Kerangka siklus hidup ketahanan

AWS Bimbingan Preskriptif

AWS Bimbingan Preskriptif: Kerangka siklus hidup ketahanan

Table of Contents

Kerangka siklus hidup ketahanan: Pendekatan berkelanjutan untuk peningkatan ketahanan

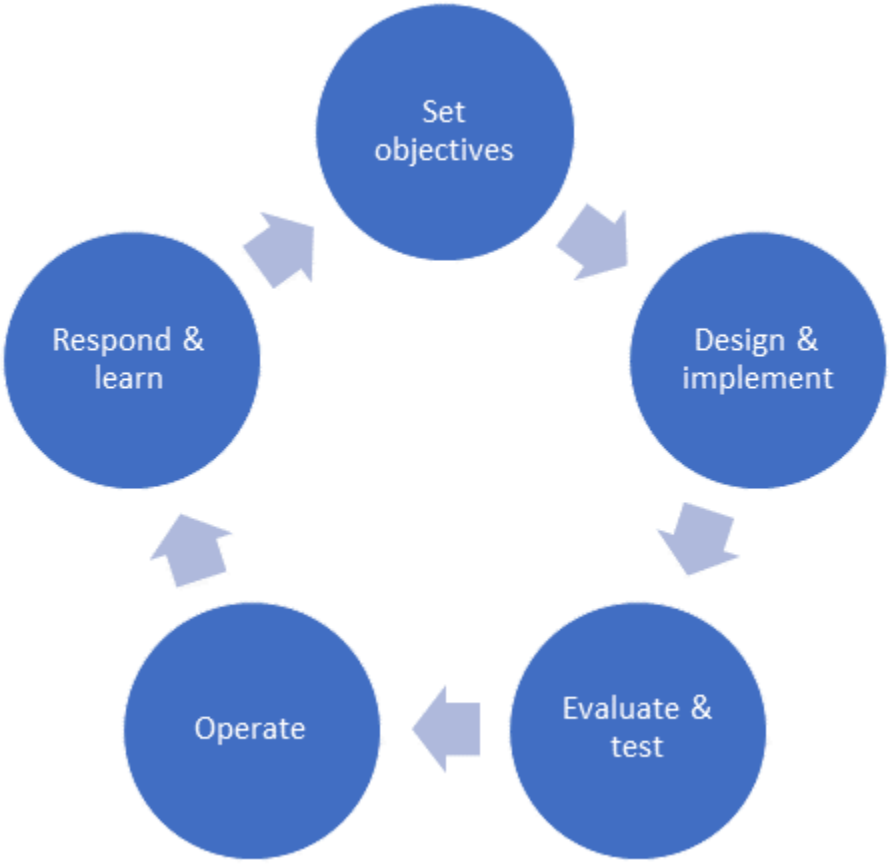
Amazon Web Services (kontributor)

Oktober 2023 (sejarah dokumen)

Organisasi modern saat ini menghadapi tantangan terkait ketahanan yang terus meningkat, terutama karena harapan dari pelanggan bergeser ke arah pola pikir yang selalu aktif dan selalu tersedia. Tim jarak jauh dan aplikasi yang kompleks dan terdistribusi digabungkan dengan meningkatnya kebutuhan akan rilis yang sering. Akibatnya, organisasi dan aplikasinya harus lebih tangguh dari sebelumnya.

AWS mendefinisikan ketahanan sebagai kemampuan aplikasi untuk menolak atau memulihkan dari gangguan, termasuk yang terkait dengan infrastruktur, layanan dependen, kesalahan konfigurasi, dan masalah jaringan sementara. (Lihat Ketahanan, dan komponen keandalan dalam dokumentasi Pilar Keandalan Kerangka AWS Well-Architected Framework.) Namun, untuk mencapai tingkat ketahanan yang diinginkan, pertukaran sering diperlukan. Kompleksitas operasional, kompleksitas teknik, dan biaya perlu dinilai dan disesuaikan.

Berdasarkan bertahun-tahun bekerja dengan pelanggan dan tim internal, AWS telah mengembangkan kerangka siklus hidup ketahanan yang menangkap pembelajaran ketahanan dan praktik terbaik. Kerangka kerja menguraikan lima tahap kunci yang diilustrasikan dalam diagram berikut. Pada setiap tahap Anda dapat menggunakan strategi, layanan, dan mekanisme untuk meningkatkan postur ketahanan Anda.



Tahapan-tahapan ini dibahas di bagian berikut dari panduan ini:

- Tahap 1: Tetapkan tujuan
- Tahap 2: Desain dan implementasi
- Tahap 3: Evaluasi dan uji
- Tahap 4: Beroperasi
- Tahap 5: Menanggapi dan belajar

Istilah dan definisi

Konsep ketahanan setiap tahap diterapkan pada tingkat yang berbeda, mulai dari komponen individu hingga seluruh sistem. Menerapkan konsep-konsep ini membutuhkan definisi yang jelas dari beberapa istilah:

- Komponen adalah elemen yang melakukan fungsi, dan terdiri dari sumber daya perangkat lunak dan teknologi. Contoh komponen termasuk konfigurasi kode, infrastruktur seperti jaringan, atau

bahkan server, penyimpanan data, dan dependensi eksternal seperti perangkat otentikasi multi-faktor (MFA).

- Aplikasi adalah kumpulan komponen yang memberikan nilai bisnis, seperti etalase web yang menghadap pelanggan atau proses backend yang meningkatkan model pembelajaran mesin. Aplikasi mungkin terdiri dari subset komponen dalam satu AWS akun, atau mungkin kumpulan beberapa komponen yang menjangkau beberapa Akun AWS dan Wilayah.
- Sistem adalah kumpulan aplikasi, orang, dan proses yang diperlukan untuk mengelola fungsi bisnis tertentu. Ini mencakup aplikasi yang diperlukan untuk menjalankan fungsi; proses operasional seperti integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD), observabilitas, manajemen konfigurasi, respons insiden, dan pemulihan bencana; dan operator yang mengelola tugas-tugas tersebut.
- Gangguan adalah peristiwa yang mencegah aplikasi Anda menjalankan fungsi bisnisnya dengan benar.
- Kerusakan adalah efek gangguan pada aplikasi jika tidak dikurangi. Aplikasi dapat terganggu jika mereka mengalami serangkaian gangguan.

Ketahanan berkelanjutan

Siklus hidup ketahanan adalah proses yang berkelanjutan. Bahkan dalam organisasi yang sama, tim aplikasi Anda mungkin tampil pada tingkat kelengkapan yang berbeda dalam setiap tahap, tergantung pada persyaratan aplikasi Anda. Namun, semakin lengkap setiap tahap, semakin tinggi tingkat ketahanan aplikasi Anda.

Anda harus menganggap siklus hidup ketahanan sebagai proses standar yang dapat dioperasionalkan oleh organisasi Anda. AWS telah dengan sengaja memodelkan siklus hidup ketahanan agar serupa dengan siklus hidup pengembangan perangkat lunak (SDLC), dengan tujuan menggabungkan perencanaan, pengujian, dan pembelajaran di seluruh proses operasi saat Anda mengembangkan dan mengoperasikan aplikasi Anda. Seperti banyak proses pengembangan tangkas, siklus hidup ketahanan dapat diulang dengan setiap iterasi proses pengembangan. Kami menyarankan Anda memperdalam praktik dalam setiap tahap siklus hidup secara progresif dari waktu ke waktu.

