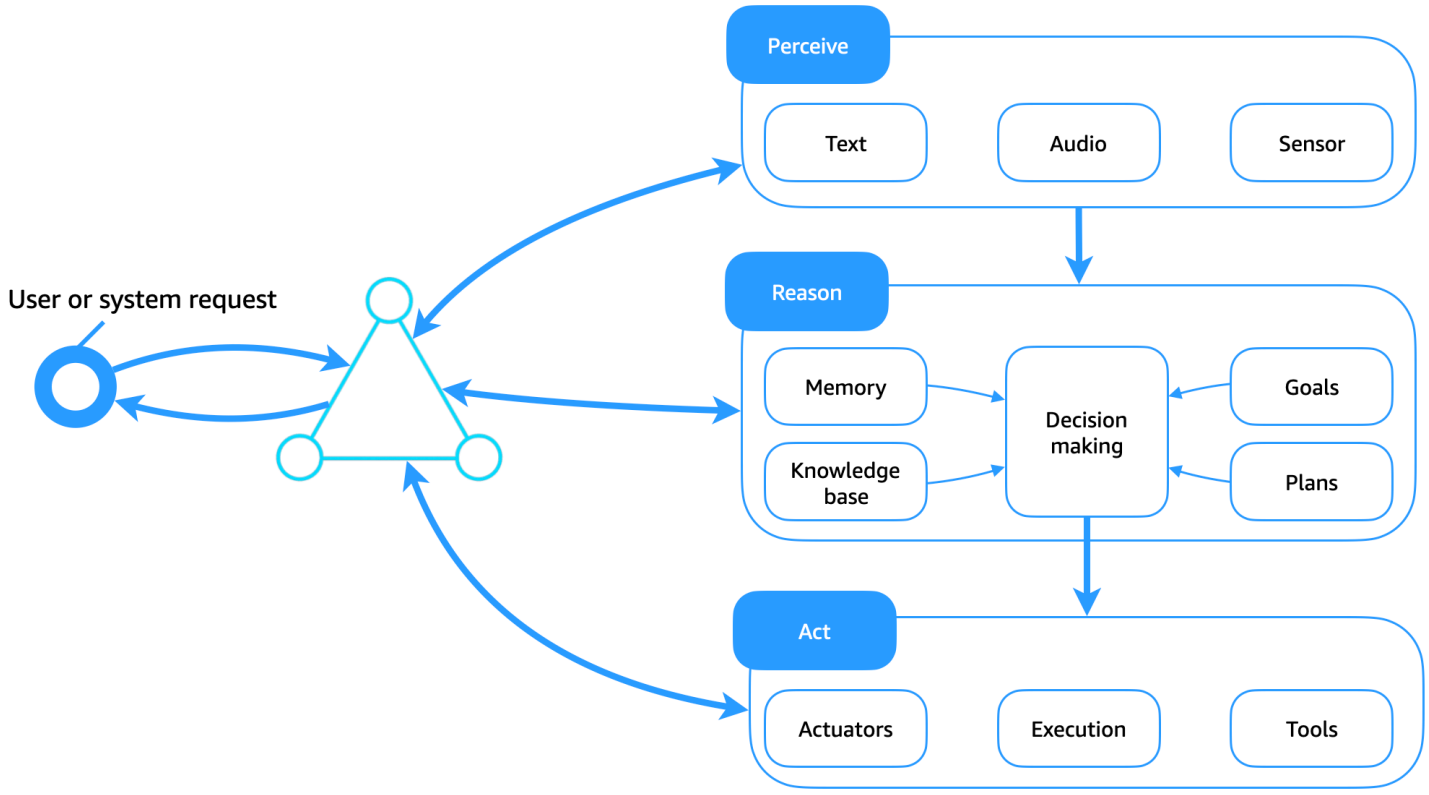
- Pembelajaran penguatan: Mengoptimalkan perilaku melalui coba-coba dengan memaksimalkan imbalan kumulatif di lingkungan interaktif.

