Menguji aplikasi tanpa server pada AWS

AWS Panduan Preskriptif

AWS Panduan Preskriptif: Menguji aplikasi tanpa server pada AWS

Table of Contents

Menguji aplikasi tanpa server pada AWS

Amazon Web Services (???kontributor)

Maret 2026 (sejarah dokumen)

Panduan ini membahas metodologi untuk menguji aplikasi tanpa server, menjelaskan tantangan yang mungkin Anda temui selama pengujian, dan memperkenalkan praktik terbaik. Teknik pengujian ini dimaksudkan untuk membantu Anda mengulangi lebih cepat dan melepaskan kode Anda dengan lebih percaya diri.

Panduan ini untuk pengembang yang ingin membuat strategi pengujian untuk aplikasi tanpa server mereka. Anda dapat menggunakan panduan ini sebagai titik awal untuk mempelajari strategi pengujian, dan kemudian mengunjungi repositori Sampel Uji Tanpa Server untuk melihat contoh pengujian yang mengikuti pola dan praktik terbaik yang dijelaskan dalam panduan ini. Panduan ini menjelaskan metodologi pengujian tanpa server, menjelaskan tantangan yang dihadapi pelanggan saat menguji aplikasi tanpa server, dan memperkenalkan praktik terbaik untuk menguji aplikasi tanpa server. Teknik-teknik ini dimaksudkan untuk membantu pengembang mengulangi lebih cepat dan melepaskan lebih percaya diri.

Ikhtisar

Pengujian otomatis adalah investasi penting yang membantu memastikan kualitas aplikasi dan kecepatan pengembangan. Pengujian juga mempercepat umpan balik pengembang. Sebagai pengembang, Anda ingin dapat melakukan iterasi dengan cepat pada aplikasi Anda dan mendapatkan umpan balik tentang kualitas kode Anda. Banyak pengembang terbiasa menulis aplikasi yang mereka terapkan ke lingkungan di desktop mereka, baik langsung ke sistem operasi mereka atau dalam lingkungan berbasis kontainer. Ketika Anda bekerja di desktop atau lingkungan berbasis container, Anda biasanya menulis tes terhadap kode yang di-host sepenuhnya di desktop Anda. Namun, dalam aplikasi tanpa server, komponen arsitektur mungkin tidak dapat diterapkan ke lingkungan desktop tetapi mungkin hanya ada di cloud. Arsitektur berbasis cloud mungkin mencakup lapisan persistensi, sistem pesan, konstruksi keamanan, dan komponen lainnya APIs. Ketika Anda menulis kode aplikasi yang bergantung pada komponen-komponen ini, mungkin sulit untuk menentukan cara terbaik untuk merancang dan menjalankan tes.

Panduan ini membantu Anda menyelaraskan dengan strategi pengujian yang mengurangi gesekan dan kebingungan, serta meningkatkan kualitas kode.

Prasyarat

Panduan ini mengasumsikan bahwa Anda terbiasa dengan dasar-dasar pengujian otomatis, termasuk bagaimana pengujian perangkat lunak otomatis digunakan untuk memastikan kualitas perangkat lunak. Panduan ini memberikan pengenalan tingkat tinggi untuk strategi pengujian aplikasi tanpa server, dan tidak memerlukan pengalaman menulis tes langsung.

Definisi

Panduan ini menggunakan istilah-istilah berikut:

- Unit test adalah tes yang dijalankan terhadap kode untuk satu komponen arsitektur secara terpisah.
- Tes integrasi dijalankan terhadap dua atau lebih komponen arsitektur, biasanya di lingkungan cloud.
- End-to-end tes memverifikasi perilaku di seluruh aplikasi atau alur kerja.
- Emulator adalah aplikasi (sering disediakan oleh pihak ketiga) yang dirancang untuk meniru layanan cloud tanpa menyediakan atau menggunakan sumber daya cloud apa pun.
- Mocks (juga disebut palsu) adalah implementasi dalam aplikasi pengujian yang menggantikan ketergantungan dengan simulasi ketergantungan itu.

Tujuan

Praktik terbaik dalam panduan ini dimaksudkan untuk membantu Anda mencapai dua tujuan utama:

- Meningkatkan kualitas aplikasi tanpa server
  - Pengujian pada batas arsitektur
  - Pengujian pada batas kode
- Kurangi waktu untuk mengimplementasikan atau mengubah fitur

Meningkatkan kualitas perangkat lunak

Kualitas aplikasi sangat bergantung pada kemampuan pengembang untuk menguji berbagai skenario untuk memverifikasi fungsionalitas. Jika Anda tidak menerapkan pengujian otomatis, atau, lebih

umum, jika pengujian Anda tidak mencakup skenario yang diperlukan secara memadai, kualitas aplikasi Anda tidak dapat ditentukan atau dijamin.

Dalam arsitektur berbasis server, tim dapat dengan mudah menentukan ruang lingkup untuk pengujian: Setiap kode yang berjalan pada server aplikasi harus diuji. Komponen lain yang memanggil ke server, atau dependensi yang dipanggil server, sering dianggap eksternal dan di luar ruang lingkup untuk pengujian oleh tim yang bertanggung jawab atas aplikasi di server.

Aplikasi tanpa server sering terdiri dari unit kerja yang lebih kecil, seperti AWS Lambda fungsi, yang berjalan di lingkungan mereka sendiri. Tim kemungkinan akan bertanggung jawab atas kelipatan unit yang lebih kecil ini dalam satu aplikasi. Beberapa fungsi aplikasi dapat didelegasikan sepenuhnya ke layanan terkelola seperti Amazon Simple Storage Service (Amazon S3) atau Amazon Simple Queue Service (Amazon SQS) tanpa menggunakan kode yang dikembangkan secara internal. Model berbasis server tradisional untuk pengujian perangkat lunak mungkin mengecualikan layanan terkelola dengan mempertimbangkannya di luar aplikasi. Hal ini dapat menyebabkan cakupan yang tidak memadai, di mana skenario kritis mungkin terbatas pada pengujian eksplorasi manual atau beberapa kasus uji integrasi di mana hasilnya bervariasi menurut lingkungan. Oleh karena itu, mengadopsi strategi pengujian yang mencakup perilaku layanan terkelola dan konfigurasi cloud dapat meningkatkan kualitas perangkat lunak.