Tahap 1: Tetapkan tujuan

Memahami tingkat ketahanan apa yang dibutuhkan dan bagaimana Anda akan mengukurnya adalah dasar untuk tahap tujuan yang ditetapkan. Sulit untuk meningkatkan sesuatu jika Anda tidak memiliki tujuan dan Anda tidak dapat mengukurnya.

Tidak semua aplikasi membutuhkan tingkat ketahanan yang sama. Ketika Anda menetapkan tujuan, pertimbangkan tingkat yang diperlukan untuk melakukan investasi dan trade-off yang benar. Analogi yang bagus untuk ini adalah mobil: Ini memiliki empat ban tetapi hanya membawa satu ban serep. Peluang mendapatkan beberapa ban kempes selama perjalanan rendah, dan memiliki suku cadang tambahan dapat menghilangkan fitur lain, seperti ruang kargo atau efisiensi bahan bakar, jadi ini adalah trade-off yang masuk akal.

Setelah Anda menentukan tujuan, Anda menerapkan kontrol observabilitas di tahap selanjutnya (Tahap 2: desain dan implementasi dan Tahap 4: Operasikan) untuk memahami apakah tujuan terpenuhi.

Pemetaan aplikasi penting

Mendefinisikan tujuan ketahanan seharusnya tidak secara eksklusif menjadi percakapan teknis. Sebaliknya, mulailah dengan fokus berorientasi bisnis untuk memahami apa yang harus diberikan aplikasi dan konsekuensi dari penurunan nilai. Pemahaman tentang tujuan bisnis ini kemudian mengalir ke bidang-bidang seperti arsitektur, teknik, dan operasi. Tujuan ketahanan apa pun yang Anda tetapkan dapat diterapkan pada semua aplikasi Anda, tetapi cara tujuan diukur seringkali bervariasi tergantung pada fungsi aplikasi. Anda mungkin menjalankan aplikasi yang penting untuk bisnis, dan jika aplikasi ini terganggu, organisasi Anda dapat kehilangan pendapatan yang signifikan atau menderita kerugian reputasi. Sebagai alternatif, Anda mungkin memiliki aplikasi lain yang tidak terlalu penting dan dapat mentolerir beberapa downtime tanpa berdampak negatif pada kemampuan organisasi Anda untuk melakukan bisnis.

Sebagai contoh, pikirkan aplikasi manajemen pesanan untuk perusahaan ritel. Jika komponen aplikasi manajemen pesanan terganggu dan tidak berjalan dengan baik, penjualan baru tidak akan berhasil. Perusahaan ritel ini juga memiliki kedai kopi untuk karyawannya yang berlokasi di salah satu bangunannya. Kedai kopi memiliki menu online yang dapat diakses karyawan di halaman web statis. Jika halaman web ini menjadi tidak tersedia, beberapa karyawan mungkin mengeluh, tetapi itu tidak akan menyebabkan kerugian finansial bagi perusahaan. Berdasarkan contoh ini, bisnis kemungkinan

akan memilih untuk memiliki tujuan ketahanan yang lebih agresif untuk aplikasi manajemen pesanan tetapi tidak akan melakukan investasi yang signifikan untuk memastikan ketahanan aplikasi web.

Mengidentifikasi aplikasi yang paling penting, di mana harus menerapkan upaya paling banyak, dan di mana melakukan trade-off sama pentingnya dengan mampu mengukur ketahanan aplikasi dalam produksi. Untuk lebih memahami dampak penurunan nilai, Anda dapat melakukan analisis dampak bisnis (BIA). BIA menyediakan pendekatan terstruktur dan sistematis untuk mengidentifikasi dan memprioritaskan aplikasi bisnis penting, menilai potensi risiko dan dampak, dan mengidentifikasi dependensi pendukung. BIA membantu mengukur biaya downtime untuk aplikasi terpenting organisasi Anda. Metrik ini membantu menguraikan berapa biayanya jika aplikasi tertentu terganggu dan tidak dapat menyelesaikan fungsinya. Pada contoh sebelumnya, jika aplikasi manajemen pesanan terganggu, bisnis ritel bisa kehilangan pendapatan yang signifikan.

Memetakan cerita pengguna

Selama proses BIA, Anda mungkin menemukan bahwa aplikasi bertanggung jawab atas lebih dari satu fungsi bisnis, atau bahwa fungsi bisnis memerlukan banyak aplikasi. Menggunakan contoh perusahaan ritel sebelumnya, fungsi manajemen pesanan mungkin memerlukan aplikasi terpisah untuk checkout, promosi, dan harga. Jika salah satu aplikasi gagal, dampaknya bisa dirasakan oleh bisnis dan oleh pengguna yang berinteraksi dengan perusahaan. Misalnya, perusahaan mungkin tidak dapat menambahkan pesanan baru, memberikan akses ke promosi dan diskon, atau memperbarui harga produk mereka. Fungsi berbeda yang diperlukan oleh fungsi manajemen pesanan ini mungkin bergantung pada beberapa aplikasi. Fungsi-fungsi ini mungkin juga memiliki beberapa dependensi eksternal, yang membuat proses mencapai ketahanan yang berfokus pada komponen murni terlalu kompleks. Cara yang lebih baik untuk menangani skenario ini adalah dengan fokus pada cerita pengguna, yang menguraikan pengalaman yang diharapkan pengguna saat berinteraksi dengan satu aplikasi atau serangkaian aplikasi.

Berfokus pada cerita pengguna membantu Anda memahami bagian mana dari pengalaman pelanggan yang paling penting, sehingga Anda dapat membangun mekanisme untuk melindungi terhadap ancaman tertentu. Pada contoh sebelumnya, satu cerita pengguna bisa berupa checkout, yang melibatkan aplikasi checkout dan memiliki ketergantungan pada aplikasi penetapan harga. Kisah pengguna lain bisa melihat promosi, yang melibatkan aplikasi promosi. Setelah Anda memetakan aplikasi yang paling penting dan cerita pengguna mereka, Anda dapat mulai menentukan metrik yang akan Anda gunakan untuk mengukur ketahanan untuk cerita pengguna ini. Metrik ini dapat diterapkan di seluruh portofolio atau ke cerita pengguna individu.

Mendefinisikan pengukuran

Tujuan titik pemulihan (RPO), tujuan waktu pemulihan (RTO), dan tujuan tingkat layanan (SLO) adalah pengukuran industri standar yang digunakan untuk menilai ketahanan sistem tertentu. RPO mengacu pada seberapa banyak kehilangan data yang dapat ditoleransi bisnis jika terjadi kegagalan, sedangkan RTO adalah ukuran seberapa cepat aplikasi harus tersedia lagi setelah pemadaman. Kedua metrik ini diukur dalam satuan waktu: detik, menit, dan jam. Anda juga dapat mengukur jumlah waktu selama aplikasi berfungsi dengan baik; yaitu, ia menjalankan fungsinya sebagaimana dirancang dan dapat diakses oleh penggunanya. SLOs ini merinci tingkat layanan yang diharapkan akan diterima pelanggan dan diukur dengan metrik seperti persentase (%) permintaan yang dilayani tanpa kesalahan dalam waktu respons yang kurang dari satu detik (misalnya, 99,99% permintaan akan menerima respons setiap bulan). RPO dan RTO terkait dengan strategi pemulihan bencana, dengan asumsi bahwa akan ada gangguan dalam operasi aplikasi dan proses pemulihan yang berkisar dari memulihkan cadangan hingga mengarahkan lalu lintas pengguna. SLO ditangani dengan menerapkan kontrol ketersediaan tinggi, yang cenderung mengurangi waktu henti untuk suatu aplikasi.