  Contoh: mempelajari strategi mana yang mengarah pada penyelesaian tugas tercepat.

Pembelajaran terintegrasi erat dengan modul kognitif agen. Ini menyempurnakan strategi perencanaan berdasarkan hasil masa lalu, meningkatkan pengambilan keputusan melalui evaluasi keberhasilan historis, dan terus meningkatkan pemetaan antara persepsi dan tindakan. Melalui loop pembelajaran dan umpan balik tertutup ini, agen berkembang melampaui eksekusi reaktif untuk menjadi sistem peningkatan diri yang mampu beradaptasi dengan tujuan, kondisi, dan konteks baru dari waktu ke waktu.

Arsitektur agen tradisional: persepsi, alasan, tindakan

Diagram berikut menggambarkan bagaimana blok bangunan yang dibahas di bagian sebelumnya beroperasi di bawah siklus persepsi, alasan, tindakan.



Persepsi modul

Modul persepsi bertindak sebagai antarmuka sensorik agen dengan dunia luar. Ini mengubah input lingkungan mentah menjadi representasi terstruktur yang menginformasikan penalaran. Ini termasuk menangani data multimodal seperti teks, audio, atau sinyal sensor.

- Input teks dapat berasal dari perintah pengguna, dokumen, atau dialog.
- Input audio termasuk instruksi lisan atau suara lingkungan.
- Input sensor menangkap sinyal dunia nyata seperti gerakan, umpan visual, atau GPS.

Ketika input mentah telah dicerna, proses persepsi melakukan ekstraksi fitur, diikuti oleh pengenalan objek atau peristiwa dan interpretasi semantik untuk menciptakan model yang bermakna dari situasi saat ini. Output ini memberikan konteks terstruktur untuk pengambilan keputusan hilir dan menambatkan alasan agen dalam pengamatan dunia nyata.

Modul alasan

Modul alasan adalah inti kognitif agen. Ini mengevaluasi konteks, merumuskan niat, dan menentukan tindakan yang tepat. Modul ini mengatur perilaku yang didorong oleh tujuan dengan menggunakan pengetahuan dan penalaran yang dipelajari.

Modul alasan terdiri dari submodul yang terintegrasi erat:

- Memori: Mempertahankan status dialog, konteks tugas, dan sejarah episodik dalam format jangka pendek dan jangka panjang.
- Basis pengetahuan: Menyediakan akses ke aturan simbolis, ontologi, atau model yang dipelajari (seperti penyematan, fakta, dan kebijakan).
- Tujuan dan rencana: Mendefinisikan hasil yang diinginkan dan membangun strategi tindakan untuk mencapainya. Tujuan dapat diperbarui secara dinamis dan rencana dapat dimodifikasi secara adaptif berdasarkan umpan balik.
- Pengambilan keputusan: Bertindak sebagai mesin arbitrase pusat dengan menimbang opsi, mengevaluasi trade-off, dan memilih tindakan selanjutnya. Faktor submodul ini dalam ambang kepercayaan, penyelarasan tujuan, dan kendala kontekstual.

Bersama-sama, komponen-komponen ini memungkinkan agen untuk bernalar tentang lingkungannya, memperbarui keyakinan, memilih jalur, dan berperilaku dengan cara yang koheren dan adaptif. Modul alasan menutup kesenjangan antara persepsi dan perilaku.

Modul Act

Modul tindakan mengeksekusi keputusan yang dipilih agen dengan berinteraksi dengan lingkungan digital atau fisik untuk melaksanakan tugas. Di sinilah niat menjadi tindakan.