Pengujian pada batas arsitektur

Saat aplikasi tanpa server tumbuh, mereka secara alami menyebar di beberapa komponen arsitektur. Meskipun ini menggunakan kemampuan AWS terdistribusi, itu dapat membuat end-to-end perilaku sulit dipahami.

Mengidentifikasi batas-batas alam

Saat merancang arsitektur Anda mengikuti praktik terbaik tanpa server (satu fungsi = satu pekerjaan, decoupling), Anda akan melihat batasan alami di sekitar subsistem. Batas-batas ini mewakili titik pemisahan logis dalam aplikasi Anda.

Batas sebagai kontrak pengujian

Batas-batas arsitektur ini adalah kandidat yang sangat baik untuk menguji tepi. Perlakukan setiap batas sebagai kontrak dan validasi bahwa ia berperilaku sesuai dengan spesifikasi yang ditentukan. Pikirkan batas-batas ini sebagai jahitan dalam aplikasi Anda di mana Anda dapat memasukkan validasi pengujian.

Manfaat utama

Berikut ini adalah manfaat utama pengujian pada batas arsitektur:

- Lingkup pengujian terfokus - Uji subsistem secara independen tanpa perlu memahami seluruh aplikasi.
- Validasi kontrak — Pastikan setiap batas mempertahankan perilaku yang diharapkan saat sistem berkembang
- Instrumentasi tujuan ganda - Batasan yang sama ini membuat kait observabilitas yang sangat baik dalam produksi
- Test harness - Memungkinkan Anda menguji sistem tanpa server asinkron. Ini membantu Anda menguji arsitektur berbasis peristiwa dengan menangkap dan memvalidasi peristiwa saat mereka mengalir melalui subsistem Anda.

Pengujian pada batas kode

Tentukan batas kode yang jelas dengan memisahkan kode infrastruktur, seperti kode Lambda, dari logika bisnis inti Anda. Pemisahan ini menciptakan cakupan pengujian berbeda yang menyederhanakan strategi pengujian Anda.

Pola batas

Tetapkan dua batas kode yang jelas dalam fungsi Lambda Anda:

- Batas luar (Lambda handler) - Lapisan adaptor ramping yang menangani masalah khusus AWS Lambda
- Batas dalam (logika bisnis) - Metode logika bisnis murni yang independen dari runtime Lambda

Handler sebagai adaptor (lingkup luar)

Handler fungsi Lambda Anda harus berupa lapisan tipis yang:

- Mengekstrak data dari yang masuk `event` dan objek `context`
- Memvalidasi data yang diekstraksi
- Hanya meneruskan detail yang relevan ke metode logika bisnis
- Mengembalikan hasil dalam format yang diharapkan untuk Lambda

Logika bisnis (lingkup dalam)

Logika bisnis inti Anda harus:

- Beroperasi secara independen dari detail khusus untuk Lambda
- Terima input sederhana dan tervalidasi
- Kembalikan output yang dapat diprediksi
- Memerlukan dependensi minimal untuk inisialisasi

Menguji manfaat berdasarkan ruang lingkup

- Tes batas dalam — Pengujian unit komprehensif seputar logika bisnis tanpa kompleksitas Lambda atau pengaturan lingkungan
- Tes batas luar - Tes integrasi terfokus yang memvalidasi penanganan peristiwa dan ekstraksi data lapisan adaptor
- Overhead pengujian minimal - Tidak ada lingkungan yang kompleks atau dependensi ekstensif yang diperlukan untuk sebagian besar pengujian Anda

Pendekatan berbasis batas ini memungkinkan Anda untuk menguji sebagian besar kode Anda sebagai fungsi murni sambil menjaga pengujian Lambda tetap minimal dan ditargetkan.

Kurangi waktu untuk mengimplementasikan atau mengubah fitur

Anda dapat meminimalkan efek bug perangkat lunak dan masalah konfigurasi pada biaya dan jadwal dengan menangkap masalah ini selama siklus pengembangan berulang. Ketika pengembang gagal mendeteksi masalah ini, lebih banyak orang harus menginvestasikan upaya tambahan untuk mengidentifikasi masalah.

Arsitektur tanpa server mungkin mencakup layanan terkelola yang menyediakan fungsionalitas aplikasi penting melalui panggilan API. Untuk alasan ini, siklus pengembangan Anda harus mencakup pengujian yang memvalidasi jalur bahagia (di mana interaksi dengan layanan ini berperilaku seperti yang diharapkan) dan jalur sedih (di mana panggilan gagal, mengembalikan respons yang tidak terduga, atau berperilaku berbeda di seluruh lingkungan). Tanpa pengujian ini, Anda mungkin mengalami masalah yang berasal dari perbedaan antara lingkungan lokal Anda dan lingkungan yang diterapkan. Ketika itu terjadi, Anda harus meluangkan waktu tambahan untuk mencoba mereproduksi dan memverifikasi perbaikan, karena setiap iterasi sekarang memerlukan validasi perubahan terhadap lingkungan yang berbeda dari pengaturan pilihan Anda.

Strategi pengujian tanpa server yang tepat meningkatkan waktu iterasi Anda dengan memberikan hasil yang akurat untuk pengujian yang menyertakan panggilan ke layanan lain.

Teknik pengujian untuk aplikasi tanpa server

Ada tiga pendekatan utama untuk menjalankan pengujian terhadap kode aplikasi tanpa server: pengujian tiruan, pengujian emulasi, dan pengujian di cloud. Anda biasanya dapat menemukan campuran dari pendekatan ini yang digunakan dalam lingkup proyek tunggal. Misalnya, Anda dapat menggunakan pengujian tiruan saat mengembangkan kode secara lokal, pengujian emulasi untuk mereplikasi perilaku layanan lebih dekat sebelum penerapan, dan pengujian cloud sebagai bagian dari proses integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD) malam hari.

Pengujian tiruan

Pengujian tiruan adalah strategi di mana Anda membuat objek pengganti dalam kode Anda yang mensimulasikan perilaku layanan cloud. Misalnya, Anda dapat menulis pengujian yang menggunakan tiruan layanan Amazon S3, dan Anda dapat mengonfigurasi tiruan tersebut untuk mengembalikan objek respons tertentu setiap kali metode dipanggil`PutObject`. Saat pengujian berjalan, tiruan mengembalikan respons tanpa memanggil Amazon S3 atau titik akhir layanan lainnya.