Metrik SLO umumnya digunakan dalam definisi perjanjian tingkat layanan (SLA), yang merupakan kontrak antara penyedia layanan dan pengguna akhir. SLA biasanya datang dengan komitmen keuangan dan menguraikan hukuman yang perlu dibayar oleh penyedia jika perjanjian ini tidak dipenuhi. Namun, SLA bukanlah ukuran postur ketahanan Anda, dan meningkatkan SLA tidak membuat aplikasi Anda lebih tangguh.

Anda dapat mulai menetapkan tujuan Anda berdasarkan SLO, RPO, dan RTO. Setelah Anda menentukan tujuan ketahanan Anda dan mendapatkan pemahaman yang jelas tentang target RPO dan RTO Anda, Anda dapat menggunakan untuk menjalankan penilaian arsitektur Anda AWS Resilience Hubuntuk mengungkap potensi kelemahan terkait ketahanan. AWS Resilience Hub menilai arsitektur aplikasi terhadap praktik terbaik AWS Well-Architected Framework dan berbagi panduan remediasi dalam konteks apa yang secara khusus perlu ditingkatkan untuk memenuhi target RTO dan RPO yang Anda tentukan.

Membuat pengukuran tambahan

RPO, RTO, dan SLO adalah indikator ketahanan yang baik, tetapi Anda juga dapat memikirkan tujuan dari perspektif bisnis dan menentukan tujuan di sekitar fungsi aplikasi Anda. Misalnya, tujuan Anda adalah: Pesanan yang berhasil per menit akan tetap di atas 98% jika latensi antara frontend dan backend saya meningkat sebesar 40%.Atau: Aliran yang dimulai per detik akan tetap dalam

standar deviasi dari rata-rata bahkan jika komponen tertentu hilang. Anda juga dapat membuat tujuan untuk mencapai pengurangan waktu rata-rata untuk memulihkan (MTTR) di seluruh jenis kegagalan yang diketahui; misalnya: Waktu pemulihan akan berkurang sebesar x% jika salah satu dari masalah yang diketahui ini terjadi. Menciptakan tujuan yang selaras dengan kebutuhan bisnis membantu Anda mengantisipasi jenis kegagalan yang harus ditoleransi aplikasi Anda. Ini juga membantu Anda mengidentifikasi pendekatan untuk mengurangi kemungkinan gangguan pada aplikasi Anda.

Jika Anda berpikir tentang tujuan untuk terus beroperasi jika Anda kehilangan 5% dari instance yang memberi daya pada aplikasi Anda, Anda mungkin menentukan bahwa aplikasi Anda harus di-prescaled atau memiliki kemampuan untuk menskalakan cukup cepat untuk mendukung lalu lintas tambahan yang disebabkan selama acara tersebut. Atau, Anda mungkin menentukan bahwa Anda harus memanfaatkan pola arsitektur yang berbeda, seperti yang dijelaskan di Tahap 2: Bagian Desain dan implementasi.

Anda juga harus menerapkan langkah-langkah observabilitas untuk tujuan bisnis spesifik Anda. Misalnya, Anda dapat melacak tingkat pesanan rata-rata, harga pesanan rata-rata, jumlah rata-rata langganan, atau metrik lain yang dapat memberikan wawasan tentang kesehatan bisnis berdasarkan perilaku aplikasi Anda. Dengan menerapkan kemampuan observabilitas untuk aplikasi Anda, Anda dapat membuat alarm dan mengambil tindakan jika metrik ini melebihi batas yang ditentukan. Observabilitas dibahas secara lebih rinci di bagian Tahap 4: Operate.

Tahap 2: Desain dan implementasi

Pada tahap sebelumnya, Anda menetapkan tujuan ketahanan Anda. Sekarang pada tahap desain dan implementasi, Anda mencoba mengantisipasi mode kegagalan dan mengidentifikasi pilihan desain, sebagaimana dipandu oleh tujuan yang Anda tetapkan pada tahap sebelumnya. Anda juga menentukan strategi untuk manajemen perubahan dan mengembangkan kode perangkat lunak dan konfigurasi infrastruktur. Bagian berikut menyoroti praktik AWS terbaik yang harus Anda pertimbangkan saat mengambil trade-off seperti biaya, kompleksitas, dan overhead operasional.

AWS Kerangka Well-Architected

Ketika Anda merancang aplikasi Anda berdasarkan tujuan ketahanan yang Anda inginkan, Anda perlu mengevaluasi beberapa faktor dan membuat trade-off pada arsitektur yang paling optimal. Untuk membangun aplikasi yang sangat tangguh, Anda harus mempertimbangkan aspek desain, bangunan dan penyebaran, keamanan, dan operasi. The AWS Well-Architected Framework menyediakan seperangkat praktik terbaik, prinsip desain, dan pola arsitektur untuk membantu Anda merancang aplikasi yang tangguh. AWS Enam pilar Kerangka AWS Well-Architected memberikan praktik terbaik untuk merancang dan mengoperasikan sistem yang tangguh, aman, efisien, hemat biaya, dan berkelanjutan. Kerangka kerja ini menyediakan cara untuk secara konsisten mengukur arsitektur Anda terhadap praktik terbaik dan mengidentifikasi area untuk perbaikan.

Berikut ini adalah contoh bagaimana AWS Well-Architected Framework dapat membantu Anda merancang dan mengimplementasikan aplikasi yang memenuhi tujuan ketahanan Anda:

- Pilar keandalan: Pilar keandalan menekankan pentingnya membangun aplikasi yang dapat beroperasi dengan benar dan konsisten, bahkan selama kegagalan atau gangguan. Misalnya, AWS Well-Architected Framework merekomendasikan agar Anda menggunakan arsitektur microservices untuk membuat aplikasi Anda lebih kecil dan lebih sederhana, sehingga Anda dapat membedakan antara kebutuhan ketersediaan komponen yang berbeda dalam aplikasi Anda. Anda juga dapat menemukan deskripsi rinci tentang praktik terbaik untuk membangun aplikasi dengan menggunakan pelambatan, coba lagi dengan mundur eksponensial, gagal cepat (pelepasan beban), idempotensi, kerja konstan, pemutus sirkuit, dan stabilitas statis.
- Tinjauan komprehensif: The AWS Well-Architected Framework mendorong tinjauan komprehensif arsitektur Anda terhadap praktik terbaik dan prinsip desain. Ini menyediakan cara untuk secara konsisten mengukur arsitektur Anda dan mengidentifikasi area untuk perbaikan.

- Manajemen risiko: The AWS Well-Architected Framework membantu Anda mengidentifikasi dan mengelola risiko yang mungkin memengaruhi keandalan aplikasi Anda. Dengan mengatasi skenario kegagalan potensial secara proaktif, Anda dapat mengurangi kemungkinannya atau kerusakan yang diakibatkannya.
- Peningkatan berkelanjutan: Ketahanan adalah proses yang berkelanjutan, dan Kerangka AWS Well-Architected menekankan perbaikan berkelanjutan. Dengan secara teratur meninjau dan menyempurnakan arsitektur dan proses Anda berdasarkan panduan AWS Well-Architected Framework, Anda dapat memastikan bahwa sistem Anda tetap tangguh dalam menghadapi tantangan dan persyaratan yang berkembang.

