Membangun arsitektur multi-tenant untuk AI agen di AWS

AWS Bimbingan Preskriptif

AWS Bimbingan Preskriptif: Membangun arsitektur multi-tenant untuk AI agen di AWS

Table of Contents

Membangun arsitektur multi-tenant untuk AI agen di AWS

Aaron Sempf dan Tod Golding, Amazon Web Services

Juli 2025 (sejarah dokumen)

Agentic AI mewakili perubahan paradigma yang mengganggu yang mengharuskan organisasi untuk memikirkan kembali bagaimana membangun, menyampaikan, dan mengoperasikan sistem mereka. Model agen memiliki tim yang mengeksplorasi cara-cara baru untuk menguraikan sistem menjadi satu atau lebih agen yang menciptakan jalur, kemungkinan, dan nilai baru.

Sebagian besar diskusi agen berpusat di sekitar alat, kerangka kerja, dan pola yang digunakan untuk membangun dan mengimplementasikan agen. Kita tidak hanya harus mengadopsi alat yang baik untuk membuat agen tetapi juga protokol integrasi baru, strategi otentikasi, dan mekanisme penemuan yang dapat berfungsi sebagai dasar arsitektur agen.

Sementara jumlah alat agen bertambah, tim juga harus mempertimbangkan bagaimana agen mereka mengatasi tantangan arsitektur yang lebih tradisional. Skala, tetangga yang bising, ketahanan, biaya, dan efisiensi operasional adalah topik mendasar yang harus dievaluasi saat Anda merancang, membangun, dan menyebarkan agen. Terlepas dari seberapa otonom dan cerdas agen, kita juga harus memastikan bahwa mereka mencapai skala ekonomi, efisiensi, dan kelincahan yang selaras dengan kebutuhan bisnis.

Tujuan panduan ini adalah untuk mengeksplorasi berbagai dimensi jejak kaki agen. Ini termasuk meninjau berbagai penyebaran agen dan pola konsumsi dan menyoroti berbagai strategi untuk membuat agen yang membahas tujuan arsitektur. Ini juga berarti melihat bagaimana agen dapat dikonsumsi dalam lingkungan multi-penyewa dengan memperkenalkan konstruksi internal yang biasanya diperlukan dalam pengaturan multi-penyewa.

Audiens yang dituju

Panduan ini ditujukan untuk arsitek, pengembang, dan pemimpin teknologi yang ingin membangun sistem multi-tenant berbasis AI.

Tujuan

Panduan ini membantu Anda melakukan hal berikut:

- Pahami penerapan agen multi-penyewa, jelajahi model silo dan gabungan, dan bagaimana konteks penyewa memengaruhi implementasi agen
- Jelajahi manajemen agen, termasuk orientasi, isolasi penyewa, dan manajemen sumber daya di lingkungan penyedia tunggal dan multi-penyedia
- Mengevaluasi aspek agen multi-penyewa, termasuk kepemilikan data, pemantauan, dan pengujian

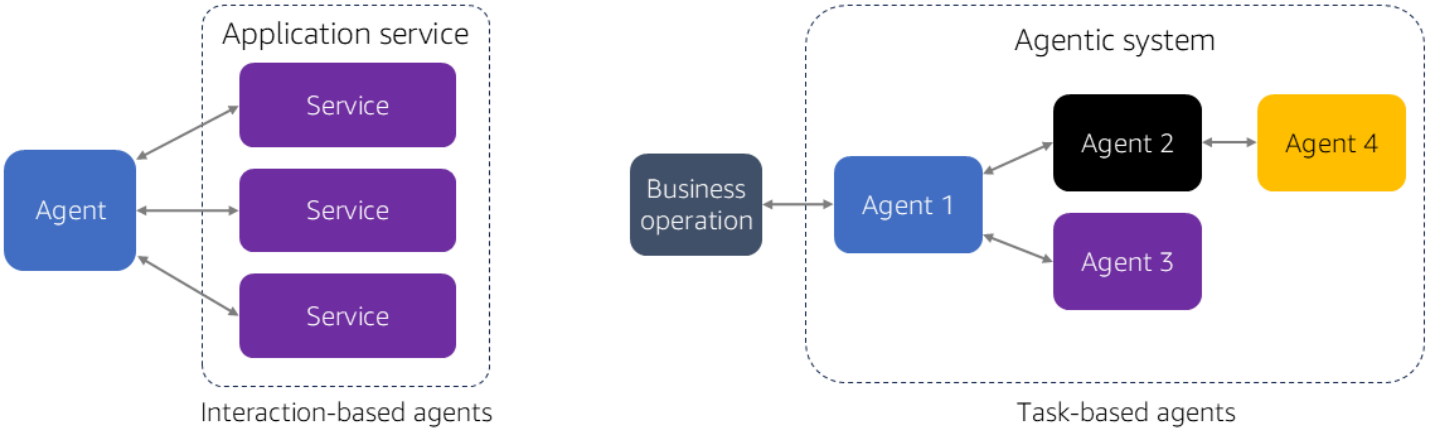
Tentang seri konten ini

Panduan ini adalah bagian dari seri tentang AI agen di AWS. Untuk informasi lebih lanjut dan untuk melihat panduan lain dalam seri ini, lihat Agentic AI di situs web AWS Prescriptive Guidance.

Dasar-dasar agen

Sebelum kita membahas detail arsitektur, kita harus menguraikan peran berbeda yang dimainkan agen karena “agen” adalah istilah kelebihan beban yang dapat diterapkan pada banyak kasus penggunaan. Mari kita mulai dengan beberapa istilah luas yang dapat membantu mengkategorikannya.

Pada tingkat terluar, kita perlu mulai dengan mengklasifikasikan peran dan sifat agen. Ini menantang karena ada berbagai skenario di mana agen dapat diterapkan pada sejumlah masalah. Namun, untuk diskusi ini, kami fokus pada apa artinya memperkenalkan agen ke dalam aplikasi atau sistem. Dalam model ini, kami menekankan bagaimana dan di mana agen dapat memperkaya pengalaman sistem Anda dengan sebaik-baiknya. Opsi yang Anda pilih memengaruhi cara agen Anda dibangun, diintegrasikan, dan diterapkan ke berbagai domain dan kasus penggunaan. Diagram berikut menunjukkan dua pola agen yang digunakan pembangun.

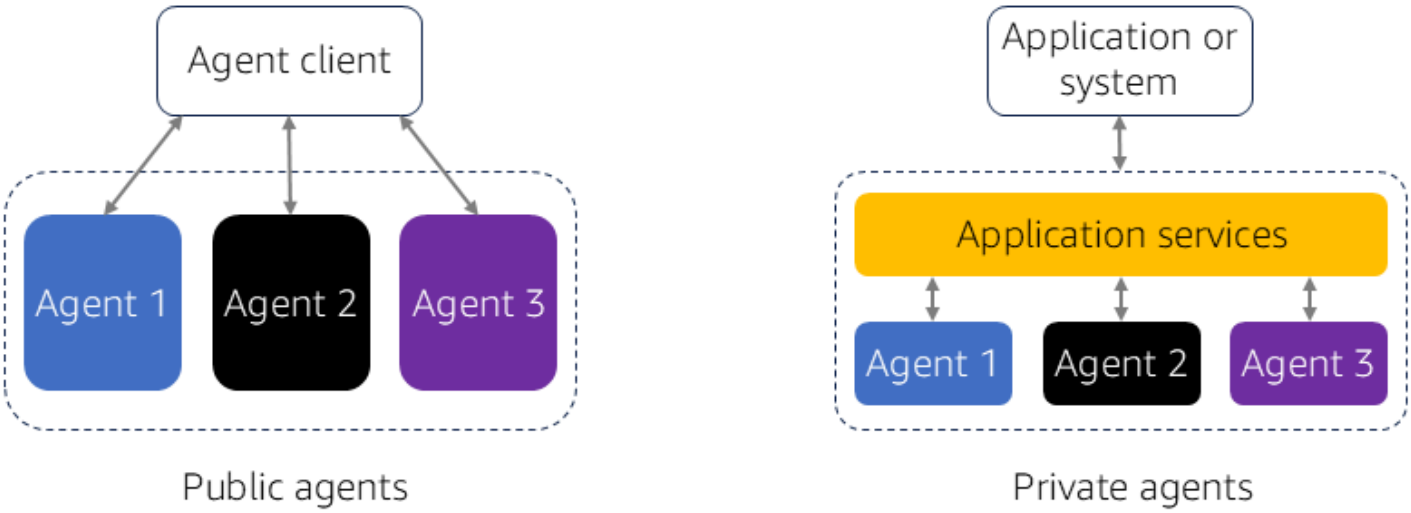


Di sisi kiri diagram adalah agen berbasis interaksi. Dalam mode ini, agen menciptakan pandangan ke dalam sistem yang ada untuk mengatur interaksi dengan layanan yang mendasarinya untuk mencapai tujuan atau hasil. Kuncinya adalah bahwa agen ditambahkan ke sistem sebagai pendekatan alternatif untuk mendorong fitur dan kemampuan sistem. Bayangkan, misalnya, bahwa vendor perangkat lunak independen (ISV) memiliki sistem akuntansi dengan UX yang digunakan untuk melakukan operasi. Agen berbasis interaksi menyederhanakan interaksi dengan kemampuan yang ada ini. Ini bukan tentang belajar bagaimana mencapai tujuan yang ditentukan secara longgar dan lebih banyak tentang menyediakan cara mengatur jalur yang diketahui.

Sebaliknya, sistem berbasis tugas di sisi kanan diagram mewakili pendekatan yang berbeda. Agen dalam sistem itu menggunakan pengetahuan dan kemampuan mereka untuk belajar menyelesaikan tugas dan mendorong hasil bisnis. Anda dapat berargumen bahwa kedua model mencapai hasil

bisnis, tetapi model berbasis tugas bergantung pada agen itu sendiri untuk menentukan bagaimana mencapai hasil. Agen semacam itu kurang deterministik dan malah mengandalkan kemampuan mereka untuk belajar dan berkembang. Sebaliknya, agen berbasis interaksi sebagian besar dirancang untuk mengatur serangkaian kemampuan yang diketahui. Perbedaan ini memengaruhi cara Anda membangun, lingkup, dan mengintegrasikan agen untuk mendukung bisnis Anda.

Kami juga membutuhkan istilah yang mencirikan bagaimana dan di mana kami menyebarkan agen. Dimana agen tinggal dalam jejak sistem Anda dapat mempengaruhi bagaimana itu dibangun, dicakup, dan diamankan. Diagram berikut menguraikan dua model berbeda yang dapat diterapkan pada agen.



Di sisi kiri diagram adalah sistem penyebaran dengan tiga agen berbeda. Agen terpapar ke klien eksternal yang mungkin agen atau aplikasi lain. Untuk model ini, agen disebut sebagai agen publik.

Sebaliknya, diagram di sisi kanan menunjukkan agen dalam implementasi solusi. Dalam hal ini, ada serangkaian layanan aplikasi yang dikonsumsi oleh pengguna atau sistem. Pengguna ini berinteraksi dengan aplikasi tanpa kesadaran akan fakta bahwa agen adalah bagian dari pengalaman. Agen kemudian dipanggil dan diatur oleh layanan sistem yang mendasarinya. Agen yang dikerahkan dengan cara ini disebut sebagai agen swasta.

Sebagian besar nilai agen berfokus pada model publik di mana penyedia dapat mempublikasikan agen mereka dengan maksud mengintegrasikan mereka dengan agen pihak ketiga lainnya. Agen kemudian akan menjadi bagian dari mesh atau web layanan yang saling berhubungan yang, secara kolektif, mampu mengatasi banyak kasus penggunaan. Meskipun agen ini dapat digunakan di banyak domain, kasus business-to-business penggunaannya sangat cocok. Diagram berikut memberikan

pandangan konseptualisasi tentang bagaimana tampilannya untuk merakit agen koleksi yang memecahkan masalah tertentu.