Objek pengganti ini sering dihasilkan oleh kerangka kerja tiruan untuk mengurangi jumlah implementasi yang diperlukan untuk pengujian yang diberikan. Beberapa kerangka kerja tiruan bersifat generik dan yang lainnya dirancang khusus untuk. AWS SDKs

Mocks, bersama dengan rintisan, termasuk dalam kategori palsu yang lebih luas. Mocks berbeda dari emulator (dibahas nanti di bagian ini) karena tiruan biasanya dibuat atau dikonfigurasi oleh pengembang sebagai bagian dari kode pengujian, sedangkan emulator adalah aplikasi mandiri yang mengekspos APIs (seperti REST APIs) dengan cara yang sama seperti sistem yang mereka tiru.

Keuntungan menggunakan tiruan antara lain sebagai berikut:

- Mocks dapat mensimulasikan layanan pihak ketiga yang berada di luar kendali aplikasi Anda, seperti APIs dan penyedia perangkat lunak sebagai layanan (SaaS), tanpa memerlukan akses langsung ke layanan tersebut.
- Mocks juga berguna untuk menguji kondisi kegagalan, terutama ketika kondisi seperti itu (seperti pemadaman layanan) sulit untuk disimulasikan.
- Seperti emulator, kerangka kerja tiruan dapat menyediakan pengembangan lokal yang cepat setelah dikonfigurasi.
- Mocks dapat memberikan perilaku pengganti untuk hampir semua jenis objek, sehingga strategi mengejek dapat menciptakan cakupan untuk berbagai layanan yang lebih luas daripada emulator.

- Ketika fitur atau perilaku baru tersedia, pengujian tiruan dapat bereaksi lebih cepat dengan menggunakan (atau kembali ke) kerangka kerja tiruan generik, yang dapat mensimulasikan fitur baru segera setelah SDK yang diperbarui AWS tersedia.

Pengujian tiruan memiliki kelemahan ini:

- Mocks umumnya memerlukan upaya pengaturan dan konfigurasi yang tidak sepele, khususnya ketika mencoba menentukan nilai pengembalian dari layanan yang berbeda untuk mengejek respons dengan benar.
- Karena tiruan ditulis atau dikonfigurasi oleh pengembang yang menulis tes, ini merupakan tanggung jawab tambahan bagi pengembang.
- Anda mungkin perlu memiliki akses ke cloud untuk memahami APIs dan mengembalikan nilai layanan.
- Mocks juga bisa sulit dipertahankan, karena memerlukan pembaruan setiap kali tanda tangan API cloud yang diejek berubah, skema nilai pengembalian berkembang, atau logika aplikasi yang diuji diperluas untuk melakukan panggilan ke yang baru. APIs Perubahan ini menciptakan beban kerja pengembangan pengujian tambahan untuk pengembang.
- Tes tiruan mungkin lulus secara lokal tetapi gagal di cloud karena mensimulasikan — bukan mereplikasi — perilaku layanan nyata, sehingga masalah khusus lingkungan tidak terdeteksi.
- Kerangka kerja tiruan, seperti emulator, tidak dapat mendeteksi pelanggaran kebijakan AWS Identity and Access Management (IAM), batas kuota layanan, atau batasan ukuran muatan, juga tidak memicu layanan tambahan seperti Amazon CloudWatch, AWS CloudTrail atau Amazon GuardDuty yang dapat memengaruhi perilaku aplikasi di lingkungan yang diterapkan.
- Meskipun tiruan lebih baik dalam mensimulasikan apa yang akan dilakukan aplikasi ketika otorisasi gagal atau kuota terlampaui, pengujian tiruan tidak dapat menentukan hasil mana yang benar-benar akan terjadi ketika kode diterapkan ke lingkungan produksi.

Pengujian emulasi

Pengujian emulasi diaktifkan oleh aplikasi yang berjalan secara lokal yang dikenal sebagai emulator yang menyerupai. Layanan AWS Emulator memiliki APIs yang mirip dengan rekan cloud mereka dan memberikan nilai pengembalian yang serupa. Mereka juga dapat mensimulasikan perubahan status yang diprakarsai oleh panggilan API. Misalnya, emulator Amazon S3 mungkin menyimpan objek pada disk lokal saat `PutObject` metode dipanggil. Ketika `GetObject` dipanggil dengan kunci yang sama, emulator membaca objek yang sama dari disk dan mengembalikannya.

Keuntungan dari pengujian emulasi meliputi:

- Ini menyediakan lingkungan yang paling akrab bagi pengembang yang terbiasa mengembangkan kode secara eksklusif di lingkungan lokal. Misalnya, jika Anda terbiasa dengan pengembangan aplikasi n-tier, Anda mungkin memiliki mesin database dan server web, mirip dengan yang berjalan dalam produksi, berjalan di mesin lokal Anda untuk menyediakan kemampuan pengujian cepat, lokal, dan terisolasi.
- Ini tidak memerlukan perubahan apa pun pada infrastruktur cloud (seperti akun cloud pengembang), sehingga mudah diterapkan dengan pola pengujian yang ada. Pengujian emulasi memberikan manfaat dari iterasi pengembangan lokal yang cepat.

Namun, emulator memiliki beberapa kelemahan:

- Mereka sering sulit untuk mengatur dan mereplikasi, terutama ketika mereka digunakan sebagai bagian dari CI/CD jaringan pipa. Ini dapat meningkatkan beban kerja dan pemeliharaan untuk staf TI atau untuk pengembang yang mengelola instalasi perangkat lunak mereka sendiri.
- Fitur yang ditiru dan APIs biasanya tertinggal dari perubahan layanan dan mungkin menghambat adopsi fitur baru sampai dukungan ditambahkan.
- Emulator mungkin memerlukan dukungan, pembaruan, perbaikan bug, atau peningkatan paritas fitur, yang merupakan tanggung jawab pembuat emulator (yang seringkali merupakan perusahaan pihak ketiga).
- Pengujian yang mengandalkan emulator dapat memberikan hasil yang sukses secara lokal, tetapi mungkin gagal di cloud karena interaksi dengan aspek lain AWS seperti kebijakan dan kuota IAM, yang umumnya tidak ditiru.
- Beberapa Layanan AWS tidak memiliki emulator yang tersedia, jadi jika Anda hanya mengandalkan emulasi, Anda mungkin tidak memiliki opsi pengujian yang memuaskan untuk bagian dari aplikasi Anda.