Memahami dependensi

Memahami dependensi sistem adalah kunci untuk ketahanan. Dependensi mencakup koneksi antara komponen dalam aplikasi, dan koneksi ke komponen di luar aplikasi, seperti API pihak ketiga dan layanan bersama milik bisnis. Memahami koneksi ini membantu Anda mengisolasi dan mengelola gangguan, karena gangguan pada satu komponen dapat memengaruhi komponen lainnya. Pengetahuan ini membantu para insinyur menilai dampak gangguan dan merencanakan yang sesuai, dan memastikan bahwa sumber daya digunakan secara efektif. Memahami dependensi membantu Anda membuat strategi alternatif dan mengoordinasikan proses pemulihan. Ini juga membantu Anda menentukan kasus di mana Anda dapat mengganti ketergantungan keras dengan dependensi lunak, sehingga aplikasi Anda dapat terus melayani fungsi bisnisnya ketika ada gangguan ketergantungan. Dependensi juga memengaruhi keputusan tentang load balancing dan penskalaan aplikasi. Memahami dependensi sangat penting ketika Anda membuat perubahan pada aplikasi Anda, karena dapat membantu Anda menentukan potensi risiko dan dampak. Pengetahuan ini membantu Anda membuat aplikasi yang stabil dan tangguh, membantu dalam manajemen kesalahan, penilaian dampak, pemulihan gangguan, penyeimbangan beban, penskalaan, dan manajemen perubahan. Anda dapat melacak dependensi secara manual atau menggunakan alat dan layanan seperti AWS X-Rayuntuk memahami dependensi aplikasi terdistribusi Anda.

Strategi pemulihan bencana

Strategi pemulihan bencana (DR) memainkan peran penting dalam membangun dan mengoperasikan aplikasi yang tangguh, terutama dengan memastikan kelangsungan bisnis. Ini menjamin bahwa operasi bisnis yang penting dapat bertahan dengan penurunan sesedikit mungkin, bahkan selama peristiwa bencana, sehingga meminimalkan waktu henti dan potensi hilangnya pendapatan. Strategi DR sangat penting untuk perlindungan data karena sering menggabungkan

cadangan data reguler dan replikasi data di beberapa lokasi, yang membantu melindungi informasi bisnis yang berharga dan membantu mencegah kerugian total selama bencana. Selain itu, banyak industri diatur oleh kebijakan yang mengharuskan bisnis untuk memiliki strategi DR untuk melindungi data sensitif dan untuk memastikan bahwa layanan tetap tersedia selama bencana. Dengan memastikan gangguan layanan minimal, strategi DR juga meningkatkan kepercayaan dan kepuasan pelanggan. Strategi DR yang diterapkan dengan baik dan sering dipraktikkan mengurangi waktu pemulihan setelah bencana, dan membantu memastikan bahwa aplikasi dengan cepat dibawa kembali online. Selain itu, bencana dapat menimbulkan biaya besar, tidak hanya dari kehilangan pendapatan karena downtime, tetapi juga dari biaya pemulihan aplikasi dan data. Strategi DR yang dirancang dengan baik membantu melindungi dari kerugian finansial ini.

Strategi yang Anda pilih tergantung pada kebutuhan spesifik aplikasi Anda, RTO dan RPO Anda, dan anggaran Anda. AWS Elastic Disaster Recoveryadalah layanan ketahanan yang dibuat khusus yang dapat Anda gunakan untuk membantu mengimplementasikan strategi DR untuk aplikasi lokal dan berbasis cloud.

Untuk informasi selengkapnya, lihat Disaster Recovery of Workloads on AWS and AWS Multi-Region Fundamentals di situs web. AWS

Mendefinisikan strategi CI/CD

Salah satu penyebab umum gangguan aplikasi adalah kode atau perubahan lain yang mengubah aplikasi dari keadaan kerja yang diketahui sebelumnya. Jika Anda tidak menangani manajemen perubahan dengan hati-hati, itu dapat menyebabkan kerusakan yang sering terjadi. Frekuensi perubahan meningkatkan peluang untuk dampak. Namun, membuat perubahan lebih jarang menghasilkan set perubahan yang lebih besar, yang jauh lebih mungkin mengakibatkan penurunan karena kompleksitasnya yang tinggi. Praktik Continuous Integration and Continuous Delivery (CI/CD) dirancang untuk menjaga perubahan kecil dan sering (menghasilkan peningkatan produktivitas) sambil menundukkan setiap perubahan ke tingkat inspeksi yang tinggi melalui otomatisasi. Beberapa strategi dasar adalah:

- Otomatisasi penuh: Konsep dasar CI/CD adalah untuk mengotomatiskan proses build dan deployment sebanyak mungkin. Ini termasuk pembangunan, pengujian, penyebaran, dan bahkan pemantauan. Pipa otomatis membantu mengurangi kemungkinan kesalahan manusia, memastikan konsistensi, dan membuat proses lebih andal dan efisien.
- Test-driven development (TDD): Tulis tes sebelum menulis kode aplikasi. Praktik ini memastikan bahwa semua kode memiliki pengujian terkait, yang meningkatkan keandalan kode dan kualitas inspeksi otomatis. Pengujian ini dijalankan di pipeline CI untuk memvalidasi perubahan.

- Komit dan integrasi yang sering: Dorong pengembang untuk sering melakukan kode dan sering melakukan integrasi. Perubahan kecil dan sering lebih mudah untuk diuji dan di-debug, yang mengurangi risiko masalah yang signifikan. Otomatisasi mengurangi biaya setiap komit dan penyebaran, membuat integrasi yang sering menjadi mungkin.
- Infrastruktur yang tidak dapat diubah: Perlakukan server Anda dan komponen infrastruktur lainnya seperti entitas statis dan tidak dapat diubah. Ganti infrastruktur alih-alih memodifikasinya sebanyak mungkin, dan bangun infrastruktur baru melalui kode yang diuji, dan diterapkan melalui pipeline Anda.
- Mekanisme rollback: Selalu miliki cara yang mudah, andal, dan sering diuji untuk mengembalikan perubahan jika terjadi kesalahan. Mampu dengan cepat kembali ke keadaan baik yang diketahui sebelumnya dengan cepat sangat penting untuk keselamatan penerapan. Ini bisa menjadi tombol sederhana untuk kembali ke keadaan sebelumnya, atau dapat sepenuhnya otomatis dan diprakarsai oleh alarm.
- Kontrol versi: Pertahankan semua kode aplikasi, konfigurasi, dan bahkan infrastruktur sebagai kode dalam repositori yang dikendalikan versi. Praktik ini membantu memastikan bahwa Anda dapat dengan mudah melacak perubahan dan mengembalikannya jika diperlukan.
- Penerapan Canary dan penerapan biru/hijau: Menerapkan versi baru aplikasi Anda ke subset infrastruktur Anda terlebih dahulu, atau mempertahankan dua lingkungan (biru/hijau), memungkinkan Anda memverifikasi perilaku perubahan dalam produksi dan memutar kembali dengan cepat jika perlu.

CI/CD bukan hanya tentang alat tetapi juga tentang budaya. Menciptakan budaya yang menghargai otomatisasi, pengujian, dan pembelajaran dari kegagalan sama pentingnya dengan menerapkan alat dan proses yang tepat. Rollback, jika dilakukan dengan sangat cepat dengan dampak minimal, tidak boleh dianggap sebagai kegagalan tetapi pengalaman belajar.