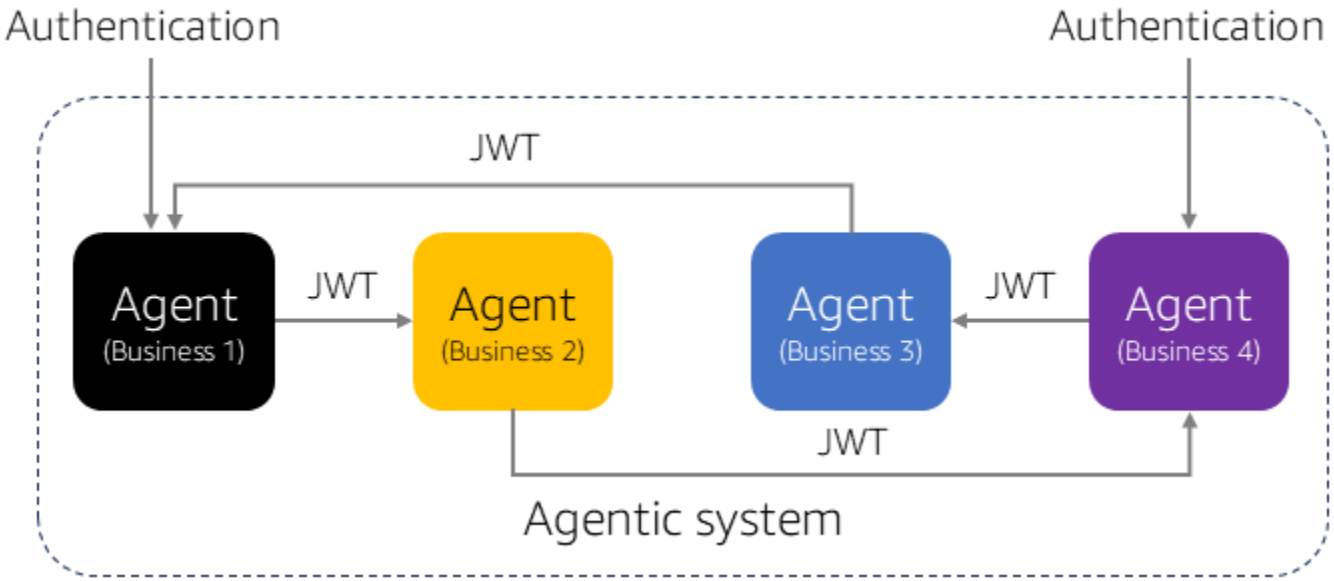
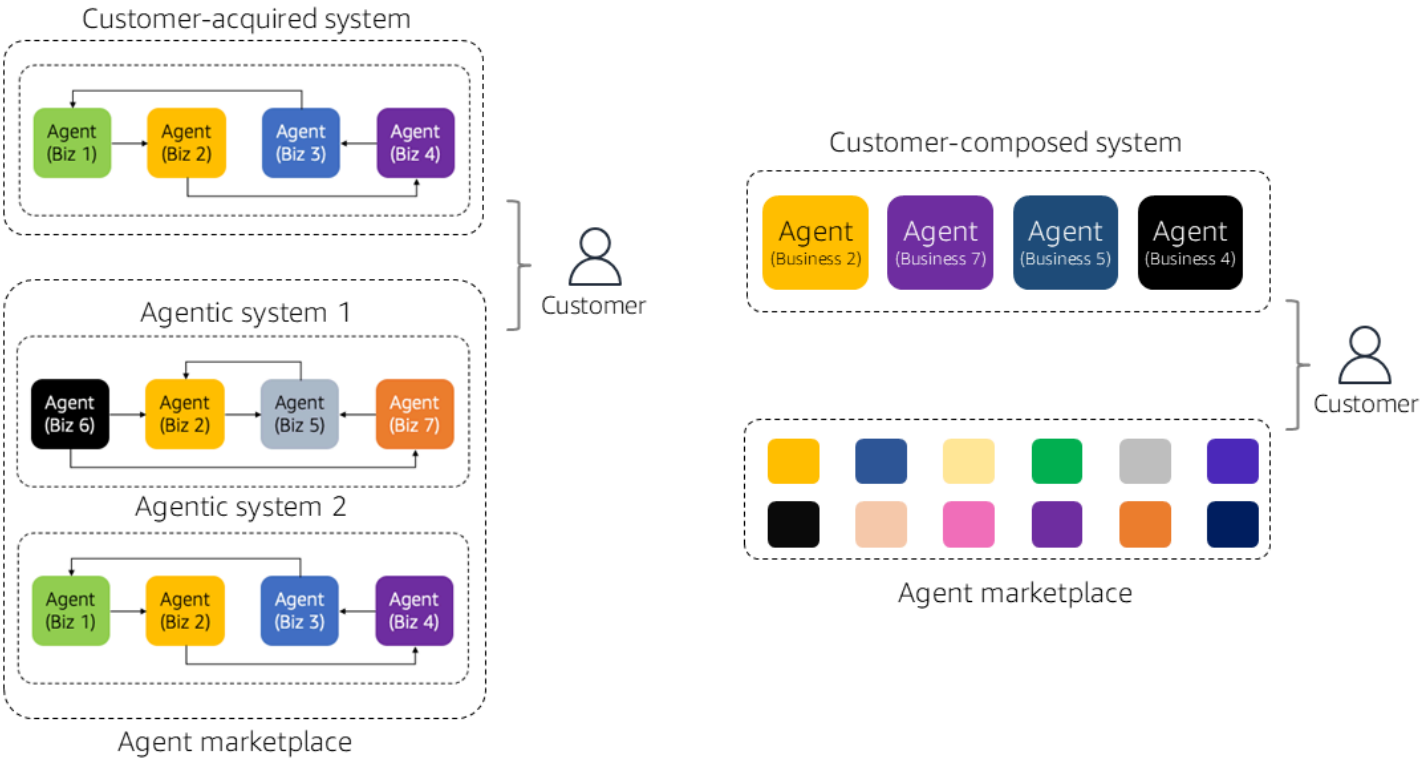


Diagram menunjukkan empat agen bisnis yang bekerja sama untuk mencapai serangkaian tujuan. Ketika agen disusun dengan cara ini, mereka mewakili sistem agen, dan ada banyak rasa dari sistem tersebut. Mereka bisa menjadi satu set agen kolaborasi yang dikemas sebelumnya yang umumnya dikonsumsi sebagai satu unit. Atau sistem dapat dirakit secara dinamis oleh pelanggan yang ingin memilih dan memilih kombinasi agen yang paling memenuhi kebutuhan mereka.

Kedua pendekatan menawarkan jalur yang layak untuk integrasi agen. Beberapa agen dibangun dengan harapan bahwa mereka akan diintegrasikan ke dalam sistem tertentu di mana mereka dapat memaksimalkan nilai, jangkauan, dan dampaknya. Gagasan tentang sistem agen ini juga menimbulkan pertanyaan tentang bagaimana agen diperoleh, dan mungkin ada banyak cara untuk mengatasi hal ini. Diagram berikut memberikan contoh bagaimana agen dan sistem ini dapat dibuat melalui pengalaman transaksional.

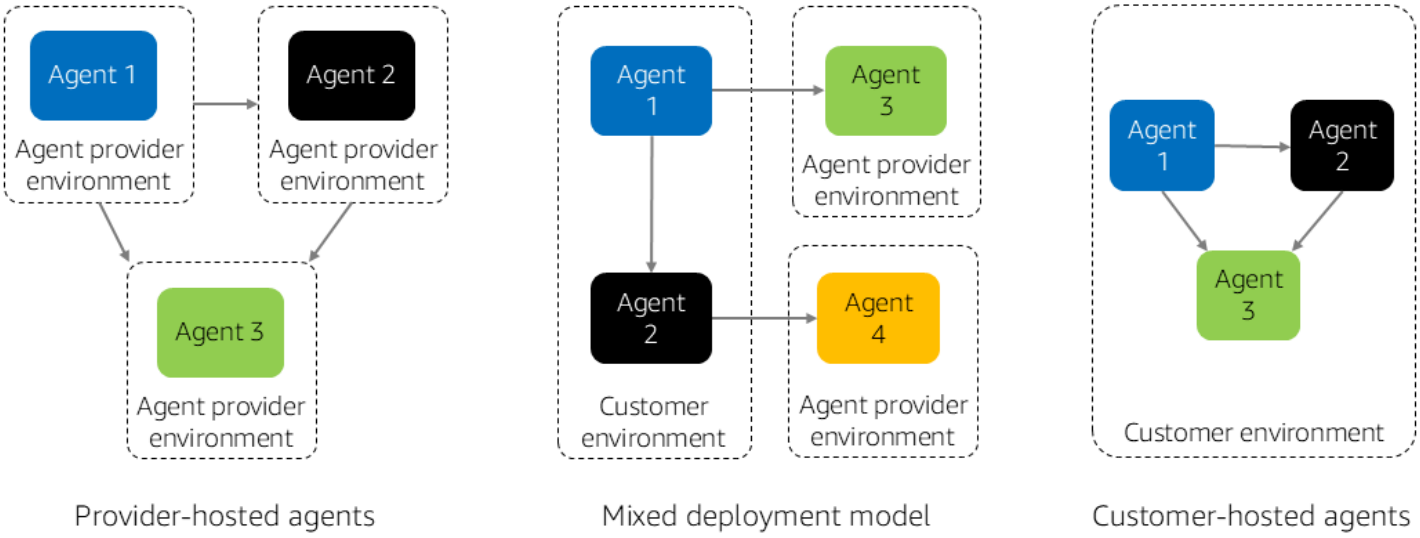


Dua contoh pengalaman pasar ditampilkan. Di sisi kiri, pasar digunakan untuk memperoleh sistem yang dikemas. Dalam skenario ini, pasar menemukan dan menerapkan sistem yang membahas tujuan yang lebih luas yang memerlukan integrasi dan orkestrasi beberapa agen.

Contoh di sisi kanan menunjukkan pasar di mana agen ditemukan dan disusun ke dalam sistem agen. Dalam skenario ini, pelanggan dapat membangun sistem agen terintegrasi yang kompatibel untuk memenuhi kebutuhan mereka. Kemampuan untuk merakit agen dengan cara ini tergantung pada model kompatibilitas dan persyaratan integrasi agen individu.

Pertimbangan hosting agen

Sekarang setelah Anda memahami konsep agen yang lebih luas, mari kita bahas apa artinya menjadi tuan rumah dan menjalankan agen ini. Kita harus memikirkan bagaimana dan di mana komputasi berjalan, bagaimana skalanya, bagaimana mereka beroperasi, dan bagaimana mereka dikelola. Pada saat yang sama, beberapa pola yang kami harapkan untuk dilihat sebagai agen lebih banyak diterapkan dan diadopsi. Diagram berikut menunjukkan contoh kemungkinan permutasi.



Tiga strategi berbeda diwakili di sini. Di sisi kiri diagram, Anda melihat model di mana agen kami di-host, diskalakan, dan dikelola dalam lingkungan masing-masing penyedia agen. Agen-agen ini diterbitkan dan dikonsumsi sebagai layanan, beroperasi dalam model yang diberi label sebagai agen sebagai layanan (AaaS). Di sisi kanan adalah model di mana agen penyedia semuanya dihosting di lingkungan pelanggan yang berdedikasi.

Di tengah diagram adalah model penyebaran campuran yang menggabungkan dua strategi ini, menampung beberapa agen secara lokal di lingkungan pelanggan dan berinteraksi dengan beberapa agen yang di-host dari jarak jauh di lingkungan penyedia.

Opsi keempat (tidak ditampilkan) dapat berupa di mana agen dibangun sebagai layanan rendah atau tanpa kode yang diskalakan dan dikelola oleh layanan infrastruktur agen. Kami tidak akan membahas ini secara rinci karena arsitektur dan hosting agen terkelola ditentukan terutama oleh organisasi yang memiliki layanan.

Anda dapat membayangkan berbagai faktor yang mungkin mempengaruhi adopsi salah satu model ini. Kepatuhan, peraturan, dan batasan keamanan, misalnya, dapat mendorong seseorang menuju

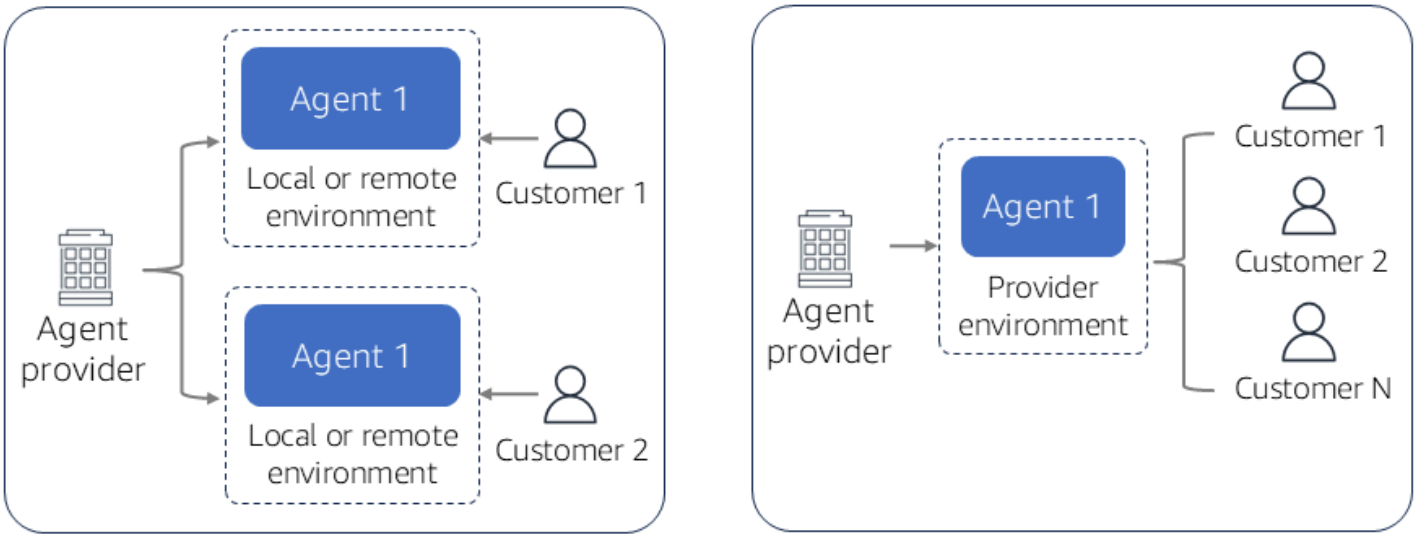
agen yang dihosting pelanggan. Skala, kelincahan, dan efisiensi dapat mendorong organisasi lebih ke arah model AaaS.

Konsep kuncinya di sini adalah bahwa agen dapat dan dikerahkan dan dihosting dalam banyak cara. Adalah tugas Anda untuk menentukan bagaimana agen dapat diterapkan dengan baik. Jejak, keamanan, dan penyebaran, di antara faktor-faktor lain, secara signifikan memengaruhi cara Anda mendekati agen bangunan dan operasi. Agen swasta dan publik, misalnya, mungkin memiliki desain dan siklus hidup rilis yang berbeda.

Agen bertemu multi-tenancy

Sangat mudah untuk menganggap agen sebagai blok bangunan di mana agen dipandang sebagai serangkaian komponen otonom yang dirakit untuk mendukung kebutuhan domain atau masalah bisnis tertentu. Yang menjadi lebih menarik adalah ketika kita mulai berpikir tentang bagaimana agen ini dikemas dan dikonsumsi oleh penyedia. Dalam banyak hal, agen menjadi sumber biaya dan pendapatan untuk bisnis. Penyedia agen harus mempertimbangkan persona berbeda yang menggunakan layanan mereka, profil konsumsi persona, dan strategi monetisasi yang memungkinkan penyedia agen untuk membuat model penetapan harga dan tingkat yang selaras dengan konsumen.

Penyedia agen dapat mendukung beberapa model untuk menyebarkan agen mereka untuk memenuhi kebutuhan pelanggan. Diagram berikut menunjukkan pandangan konseptual dari dua model penyebaran agen utama.