Pengujian di cloud

Pengujian di cloud adalah proses menjalankan pengujian terhadap kode yang diterapkan ke lingkungan cloud alih-alih lingkungan desktop. Nilai pengujian di cloud meningkat dengan pengembangan aplikasi cloud-native. Contoh:

- Anda dapat menjalankan pengujian di cloud terhadap fitur terbaru APIs dan layanan, memastikan pengujian Anda mencerminkan nilai dan perilaku pengembalian terbaru yang AWS terus meluncurkan layanan dan kemampuan baru.
- Pengujian Anda dapat mencakup kebijakan IAM, kuota layanan, konfigurasi, dan semua layanan.
- Lingkungan pengujian cloud Anda paling mirip dengan lingkungan runtime produksi Anda, jadi pengujian yang Anda jalankan di cloud kemungkinan akan mencapai hasil yang paling konsisten saat mereka berkembang melalui lingkungan.

Kelemahan pengujian di cloud meliputi yang berikut:

- Penerapan ke lingkungan cloud biasanya membutuhkan waktu lebih lama daripada penerapan ke lingkungan desktop. Alat seperti AWS Serverless Application Model (AWS SAM) Accelerate, AWS Cloud Development Kit (AWS CDK) watch mode, dan SST membantu mengurangi latensi yang terlibat dengan iterasi penerapan cloud.
- Pengujian cloud dapat menimbulkan biaya layanan tambahan, tidak seperti pengujian lokal, yang menggunakan lingkungan pengembangan lokal Anda.
- Pengujian di cloud mungkin kurang layak jika Anda tidak memiliki akses internet berkecepatan tinggi.
- Dalam industri yang diatur, kebijakan keamanan perusahaan dapat membatasi akses pengembang ke lingkungan cloud, sehingga sulit atau tidak mungkin untuk menjalankan pengujian cloud sebagai bagian dari alur kerja pengembangan lokal.
- Batas lingkungan sering ditarik pada tingkat tumpukan di akun bersama untuk lingkungan pengembang, terkadang dengan menggunakan strategi tipe namespace seperti menggunakan awalan untuk mengidentifikasi kepemilikan. Untuk lingkungan pra-produksi dan produksi, batasan biasanya ditarik pada tingkat akun untuk melindungi beban kerja dari masalah tetangga yang bising, untuk mendukung kontrol keamanan yang paling tidak istimewa, dan untuk melindungi data sensitif. Persyaratan untuk menciptakan lingkungan yang terisolasi dapat menempatkan beban tambahan pada DevOps tim, terutama jika mereka berada di perusahaan yang memiliki kontrol ketat seputar akun dan infrastruktur.

Tantangan saat menguji aplikasi tanpa server

Saat Anda menggunakan emulator dan panggilan tiruan untuk menguji aplikasi tanpa server di desktop lokal, Anda mungkin mengalami inkonsistensi pengujian saat kode Anda berkembang dari lingkungan ke lingkungan di pipeline Anda. CI/CD Pengujian unit yang Anda buat di desktop untuk memvalidasi logika bisnis aplikasi Anda mungkin tidak menyertakan atau secara akurat mewakili aspek penting dari layanan cloud. Tes lengkap tidak dapat dilakukan secara lokal dalam isolasi. Mereka memerlukan verifikasi izin dan konfigurasi di antara beberapa layanan.

Bagian berikut menguraikan tantangan yang mungkin Anda alami saat menerapkan strategi pengujian cloud. Bagian berikut menguraikan tantangan yang dialami pelanggan ketika mencoba menerapkan strategi pengujian cloud, dan panduan kami tentang praktik terbaik untuk mencapai cakupan pengujian yang efektif.

Contoh: Fungsi Lambda yang membuat bucket Amazon S3

Jika logika fungsi Lambda bergantung pada pembuatan bucket Amazon S3, pengujian lengkap akan mengonfirmasi bahwa Amazon S3 berhasil dipanggil dan bucket berhasil dibuat. Dalam pengaturan pengujian tiruan, Anda mungkin mengejek respons sukses dan berpotensi menambahkan kasus uji untuk menangani respons kegagalan. Dalam skenario pengujian emulasi, `CreateBucket` API mungkin dipanggil, tetapi identitas yang membuat panggilan tidak akan berasal dari layanan Lambda dengan asumsi peran, dan otentikasi placeholder akan digunakan sebagai gantinya – ini sering kali peran atau identitas pengguna Anda yang lebih permisif.

Pengaturan tiruan dan emulasi yang dibahas sebelumnya kemungkinan besar akan menguji apa yang akan dilakukan fungsi Lambda jika berhasil (atau tidak berhasil) memanggil Amazon S3. Namun, pengujian tersebut akan gagal untuk menangkap apakah fungsi Lambda mampu berhasil membuat bucket, mengingat konfigurasi fungsi. Konfigurasi ini kemungkinan diwakili oleh infrastruktur sebagai kode (IAc) untuk produk dan layanan seperti AWS CloudFormation, AWS SAM, atau HashiCorp Terraform. Satu masalah yang mungkin terjadi adalah bahwa peran yang ditetapkan ke fungsi tidak memiliki kebijakan terlampir yang memungkinkan `s3:CreateBucket` tindakan tersebut, dan oleh karena itu fungsi tersebut akan selalu gagal saat diterapkan ke lingkungan cloud.

Contoh: Fungsi Lambda yang memproses pesan dari Amazon SQS

Jika antrean Amazon Simple Queue Service (Amazon SQS) adalah sumber fungsi Lambda, pengujian lengkap harus memverifikasi bahwa fungsi Lambda berhasil dipanggil saat pesan

dimasukkan dalam antrean. Pengujian emulasi dan pengujian tiruan umumnya diatur untuk menjalankan kode fungsi Lambda secara langsung, dan untuk mensimulasikan integrasi Amazon SQS dengan meneruskan muatan peristiwa JSON (atau objek deserialisasi) sebagai input penangan fungsi.

Pengujian lokal yang mensimulasikan integrasi Amazon SQS akan menguji apa yang akan dilakukan fungsi Lambda saat dipanggil oleh Amazon SQS dengan muatan tertentu, tetapi tidak akan memastikan bahwa Amazon SQS akan berhasil menjalankan fungsi Lambda saat dikerahkan ke lingkungan cloud.