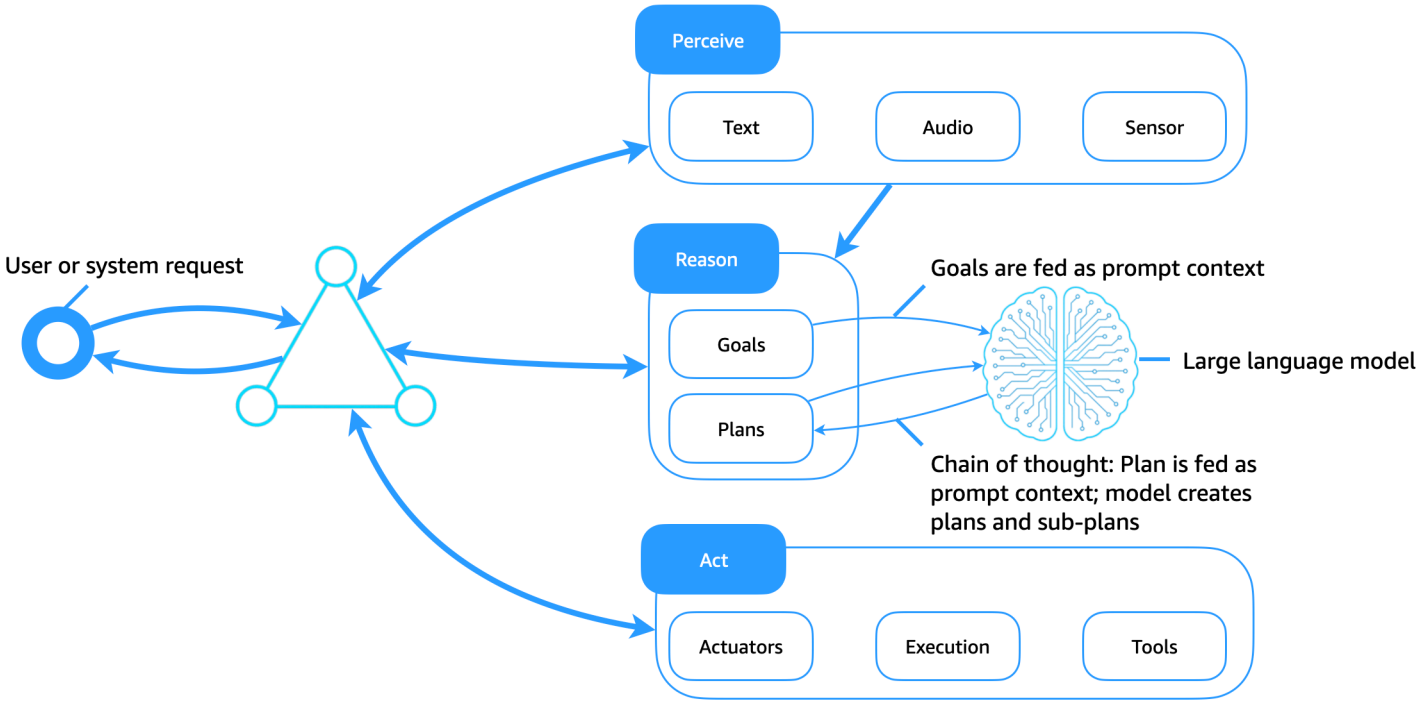
Modul ini mencakup tiga saluran fungsional:

- Aktuator: Untuk agen yang memiliki kehadiran fisik (seperti robot dan perangkat IoT), mengontrol interaksi tingkat perangkat keras seperti gerakan, manipulasi, atau pensinyalan.
- Eksekusi: Menangani tindakan berbasis perangkat lunak, termasuk memanggil, mengirim perintah APIs, dan memperbarui sistem.
- Alat: Memungkinkan kemampuan fungsional seperti pencarian, ringkasan, eksekusi kode, perhitungan, dan penanganan dokumen. Alat-alat ini sering dinamis dan sadar konteks, yang memperluas utilitas agen.

Output dari modul tindakan memberi umpan balik ke lingkungan dan menutup loop. Hasil ini dirasakan oleh agen lagi. Mereka memperbarui keadaan internal agen dan menginformasikan keputusan masa depan, sehingga menyelesaikan siklus persepsi, alasan, tindakan.