Customer-dedicated agents | Agent as a service (AaaS)

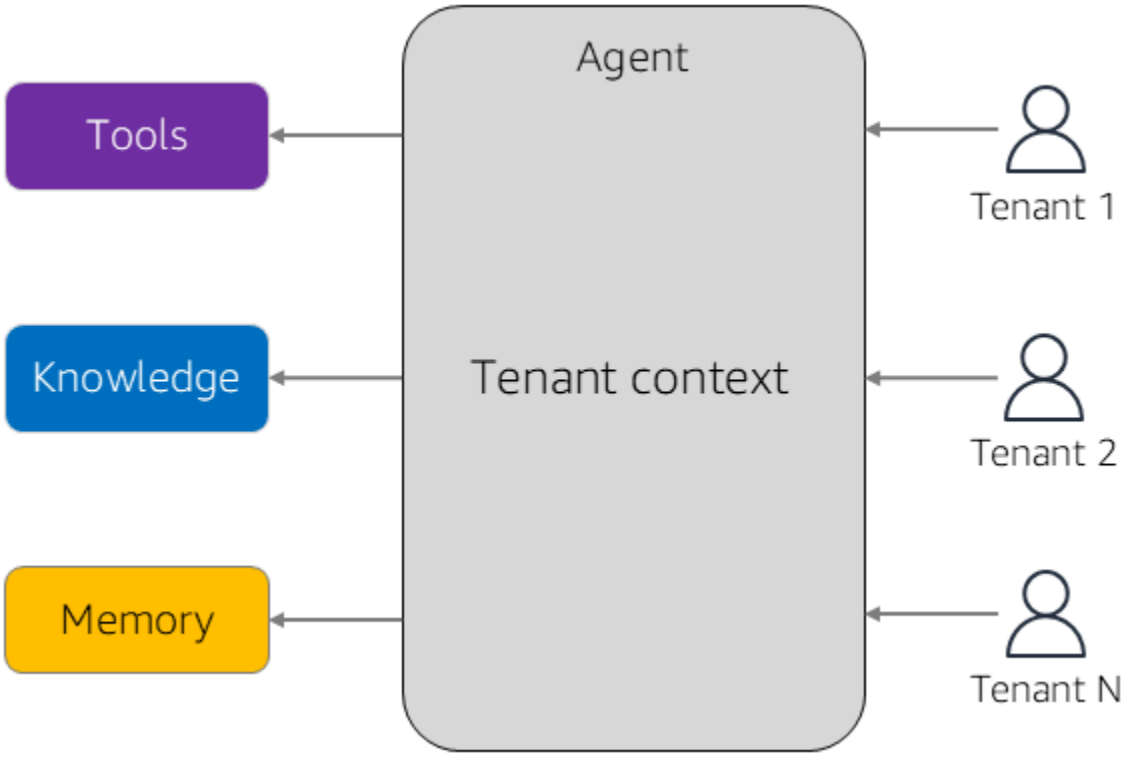
Sisi kiri diagram menunjukkan model agen khusus pelanggan. Penyedia agen membangun agen dengan menerapkan instance agen terpisah untuk setiap pelanggan onboard. Dengan pendekatan ini, kemampuan agen dan kemampuannya untuk memperoleh pengetahuan akan terbatas pada ruang lingkup lingkungan pelanggan tertentu. Ini akhirnya mewakili pengalaman per pelanggan yang mewarisi beberapa kompleksitas dan keuntungan dari mendukung lingkungan pelanggan yang berdedikasi.

Sebaliknya, diagram di sisi kanan diagram memiliki agen tunggal yang digunakan di lingkungan penyedia. Agen memproses permintaan dari banyak pelanggan, berkembang dan belajar

berdasarkan pengalaman kolektif semua pelanggan. Setiap pelanggan baru yang ditambahkan hanya akan mewakili klien valid lain dari agen. Agen berjalan seperti model agen sebagai layanan (AaaS), menggunakan konstruksi bersama untuk mendukung kebutuhan klien. Dalam kedua kasus, konsumen agen dapat berupa aplikasi, sistem, atau bahkan agen lainnya.

Ada dua cara Anda dapat melihat model AaaS. Model di atas memberikan pengalaman yang sama kepada semua pelanggan. Ini berarti internal agen tidak akan mencakup tingkat spesialisasi apa pun yang mempertimbangkan konteks klien yang meminta. Umumnya, untuk mode ini, asumsinya adalah bahwa sifat ruang lingkup, tujuan, dan nilai agen berpusat di sekitar serangkaian sumber daya, pengetahuan, dan hasil bersama yang diterapkan secara universal untuk semua klien.

Pendekatan alternatif untuk AaaS adalah di mana konteks klien mempengaruhi pengalaman dan implementasi agen. Diagram berikut memberikan pandangan konseptual dari jejak agen AaaS dalam konteks ini.



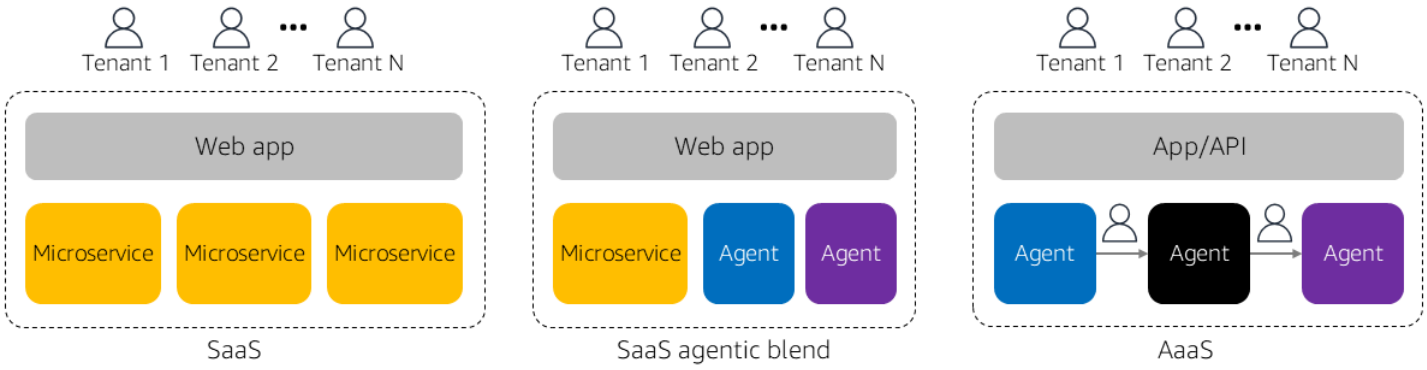
Dalam tampilan AaaS ini, asal dan konteks permintaan yang masuk secara signifikan mempengaruhi jejak agen. Sumber daya, tindakan, dan alat yang merupakan bagian dari implementasi dasar agen dapat bervariasi untuk setiap permintaan penyewa yang masuk. Nilai agen terhubung dengan kemampuannya untuk menggunakan konteks penyewa untuk sampai pada tindakan dan hasil yang dipengaruhi oleh keadaan penyewa, pengetahuan, dan faktor lainnya. Beberapa permintaan dapat menghasilkan hasil penyewa yang unik, dan yang lain dapat menghasilkan hasil per penyewa

yang lebih disesuaikan. Ini menambah dimensi baru pada kemampuan agen untuk belajar, yang dapat mencakup menjadi lebih kontekstual dan memperoleh dan menerapkan pengetahuan yang meningkatkan hasil yang ditargetkan.

Untuk penyedia, model AaaS menawarkan banyak keuntungan. Dengan banyak pelanggan yang mengkonsumsi satu agen, penyedia memiliki peluang yang lebih baik untuk mencapai skala ekonomi, mendorong efisiensi operasional, mengendalikan biaya, dan menciptakan pengalaman manajemen terpadu. Ini memiliki potensi untuk kelincahan, inovasi, dan pertumbuhan yang lebih besar untuk bisnis agen.

Kualitas ini tumpang tindih dengan prinsip yang sama yang mendorong adopsi model perangkat lunak sebagai layanan (SaaS). Pada dasarnya, model AaaS dibangun sebagai layanan multi-penyewa yang mewarisi banyak skala, ketahanan, isolasi, orientasi, dan atribut operasional yang sama yang ditemukan di lingkungan SaaS. Dalam banyak hal, pengalaman AaaS banyak meminjam dari strategi dan praktik yang digunakan oleh penyedia SaaS, tetapi masuk akal untuk memisahkan persyaratan ini. Untuk tujuan kami, penekanannya sebagian besar pada implikasi yang datang dengan agen bangunan dan operasi yang membutuhkan dukungan multi-penyewa.

Untuk sistem yang dapat memperlakukan semua pengguna secara setara dan tidak memerlukan pengelolaan data yang persisten, sensitif, atau spesifik pelanggan, gagasan penyewaan minimal akan mempengaruhi agen mereka. Untuk sistem yang diharapkan dapat melayani banyak pelanggan sambil mempertahankan isolasi data, penyesuaian, dan kesadaran konteks, mendukung banyak penyewa dapat menjadi elemen penting dari desain, strategi, dan tujuan agen. Diagram berikut menunjukkan bagaimana multi-tenancy dapat digunakan di lingkungan agen.



Di sisi kiri diagram ini adalah arsitektur multi-tenant klasik. Ini termasuk aplikasi web dan serangkaian layanan mikro yang menerapkan logika bisnis. Beberapa penyewa mengkonsumsi infrastruktur bersama dari lingkungan ini, skala untuk memenuhi beban kerja yang bergeser dari populasi penyewa yang berkembang. Lingkungan dioperasikan dan dikelola melalui satu panel kaca untuk semua penyewa.

Bayangkan bagaimana model mental ini memetakan ke agen di sisi kanan diagram ini. Satu agen menjalankan model AaaS yang dikonsumsi oleh satu atau lebih penyewa. Agen dapat berasal dari beberapa penyedia dengan konteks penyewa yang mengalir di antara mereka karena satu contoh dari satu agen harus memproses permintaan dari beberapa penyewa.

Contoh di tengah diagram ini adalah model hibrida di mana agen adalah bagian dari pengalaman SaaS secara keseluruhan. Beberapa bagian dari sistem diimplementasikan dalam model yang lebih tradisional dan bagian lain dari sistem bergantung pada agen. Pola ini cenderung umum untuk banyak penawaran SaaS—terutama untuk organisasi yang beralih ke pengalaman agen. Sudah umum model ini bertahan karena tidak semua sistem dikirimkan sebagai AAA murni. Perhatikan juga bahwa multi-tenancy masih berlaku untuk agen model. Sementara agen mungkin tertanam dalam suatu sistem, mereka masih dapat memproses permintaan dari beberapa penyewa.

Wajar untuk bertanya apakah multi-tenancy benar-benar penting. Anda dapat berargumen bahwa agen memproses permintaan, sehingga mendukung penyewaan mungkin memiliki sedikit efek. Tetapi saat kami menggali lebih dalam implikasi agen multi-penyewa, penyewaan dapat secara langsung memengaruhi bagaimana agen memengaruhi bagaimana alat, memori, data, dan bagian agen lainnya diakses, digunakan, dan dikonfigurasi untuk mendukung penyewa individu. Tenancy juga memengaruhi bagaimana penskalaan, pelambatan, penetapan harga, tiering, dan aspek bisnis lainnya berlaku untuk arsitektur agen Anda.

Satu hal yang dapat diambil dari ini adalah bahwa ada kasus penggunaan agen yang memerlukan dukungan multi-tenancy. Tantangannya adalah menentukan bagaimana multi-tenancy membentuk keseluruhan desain dan arsitektur pengalaman agen Anda. Untuk beberapa agen, dukungan multi-penyewa mewakili kemampuan pembeda, memungkinkan agen untuk menerapkan konteks khusus penyewa ke agen yang memberikan hasil yang ditargetkan.

Di bagian selanjutnya, Anda akan melihat bagaimana terminologi dan pola desain yang kami buat untuk menggambarkan arsitektur SaaS multi-penyewa akan berguna. Konsep-konsep ini dapat diadopsi oleh model AaaS dengan meminjam aspek yang berguna, yang memperkenalkan konsep khusus agen baru di mana mereka dibutuhkan.

Identitas, konteks penyewa, dan sistem agen

Menambahkan konteks penyewa ke agen individu tidak terlalu menantang. Dalam banyak kasus, tim dapat mengandalkan mekanisme khas yang mengikat pengguna dan sistem ke penyewa dan meneruskan token sadar penyewa ke agen. Ini relevan ketika kami mempertimbangkan bagaimana konteks dan identitas penyewa mendukung banyak agen. Dalam model ini, penyewa harus terikat pada identitas yang mencakup semua agen yang berkolaborasi.

Secara umum, domain agen membutuhkan model identitas yang lebih lintas sektoral yang selaras dengan kebutuhan sistem agen saat ini dan yang muncul. Penyedia agen memerlukan mekanisme identitas yang mendukung model keamanan, kepatuhan, dan otorisasi unik yang disertakan dengan sistem agen operasi. Ini sangat menantang di lingkungan di mana sistem disusun oleh pelanggan atau agen lain. Setiap agen onboard harus menghubungkan identitas dan konteks penyewa dengan interaksi agen. Diagram berikut menyoroti potensi identitas dan tantangan konteks penyewa yang merupakan bagian dari interaksi agent-to-agent (a2a).