Beberapa contoh masalah konfigurasi yang mungkin Anda temui dengan Amazon SQS dan Lambda termasuk yang berikut:

- Batas waktu visibilitas Amazon SQS terlalu rendah, menghasilkan beberapa pemanggilan ketika hanya satu yang dimaksudkan.
- Peran eksekusi fungsi Lambda tidak mengizinkan membaca pesan dari antrian (melalui`sqs:ReceiveMessage`,`sqs:DeleteMessage`, atau). `sqs:GetQueueAttributes`
- Contoh peristiwa yang diteruskan ke fungsi Lambda melebihi kuota ukuran pesan Amazon SQS. Oleh karena itu, pengujian tidak valid karena Amazon SQS tidak akan pernah dapat mengirim pesan sebesar itu.

Seperti yang ditunjukkan oleh contoh-contoh ini, tes yang mencakup logika bisnis tetapi bukan konfigurasi antara layanan cloud cenderung memberikan hasil yang tidak dapat diandalkan.

Praktik terbaik untuk menguji aplikasi tanpa server

Bagian berikut menguraikan praktik terbaik untuk mencapai cakupan yang efektif saat menguji aplikasi tanpa server.

Prioritaskan pengujian di cloud

Untuk aplikasi yang dirancang dengan baik, Anda dapat menggunakan berbagai teknik pengujian untuk memenuhi berbagai persyaratan dan kondisi. Namun, berdasarkan perkakas saat ini, kami menyarankan Anda untuk fokus pada pengujian di cloud sebanyak mungkin. Meskipun pengujian di cloud dapat menciptakan latensi pengembang, meningkatkan biaya, dan terkadang memerlukan investasi dalam DevOps kontrol tambahan, teknik ini memberikan cakupan pengujian yang paling andal, akurat, dan lengkap.

Anda harus memiliki akses ke lingkungan terisolasi untuk melakukan pengujian. Idealnya, setiap pengembang harus memiliki dedicated Akun AWS untuk menghindari masalah dengan penamaan sumber daya yang dapat terjadi ketika beberapa pengembang yang bekerja dalam kode yang sama mencoba menerapkan atau memanggil panggilan API pada sumber daya yang memiliki nama yang identik. Lingkungan ini harus dikonfigurasi dengan peringatan dan kontrol yang sesuai untuk menghindari pengeluaran yang tidak perlu. Misalnya, Anda dapat membatasi jenis, tingkat, atau ukuran sumber daya yang dapat dibuat, dan mengatur peringatan email ketika perkiraan biaya melebihi ambang batas tertentu.

Jika Anda harus berbagi satu Akun AWS dengan pengembang lain, proses pengujian otomatis harus memberi nama sumber daya yang unik untuk setiap pengembang. Misalnya, memperbarui skrip atau file konfigurasi TOMB yang menyebabkan perintah AWS SAM CLI sam deploy atau sam sync dapat secara otomatis menentukan nama tumpukan yang menyertakan nama pengguna pengembang lokal.

Pengujian di cloud sangat berharga untuk semua fase pengujian, termasuk pengujian unit, pengujian integrasi, dan end-to-end pengujian.

Gunakan tiruan jika perlu

Kerangka kerja tiruan adalah alat yang berharga untuk menulis tes unit cepat. Mereka sangat berharga ketika tes perlu mencakup logika bisnis internal yang kompleks, seperti perhitungan atau simulasi matematika atau keuangan. Cari pengujian unit yang memiliki sejumlah besar kasus uji atau

variasi input, di mana input tidak mengubah pola atau konten panggilan ke layanan cloud lainnya. Membuat tes tiruan untuk skenario ini dapat meningkatkan waktu iterasi pengembang.

Kode yang dicakup oleh pengujian unit yang menggunakan pengujian tiruan juga harus dicakup oleh pengujian di cloud. Ini diperlukan karena tiruan masih berjalan di laptop pengembang atau mesin build, dan lingkungan mungkin dikonfigurasi secara berbeda dari yang ada di cloud. Misalnya, kode Anda mungkin menyertakan AWS Lambda fungsi yang menggunakan lebih banyak memori atau membutuhkan waktu lebih lama daripada yang dikonfigurasi Lambda untuk dialokasikan saat dijalankan dengan parameter input tertentu. Atau kode Anda mungkin menyertakan variabel lingkungan yang tidak dikonfigurasi dengan cara yang sama (atau sama sekali), dan perbedaannya dapat menyebabkan kode berperilaku berbeda atau gagal.

Jangan gunakan tiruan layanan cloud untuk memvalidasi implementasi yang tepat dari integrasi layanan tersebut. Meskipun mungkin dapat diterima untuk mengejek layanan cloud saat Anda menguji fungsionalitas lain, Anda harus menguji panggilan layanan cloud di cloud untuk memvalidasi konfigurasi dan implementasi fungsional yang benar.

Mocks dapat menambah nilai pada pengujian unit, terutama ketika Anda sering menguji sejumlah besar kasus. Manfaat ini berkurang untuk tes integrasi, karena tingkat upaya untuk mengimplementasikan tiruan yang diperlukan meningkat dengan jumlah titik koneksi. End-to-endpengujian tidak boleh menggunakan tiruan, karena pengujian ini umumnya berurusan dengan status dan logika kompleks yang tidak dapat dengan mudah disimulasikan dengan kerangka kerja tiruan.

Memahami pengorbanan pengujian emulasi

Emulator dapat menjadi pilihan praktis untuk kasus penggunaan tertentu. Misalnya, tim pengembangan dengan akses internet yang terbatas, tidak konsisten, atau lambat mungkin menemukan bahwa pengujian emulasi adalah cara yang paling dapat diandalkan untuk mengulangi kode sebelum pindah ke lingkungan cloud.

Untuk sebagian besar keadaan lain, gunakan emulator secara selektif. Ketika Anda sangat bergantung pada emulator, akan sulit untuk memasukkan fitur AWS layanan baru ke dalam pengujian Anda sampai vendor emulasi merilis pembaruan untuk memberikan paritas fitur. Emulator juga memerlukan investasi awal dan berkelanjutan untuk pengaturan dan konfigurasi di seluruh sistem pengembangan dan membangun mesin. Selain itu, banyak layanan cloud tidak memiliki emulator yang tersedia; memilih strategi emulasi-pertama dapat menghalangi penggunaan layanan tersebut atau menghasilkan kode dan konfigurasi yang tidak diuji dengan baik terhadap perilaku layanan nyata.