Agen AI generatif: mengganti logika simbolik dengan LLMs

Diagram berikut menggambarkan bagaimana model bahasa besar (LLMs) sekarang berfungsi sebagai inti kognitif yang fleksibel dan cerdas untuk agen perangkat lunak. Berbeda dengan sistem logika simbolis tradisional, yang mengandalkan perpustakaan rencana statis dan aturan kode tangan, LLMs memungkinkan penalaran adaptif, perencanaan kontekstual, dan penggunaan alat dinamis, yang mengubah cara agen memandang, bernalar, dan bertindak.



Peningkatan utama

Arsitektur ini meningkatkan arsitektur agen tradisional sebagai berikut:

- LLMs sebagai mesin kognitif: Tujuan, rencana, dan kueri diteruskan ke dalam model sebagai konteks yang cepat. LLM menghasilkan jalur penalaran (seperti rantai pemikiran), menguraikan tugas menjadi sub-tujuan, dan memutuskan tindakan selanjutnya.
- Penggunaan alat melalui prompt: LLMs dapat diarahkan melalui agen penggunaan alat atau penalaran dan acting (ReAct) meminta untuk memanggil APIs dan mencari, menanyakan, menghitung, dan menafsirkan output.
- Perencanaan sadar konteks: Agen menghasilkan atau merevisi rencana secara dinamis berdasarkan tujuan agen saat ini, lingkungan input, dan umpan balik, tanpa memerlukan pustaka rencana hardcode.
- Konteks cepat sebagai memori: Alih-alih menggunakan basis pengetahuan simbolis, agen menyandikan memori, rencana, dan tujuan sebagai token prompt yang diteruskan ke model.
- Belajar melalui beberapa kesempatan, pembelajaran dalam konteks: LLMs mengadaptasi perilaku melalui rekayasa yang cepat, yang mengurangi kebutuhan akan pelatihan ulang eksplisit atau perpustakaan rencana yang kaku.

Mencapai memori jangka panjang pada agen berbasis LLM

Tidak seperti agen tradisional, yang menyimpan memori jangka panjang dalam basis pengetahuan terstruktur, agen AI generatif harus bekerja dalam batasan jendela konteks. LLMs Untuk memperluas memori dan mendukung kecerdasan persisten, agen AI generatif menggunakan beberapa teknik pelengkap: penyimpanan agen, Retrieval-Augmented Generation (RAG), pembelajaran dalam konteks dan rantai cepat, dan pra-pelatihan.

Toko agen: memori jangka panjang eksternal

Status agen, riwayat pengguna, keputusan, dan hasil disimpan dalam penyimpanan memori agen jangka panjang (seperti database vektor, penyimpanan objek, atau penyimpanan dokumen). Memori yang relevan diambil sesuai permintaan dan disuntikkan ke dalam konteks prompt LLM saat runtime. Ini menciptakan loop memori persisten, di mana agen mempertahankan kontinuitas di seluruh sesi, tugas, atau interaksi.

KAIN

RAG meningkatkan kinerja LLM dengan menggabungkan pengetahuan yang diambil dengan kemampuan generatif. Ketika tujuan atau kueri dikeluarkan, agen mencari indeks pengambilan (misalnya, melalui pencarian semantik dokumen, percakapan sebelumnya, atau pengetahuan terstruktur). Hasil yang diambil ditambahkan ke prompt LLM, yang mendasari generasi dalam fakta eksternal atau konteks yang dipersonalisasi. Metode ini memperluas memori efektif agen dan meningkatkan keandalan dan kebenaran faktual.

Pembelajaran dalam konteks dan rantai cepat

Agen mempertahankan memori jangka pendek dengan menggunakan konteks token dalam sesi dan rantai prompt terstruktur. Elemen kontekstual, seperti rencana saat ini, hasil tindakan sebelumnya, dan status agen, diteruskan di antara panggilan untuk memandu perilaku.