Melakukan ORR

Tinjauan kesiapan operasional (ORR) membantu mengidentifikasi kesenjangan operasional dan prosedural. Di Amazon, kami menciptakan ORR untuk menyaring pembelajaran dari beberapa dekade mengoperasikan layanan skala tinggi menjadi pertanyaan yang dikuratori dengan panduan praktik terbaik. ORR menangkap pelajaran sebelumnya dan membutuhkan tim baru untuk memastikan bahwa mereka telah memperhitungkan pelajaran ini dalam aplikasi mereka. ORR dapat memberikan daftar mode kegagalan atau penyebab kegagalan yang dapat dibawa ke dalam aktivitas pemodelan ketahanan yang dijelaskan di bagian pemodelan ketahanan di bawah ini. Untuk informasi lebih lanjut, lihat Ulasan Kesiapan Operasional (ORR) di situs web Well-Architected AWS Framework.

Memahami batas-batas isolasi AWS kesalahan

AWS menyediakan beberapa batasan isolasi kesalahan untuk membantu Anda mencapai tujuan ketahanan Anda. Anda dapat menggunakan batas-batas ini untuk memanfaatkan ruang lingkup penahanan dampak yang dapat diprediksi yang mereka berikan. Anda harus terbiasa dengan bagaimana AWS layanan dirancang dengan menggunakan batas-batas ini sehingga Anda dapat membuat pilihan yang disengaja tentang dependensi yang Anda pilih untuk aplikasi Anda. Untuk memahami cara menggunakan batasan dalam aplikasi Anda, lihat Batas Isolasi AWS Kesalahan di AWS situs web.

Memilih tanggapan

Sebuah sistem dapat merespons dengan berbagai cara untuk alarm. Beberapa alarm mungkin memerlukan respons dari tim operasi sedangkan yang lain mungkin memicu mekanisme penyembuhan diri dalam aplikasi. Anda mungkin memutuskan untuk menyimpan tanggapan yang dapat diotomatisasi sebagai operasi manual untuk mengontrol biaya otomatisasi atau untuk mengelola kendala teknik. Jenis respons terhadap alarm kemungkinan akan dipilih sebagai fungsi dari biaya penerapan respons, frekuensi alarm yang diantisipasi, keakuratan alarm, dan potensi kerugian bisnis karena tidak menanggapi alarm sama sekali.

Misalnya, ketika proses server mogok, proses mungkin dimulai ulang oleh sistem operasi, atau server baru mungkin disediakan dan yang lama dihentikan, atau operator mungkin diinstruksikan untuk terhubung dari jarak jauh ke server dan memulai ulang. Respons ini memiliki hasil yang sama — memulai kembali proses server aplikasi — tetapi memiliki berbagai tingkat implementasi dan biaya pemeliharaan.

> **Note**
>
> Anda dapat memilih beberapa tanggapan untuk mengambil pendekatan ketahanan yang mendalam. Misalnya, dalam skenario sebelumnya, tim aplikasi mungkin memilih untuk mengimplementasikan ketiga respons dengan waktu tunda di antara masing-masing. Jika indikator proses server gagal masih dalam keadaan khawatir setelah 30 detik, tim dapat berasumsi bahwa sistem operasi telah gagal me-restart server aplikasi. Oleh karena itu, mereka mungkin membuat grup penskalaan otomatis untuk membuat server virtual baru dan memulihkan proses server aplikasi. Jika indikator masih dalam keadaan alarm setelah 300 detik, peringatan mungkin dikirim ke staf operasional untuk terhubung ke server asli dan mencoba memulihkan proses.

Tanggapan bahwa tim aplikasi dan pilihan bisnis harus mencerminkan selera bisnis untuk mengimbangi overhead operasional dengan investasi di muka dalam waktu rekayasa. Anda harus memilih respons — pola arsitektur seperti stabilitas statis, pola perangkat lunak seperti pemutus sirkuit, atau prosedur operasional–dengan mempertimbangkan secara cermat kendala dan pemeliharaan yang diantisipasi dari setiap opsi respons. Beberapa tanggapan standar mungkin ada untuk memandu tim aplikasi, sehingga Anda dapat menggunakan pustaka dan pola yang dikelola oleh fungsi arsitektur terpusat Anda sebagai masukan untuk pertimbangan ini.

Pemodelan ketahanan

Pemodelan ketahanan mendokumentasikan bagaimana aplikasi akan merespons berbagai gangguan yang diantisipasi. Dengan mengantisipasi gangguan, tim Anda dapat menerapkan observabilitas, kontrol otomatis, dan proses pemulihan untuk mengurangi atau mencegah gangguan meskipun ada gangguan. AWS telah menciptakan panduan untuk mengembangkan model ketahanan dengan menggunakan kerangka analisis ketahanan. Kerangka kerja ini dapat membantu Anda mengantisipasi gangguan dan dampaknya terhadap aplikasi Anda. Dengan mengantisipasi gangguan, Anda dapat mengidentifikasi mitigasi yang diperlukan untuk membangun aplikasi yang tangguh dan andal. Kami menyarankan Anda menggunakan kerangka analisis ketahanan untuk memperbarui model ketahanan Anda dengan setiap iterasi siklus hidup aplikasi Anda. Menggunakan kerangka kerja ini dengan setiap iterasi membantu mengurangi insiden dengan mengantisipasi gangguan selama fase desain dan menguji aplikasi sebelum dan sesudah penerapan produksi. Mengembangkan model ketahanan dengan menggunakan kerangka kerja ini membantu Anda memastikan bahwa Anda memenuhi tujuan ketahanan Anda.

Gagal dengan aman

Jika Anda tidak dapat menghindari gangguan, gagal dengan aman. Pertimbangkan untuk membuat aplikasi Anda dengan mode operasi fail-safe default, di mana tidak ada kerugian bisnis yang signifikan yang dapat terjadi. Contoh status fail-safe untuk database adalah default ke operasi hanya-baca, di mana pengguna tidak diizinkan untuk membuat atau mengubah data apa pun. Bergantung pada sensitivitas data, Anda bahkan mungkin ingin aplikasi default ke status shutdown dan bahkan tidak melakukan kueri hanya-baca. Pertimbangkan status fail-safe untuk aplikasi Anda seharusnya, dan default ke mode operasi ini dalam kondisi ekstrim.

Tahap 3: Evaluasi dan uji

Selama tahap evaluasi dan pengujian siklus hidup, aplikasi, atau perubahan pada aplikasi yang ada, telah dirancang tetapi belum dirilis ke produksi. Pada tahap ini, Anda menerapkan kegiatan untuk menguji praktik yang telah dilakukan pada tahap sebelumnya dan mengevaluasi hasilnya. Aplikasi mungkin masih dalam pengembangan aktif, atau pengembangan primer mungkin lengkap dan aplikasi mungkin sedang menjalani pengujian sebelum dirilis ke produksi. Selama tahap ini, Anda fokus pada pengembangan dan menjalankan tes yang mengkonfirmasi atau membantah harapan bahwa aplikasi akan memenuhi tujuan yang ditentukan untuk ketahanan. Selain itu, Anda mengembangkan dan menguji prosedur operasional sistem. Prosedur penyebaran yang Anda kembangkan di Tahap 2: Tahap desain dan implementasi dipraktikkan dan hasilnya dievaluasi. Meskipun kegiatan pengujian dan evaluasi ini dimulai selama bagian siklus hidup ini, mereka tidak berakhir di sini. Pengujian dan evaluasi berlanjut saat Anda pindah ke Tahap 4: Mengoperasikan tahap.