Jika Anda menggunakan pengujian emulasi, lengkapi dengan pengujian cloud sebanyak mungkin untuk memvalidasi bahwa konfigurasi cloud yang tepat sudah ada dan untuk menguji interaksi dengan layanan yang hanya dapat disimulasikan atau diejek di lingkungan yang ditiru.

Pengujian emulasi dapat memberikan umpan balik cepat untuk pengujian unit dan, tergantung pada fitur dan paritas perilaku perangkat lunak emulasi, dapat mendukung beberapa integrasi dan end-to-end pengujian juga.

Tes lingkup melalui batas-batas alam

Ketika aplikasi tanpa server tumbuh di lebih banyak komponen arsitektur, batas-batas alami muncul di sekitar subsistem—terutama ketika mengikuti praktik terbaik seperti fungsi tujuan tunggal dan decoupling berbasis peristiwa. Batas-batas ini berfungsi sebagai tepi pengujian yang efektif di mana Anda dapat memvalidasi kontrak antar komponen.

Identifikasi batas-batas arsitektur

Cari jahitan alami dalam desain aplikasi Anda:

- Antara layanan, seperti EventBridge aturan Amazon yang menghubungkan penerbit ke konsumen
- Di tepi API, seperti titik akhir Amazon API Gateway yang mengedepankan fungsi Lambda
- Sekitar alur kerja, seperti AWS Step Functions mengatur beberapa layanan
- Pada lapisan penyimpanan, seperti aliran Amazon DynamoDB yang memicu pemrosesan hilir

Pisahkan kode Lambda dari logika bisnis

Sederhanakan pengujian Anda dengan mengisolasi kode Lambda dari logika bisnis inti. Handler Lambda Anda harus bertindak sebagai adaptor tipis antara AWS runtime dan logika aplikasi Anda. Itu harus mengekstrak dan memvalidasi data peristiwa dan kemudian mendelegasikan ke fungsi yang dapat diuji yang tidak memiliki dependensi Lambda. Ini membuat logika bisnis Anda portabel, lebih mudah dipikirkan, dan mudah diuji tanpa mengejek objek Lambda atau menyiapkan lingkungan yang kompleks.

Perlakukan batas sebagai kontrak

Uji di batas, bukan melalui itu. Validasi apa yang melintasi tepi tanpa memerlukan seluruh sistem hilir. Batas-batas yang sama ini juga berfungsi sebagai kait observabilitas dalam produksi. Lapisan

arsitektur tempat Anda menguji dapat diinstrumentasi untuk pemantauan menggunakan Amazon CloudWatch Log, AWS X-Ray jejak, dan EventBridge peristiwa.

Gunakan test harness untuk alur kerja asinkron

Aplikasi tanpa server sering mengandalkan pola asinkron, di mana peristiwa memicu pemrosesan, pesan mengalir melalui antrian, dan alur kerja menjangkau beberapa layanan tanpa tanggapan langsung. Anda tidak bisa begitu saja memanggil fungsi dan memeriksa nilai yang dikembalikan. Hasilnya mungkin muncul kemudian dalam database, aliran log, atau layanan lain.

Test harness sedang menguji infrastruktur yang Anda terapkan bersama aplikasi Anda untuk mengamati dan memvalidasi perilaku asinkron ini. Test harness biasanya meliputi:

- Pendengar acara yang berlangganan acara yang sama yang dihasilkan aplikasi Anda
- Mekanisme penyimpanan (seperti tabel DynamoDB atau bucket Amazon S3) di mana hasil pengujian dapat ditangkap
- Logika polling dalam kode pengujian Anda yang menunggu hasil yang diharapkan muncul

Kode pengujian Anda memulai suatu peristiwa, menunggu alur kerja selesai, lalu menanyakan harness pengujian untuk memverifikasi hasil yang diharapkan terjadi.

Berikut ini adalah praktik terbaik:

- Tentukan jelas SLAs untuk operasi asinkron — Tetapkan berapa lama alur kerja harus berlangsung dan gunakan ini sebagai batas waktu polling dalam pengujian Anda
- Gunakan pengidentifikasi unik untuk isolasi pengujian — Hasilkan nama file, pesan IDs, atau token korelasi unik per pengujian yang dijalankan untuk mencegah interferensi antar pengujian
- Terapkan infrastruktur pengujian di samping aplikasi Anda — Sertakan sumber daya test harness di infrastructure-as-code template Anda agar tetap sinkron saat aplikasi Anda berkembang
- Bersihkan data pengujian setelah pengujian dijalankan — Ini mencegah akumulasi artefak pengujian di lingkungan cloud Anda

Test harness paling berharga untuk pengujian integrasi yang memvalidasi alur kerja di beberapa layanan, end-to-end pengujian yang memverifikasi perjalanan pengguna yang lengkap, dan arsitektur berbasis peristiwa tempat layanan berkomunikasi melalui, Amazon SNS, Amazon SQS, atau EventBridge Amazon Kinesis.

Mengatur lingkungan cloud untuk isolasi pengembang

Pengujian di cloud membutuhkan lingkungan yang terisolasi satu sama lain. Ketika pengembang berbagi satu Akun AWS, seperti akun pengembangan tim, pertimbangkan untuk membuat tumpukan aplikasi terpisah untuk setiap pengembang atau cabang fitur. Ini mengisolasi sumber daya, mencegah tabrakan penamaan, dan menghindari pertengkaran kuota atau masalah tetangga yang bising selama pengujian.

Gunakan AWS Systems Manager Parameter Store atau perkakas serupa untuk mengelola konfigurasi khusus tumpukan, seperti titik akhir API dan nama antrian. Untuk efisiensi biaya, bagikan sumber daya yang mahal seperti kluster Amazon Relational Database Service (Amazon RDS) di seluruh tumpukan pengembang sambil menjaga sumber daya tanpa server yang ringan (seperti fungsi Lambda, tahapan API Gateway, dan tabel DynamoDB) terisolasi per tumpukan.