Pretraining lanjutan dan fine-tuning

Untuk agen khusus domain, LLMs dapat dilanjutkan pra-pelatihan pada koleksi kustom seperti log, data perusahaan, atau dokumentasi produk. Atau, penyetelan instruksi atau pembelajaran penguatan dari umpan balik manusia (RLHF) dapat menanamkan perilaku seperti agen langsung ke dalam model. Ini menggeser pola penalaran dari logika waktu cepat ke representasi internal model, mengurangi panjang yang cepat, dan meningkatkan efisiensi.

Manfaat gabungan dalam AI agen

Teknik-teknik ini, ketika digunakan bersama, memungkinkan agen AI generatif untuk:

- Pertahankan kesadaran kontekstual dari waktu ke waktu.
- Sesuaikan perilaku berdasarkan riwayat atau preferensi pengguna.
- Membuat keputusan dengan menggunakan up-to-date, pengetahuan faktual, atau pribadi.
- Skala ke kasus penggunaan perusahaan dengan perilaku yang persisten, patuh, dan dapat dijelaskan.

Dengan menambah LLMs dengan memori eksternal, lapisan pengambilan, dan pelatihan lanjutan, agen dapat mencapai tingkat kontinuitas kognitif dan tujuan yang tidak dapat dicapai sebelumnya melalui sistem simbolik saja.

Membandingkan AI tradisional dengan agen perangkat lunak dan AI agen

Tabel berikut memberikan perbandingan rinci AI tradisional, agen perangkat lunak, dan AI agen.

Karakteristik	AI tradisional	Agen perangkat lunak	AI Agen
Contoh	Filter spam, pengklasifikasi gambar, mesin rekomendasi	Chatbots, penjadwal tugas, agen pemantauan	Asisten AI, agen pengembang otonom, orkestrasi LLM multi-agen
Model eksekusi	Batch atau sinkron	Digerakkan oleh acara atau terjadwal	Asinkron, didorong oleh peristiwa, dan didorong oleh tujuan
Otonomi	Terbatas; sering membutuhkan orkestrasi manusia atau eksternal	Medium; beroperasi secara independen dalam batas yang telah ditentukan	Tinggi; bertindak secara independen dengan strategi adaptif

Karakteristik	AI tradisional	Agen perangkat lunak	AI Agen
Reaktivitas	Reaktif terhadap input data	Reaktif terhadap lingkungan dan peristiwa	Reaktif dan proaktif; mengantisipasi dan memulai tindakan
Proaktif	Langka	Hadir dalam beberapa sistem	Atribut inti; mendorong perilaku yang diarahkan pada tujuan
Komunikasi	Minimal; biasanya mandiri atau terikat API	Pesan antar-agen atau agen-manusia	Multi-agen dan human-in-the-loop interaksi yang kaya
Pengambilan keputusan	Inferensi model saja (klasifikasi, prediksi, dan sebagainya)	Penalaran simbolis, atau keputusan berbasis aturan atau naskah	Penalaran kontekstu al, berbasis tujuan, dinamis (sering kali ditingkatkan LLM)
Niat yang didelegas ikan	Tidak; melakukan tugas yang ditentuka n langsung oleh pengguna	Sebagian; bertindak atas nama pengguna atau sistem yang memiliki ruang lingkup terbatas	Ya; bertindak dengan tujuan yang didelegas ikan, seringkali di seluruh layanan, pengguna, atau sistem
Pembelajaran dan adaptasi	Seringkali model-sentris (misalnya., pelatihan ML)	Terkadang adaptif	Pembelajaran, memori, atau penalaran tertanam (misalnya, umpan balik, koreksi diri)
Agensi	Tidak ada; alat untuk manusia	Implisit atau dasar	Eksplisit; beroperas i dengan tujuan, sasaran, dan pengarahan diri