Tahap evaluasi dan pengujian dibagi menjadi dua fase: kegiatan pra-penyebaran dan kegiatan pasca-penyebaran. Aktivitas pra-penerapan terdiri dari tugas-tugas yang harus diselesaikan sebelum Anda menyebarkan aplikasi ke lingkungan apa pun, termasuk menerapkan versi baru perangkat lunak serta penerapan awal ke lingkungan pengujian. Kegiatan pasca-penyebaran terjadi setelah perangkat lunak telah digunakan ke dalam lingkungan pengujian atau produksi. Bagian berikut membahas fase-fase ini secara lebih rinci.

Kegiatan pra-penyebaran

Desain lingkungan

Lingkungan di mana Anda menguji dan mengevaluasi aplikasi Anda memengaruhi seberapa teliti Anda dapat mengujinya, dan seberapa besar keyakinan yang Anda miliki bahwa hasil tersebut secara akurat mencerminkan apa yang akan terjadi dalam produksi. Anda mungkin dapat melakukan beberapa pengujian integrasi secara lokal pada mesin pengembang dengan menggunakan layanan seperti Amazon DynamoDB (lihat Menyiapkan DynamoDB lokal dalam dokumentasi DynamoDB). Namun, pada titik tertentu Anda perlu menguji di lingkungan yang mereplikasi lingkungan produksi Anda untuk mencapai kepercayaan tertinggi pada hasil Anda. Lingkungan ini akan dikenakan biaya, jadi kami sarankan Anda mengambil pendekatan bertahap, atau pipa, ke lingkungan Anda, di mana lingkungan seperti produksi muncul kemudian dalam pipa.

Pengujian integrasi

Pengujian integrasi adalah proses pengujian bahwa komponen aplikasi yang terdefinisi dengan baik menjalankan fungsinya dengan benar ketika beroperasi dengan dependensi eksternal. Dependensi eksternal tersebut dapat berupa komponen lain yang dikembangkan khusus, AWS layanan yang Anda gunakan untuk aplikasi, dependensi pihak ketiga, dan dependensi lokal. Panduan ini berfokus pada tes integrasi yang menunjukkan ketahanan aplikasi Anda. Ini mengasumsikan bahwa tes unit dan integrasi sudah ada yang menunjukkan akurasi fungsional perangkat lunak Anda.

Kami menyarankan Anda merancang tes integrasi yang secara khusus menguji pola ketahanan yang telah Anda terapkan, seperti pola pemutus sirkuit atau pelepasan beban (lihat Tahap 2: Desain dan implementasi). Tes integrasi berorientasi ketahanan sering melibatkan penerapan beban tertentu ke aplikasi atau sengaja memperkenalkan gangguan ke lingkungan dengan menggunakan kemampuan seperti ().AWS Fault Injection ServiceAWS FIS Idealnya, Anda harus menjalankan semua tes integrasi sebagai bagian dari pipeline CI/CD Anda dan memastikan bahwa Anda menjalankan pengujian setiap kali kode dilakukan. Ini membantu Anda mendeteksi dan bereaksi dengan cepat terhadap setiap perubahan kode atau konfigurasi yang mengakibatkan pelanggaran tujuan ketahanan Anda. Aplikasi terdistribusi skala besar sangat kompleks, dan bahkan perubahan kecil dapat secara signifikan memengaruhi ketahanan bagian aplikasi Anda yang tampaknya tidak terkait. Cobalah untuk menjalankan tes Anda pada setiap komit. AWS menyediakan seperangkat alat yang sangat baik untuk mengoperasikan pipa CI/CD Anda dan alat lainnya DevOps . Untuk informasi lebih lanjut, lihat Pengantar DevOps AWS di AWS situs web.

Pipa penyebaran otomatis

Penerapan ke, dan pengujian di, lingkungan pra-produksi Anda adalah tugas yang berulang dan kompleks yang sebaiknya diserahkan kepada otomatisasi. Otomatisasi proses ini membebaskan sumber daya manusia dan mengurangi peluang kesalahan. Mekanisme untuk mengotomatisasi proses ini sering disebut sebagai pipa. Saat Anda membuat pipeline, kami sarankan Anda menyiapkan serangkaian lingkungan pengujian yang semakin dekat dengan konfigurasi produksi Anda. Anda menggunakan rangkaian lingkungan ini untuk berulang kali menguji aplikasi Anda. Lingkungan pertama menyediakan serangkaian kemampuan yang lebih terbatas daripada lingkungan produksi tetapi menimbulkan biaya yang jauh lebih rendah. Lingkungan selanjutnya harus menambahkan layanan dan skala untuk lebih mencerminkan lingkungan produksi.

Mulailah dengan menguji di lingkungan pertama. Setelah penerapan Anda lulus semua pengujian Anda di lingkungan pengujian pertama, biarkan aplikasi berjalan di bawah sejumlah beban untuk jangka waktu tertentu untuk melihat apakah ada masalah yang terjadi seiring waktu. Konfirmasikan

bahwa Anda telah mengonfigurasi observabilitas dengan benar (lihat Ketepatan alarm nanti di panduan ini) sehingga Anda dapat mendeteksi masalah apa pun yang muncul. Ketika periode pengamatan ini telah berhasil diselesaikan, terapkan aplikasi Anda ke lingkungan pengujian berikutnya dan ulangi prosesnya, tambahkan pengujian atau beban tambahan yang didukung oleh lingkungan. Setelah Anda cukup menguji aplikasi Anda dengan cara ini, Anda dapat menggunakan metode penyebaran yang sebelumnya Anda siapkan untuk menyebarkan aplikasi ke dalam produksi (lihat Tentukan strategi CI/CD sebelumnya dalam panduan ini). Artikel Mengotomatiskan penerapan hands-off yang aman di Amazon Builders' Library adalah sumber daya luar biasa yang menjelaskan cara Amazon mengotomatiskan penerapan kode. Jumlah lingkungan yang mendahului penerapan produksi Anda akan bervariasi, tergantung pada kompleksitas aplikasi Anda dan jenis dependensi yang dimilikinya.

Pengujian beban

Di permukaan, pengujian beban menyerupai pengujian integrasi. Anda menguji fungsi diskrit aplikasi Anda dan dependensi eksternalnya untuk memverifikasi bahwa ia beroperasi seperti yang diharapkan. Pengujian beban kemudian melampaui pengujian integrasi untuk fokus pada bagaimana aplikasi berfungsi di bawah beban yang terdefinisi dengan baik. Pengujian beban memerlukan verifikasi fungsionalitas yang benar, sehingga harus terjadi setelah tes integrasi yang berhasil. Penting untuk memahami seberapa baik aplikasi merespons di bawah beban yang diharapkan serta bagaimana perilakunya ketika beban melebihi harapan. Ini membantu Anda memverifikasi bahwa Anda telah menerapkan mekanisme yang diperlukan untuk memastikan bahwa aplikasi Anda tetap tangguh di bawah beban ekstrim. Untuk panduan komprehensif untuk memuat pengujian AWS, lihat Pengujian Beban Terdistribusi AWS di Pustaka AWS Solusi.

Kegiatan pasca-penyebaran

Ketahanan adalah proses yang berkelanjutan dan evaluasi ketahanan aplikasi Anda harus dilanjutkan setelah aplikasi diterapkan. Hasil aktivitas pasca-penerapan Anda, seperti penilaian ketahanan yang sedang berlangsung, mungkin mengharuskan Anda mengevaluasi ulang dan memperbarui beberapa aktivitas ketahanan yang Anda lakukan sebelumnya dalam siklus hidup ketahanan.