Dalam industri yang diatur, kebijakan keamanan perusahaan dapat membatasi akses pengembang ke lingkungan cloud, sehingga sulit untuk menjalankan pengujian cloud sebagai bagian dari alur kerja pengembangan lokal. Dalam kasus ini, pengujian emulasi dapat mengisi kesenjangan antara pengujian tiruan lokal dan validasi cloud penuh, meskipun harus dilengkapi dengan pengujian cloud kapan pun akses memungkinkan.

Mempercepat loop umpan balik

Saat Anda menguji di cloud, gunakan alat dan teknik untuk mempercepat loop umpan balik pengembangan. Misalnya, gunakan mode AWS SAM Accelerate dan AWS CDK watch untuk mengurangi waktu yang diperlukan untuk mendorong modifikasi kode ke lingkungan cloud. Sampel dalam repositori Sampel Uji GitHub Tanpa Server mengeksplorasi beberapa teknik ini.

Kami juga menyarankan Anda membuat dan menguji sumber daya cloud dari komputer lokal Anda sedini mungkin selama pengembangan – tidak hanya setelah check-in ke kontrol sumber. Praktik ini memungkinkan eksplorasi dan eksperimen yang lebih cepat saat mengembangkan solusi. Selain itu, kemampuan untuk mengotomatiskan penerapan dari mesin pengembangan membantu Anda menemukan masalah konfigurasi cloud lebih cepat dan mengurangi upaya yang sia-sia dari memperbarui dan menyetujui modifikasi ke kontrol sumber.

Pertanyaan yang Sering Diajukan

Saya memiliki fungsi Lambda yang melakukan perhitungan dan mengembalikan hasil tanpa memanggil layanan lain. Apakah saya benar-benar perlu menguji ini di cloud?

Ya. AWS Lambda fungsi memiliki parameter konfigurasi yang dapat mengubah hasil tes. Semua kode fungsi Lambda memiliki ketergantungan pada pengaturan batas waktu dan memori, yang dapat menyebabkan fungsi gagal jika tidak disetel dengan benar. Kebijakan Lambda juga memungkinkan pencatatan keluaran standar ke Amazon. CloudWatch Bahkan jika kode Anda tidak menelepon CloudWatch secara langsung, izin diperlukan untuk mengaktifkan logging, dan izin itu tidak dapat diejek atau ditiru secara akurat.

Bagaimana pengujian di cloud dapat membantu pengujian unit? Jika ada di cloud dan terhubung ke sumber daya lain, bukankah itu tes integrasi?

Kami mendefinisikan pengujian unit sebagai tes yang beroperasi pada komponen arsitektur secara terpisah. Definisi ini tidak selalu menghalangi penggunaan panggilan layanan atau komunikasi jaringan lainnya.

Banyak aplikasi tanpa server memang memiliki komponen arsitektur yang dapat diuji secara terpisah, bahkan di cloud. Contoh dasarnya adalah fungsi Lambda yang mengambil beberapa input, menafsirkannya, dan mengirim pesan ke antrian Amazon Simple Queue Service (Amazon SQS). Tes unit dari fungsi semacam itu kemungkinan akan menguji apakah nilai input menghasilkan nilai tertentu yang ada dalam pesan antrian. Pertimbangkan tes yang ditulis dengan menggunakan pola mengatur, bertindak, menegaskan:

- Atur - Alokasikan sumber daya (antrian untuk menerima pesan, dan fungsi yang sedang diuji).
- Act - Panggil fungsi yang sedang diuji.
- Tegaskan - Ambil pesan yang dikirim oleh fungsi, dan validasi output.

Pendekatan pengujian tiruan akan melibatkan mengejek antrian dengan objek tiruan dalam proses, dan membuat instance dalam proses dari kelas atau modul yang berisi kode fungsi Lambda. Selama fase assert, pesan antrian akan diambil dari objek yang diejek.

Dalam pendekatan berbasis cloud, pengujian akan membuat antrean Amazon SQS untuk keperluan pengujian, dan akan menerapkan fungsi Lambda dengan variabel lingkungan yang dikonfigurasi untuk menggunakan antrian Amazon SQS yang terisolasi sebagai tujuan keluaran. Setelah menjalankan fungsi Lambda, pengujian akan mengambil pesan dari antrian Amazon SQS.

Pengujian berbasis cloud akan menjalankan kode yang sama, menegaskan perilaku yang sama, dan memvalidasi kebenaran fungsional aplikasi. Namun, itu akan memiliki keuntungan tambahan karena dapat memvalidasi pengaturan fungsi Lambda berikut: AWS Identity and Access Management peran (IAM), kebijakan IAM, dan batas waktu dan pengaturan memori fungsi.

Langkah dan sumber daya selanjutnya

Gunakan sumber daya berikut untuk bacaan lebih lanjut dan contoh konkret tambahan.

Implementasi sampel

Repositori Sampel Uji Tanpa Server pada GitHub berisi contoh nyata pengujian yang mengikuti pola dan praktik terbaik yang dijelaskan dalam panduan ini. Repositori berisi kode sampel dan panduan panduan dari proses pengujian tiruan, emulasi, dan cloud yang dijelaskan di bagian sebelumnya. Gunakan repositori ini untuk mempercepat panduan pengujian tanpa server terbaru dari. AWS

Sumber bacaan lebih lanjut

Kunjungi Serverless Land untuk mengakses blog, video, dan pelatihan terbaru untuk teknologi tanpa AWS server.

Referensi

- Mempercepat pengembangan tanpa server dengan AWS SAM Accelerate (posting blog)AWS
- Meningkatkan kecepatan pengembangan dengan CDK Watch (posting AWS blog)
- Integrasi layanan mengejek dengan AWS Step Functions Lokal (AWS posting blog)
- Memulai dengan menguji aplikasi tanpa server (AWS posting blog)
- Mempercepat pengujian tanpa server dengan LocalStack integrasi di VS Code IDE (AWS posting blog)
- Fungsi debug lokal dengan AWS SAM(dokumentasi)AWS

Alat

- AWS SAM — Menguji dan men-debug aplikasi tanpa server
- AWS SAM — Mengintegrasikan dengan tes otomatis
- AWS Lambda — Menguji fungsi Lambda di konsol
- LocalStack Cloud Emulator — Tingkatkan pengalaman pengujian lokal untuk aplikasi tanpa server LocalStack

Kontributor

Mengotorisasi

- Dan Rubah, AWS
- Leslie Raj, AWS
- Rohan Mehta, AWS
- Bukit Rob, AWS

Meninjau

- Brian Krygsman, AWS

Penulisan teknis

- Lilly, AbouHarb AWS

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
Update	Kami menambahkan panduan tentang pengujian pada batas arsitektur dan kode, memanfaatkan pengujian untuk alur kerja asinkron, dan isolasi lingkungan pengembang. Kami juga memperbarui rekomendasi pengujian emulasi.	Maret 18, 2026
Publikasi awal	—	Desember 9, 2022

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

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

A

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 pemrosesan atau modifikasi 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 Wilayah, atau di seluruh organisasi, dengan menggunakan templat YAMM. Untuk informasi selengkapnya, lihat Paket kesesuaian dalam dokumentasi. AWS Config

integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD)

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

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

CV

Lihat visi komputer.