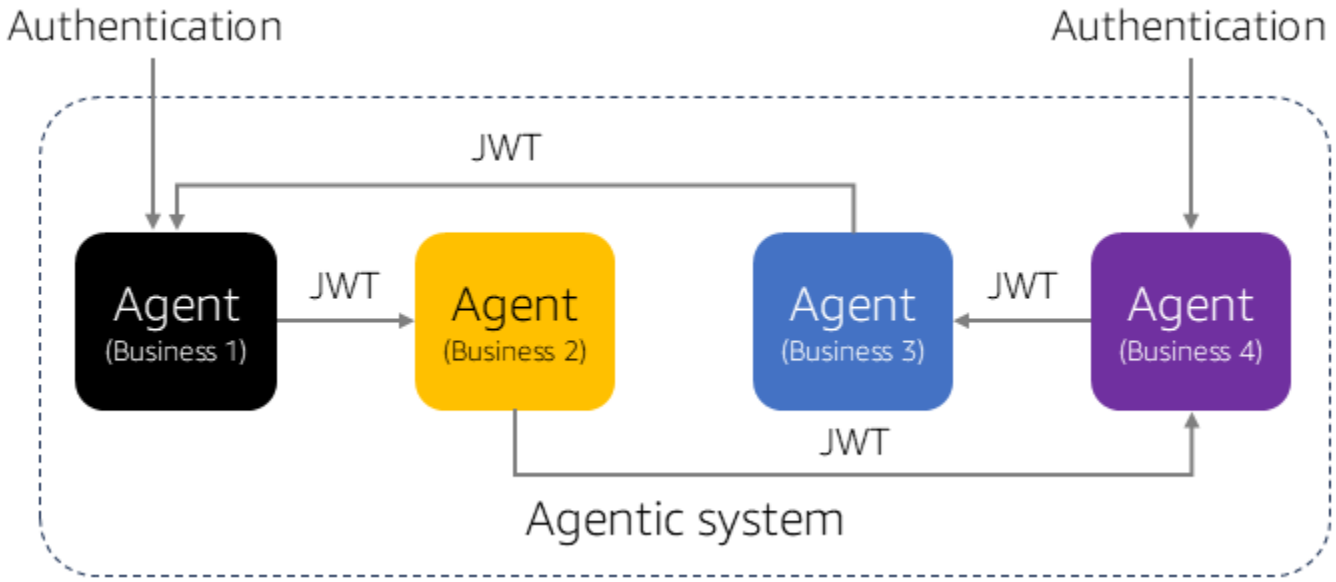


Diagram ini menunjukkan serangkaian agen buatan penyedia yang berinteraksi sebagai bagian dari sistem agen yang kami bahas. Sekarang dilengkapi dengan identitas dan konteks penyewa. Skenario ini adalah contoh dari sistem agen yang mendukung beberapa titik masuk. Kami berasumsi bahwa setiap agen dalam sistem ini memerlukan mekanisme otentikasi sendiri untuk menyelesaikan sistem atau pengguna untuk penyewa tertentu. Saat agen ini berinteraksi, konteks penyewa diteruskan ke token web JSON (JWT) yang akan digunakan untuk mengotorisasi akses dan menyuntikkan konteks penyewa ke agen.

Secara konseptual, perbedaan utama dengan skenario ini adalah bahwa agen menyebarkan dan beroperasi secara independen, yang berarti bahwa setiap agen harus dapat menyelesaikan identitasnya dan mengotorisasi akses. Kuncinya adalah bahwa identitasnya harus memiliki beberapa kemampuan terdistribusi untuk menangani kebutuhan sistem agen yang lebih luas. Juga harus ada keselarasan tentang bagaimana agen berbagi konteks penyewa.

Menerapkan nilai bisnis SaaS ke AaaS

Secara umum, ketika kita melihat menjalankan sistem apa pun dalam suatu as-a-service model, kita mempertimbangkan sifat pengalaman dan bagaimana jejak teknis dan operasionalnya mendorong hasil bisnis. Ketika mengadopsi SaaS, misalnya, organisasi menggunakan skala ekonomi, efisiensi operasional, profil biaya, dan kelincahan untuk mendorong pertumbuhan, margin, dan inovasi.

Agen yang dikirim sebagai AAA cenderung menargetkan hasil bisnis yang serupa. Dengan mendukung beberapa penyewa, agen dapat menyelaraskan konsumsi sumber daya dengan aktivitas penyewa. Ini menghasilkan skala ekonomi yang datang dengan lingkungan SaaS tradisional. AaaS juga memungkinkan organisasi untuk mengelola, mengoperasikan, dan menyebarkan agen dengan cara yang memungkinkan rilis yang sering dan mendorong kelincahan bagi penyedia agen. Kuncinya adalah model AaaS tidak bergantung pada teknologi. Ini menciptakan dan mendorong strategi bisnis yang mendorong pertumbuhan, merampingkan adopsi, dan menyederhanakan operasi.

Model penyebaran agen

Dalam pengalaman AaaS dasar, penyedia dapat menyebarkan agen menggunakan berbagai pola. Ada banyak faktor yang mempengaruhi bagaimana agen dikerahkan untuk memenuhi kebutuhan pelanggan, kinerja, kepatuhan, geografi, dan keamanan. Strategi penyebaran yang berbeda mempengaruhi bagaimana agen dirancang, diimplementasikan, dan dikonsumsi. Di sinilah kami dapat memperkenalkan istilah multi-tenant klasik untuk memberi label strategi penerapan yang berbeda. Diagram berikut menunjukkan permutasi yang berbeda untuk menyebarkan agen di lingkungan AaaS.

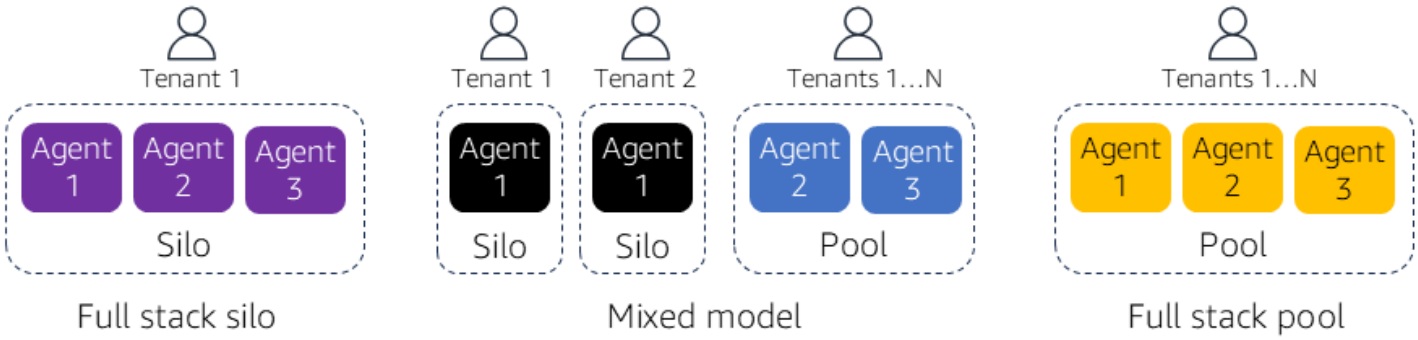
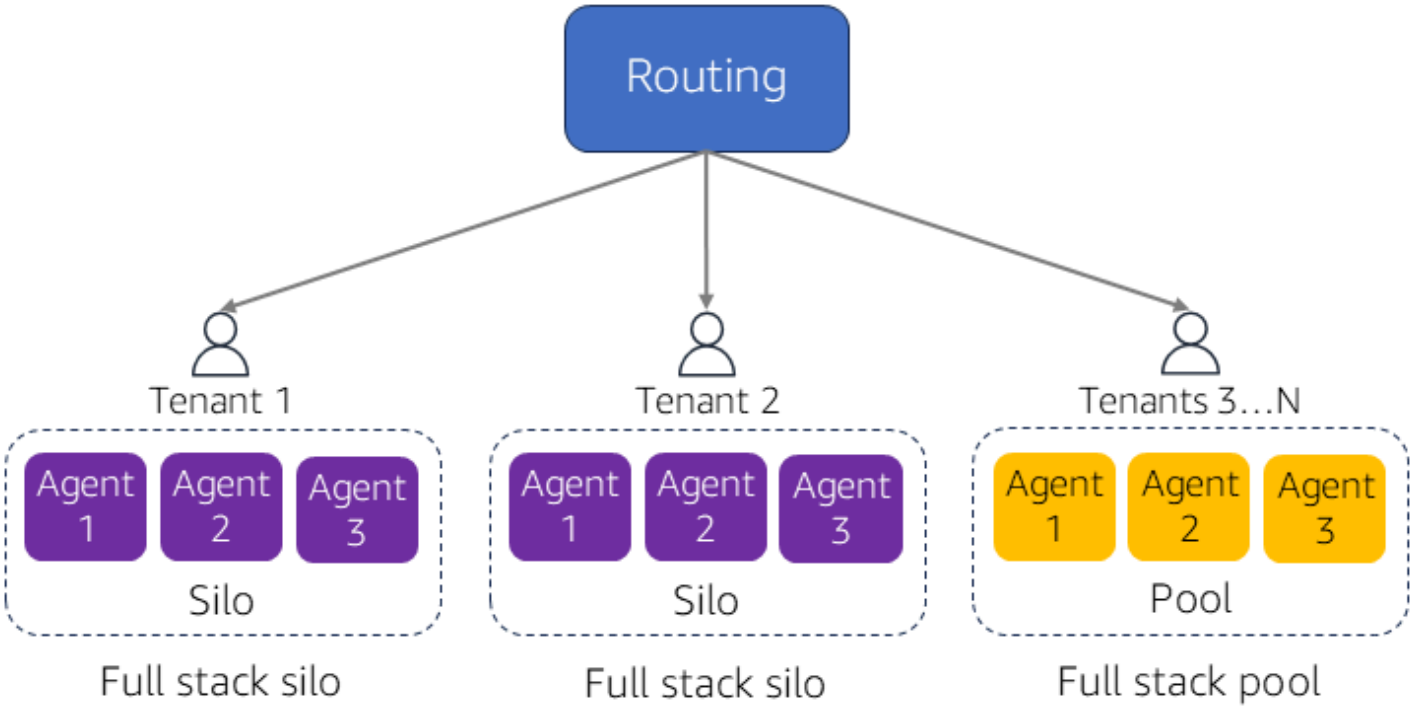


Diagram ini mewakili tiga mode penyebaran agen. Di sisi kiri adalah model silo, di mana setiap penyewa diberikan pengalaman yang sepenuhnya terisolasi dan satu set agen khusus. Dalam skenario ini, agen tidak berbagi komputasi, sumber daya, atau lingkungan eksekusi di seluruh penyewa.

Contoh tengah menggambarkan model hibrida, di mana penyewa menggunakan kombinasi agen silo dan gabungan. Misalnya, Agen 1 digunakan dalam mode siloed—setiap penyewa menerima instance khusus—sementara agen 2 dan 3 beroperasi dalam model gabungan, berbagi sumber daya di seluruh penyewa.

Di sisi kanan adalah model yang sepenuhnya dikumpulkan, di mana semua agen dibagikan di seluruh penyewa, menawarkan penyebaran multi-penyewa klasik. Dalam skenario ini, penyewa memanfaatkan komputasi umum, memori, dan infrastruktur layanan untuk eksekusi agen.

Idenya adalah bahwa agen dapat beroperasi dalam model penyebaran yang berbeda, dengan sumber daya komputasi dan dependen baik yang didedikasikan (siloed) atau dibagikan (dikumpulkan) di seluruh penyewa. Strategi penyebaran ini tidak saling eksklusif. Layanan agen sering mendukung spektrum kebutuhan pelanggan, menggabungkan kedua model untuk menyeimbangkan kinerja, isolasi, biaya, dan skalabilitas. Diagram berikut menunjukkan sistem agen yang mendukung beberapa konfigurasi penerapan dalam lingkungan operasional yang sama.

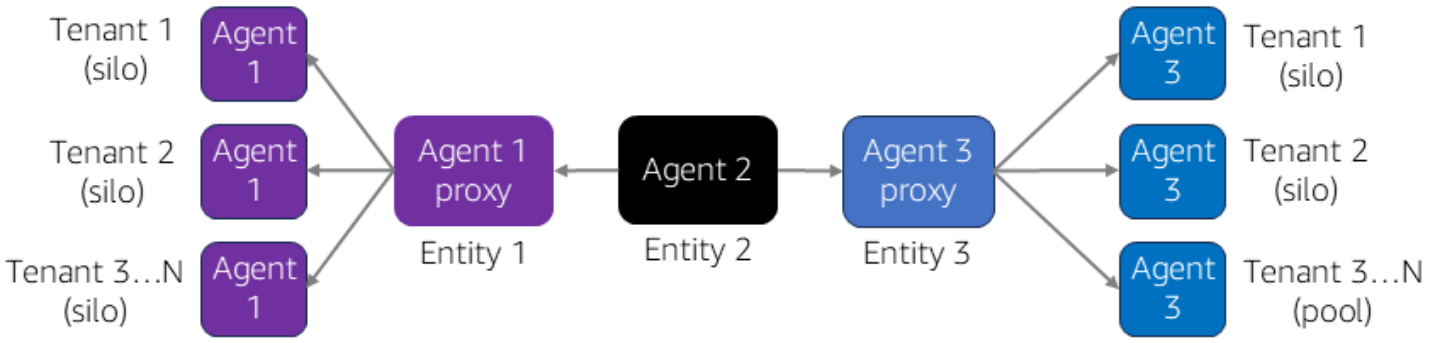


Dalam diagram ini, penyedia agen memiliki tiga agen yang digunakan melalui agen sebagai layanan (AaaS). Mereka mendukung dua jenis penyewa. Di sisi kiri, dua penyewa memiliki persyaratan kepatuhan dan kinerja yang mereka tangani melalui model silo full-stack. Penyewa yang tersisa di sisi kanan berjalan dalam model gabungan di mana penyewa berbagi sumber daya.

Jika tujuannya adalah kelincahan dan efisiensi operasional, cobalah untuk membatasi efek yang terkait dengan mendukung model penyebaran per-penyewa. Ini berarti menempatkan routing dan mekanisme pengalaman lainnya di tempat yang memungkinkan agen untuk dikelola, dioperasikan, dan digunakan melalui satu panel kaca.

Jika Anda membangun agen di lingkungan rendah atau tanpa kode, tidak akan ada gagasan tentang agen yang dibungkam atau dikumpulkan. Sebaliknya, agen dapat sepenuhnya dikelola oleh agen lain. Model silo dan gabungan lebih diterapkan pada lingkungan di mana organisasi mengontrol konstruksi dan jejak agen. Dalam hal ini, tim harus mempertimbangkan model penerapan mana yang akan didukung.

Di permukaan, model penerapan ini tidak secara langsung memengaruhi cara agen berfungsi dalam sistem yang lebih luas. Seorang agen mungkin tidak memiliki kesadaran langsung tentang agen lain yang digunakan dalam silo atau model gabungan. Sebaliknya, strategi penyebaran ini dapat diimplementasikan sebagai bagian dari konstruksi perutean dalam suatu lingkungan. Diagram berikut menunjukkan contoh bagaimana model siloed dan pooled dapat diimplementasikan menggunakan strategi routing.