Melakukan penilaian ketahanan

Menilai ketahanan tidak berhenti setelah Anda menerapkan aplikasi Anda ke dalam produksi. Bahkan jika Anda memiliki pipeline penyebaran yang terdefinisi dengan baik dan otomatis, perubahan

terkadang dapat terjadi secara langsung di lingkungan produksi. Selain itu, mungkin ada faktor-faktor yang belum Anda pertimbangkan dalam verifikasi ketahanan pra-penerapan Anda. AWS Resilience Hubmenyediakan tempat sentral di mana Anda dapat menilai apakah arsitektur yang Anda gunakan memenuhi kebutuhan RPO dan RTO yang Anda tentukan. Anda dapat menggunakan layanan ini untuk menjalankan penilaian berdasarkan permintaan atas ketahanan aplikasi Anda, mengotomatiskan penilaian, dan bahkan mengintegrasikannya ke dalam alat CI/CD Anda, seperti yang dibahas dalam AWS posting blog Menilai ketahanan aplikasi secara terus-menerus dengan dan. AWS Resilience HubAWS CodePipeline Mengotomatiskan penilaian ini adalah praktik terbaik karena membantu memastikan bahwa Anda terus mengevaluasi postur ketahanan Anda dalam produksi.

Pengujian DR

Pada Tahap 2: Merancang dan menerapkan, Anda mengembangkan strategi pemulihan bencana (DR) sebagai bagian dari sistem Anda. Selama Tahap 4, Anda harus menguji prosedur DR Anda untuk memastikan bahwa tim Anda sepenuhnya siap untuk suatu insiden dan prosedur Anda bekerja seperti yang diharapkan. Anda harus menguji semua prosedur DR Anda, termasuk failover dan failback, secara teratur dan meninjau hasil setiap latihan untuk menentukan apakah dan bagaimana prosedur sistem Anda harus diperbarui untuk hasil terbaik. Ketika Anda awalnya mengembangkan tes DR Anda, jadwalkan tes jauh sebelumnya dan pastikan bahwa seluruh tim memahami apa yang diharapkan, bagaimana hasil akan diukur, dan mekanisme umpan balik apa yang akan digunakan untuk memperbarui prosedur berdasarkan hasil. Setelah Anda menjadi mahir dalam menjalankan tes DR terjadwal, pertimbangkan untuk menjalankan tes DR yang tidak diumumkan. Bencana nyata tidak terjadi sesuai jadwal, jadi Anda harus siap untuk melaksanakan rencana Anda kapan saja. Namun, tanpa pemberitahuan tidak berarti tidak direncanakan. Pemangku kepentingan utama masih perlu merencanakan acara untuk memastikan bahwa pemantauan yang tepat telah dilakukan dan bahwa pelanggan dan aplikasi penting tidak terkena dampak buruk.

Deteksi drift

Perubahan konfigurasi yang tidak terduga dalam aplikasi produksi dapat terjadi bahkan ketika otomatisasi dan prosedur yang terdefinisi dengan baik sudah ada. Untuk mendeteksi perubahan pada konfigurasi aplikasi Anda, Anda harus memiliki mekanisme untuk mendeteksi drift, yang mengacu pada penyimpangan dari konfigurasi dasar. Untuk mempelajari cara mendeteksi drift di AWS CloudFormation tumpukan, lihat Mendeteksi perubahan konfigurasi yang tidak dikelola pada tumpukan dan sumber daya dalam dokumentasi. AWS CloudFormation Untuk mendeteksi drift di AWS lingkungan aplikasi Anda, lihat Mendeteksi dan menyelesaikan drift AWS Control Tower dalam dokumentasi. AWS Control Tower

Pengujian sintetis

Pengujian sintetis adalah proses pembuatan perangkat lunak yang dapat dikonfigurasi yang berjalan dalam produksi, secara terjadwal, untuk menguji API aplikasi Anda dengan cara yang mensimulasikan pengalaman pengguna akhir. Tes ini kadang-kadang disebut sebagai kenari, mengacu pada penggunaan asli istilah dalam penambangan batubara. Tes sintetis seringkali dapat memberikan peringatan dini ketika aplikasi mengalami gangguan, bahkan jika kerusakannya sebagian atau intermiten, seperti yang sering terjadi pada kegagalan abu-abu.

Rekayasa kekacauan

Rekayasa kekacauan adalah proses sistematis yang melibatkan sengaja menerapkan aplikasi pada peristiwa yang mengganggu dengan cara yang dikurangi risiko, memantau responsnya dengan cermat, dan menerapkan perbaikan yang diperlukan. Tujuannya adalah untuk memvalidasi atau menantang asumsi tentang kemampuan aplikasi untuk menangani gangguan tersebut. Alih-alih meninggalkan peristiwa ini secara kebetulan, rekayasa kekacauan memberdayakan para insinyur untuk mengatur eksperimen dalam lingkungan yang terkendali, biasanya selama periode lalu lintas rendah dan dengan dukungan teknik yang tersedia untuk mitigasi yang efektif.

Rekayasa kekacauan dimulai dengan memahami kondisi operasi normal, yang dikenal sebagai kondisi tunak, dari aplikasi yang sedang dipertimbangkan. Dari sana, Anda merumuskan hipotesis yang merinci perilaku sukses aplikasi di hadapan gangguan. Anda menjalankan eksperimen, yang melibatkan injeksi gangguan yang disengaja, termasuk, namun tidak terbatas pada, latensi jaringan, kegagalan server, kesalahan hard drive, dan gangguan dependensi eksternal. Anda kemudian menganalisis hasil percobaan dan meningkatkan ketahanan aplikasi berdasarkan pembelajaran Anda. Eksperimen ini berfungsi sebagai alat yang berharga untuk meningkatkan berbagai aspek aplikasi, termasuk kinerjanya, dan mengungkap masalah laten yang mungkin tetap tersembunyi. Selain itu, rekayasa kekacauan membantu mengungkapkan kekurangan dalam observabilitas dan alat yang mengkhawatirkan, dan membantu Anda memperbaikinya. Ini juga berkontribusi untuk mengurangi waktu pemulihan dan meningkatkan keterampilan operasional. Rekayasa kekacauan mempercepat penerapan praktik terbaik dan menumbuhkan pola pikir perbaikan berkelanjutan. Pada akhirnya, ini memungkinkan tim untuk membangun dan mengasah keterampilan operasional mereka melalui latihan reguler dan pengulangan.

AWS merekomendasikan agar Anda memulai upaya rekayasa kekacauan Anda di lingkungan non-produksi. Anda dapat menggunakan AWS Fault Injection Service (AWS FIS) untuk menjalankan eksperimen rekayasa kekacauan dengan kesalahan tujuan umum serta kesalahan yang unik. AWS Layanan yang dikelola sepenuhnya ini mencakup alarm kondisi berhenti dan kontrol izin penuh

sehingga Anda dapat dengan mudah mengadopsi rekayasa kekacauan dengan aman dan percaya diri.

Tahap 4: Beroperasi

Setelah Anda menyelesaikan Tahap 3: Evaluasi dan uji, Anda siap untuk menyebarkan aplikasi ke produksi. Pada tahap Operate, Anda menyebarkan aplikasi Anda ke produksi dan mengelola pengalaman pelanggan Anda. Desain dan implementasi aplikasi Anda menentukan banyak hasil ketahanannya, tetapi tahap ini berfokus pada praktik operasional yang digunakan sistem Anda untuk mempertahankan dan meningkatkan ketahanan. Membangun budaya keunggulan operasional membantu menciptakan standar dan konsistensi dalam praktik ini.