Karakteristik	AI tradisional	Agen perangkat lunak	AI Agen
Kesadaran konteks	Rendah; tanpa kewarganegaraan atau berbasis snapshot	Sedang; beberapa pelacakan negara	Tinggi; menggunakan memori, konteks situasional, dan model lingkungan
Peran infrastruktur	Tertanam dalam aplikasi atau pipeline analitik	Middleware atau komponen lapisan layanan	Jaring agen yang dapat dikomposisikan terintegrasi dengan sistem cloud, tanpa server, atau edge

Ringkasnya:

- AI tradisional berpusat pada alat dan sempit secara fungsional. Ini berfokus pada prediksi atau klasifikasi.
- Agen perangkat lunak tradisional memperkenalkan otonomi dan komunikasi dasar, tetapi mereka sering terikat aturan atau statis.
- Agentic AI menyatukan otonomi, asinkron, dan agensi. Ini memungkinkan entitas cerdas yang didorong oleh tujuan yang dapat bernalar, bertindak, dan beradaptasi dalam sistem yang kompleks. Ini membuat AI agen ideal untuk masa depan cloud-native yang digerakkan oleh AI.

Langkah berikutnya

Panduan ini membahas sejarah dan fondasi AI agen, yang mewakili evolusi agen perangkat lunak tradisional menjadi sistem otonom dan cerdas yang didukung oleh AI generatif. Ini menggambarkan bagaimana agen perangkat lunak awal mengikuti aturan dan logika yang telah ditentukan untuk mengotomatiskan tugas dalam batas-batas tetap, dan menjelaskan bagaimana AI agen dibangun di atas fondasi ini dengan menggabungkan model bahasa besar, yang memungkinkan agen untuk bernalar, belajar, dan beradaptasi secara dinamis di lingkungan terbuka.

Anda dapat menjelajahi AI agen secara mendalam dengan meninjau publikasi berikut dalam seri ini:

- Mengoperasionalkan AI agen AWS menyediakan strategi organisasi untuk mengubah AI agen dari eksperimen terisolasi menjadi infrastruktur skala perusahaan yang menghasilkan nilai.
- Pola dan alur kerja AI agen AWS membahas cetak biru dasar dan konstruksi modular yang digunakan untuk merancang, menyusun, dan mengatur agen AI yang berorientasi pada tujuan.
- Kerangka kerja, protokol, dan alat AI agen AWS mencakup fondasi perangkat lunak, toolkit, dan protokol untuk dipertimbangkan saat Anda membangun solusi AI agen Anda.
- Membangun arsitektur tanpa server untuk AI agen AWS membahas arsitektur tanpa server sebagai fondasi alami dari beban kerja AI modern dan menjelaskan bagaimana Anda dapat membangun arsitektur tanpa server asli AI di. AWS Cloud
- Membangun arsitektur multi-penyewa untuk AI agen AWS menjelaskan penggunaan agen AI dalam pengaturan multi-penyewa, termasuk pertimbangan hosting, model penerapan, dan pesawat kontrol.

Sumber daya

Untuk informasi lebih lanjut tentang konsep yang dibahas dalam panduan ini, lihat panduan dan artikel berikut.