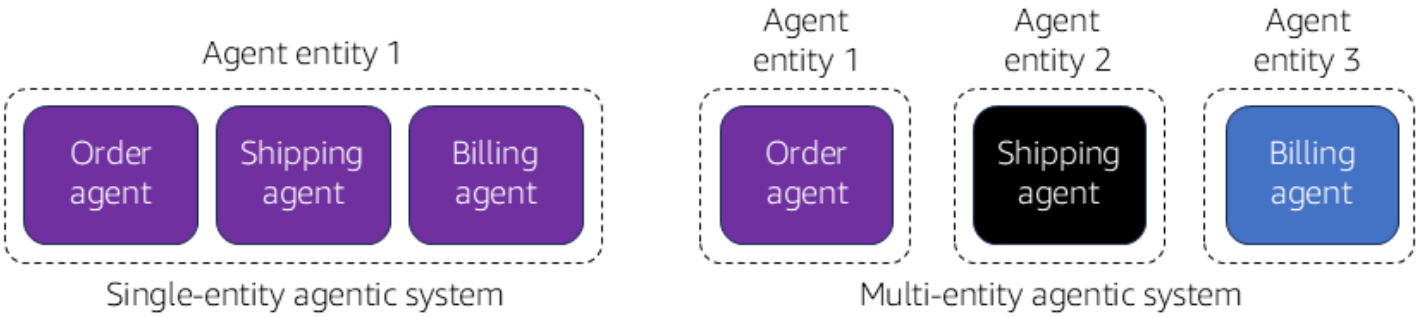
Contoh ini mencakup tiga agen dari tiga penyedia yang berbeda. Setiap penyedia agen memiliki opsi untuk menerapkan strategi penyebaran sendiri. Misalnya, agen 1 menggunakan proxy untuk mendistribusikan permintaan masuk ke satu set agen penyewa tersilo. Agen 2 tidak memerlukan perutean dan mendukung semua permintaan penyewa melalui satu agen gabungan. Agen 3 adalah penyebaran model hibrida di mana beberapa penyewa dibungkam dan yang lainnya dikumpulkan.

Jika dan bagaimana Anda memilih untuk mendukung model penerapan ini bergantung pada sifat solusi Anda. Anda mungkin tidak perlu mendukung kedua model tersebut. Namun, Anda mungkin memiliki contoh di mana Anda harus mempertimbangkan untuk mendukung strategi ini, seperti kepatuhan, tetangga yang berisik, kinerja, atau tingkatan.

Memperkenalkan dan menerapkan konteks penyewa

Jika kita membangun agen yang mendukung multi-tenancy, kita harus mulai dengan mempertimbangkan cara mengatur konteks penyewa, yang akan digunakan untuk menerapkan kebijakan, strategi, dan mekanisme khusus penyewa dalam implementasi agen.

Pada tingkat paling dasar, Anda dapat memperkenalkan konteks penyewa ke agen melalui alat dan mekanisme umum yang kami gunakan dalam arsitektur multi-penyewa klasik. Ini bisa melalui kunci API, OAuth, atau berbagai mekanisme validasi lainnya. Banyak contoh fokus ini pada penyelesaian sistem yang diautentikasi atau pengguna ke kunci token web JSON (JWT) yang memegang konteks penyewa. JWT kemudian disebarkan melalui sistem. Ini menjadi lebih menarik ketika kita mempertimbangkan cara menyusun sistem agen. Diagram berikut menunjukkan contoh dua varietas lingkungan agen.



Dalam diagram ini, model di sisi kiri mewakili sistem agen di mana semua agen dimiliki, dikelola, dan dihosting oleh satu entitas. Ketika Anda memiliki kendali penuh atas seluruh pengalaman, Anda dapat menggunakan strategi khas untuk melewati penyewa melalui setiap agen.

Model di sisi kanan, yang mungkin lebih umum, mewakili sistem agen yang menjangkau beberapa entitas. Agen secara independen dibangun, dikelola, dan dioperasikan, sehingga mereka masing-masing memiliki skema otentikasi dan otorisasi mereka sendiri. Tantangannya di sini adalah bahwa kita membutuhkan cara universal untuk menyelesaikan dan berbagi konteks penyewa di antara agen-agen ini. Ini bergantung pada model yang lebih terdistribusi di mana setiap agen harus dapat mengautentikasi sistem atau pengguna dan menyelesaikannya ke penyewa sesuai dengan mekanisme yang diterapkan.

Membangun agen yang sadar penyewa

Multi-tenancy memengaruhi cara kami menerapkan agen individu. Saat agen memproses permintaan, pertimbangkan bagaimana konteks penyewa memengaruhi cara agen mengakses data,

membuat keputusan, dan memanggil tindakan. Untuk lebih memahami bagaimana dan di mana multi-tenancy memengaruhi profil agen Anda, pertama-tama tentukan bagaimana konstruksi dapat menjadi bagian dari agen mana pun.

Tantangannya adalah bahwa ruang lingkup, sifat, dan desain agen sama sekali tidak konkret karena penyedia membuat pilihan mereka sendiri tentang desain pengalaman agen. Pada akhirnya, inti dari agen adalah bahwa itu adalah layanan pembelajaran otonom yang dapat mengakses berbagai alat, sumber data, dan memori untuk menentukan cara terbaik untuk menyelesaikan tugas.

Kurang penting untuk mengetahui dengan tepat strategi dan pola mana yang digunakan agen. Dalam model multi-tenant, lebih penting untuk mengidentifikasi bagaimana berbagai bagian agen dikonfigurasi, diakses, dan diterapkan. Pertimbangkan lingkungan agen potensial yang bergantung pada serangkaian sumber daya dan mekanisme untuk mencapai tujuannya. Diagram berikut menunjukkan contoh agen semacam itu.

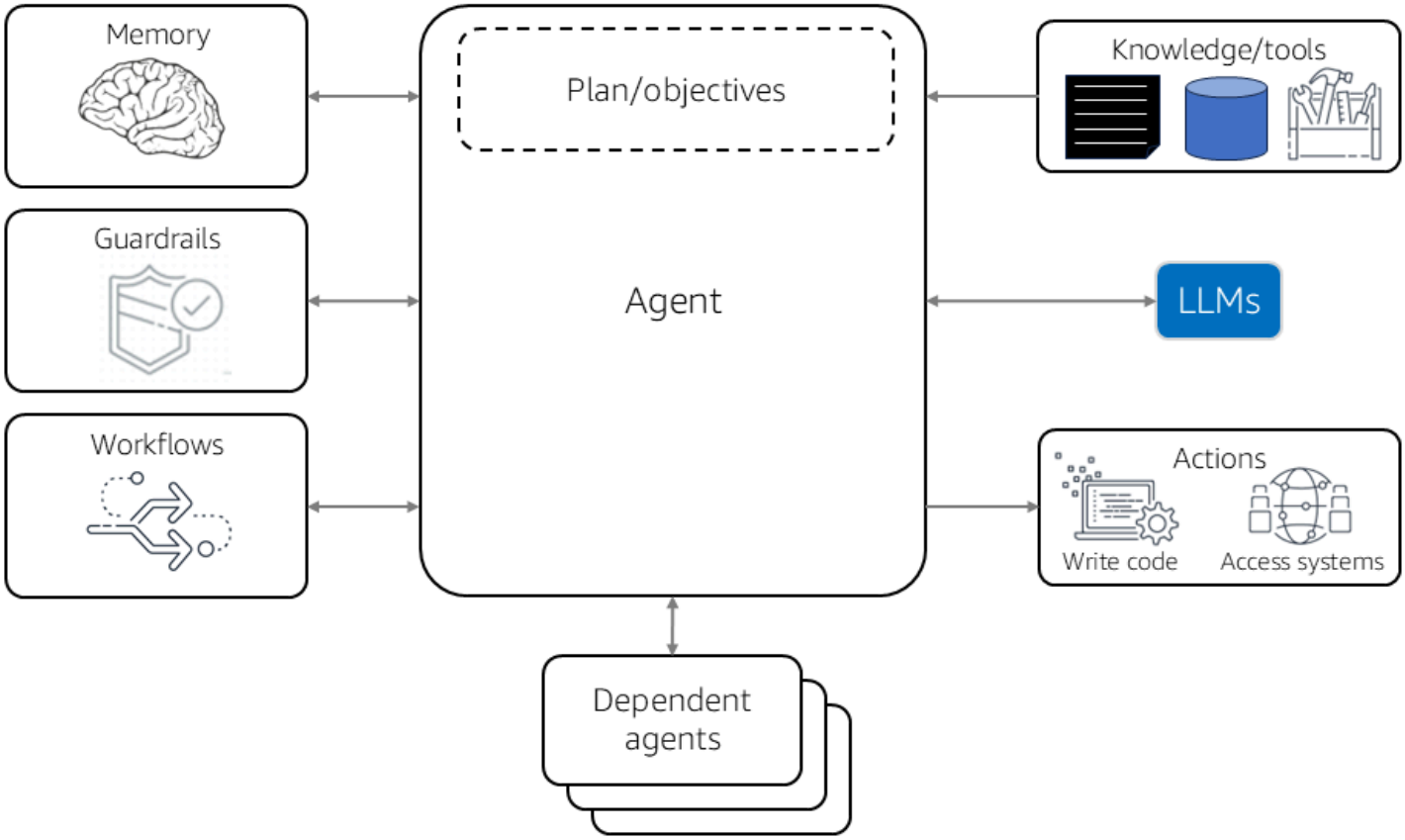


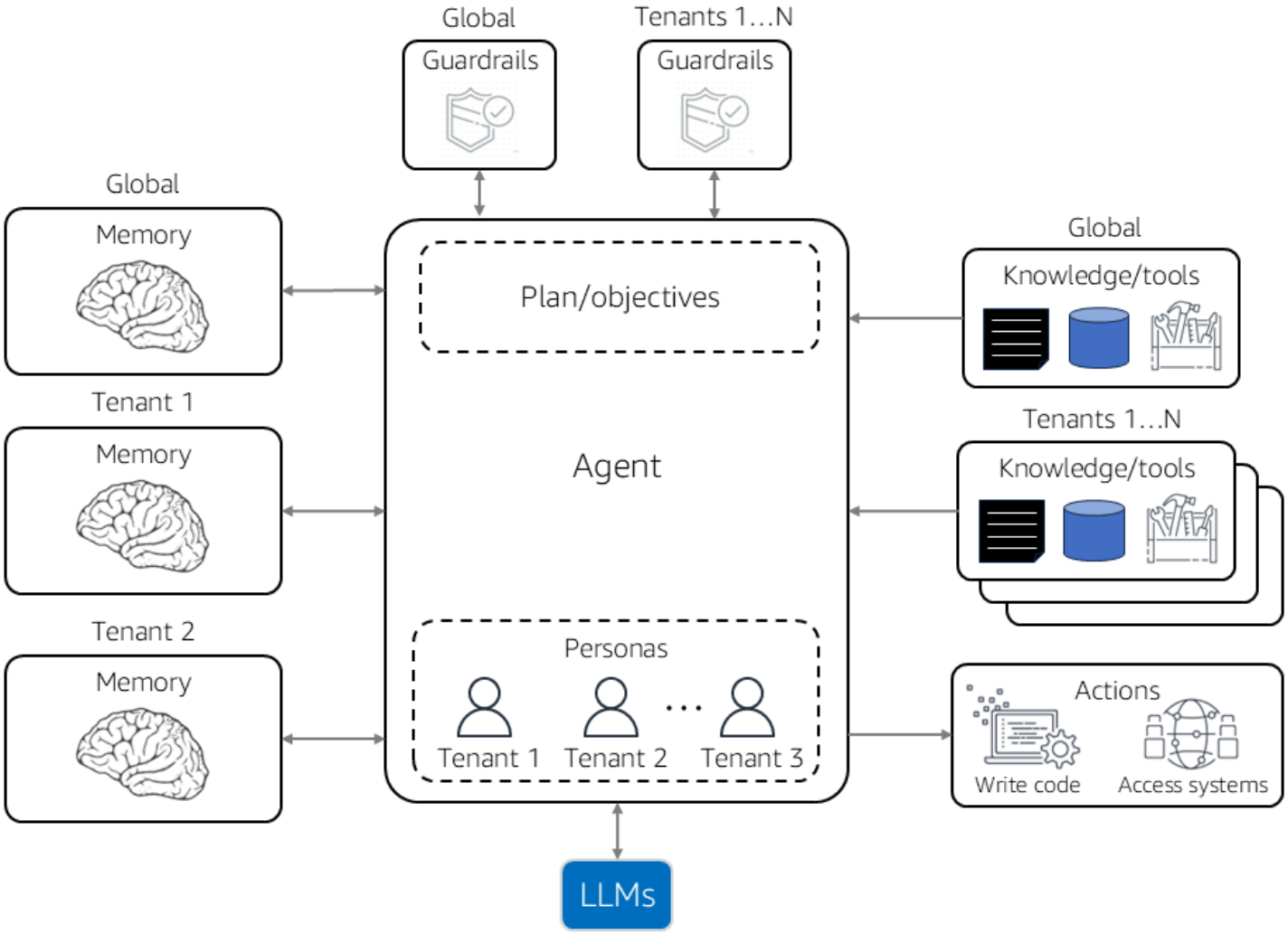
Diagram ini mewakili berbagai kemungkinan agen yang komprehensif, menampilkan berbagai alat dan mekanisme yang dapat digabungkan untuk mencapai suatu tujuan. Di sisi kiri diagram, perhatikan bagaimana agen bergantung pada memori sebagai bagian dari konteksnya, pagar pembatas untuk mendefinisikan kebijakan yang memandu aktivitasnya, dan alur kerja yang diarahkan

pada tugas tertentu. Beberapa orang mungkin berpendapat bahwa alur kerja tidak boleh disertakan dalam konteks ini, tetapi mungkin ada skenario di mana alur kerja merupakan bagian integral dari pengalaman agen.

Sisi kanan diagram menunjukkan bagaimana input seperti pengetahuan dan alat dapat memberikan wawasan dan konteks tambahan yang meningkatkan kemampuan agen. Agen kemudian mengeluarkan tindakan, seperti menulis kode atau mengakses sistem. Bagian bawah diagram menunjukkan bagaimana agen bergantung pada satu atau lebih agen internal atau pihak ketiga yang dapat diatur sebagai bagian dari sistem yang lebih luas.

Kita sekarang dapat berpikir tentang apa artinya memperkenalkan multi-tenancy. Penyewaan memaksa kita untuk mempertimbangkan bagaimana dan di mana agen memperkenalkan strategi dan mekanisme yang mendikte perilaku dan tindakan. Ini menambah dimensi lain pada bagaimana kita berpikir tentang agen dalam hal pengetahuan, pembelajaran, alat, dan memori mereka.