D

data saat istirahat

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

klasifikasi data

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

penyimpangan data

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

data dalam transit

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

jala data

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

minimalisasi data

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

perimeter data

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

prapemrosesan data

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

asal data

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

subjek data

Individu yang datanya dikumpulkan dan diproses.

gudang data

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

bahasa definisi database (DDL)

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

bahasa manipulasi basis data (DHTML)

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

DDL

Lihat bahasa definisi database.

ansambel yang dalam

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

pembelajaran mendalam

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

defense-in-depth

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

administrator yang didelegasikan

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

deployment

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

lingkungan pengembangan

Lihat lingkungan.

kontrol detektif

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

pemetaan aliran nilai pengembangan (DVSM)

Sebuah proses yang digunakan untuk mengidentifikasi dan memprioritaskan kendala yang mempengaruhi kecepatan dan kualitas dalam siklus hidup pengembangan perangkat lunak. DVSM memperluas proses pemetaan aliran nilai yang awalnya dirancang untuk praktik

manufaktur ramping. Ini berfokus pada langkah-langkah dan tim yang diperlukan untuk menciptakan dan memindahkan nilai melalui proses pengembangan perangkat lunak.

kembar digital

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

tabel dimensi

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

musibah

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

pemulihan bencana (DR)

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

DML~

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

epik

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

ERP

Lihat perencanaan sumber daya perusahaan.

analisis data eksplorasi (EDA)

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

F

tabel fakta

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

gagal cepat

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

batas isolasi kesalahan

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

cabang fitur

Lihat cabang.

fitur

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

pentingnya fitur

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

transformasi fitur

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

beberapa tembakan mendorong

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

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?

interpretasi

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 file. AWS Cloud Untuk informasi lebih lanjut, lihat entri 7 Rs di glosarium ini dan lihat Memobilisasi organisasi Anda untuk mempercepat migrasi skala besar.

ML

Lihat pembelajaran mesin.

modernisasi

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

penilaian kesiapan modernisasi

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

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

aplikasi monolitik (monolit)

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

MPA

Lihat Penilaian Portofolio Migrasi.

MQTT

Lihat Transportasi Telemetri Antrian Pesan.

klasifikasi multiclass

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

infrastruktur yang bisa berubah

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

O

OAC

Lihat kontrol akses asal.

OAI

Lihat identitas akses asal.

OCM

Lihat manajemen perubahan organisasi.

migrasi offline

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

OI

Lihat integrasi operasi.

OLA

Lihat perjanjian tingkat operasional.

migrasi online

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

OPC-UA

Lihat Komunikasi Proses Terbuka - Arsitektur Terpadu.

Komunikasi Proses Terbuka - Arsitektur Terpadu (OPC-UA)

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

perjanjian tingkat operasional (OLA)

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

Tinjauan Kesiapan Operasional (ORR)

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

teknologi operasional (OT)

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

integrasi operasi (OI)

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

jejak organisasi

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

manajemen perubahan organisasi (OCM)

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

kontrol akses asal (OAC)

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

identitas akses asal (OAI)

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

ORR

Lihat tinjauan kesiapan operasional.

OT

Lihat teknologi operasional.

keluar (jalan keluar) VPC

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

P

batas izin

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

Informasi Identifikasi Pribadi (PII)

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

PII

Lihat informasi yang dapat diidentifikasi secara pribadi.

buku pedoman

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

PLC

Lihat pengontrol logika yang dapat diprogram.

PLM

Lihat manajemen siklus hidup produk.

kebijakan

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

persistensi poliglot

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

penilaian portofolio

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

predikat

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

predikat pushdown

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

kontrol preventif

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

principal

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

privasi berdasarkan desain

Pendekatan rekayasa sistem yang memperhitungkan privasi melalui seluruh proses pengembangan.

zona host pribadi

Container yang menyimpan informasi tentang bagaimana Anda ingin Amazon Route 53 merespons kueri DNS untuk domain dan subdomainnya dalam satu atau 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). Jenis dukungan (S) adalah opsional. Jika Anda menyertakan dukungan, matriks disebut matriks RASCI, dan jika Anda mengecualikannya, itu disebut matriks RACI.

kontrol responsif

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

melestarikan

Lihat 7 Rs.

pensiun

Lihat 7 Rs.

Retrieval Augmented Generation (RAG)

Teknologi AI generatif di mana LLM mereferensikan sumber data otoritatif yang berada di luar sumber data pelatihannya sebelum menghasilkan respons. Misalnya, model RAG mungkin

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

rotasi

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

kontrol akses baris dan kolom (RCAC)

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

RPO

Lihat tujuan titik pemulihan.

RTO

Lihat tujuan waktu pemulihan.

buku runbook

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

D

SAML 2.0

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

PENIPUAN

Lihat kontrol pengawasan dan akuisisi data.

SCP

Lihat kebijakan kontrol layanan.

Rahasia

Dalam AWS Secrets Manager, informasi rahasia atau terbatas, seperti kata sandi atau kredensi 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 sumber daya AWS.

variabel target

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

daftar tugas

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

lingkungan uji

Lihat lingkungan.

pelatihan

Untuk menyediakan data bagi model ML Anda untuk dipelajari. Data pelatihan harus berisi jawaban yang benar. Algoritma pembelajaran menemukan pola dalam data pelatihan yang

memetakan atribut data input ke target (jawaban yang ingin Anda prediksi). Ini menghasilkan model ML yang menangkap pola-pola ini. Anda kemudian dapat menggunakan model ML untuk membuat prediksi pada data baru yang Anda tidak tahu targetnya.

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.

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.