AWS referensi

- Agen Amazon Bedrock
- Amazon Q Developer
- Agen Helai SDK

Referensi lainnya

- Hewitt, Carl, Peter Bishop, dan Richard Steiger. “Formalisme AKTOR Modular Universal untuk Kecerdasan Buatan.” Prosiding Konferensi Bersama Internasional ke-3 tentang Kecerdasan Buatan (1973): 235-245. https://www.ijcai. org/Proceedings/73/Papers/027B.pdf
- Lesser, Victor R., publikasi yang relevan (lihat daftar lengkap):
  - Lesser, Victor R. dan Daniel D. Corkill. “Secara Fungsional Akurat, Sistem Terdistribusi Kooperatif.” Transaksi IEEE pada Sistem, Manusia, dan Sibernetika 11, no. 1 (1981): 81-96. https://ieeexplore.ieee. org/abstract/document/4308581
  - Decker, Keith S. dan Victor R. Lesser. “Komunikasi dalam Layanan Koordinasi.” Lokakarya AAAI tentang Perencanaan Komunikasi Antaragen (1994). https://www.researchgate. net/profile/Victor-Lesser/publication/2768884_Communication_in_the_Service_of_Coordination/links/00b7d51cc2a0750cb4000000/Communication-in-the-Service-of-Coordination.pdf
  - Durfee, Edmund H., Victor R. Lesser, dan Daniel D. Corkill. “Tren Pemecahan Masalah Terdistribusi Koperasi.” Transaksi IEEE tentang pengetahuan dan Rekayasa Data (1989). http://mas.cs.umass. edu/Documents/ieee-tkde89.pdf
  - Durfee, Edmund H., V.R. Lesser, dan DD Corkill, “Kecerdasan Buatan Terdistribusi.” Kerjasama Melalui Komunikasi dalam Jaringan Pemecahan Masalah Terdistribusi (1987): 29-58. https://www.academia. edu/download/79885643/durf94_1.pdf
  - Lâasri, Brigitte, Hassan Lâasri, Susan Lander, dan Victor Lesser. “Model Generik untuk Agen Negosiasi Cerdas.” Jurnal Internasional Sistem Informasi Koperasi 01, no. 02 (1992): 291-317. https://doi.org/10.1142/S0218215792000210

  - Lander, Susan E. dan Victor R. Lesser. “Memahami Peran Negosiasi dalam Pencarian Terdistribusi Di Antara Agen Heterogen.” IJCAI'93: Prosiding konferensi bersama internasional ke-13 tentang intelijen Buatan (1993): 438-444. https://www.ijcai. org/Proceedings/93-1/Papers/062.pdf
  - Lander, Susan, Victor R. Lesser, dan Margaret E. Connell. “Strategi Resolusi Konflik untuk Agen Ahli yang Bekerja Sama” CKBS'90: Prosiding Konferensi Kerja Internasional tentang Sistem Berbasis Pengetahuan yang Bekerja Sama (Oktober 1990): 183-200. https://doi.org/10.1007/978-1-4471-1831-2_10
  - Prasad, M.V. Nagendra, Victor Lesser, dan Susan E. Lander. “Mempelajari Eksperimen dalam Sistem Multi-agen Heterogen.” Lokakarya IJCAI-95 tentang Adaptasi dan Pembelajaran dalam Sistem Multi-agen (1995): 59-64. https://www.researchgate.net/publication/2784280_Learning_Experiments_in_a_Heterogeneous_Multi-agent_System
- Nwana, Hyacinth S. “Agen Perangkat Lunak: Tinjauan Umum.” Tinjauan Rekayasa Pengetahuan 11, no. 3 (Oktober/November 1996): 205-244. https://teaching.shu.ac. uk/aces/rh1/elearning/multiagents/introduction/nwana.pdf
- Selfridge, Oliver G. “Pandemonium: Paradigma untuk Belajar.” Mekanisasi Proses Pemikiran: Prosiding Simposium yang Diadakan di Laboratorium Fisik Nasional 1 (1959): 511—529. https://aitopics. org/download/classics:504E1BAC
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, dan Illia Polosukhin. “Perhatian adalah semua yang Anda butuhkan.” Prosiding Konferensi ke-31 tentang Sistem Pemrosesan Informasi Saraf (NIPS). Kemajuan dalam Sistem Pemrosesan Informasi Saraf 30 (2017): 5998-6008. https://proceedings.neurips. cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
- Wooldridge, Michael dan Nicholas R. Jennings. “Agen Cerdas: Teori dan Praktek.” Tinjauan Rekayasa Pengetahuan 10, no. 2 (Januari 1995): 115-152. https://www.cs.cmu.edu/ ~motionplanning/papers/sbp_papers/integrated1/woodridge _intelligent_agents.pdf

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	—	Juli 14, 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 — Memindahkan aplikasi dan memodifikasi 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 PostgreSQL Compatible Edition.
- Replatform (angkat dan bentuk ulang) — Pindahkan aplikasi ke cloud, dan perkenalkan beberapa tingkat pengoptimalan untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan 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 instance 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

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 database 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.