Sekarang mari kita pertimbangkan cara memodifikasi model ini untuk mendukung multi-tenancy. Diagram berikut menunjukkan contoh model multi-agen.



Dalam diagram ini, kami memperkenalkan persona penyewa yang dimaksudkan untuk membentuk bagaimana agen mengintegrasikan konteks penyewa. Misalnya, di sisi kiri diagram, memori agen diubah untuk mendukung memori khusus penyewa. Hal yang sama berlaku di sisi kanan diagram di mana agen mendukung pengetahuan dan alat khusus penyewa. Dukungan yang sama juga diterapkan pada pagar pembatas.

Ini mungkin contoh ekstrem karena tidak semua aspek agen multi-penyewa memerlukan sumber daya per penyewa. Intinya adalah Anda harus mempertimbangkan bagaimana menyesuaikan agen Anda untuk penyewa tertentu dapat meningkatkan efektivitasnya. Pendekatan ini memungkinkan agen Anda untuk meningkatkan dampak dan nilainya, memberikan konteks yang lebih relevan dalam tanggapannya, dan mengembangkan kemampuan khusus. Agen kemudian akan dapat belajar, beradaptasi, dan melakukan tugas-tugas yang secara unik cocok untuk persona yang berbeda.

Ide utamanya adalah bahwa konteks penyewa secara langsung memengaruhi cara Anda membangun agen. Ini juga dapat membentuk interaksi penyewa dengan entitas eksternal, termasuk

agen lain. Membangun agen multi-tenant memperkenalkan tantangan tradisional seperti tetangga yang bising, isolasi penyewa, tiering, throttling, dan manajemen biaya. Desain dan arsitektur agen Anda harus membahas konsep multi-penyewa dasar ini, yang akan kita jelajahi di bagian selanjutnya.

Mempekerjakan pesawat kontrol di lingkungan agen

Praktik terbaik multi-tenant sering membagi implementasi menjadi dua bagian yang berbeda: bidang kontrol dan bidang aplikasi. Pesawat kontrol menyediakan satu panel kaca untuk mengakses mekanisme operasional, manajemen, dan orkestrasi yang menjangkau penyewa lingkungan. Bidang aplikasi adalah tempat logika bisnis, fitur, dan kemampuan fungsional berada.

Pembagian tanggung jawab ini juga berlaku untuk model agen. Agen multi-penyewa membutuhkan tingkat manajemen, operasi, dan wawasan terpusat, dan masuk akal untuk terus memenuhi kebutuhan ini melalui bidang kontrol. Diagram berikut menunjukkan pandangan konseptual tentang bagaimana bidang ini dibagi dalam lingkungan agen sebagai layanan (AaaS).

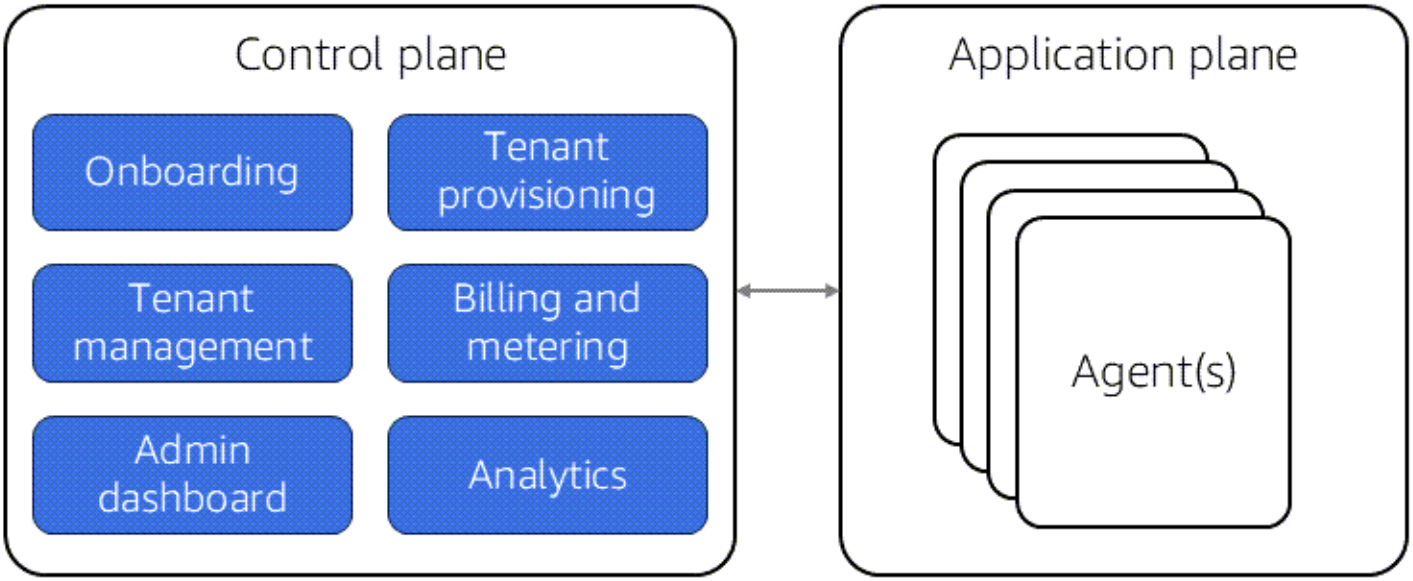


Diagram ini menunjukkan pemisahan tradisional bidang kontrol dan aplikasi. Apa yang baru adalah bahwa pesawat kontrol sekarang mengelola agen yang membentuk lingkungan AaaS. Pesawat kontrol berinteraksi dengan semua agen karena kami menganggap bahwa agen dibangun, dikelola, dan dikerahkan oleh satu penyedia.

Model ini memperkenalkan lapisan kompleksitas tambahan, terutama dalam siklus hidup agen dan koordinasi pihak ketiga, tetapi mempertahankan pemisahan mendasar dari kekhawatiran. Pesawat kontrol masih menyediakan kemampuan inti yang sama dengan mengatur konfigurasi agen, menyediakan observabilitas penyewa dan agen, mengumpulkan data konsumsi dan pengukuran untuk penagihan, dan mengelola kebijakan penyewa.

Skenario ini menjadi lebih kompleks jika Anda mempertimbangkan sistem multi-agen yang menggabungkan agen dari berbagai penyedia. Diagram berikut menunjukkan contoh model seperti itu.

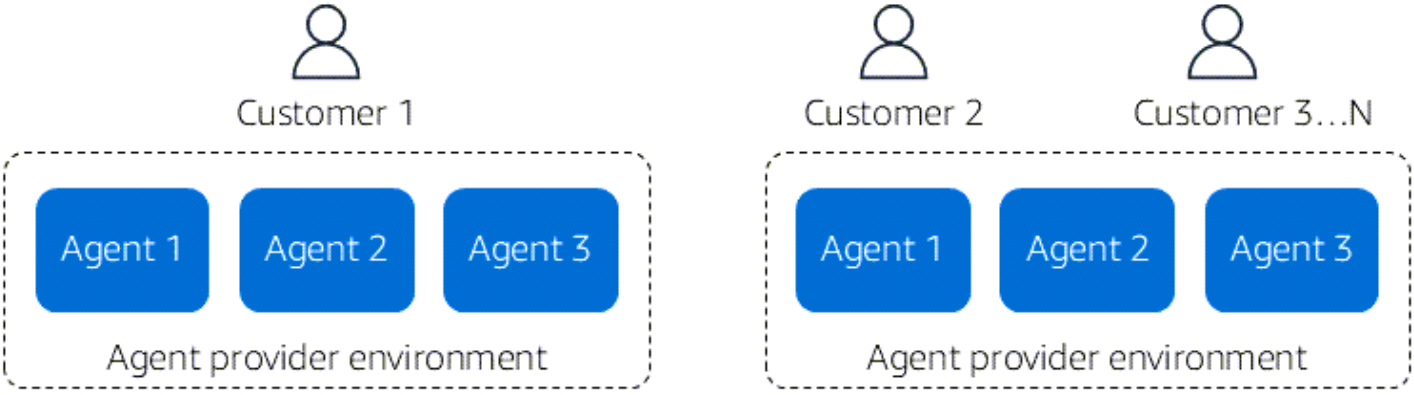


Diagram ini menggambarkan empat agen dari penyedia yang berbeda yang merupakan bagian dari sistem multi-agen. Penyedia pihak ketiga masih mengoperasikan dan menyebarkan setiap agen, yang dikonfigurasi untuk mengaktifkan akses resmi dari satu atau beberapa penyedia. Agen, bagaimanapun, tetap di bawah kendali penyedia, sehingga setiap agen mempertahankan pesawat kontrolnya sendiri.

Pada dasarnya, agen multi-penyewa ini berperilaku sebagai layanan pihak ketiga yang terintegrasi dengan agen lain. Dengan demikian, mereka harus memiliki pesawat kontrol sendiri untuk menyediakan operasi terpusat, konfigurasi, dan manajemen kemampuan agen.

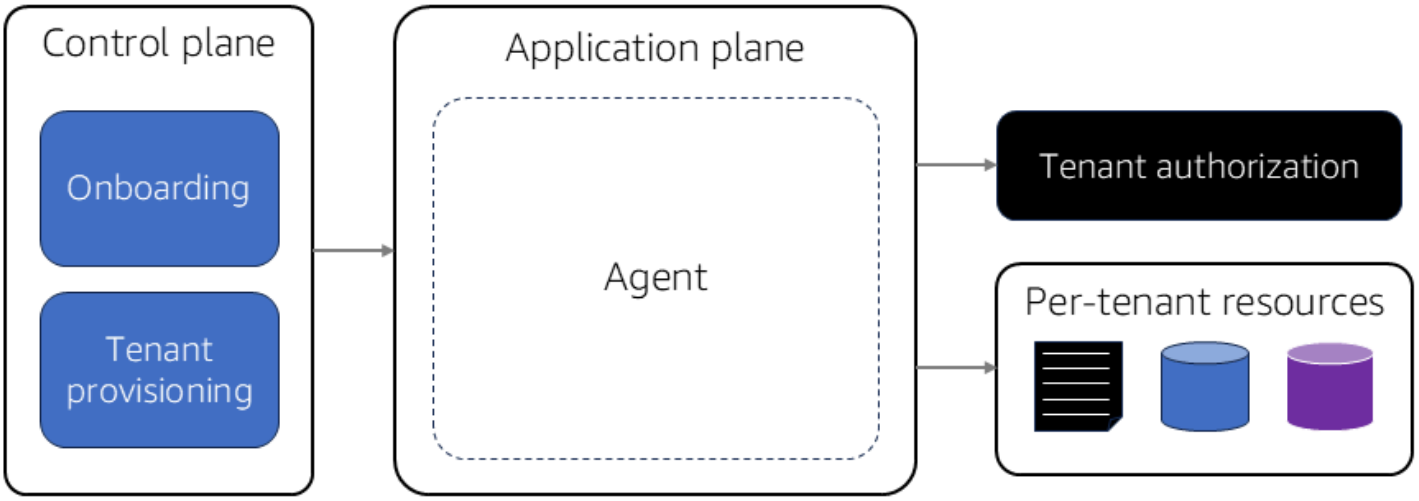
Kami berasumsi bahwa agen adalah layanan independen yang berjalan dalam pengalaman yang dihosting penyedia. Tapi ini mungkin tidak jelas dalam skenario di mana konsumen agen memaksakan lebih banyak kendala pada bagaimana dan di mana menjadi tuan rumah agen.

Penyewa orientasi ke agen

Orientasi biasanya merupakan bagian penting dari lingkungan AaaS mana pun. Cara Anda membuat, mengonfigurasi, dan menyediakan penyewa sering kali melibatkan banyak bagian, integrasi, dan alat yang bergerak. Pengalaman orientasi agen mungkin memerlukan layanan yang sama yang ditemukan di pesawat kontrol AaaS, yang mencakup identitas penyewa, tingkatan, penyediaan sumber daya per penyewa, dan konfigurasi kebijakan penyewa.

Pendekatan Anda terhadap orientasi agen dipengaruhi oleh jejak dan model penyewaan lingkungan agen Anda. Agen silo dan gabungan masing-masing memiliki nuansa tersendiri, dan pilihan untuk menggunakan agen tunggal atau beberapa agen juga memengaruhi proses orientasi. Diagram

berikut menunjukkan pandangan konseptual tentang bagaimana orientasi mempengaruhi konfigurasi agen.



Setiap kali Anda naik agen, pesawat kontrol harus mengambil langkah-langkah yang diperlukan untuk memungkinkan penyewa mengakses agen. Cara memperkenalkan penyewa bervariasi berdasarkan model otorisasi agen, tetapi asumsikan bahwa Anda akan membuat identitas penyewa yang mengaitkan permintaan agen dengan penyewa individu. Konteks penyewa ini menentukan pengalaman agen dengan menerapkannya pada rute, cakupan, dan akses kontrol.

Orientasi mungkin juga mengharuskan Anda untuk mengonfigurasi sumber daya per penyewa apa pun yang digunakan agen. Di sinilah layanan penyediaan penyewa pesawat kontrol menghubungkan agen Anda ke data dan sumber daya khusus penyewa yang dikonsultasikan agen.

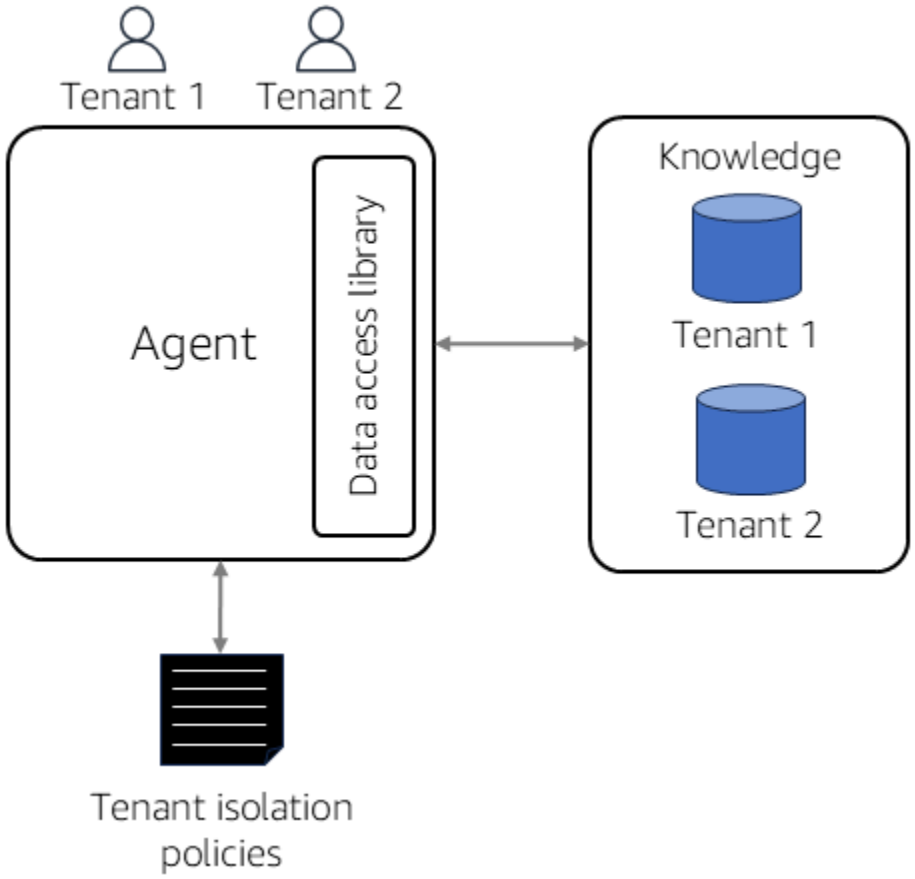
Jika sistem Anda bergantung pada integrasi agen pihak ketiga, Anda juga harus memenuhi kebutuhan agen tersebut selama proses orientasi. Cara kerjanya tergantung pada mekanisme keamanan dan integrasi untuk mengotorisasi akses antar agen. Idealnya, langkah-langkah yang diperlukan untuk mengatur dan mengonfigurasi agent-to-agent otentikasi dan otorisasi ditangani melalui orientasi otomatis.

Menegakkan isolasi penyewa

Isolasi penyewa adalah konsep yang berlaku untuk semua pengaturan multi-penyewa. Ini berarti bahwa kebijakan dan strategi Anda memastikan bahwa satu penyewa tidak dapat mengakses sumber daya penyewa lainnya. Untuk agen multi-penyewa, Anda mungkin perlu memperkenalkan konstruksi dan mekanisme yang membantu menegakkan dan persyaratan isolasi penyewa agen.

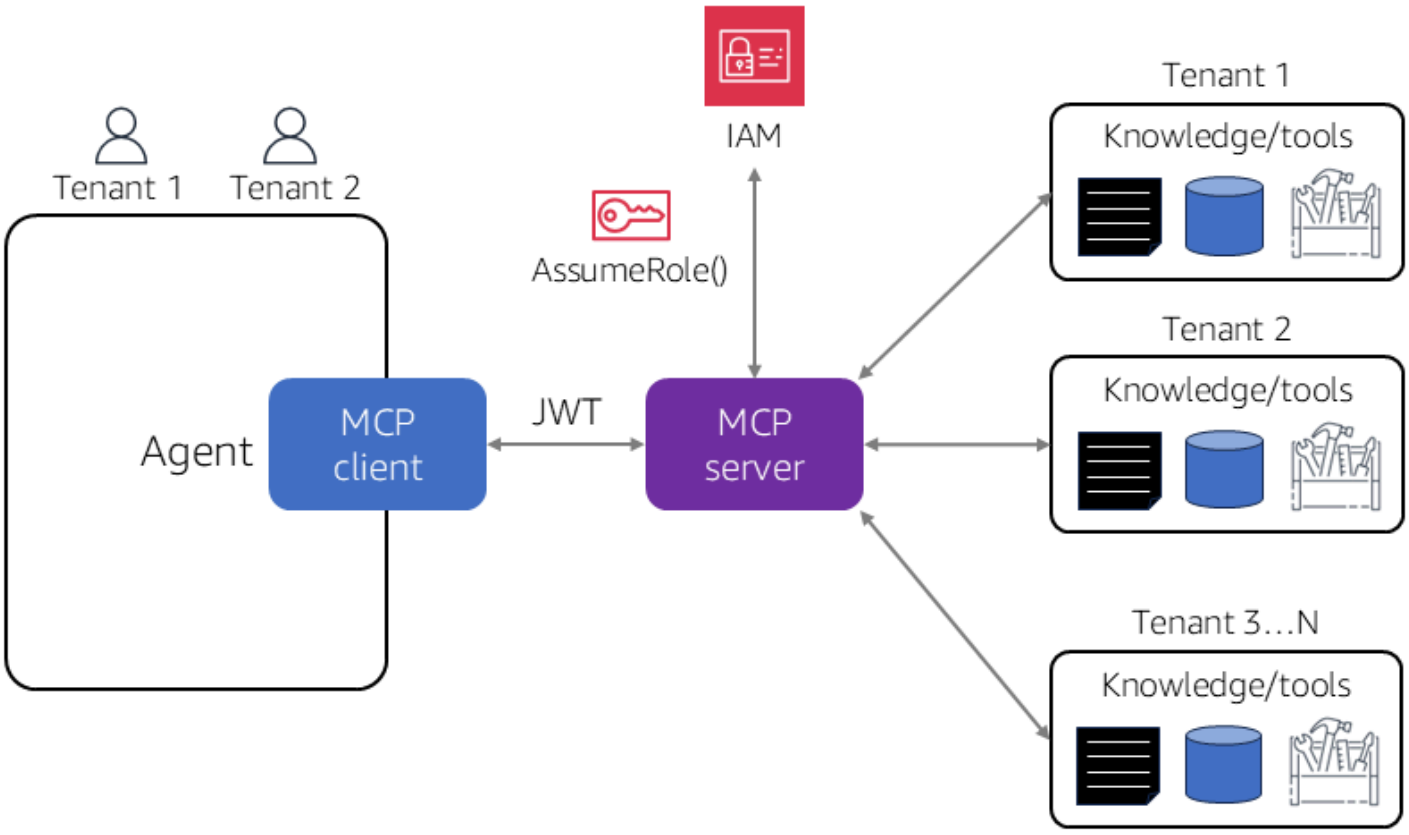
Menerapkan isolasi penyewa seperti strategi lain yang menggunakan sistem multi-tenant tradisional. Umumnya, ketika Anda membangun arsitektur AaaS, identifikasi area mana pun di sistem Anda di mana permintaan atau tindakan dapat mengakses sumber daya untuk menentukan apakah permintaan melintasi batas penyewa. Misalnya, layanan mikro mungkin memiliki dependensi pada tabel Amazon DynamoDB khusus per penyewa. Ini mengharuskan Anda untuk memperkenalkan kebijakan yang memastikan bahwa satu tabel penyewa tidak dapat diakses oleh penyewa lain.

Dalam hal ini, pertimbangkan isolasi penyewa melalui lensa agen dan interaksinya dengan sumber daya per-penyewanya. Diagram berikut menunjukkan contoh konseptual tentang bagaimana agen menerapkan kebijakan isolasi penyewa untuk mengontrol akses ke sumber daya penyewa.



Di sisi kanan diagram ini, agen memiliki pengetahuan per penyewa yang disimpan dalam database vektor terpisah. Saat agen memproses permintaan, ia memeriksa konteks penyewa yang membuat permintaan. Berdasarkan hal ini, agen menerapkan kebijakan isolasi yang tepat untuk memastikan bahwa penyewa dibatasi mengakses data atau sumber daya di luar batas yang ditentukan.

Jika agen Anda menggunakan Model Context Protocol (MCP), itu juga dapat mengimplementasikan model isolasi penyewa Anda. Diagram berikut menunjukkan contoh bagaimana memperkenalkan MCP dan menerapkan kebijakan isolasi.



MCP adalah protokol standar yang digunakan agen untuk berintegrasi dengan alat, data, dan sumber daya apa pun. Dalam contoh ini, klien MCP dan server MCP berinteraksi dengan pengetahuan dan alat khusus penyewa yang ditunjukkan di sisi kanan diagram. Konteks penyewa mengalir dari klien ke server, dan server menggunakan konteks ini untuk memperoleh kredensil cakupan penyewa dari layanan (IAM). AWS Identity and Access Management Kredensil mengontrol akses ke sumber daya masing-masing penyewa, memastikan bahwa satu penyewa dapat mengakses sumber daya penyewa lain.

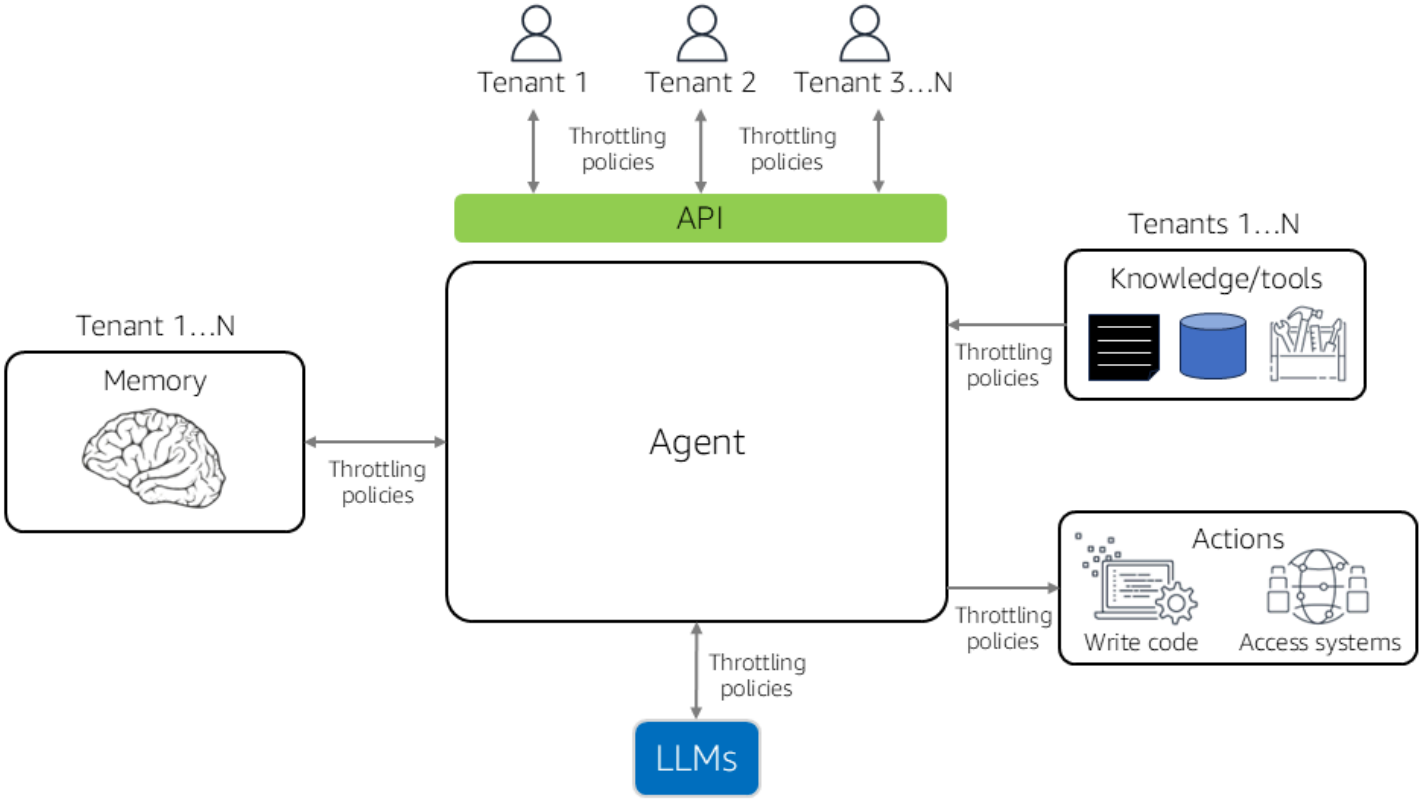
Karena agen menggabungkan multi-tenancy, mereka harus memperkenalkan mekanisme yang menerapkan kebijakan isolasi penyewa saat mereka memproses permintaan. Dalam beberapa kasus, IAM dapat membantu membatasi akses ke sumber daya penyewa. Dalam kasus lain, Anda

mungkin perlu memperkenalkan alat atau kerangka kerja lain untuk menerapkan kebijakan isolasi penyewa.

Tetangga dan agen yang berisik

Dalam lingkungan AaaS multi-penyewa di mana beberapa penyewa berbagi agen, pikirkan tentang di mana dan bagaimana memperkenalkan kebijakan yang mencegah kondisi tetangga yang bising. Kebijakan dapat memperkenalkan pembatasan tujuan umum yang berlaku untuk semua konsumsi, atau Anda dapat memiliki kebijakan penyewa atau berbasis tingkat yang menerapkan pembatasan berdasarkan persona tertentu. Anda dapat menempatkan pembatasan konsumsi yang lebih besar pada penyewa tingkat dasar daripada yang Anda lakukan pada penyewa tingkat premium.

Gagasan pelambatan ini dapat diterapkan pada beberapa titik arsitektur. Diagram berikut menunjukkan contoh beberapa area yang mungkin untuk memperkenalkan kebijakan tetangga yang bising.



Dalam tinjauan kami sebelumnya tentang implementasi multi-agen, kami memeriksa berbagai sumber daya yang dapat digunakan agen Anda, menyoroti potensi sumber daya per penyewa dalam agen. Setiap titik kontak adalah area potensial untuk memperkenalkan kebijakan pembatasan, yang

membantu memastikan bahwa penyewa tidak melebihi batas konsumsi sistem Anda atau kebijakan tingkat penyewa.

Tempat terbaik untuk memperkenalkan perlindungan tetangga yang bising adalah pada titik-titik dalam arsitektur di mana penyewa berbagi sumber daya. Komponen bersama atau gabungan ini, seperti komputasi, memori, dan model bahasa besar APIs, adalah yang paling rentan terhadap penurunan kinerja jika satu penyewa mengkonsumsi secara tidak proporsional.