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 Wilayah AWS 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.

deployment biru/hijau

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 kaca pecah dalam panduan Well-Architected AWS .

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.

Enkripsi sisi klien

Enkripsi data secara lokal, sebelum target Layanan AWS menerimanya.

Pusat Keunggulan Cloud (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 CCo E 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 CCo E, 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 Perusahaan. AWS Cloud 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. Pipa CI/CD tunggal 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 penyimpanan atau kelas 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 Wilayah, 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.

defense-in-depth

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, defense-in-depth pendekatan 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 oleh 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. Sistem big-endian menyimpan byte paling signifikan terlebih dahulu. Sistem little-endian 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 pengguna akhir dapat mengakses. 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, Wilayah AWS, 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. Beberapa bidikan dapat efektif untuk tugas-tugas yang memerlukan 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. FMs 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.

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 ()OUs. 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 .

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.

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 di AWS Well-Architected Framework.

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 inbound, outbound, dan inspeksi VPCs 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 lebih lanjut, 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 VPCs (dalam yang sama atau berbeda Wilayah AWS), internet, dan jaringan lokal. Arsitektur Referensi AWS Keamanan merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs 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 dalam bahasa lain, dan menyelesaikan kalimat. Untuk informasi lebih lanjut, lihat Apa itu LLMs.

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.

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 lebih lanjut, lihat Membangun mekanisme di AWS Well-Architected Framework.

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 ringan machine-to-machine (M2M), berdasarkan pola terbitkan/berlangganan, untuk perangkat IoT yang dibatasi sumber daya.

layanan mikro

Layanan kecil dan independen yang berkomunikasi dengan jelas APIs 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 ringan. APIs 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

Tim lintas fungsi 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 dengan Layanan Migrasi AWS 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 file. 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 machine-to-machine (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 lebih lanjut, lihat Ulasan Kesiapan Operasional (ORR) dalam Kerangka Kerja Well-Architected AWS .

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 Wilayah AWS, enkripsi sisi server dengan AWS KMS (SSE-KMS), dan dinamis dan permintaan ke bucket S3. `PUT` `DELETE`

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 inbound, outbound, dan inspeksi VPCs 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

ketekunan 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. Untuk informasi selengkapnya, lihat Mengaktifkan persistensi data di layanan mikro.

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 yang dihosting pribadi

Container yang menyimpan informasi tentang bagaimana Anda ingin Amazon Route 53 merespons kueri DNS untuk domain dan subdomainnya dalam satu atau lebih. VPCs 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 Wilayah AWS terisolasi dan independen dari yang lain untuk memberikan toleransi kesalahan, stabilitas, dan ketahanan. Untuk informasi selengkapnya, lihat Menentukan Wilayah 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 prinsip 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.

PENIPUAN

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 EC2 Amazon, 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 SCPs menentukan pagar pembatas atau menetapkan batasan pada tindakan yang dapat didelegasikan oleh administrator kepada pengguna atau peran. Anda dapat menggunakan SCPs 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 tim TI untuk diberikan kepada pelanggan mereka, seperti waktu kerja 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.

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.

split-and-seed model

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 Microsoft ASP.NET (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

Pasangan nilai kunci yang bertindak sebagai metadata untuk mengatur sumber daya Anda. AWS Tanda membantu Anda mengelola, mengidentifikasi, mengatur, dan memfilter sumber daya. Untuk informasi selengkapnya, lihat Menandai AWS sumber daya Anda.

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.

gerbang transit

Hub transit jaringan yang dapat Anda gunakan untuk menghubungkan jaringan Anda VPCs dan lokal. 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. Untuk informasi lebih lanjut, lihat panduan Mengukur ketidakpastian dalam sistem pembelajaran mendalam.

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 VPCs yang memungkinkan Anda untuk 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 saat ini 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.