Salah satu tempat alami untuk menerapkan throttling adalah pada titik masuk agen, kadang-kadang disebut “tepi luar.” Di sini, Anda dapat memperkenalkan batas global atau tenant-tier-based tarif sebelum agen mulai memproses permintaan. Throttling juga dapat diterapkan lebih dalam di jalur eksekusi, seperti ketika agen memanggil LLM, mengakses memori, atau memanggil alat bersama.

Kebijakan ini membantu Anda menegakkan penggunaan yang adil, menjaga ketahanan agen di bawah beban, dan mempertahankan pengalaman yang konsisten di seluruh penyewa. Bergantung pada tujuan Anda, Anda mungkin fokus pada perlindungan sistem umum (ketahanan) atau mengelola pengalaman penyewa secara terperinci (misalnya, dengan hak berbasis tingkatan).

Data, operasi, dan pengujian

Agen dan kepemilikan data

Tinjauan implementasi agen menyoroti skenario di mana agen bergantung pada data penyewa tertentu. Dalam hal ini, pertimbangkan siklus hidup data dan, yang lebih penting, tempat penyimpanannya. Hal ini sangat penting untuk industri dan kasus penggunaan di mana sifat data mempengaruhi bagaimana agen mengaksesnya.

Penyedia AaaS harus mengevaluasi cara menyelesaikan masalah data di lingkungan multi-penyewa, yang dapat memengaruhi orientasi, isolasi, dan operasi agen. Nuansa dan strategi yang berlaku bervariasi sesuai dengan alat, teknologi, dan data yang Anda konsumsi. Anda dapat mendekati ini dengan banyak cara, yang merupakan sesuatu yang harus diperhatikan saat Anda membuat penawaran AaaS apa pun.

Operasi agen multi-penyewa

Saat Anda membangun lingkungan agen, pikirkan cara mengoperasikan dan mengelola agen Anda. Sebagai penyedia, Anda memerlukan metrik, data, wawasan, dan log yang memungkinkan Anda memantau kesehatan, skala, dan aktivitas agen. Ini lebih terasa di lingkungan agen multi-penyewa di mana Anda ingin memahami bagaimana penyewa individu mengkonsumsi sumber daya agen.

Ini bahkan lebih signifikan dalam pengaturan multi-agen ketika Anda membutuhkan wawasan tentang interaksi agen. Mampu membuat profil dan melacak aktivitas antar agen mungkin penting untuk memecahkan masalah yang memengaruhi skala, akurasi, dan kemanjuran sistem Anda.

Tim operasi juga dapat membuat profil interaksi LLM untuk mendapatkan pemahaman yang lebih baik tentang beban yang ditempatkan agen. LLMs Data ini sangat penting untuk implementasi agen pemurnian. Ini juga dapat memberi tim operasional pandangan tentang bagaimana agen dan penyewaan mempengaruhi profil biaya keseluruhan suatu sistem.

Pelatihan dan pengujian agen multi-tenant

Salah satu tantangan yang terkait dengan agen bangunan adalah bahwa mereka diharapkan untuk belajar dan berkembang. Ini juga berarti bahwa kita harus menguji agen kita, menyempurnakannya, dan meningkatkan akurasinya sebelum memindahkannya ke produksi. Ada banyak area di mana

Anda dapat memeriksa dan menilai apakah agen Anda menilai dan mengkategorikan niat dengan benar atau memilih dan menggunakan alat dan tindakan yang tepat. Daftar variabel sangat luas, tetapi ini pada akhirnya tentang memastikan bahwa agen Anda menemukan hasil yang mencapai tujuan Anda.

Memeriksa semua bagian dan prinsip bergerak yang terkait dengan agen pengujian berada di luar cakupan dokumen ini tetapi perhatikan bahwa strategi pengujian menambah kompleksitas pada lingkungan AaaS multi-penyewa. Misalnya, jika agen memiliki data, memori, dan konstruksi lain yang diterapkan secara kontekstual untuk setiap penyewa, maka hasil agen dapat dibentuk oleh sumber daya per penyewa.

Jika Anda menggunakan agen untuk mensimulasikan skenario, Anda mungkin perlu memperluas simulasi Anda untuk kasus penggunaan khusus penyewa. Sejalan dengan itu, Anda harus memperbaiki prosedur validasi untuk memungkinkan contoh di mana kriteria validasi berbeda untuk setiap penyewa.

Pertimbangan dan diskusi

Di mana SaaS cocok?

Pakar industri secara aktif memperdebatkan bagaimana agen memengaruhi lanskap perangkat lunak sebagai layanan (SaaS). Meskipun benar bahwa agen mengubah perangkat lunak untuk banyak sistem, sulit untuk menyarankan agar agen membuat model pengiriman menjadi usang. Beberapa penyedia SaaS kemungkinan akan terganggu oleh agen adopsi, dan beberapa mungkin sepenuhnya memikirkan kembali proposisi nilai mereka, dengan condong ke model agen sebagai layanan (AaaS). Orang lain mungkin mencapai keseimbangan dengan memperkenalkan agen secara selektif untuk memenuhi kebutuhan spesifik.

Topik ini menarik karena mengadopsi prinsip-prinsip SaaS terbaik dapat mewakili evolusi SaaS berikutnya. Ini mungkin berarti bahwa SaaS sedang berjalan, atau mungkin berarti bahwa prinsip-prinsip dasar SaaS sedang dikemas dan direalisasikan dalam model berbasis agen. Mungkin kurang penting untuk memutuskan di mana terminologi itu akhirnya mendarat, tetapi tampaknya tidak mungkin SaaS sebagai sebuah konsep akan hilang. Kemungkinan besar agen akan membentuk jejak SaaS.

Pada akhirnya, kita harus memutuskan strategi mana yang dapat diterapkan pada AaaS, yang berarti memungkinkan organisasi untuk mengadopsi arsitektur agen dan strategi bisnis sehingga penyedia dapat memaksimalkan efisiensi, nilai, dan dampak sistem agen mereka. Agen bukan kotak hitam. Agen mengkonsumsi sumber daya, skala operasi, bergantung pada data, dan menghasilkan biaya—semua faktor yang harus ditangani penyedia. Penyedia agen harus mengevaluasi bagaimana prinsip multi-penyewa dapat membentuk penawaran layanan dan mengoptimalkan model operasional.

Diskusi

Lanskap agen terus berkembang dengan desain yang bervariasi berdasarkan domain, kasus penggunaan yang dimaksudkan, dan industri target. Bagian dari evolusi ini termasuk lebih menyempurnakan pandangan kita tentang strategi, pola, dan pengorbanan yang dipertimbangkan arsitek saat mereka merancang dan membangun agen.

Strategi agen yang komprehensif harus selaras dengan tujuan bisnis dan teknis. Ini termasuk mendefinisikan target pasar dan persona, menetapkan harga dan strategi manajemen sumber daya, dan menentukan bagaimana agen masuk ke dalam sistem yang lebih besar. Pertimbangan ini sangat penting ketika memberikan AAA, di mana skala, efisiensi biaya, dan inovasi adalah tujuan utama.

Kemampuan operasional sama pentingnya. Lingkungan harus mendukung pemantauan aktivitas agen, metrik kesehatan, dan pola penggunaan. Ini menjadi lebih kompleks dalam sistem multi-agen, di mana operasi harus dikoordinasikan di seluruh agen independen.

Secara keseluruhan, diskusi tentang agen ini hanya menggores permukaan berbagai pertimbangan arsitektur yang dapat menjadi bagian dari sistem agen. Selain memilih alat, kerangka kerja LLMs, dan kesuksesan yang tepat tergantung pada pembuatan arsitektur yang memenuhi persyaratan bisnis untuk skalabilitas, efisiensi, penerapan, dan multi-tenancy.

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 instans EC2 di. AWS Cloud
- Relokasi (hypervisor-level lift and shift) — Pindahkan infrastruktur ke cloud tanpa membeli perangkat keras baru, menulis ulang aplikasi, atau memodifikasi operasi yang ada. Anda memigrasikan server dari platform lokal ke layanan cloud untuk platform yang sama. Contoh: Migrasikan Microsoft Hyper-V aplikasi ke AWS.
- Pertahankan (kunjungi kembali) - Simpan aplikasi di lingkungan sumber Anda. Ini mungkin termasuk aplikasi yang memerlukan refactoring besar, dan Anda ingin menunda pekerjaan itu sampai nanti, dan aplikasi lama yang ingin Anda pertahankan, karena tidak ada pembenaran bisnis untuk memigrasikannya.

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

A

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 break-glass 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 Region, atau di seluruh organisasi, dengan menggunakan templat YAMM. Untuk informasi selengkapnya, lihat Paket kesesuaian dalam dokumentasi. AWS Config

integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD)

Proses mengotomatiskan sumber, membangun, menguji, pementasan, dan tahap produksi dari proses rilis perangkat lunak. CI/CD biasanya digambarkan sebagai pipa. CI/CD dapat membantu

Anda mengotomatiskan proses, meningkatkan produktivitas, meningkatkan kualitas kode, dan memberikan lebih cepat. Untuk informasi lebih lanjut, lihat Manfaat pengiriman berkelanjutan. CD juga dapat berarti penerapan berkelanjutan. Untuk informasi selengkapnya, lihat Continuous Delivery vs Continuous Deployment.

CV

Lihat visi komputer.

D

data saat istirahat

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

klasifikasi data

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

penyimpangan data

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

data dalam transit

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

jala data

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

minimalisasi data

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

perimeter data

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

prapemrosesan data

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

asal data

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

subjek data

Individu yang datanya dikumpulkan dan diproses.

gudang data

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

bahasa definisi database (DDL)

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

bahasa manipulasi basis data (DHTML)

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

DDL

Lihat bahasa definisi database.

ansambel yang dalam

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

pembelajaran mendalam

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

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 basis data.

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 dapat diakses oleh pengguna akhir. Dalam sebuah CI/CD pipeline, lingkungan produksi adalah lingkungan penyebaran terakhir.
- lingkungan atas — Semua lingkungan yang dapat diakses oleh pengguna selain tim pengembangan inti. Ini dapat mencakup lingkungan produksi, lingkungan praproduksi, dan lingkungan untuk pengujian penerimaan pengguna.

epik

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

ERP

Lihat perencanaan sumber daya perusahaan.

analisis data eksplorasi (EDA)

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

F

tabel fakta

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

gagal cepat

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

batas isolasi kesalahan

Dalam AWS Cloud, batas seperti Availability Zone, 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 dan pelajaran terbaik 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 AWS dengan Layanan Migrasi Aplikasi.

Penilaian Portofolio Migrasi (MPA)

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

Penilaian Kesiapan Migrasi (MRA)

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

strategi migrasi

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

ML

Lihat pembelajaran mesin.

modernisasi

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

penilaian kesiapan modernisasi

Evaluasi yang membantu menentukan kesiapan modernisasi aplikasi organisasi; mengidentifikasi manfaat, risiko, dan dependensi; dan menentukan seberapa baik organisasi dapat mendukung keadaan masa depan aplikasi tersebut. Hasil penilaian adalah cetak biru arsitektur target, peta

jalan yang merinci fase pengembangan dan tonggak untuk proses modernisasi, dan rencana aksi untuk mengatasi kesenjangan yang diidentifikasi. Untuk informasi lebih lanjut, lihat Mengevaluasi kesiapan modernisasi untuk aplikasi di. AWS Cloud

aplikasi monolitik (monolit)

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

MPA

Lihat Penilaian Portofolio Migrasi.

MQTT

Lihat Transportasi Telemetri Antrian Pesan.

klasifikasi multiclass

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

infrastruktur yang bisa berubah

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

O

OAC

Lihat kontrol akses asal.

OAI

Lihat identitas akses asal.

OCM

Lihat manajemen perubahan organisasi.

migrasi offline

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

OI

Lihat integrasi operasi.

OLA

Lihat perjanjian tingkat operasional.

migrasi online

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

OPC-UA

Lihat Komunikasi Proses Terbuka - Arsitektur Terpadu.

Komunikasi Proses Terbuka - Arsitektur Terpadu (OPC-UA)

Protokol komunikasi 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 kebijakan berbasis sumber daya), atau menentukan izin maksimum untuk semua akun di organisasi (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.

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 pengidentifikasi 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 prinsipal mana yang diizinkan mengakses, tindakan yang didukung, dan kondisi lain yang harus dipenuhi.

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

Matriks yang mendefinisikan peran dan tanggung jawab untuk semua pihak yang terlibat dalam kegiatan migrasi dan operasi cloud. Nama matriks berasal dari jenis tanggung jawab yang didefinisikan dalam matriks: bertanggung jawab (R), akuntabel (A), dikonsultasikan (C), dan diinformasikan (I). Tipe 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 merujuk sumber data otoritatif yang berada di luar sumber data pelatihannya sebelum menghasilkan respons. Misalnya, model RAG mungkin melakukan

pencarian semantik dari basis pengetahuan organisasi atau data kustom. Untuk informasi lebih lanjut, lihat Apa itu RAG.

rotasi

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

kontrol akses baris dan kolom (RCAC)

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

RPO

Lihat tujuan titik pemulihan.

RTO

Lihat tujuan waktu pemulihan.

buku runbook

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

D

SAML 2.0

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

SCADA

Lihat kontrol pengawasan dan akuisisi data.

SCP

Lihat kebijakan kontrol layanan.

Rahasia

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

keamanan dengan desain

Pendekatan rekayasa sistem yang memperhitungkan keamanan melalui seluruh proses pengembangan.

kontrol keamanan

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

pengerasan keamanan

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

sistem informasi keamanan dan manajemen acara (SIEM)

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

otomatisasi respons keamanan

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

enkripsi sisi server

Enkripsi data di tujuannya, oleh Layanan AWS yang menerimanya.

kebijakan kontrol layanan (SCP)

Kebijakan yang menyediakan kontrol terpusat atas izin untuk semua akun di organisasi. AWS Organizations 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 terkini dan relevan yang sering diakses. Instance database dapat membaca dari cache buffer, yang lebih cepat daripada membaca dari memori utama atau disk.

data hangat

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

fungsi jendela

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

beban kerja

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

aliran kerja

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

CACING

Lihat menulis sekali, baca banyak.

WQF

Lihat AWS Kerangka Kualifikasi Beban Kerja.

tulis sekali, baca banyak (WORM)

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

Z

eksploitasi zero-day

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

kerentanan zero-day

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

bisikan zero-shot

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

aplikasi zombie

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

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