Observabilitas

Bagian terpenting dari memahami pengalaman pelanggan adalah melalui pemantauan dan mengkhawatirkan. Anda perlu menginstruksikan aplikasi Anda untuk memahami keadaannya, dan Anda memerlukan perspektif yang beragam, yang berarti Anda perlu mengukur dari sisi server dan sisi klien, biasanya dengan kenari. Metrik Anda harus menyertakan data tentang interaksi aplikasi Anda dengan dependensi dan dimensinya yang selaras dengan batas isolasi kesalahan Anda. Anda juga harus membuat log yang memberikan rincian tambahan tentang setiap unit pekerjaan yang dilakukan oleh aplikasi Anda. Anda dapat mempertimbangkan untuk menggabungkan metrik dan log dengan menggunakan solusi seperti format metrik CloudWatch tertanam Amazon. Anda mungkin akan menemukan bahwa Anda selalu menginginkan lebih banyak pengamatan, jadi pertimbangkan pertukaran biaya, usaha, dan kompleksitas yang diperlukan untuk menerapkan tingkat instrumentasi yang Anda inginkan.

Tautan berikut memberikan praktik terbaik untuk menginstrumentasi aplikasi Anda dan membuat alarm:

- Memantau layanan produksi di Amazon (presentasi AWS re:Invent 2020)
- Amazon Builders' Library: Keunggulan Operasional di Amazon (presentasi re:invent 2021)AWS
- Praktik terbaik observabilitas di Amazon (AWS re:Invent 2022 presentasi)
- Instrumentasi sistem terdistribusi untuk visibilitas operasional (artikel Amazon Builders' Library)
- Membangun dasbor untuk visibilitas operasional (artikel Amazon Builders' Library)

Manajemen acara

Anda harus memiliki proses manajemen acara untuk menangani gangguan ketika alarm Anda (atau lebih buruk lagi, pelanggan Anda) memberi tahu Anda bahwa ada sesuatu yang tidak beres. Proses

ini harus mencakup melibatkan operator on-call, meningkatkan masalah, dan membuat runbook untuk pendekatan konsisten untuk pemecahan masalah yang membantu menghilangkan kesalahan manusia. Namun, gangguan biasanya tidak terjadi secara terpisah; satu aplikasi dapat memengaruhi beberapa aplikasi lain yang bergantung padanya. Anda dapat mengatasi masalah dengan cepat dengan memahami semua aplikasi yang terkena dampak dan menyatukan operator dari beberapa tim dalam satu panggilan konferensi. Namun, tergantung pada ukuran dan struktur organisasi Anda, proses ini mungkin memerlukan tim operasi terpusat.

Selain menyiapkan proses manajemen acara, Anda harus secara teratur meninjau metrik Anda melalui dasbor. Ulasan reguler membantu Anda memahami pengalaman pelanggan dan tren jangka panjang dalam kinerja aplikasi Anda. Ini membantu Anda mengidentifikasi masalah dan kemacetan sebelum menimbulkan dampak produksi yang signifikan. Meninjau metrik dengan cara yang konsisten dan terstandarisasi memberikan manfaat yang signifikan tetapi membutuhkan pembelian top-down dan investasi waktu.

Tautan berikut memberikan praktik terbaik dalam membangun dasbor dan tinjauan metrik operasional:

- Membangun dasbor untuk visibilitas operasional (artikel Amazon Builders' Library)
- Pendekatan Amazon untuk gagal dengan sukses (AWS re:invent presentasi 2019)

Ketahanan berkelanjutan

Selama Tahap 2: Desain dan implementasi dan Tahap 3: Mengevaluasi dan menguji, Anda memulai aktivitas peninjauan dan pengujian sebelum menerapkan aplikasi Anda ke produksi. Selama tahap operasi, Anda harus terus mengulangi aktivitas tersebut dalam produksi. Anda harus secara berkala meninjau postur ketahanan aplikasi Anda melalui tinjauan AWS Well-Architected Framework, TinjauanKesiapan Operasional (ORR), dan kerangka analisis ketahanan. Ini membantu memastikan bahwa aplikasi Anda tidak hanyut dari garis dasar dan standar yang ditetapkan dan membuat Anda tetap up to date dengan panduan baru atau yang diperbarui. Kegiatan ketahanan berkelanjutan ini membantu Anda menemukan gangguan yang sebelumnya tidak terduga dan membantu Anda menemukan mitigasi baru.

Anda mungkin juga ingin mempertimbangkan menjalankan hari permainan dan eksperimen rekayasa kekacauan dalam produksi setelah Anda berhasil menjalankannya di lingkungan pra-produksi. Hari permainan mensimulasikan peristiwa yang diketahui yang telah Anda bangun mekanisme ketahanan untuk memitigasi. Misalnya, hari permainan mungkin mensimulasikan gangguan layanan AWS Regional dan menerapkan failover Multi-wilayah. Meskipun menerapkan kegiatan ini dapat

memerlukan tingkat upaya yang signifikan, kedua praktik tersebut membantu Anda membangun kepercayaan bahwa sistem Anda tahan terhadap mode kegagalan yang telah Anda rancang untuk bertahan.

Dengan mengoperasikan aplikasi Anda, menghadapi peristiwa operasional, meninjau metrik, dan menguji aplikasi Anda, Anda akan menemukan banyak peluang untuk merespons dan belajar.

Tahap 5: Menanggapi dan belajar

Bagaimana aplikasi Anda merespons peristiwa yang mengganggu memengaruhi keandalannya. Belajar dari pengalaman dan bagaimana aplikasi Anda merespons gangguan di masa lalu juga penting untuk meningkatkan keandalannya.

Tahap Menanggapi dan belajar berfokus pada praktik yang dapat Anda terapkan untuk merespons peristiwa yang mengganggu dalam aplikasi Anda dengan lebih baik. Ini juga mencakup praktik untuk membantu Anda menyaring jumlah pembelajaran maksimum dari pengalaman tim operasi dan insinyur Anda.

Membuat laporan analisis insiden

Ketika suatu insiden terjadi, tindakan pertama adalah mencegah kerugian lebih lanjut bagi pelanggan dan bisnis secepat mungkin. Setelah aplikasi pulih, langkah selanjutnya adalah memahami apa yang terjadi dan mengidentifikasi langkah-langkah untuk mencegah terulangnya kembali. Analisis pasca-insiden ini biasanya ditangkap sebagai laporan yang mendokumentasikan serangkaian peristiwa yang menyebabkan gangguan aplikasi, dan efek gangguan pada aplikasi, pelanggan, dan bisnis. Laporan semacam itu menjadi artefak pembelajaran yang berharga dan harus dibagikan secara luas di seluruh bisnis.

> **Note**
>
> Sangat penting untuk melakukan analisis insiden tanpa menyalahkan apa pun. Asumsikan bahwa semua operator mengambil tindakan terbaik dan paling tepat mengingat informasi yang mereka miliki. Jangan gunakan nama operator atau insinyur dalam laporan. Mengutip kesalahan manusia sebagai alasan gangguan dapat menyebabkan anggota tim dijaga untuk melindungi diri mereka sendiri, yang mengakibatkan penangkapan informasi yang salah atau tidak lengkap.

Laporan analisis insiden yang baik, seperti yang didokumentasikan dalam proses Amazon Correction of Error (COE), mengikuti format standar dan mencoba menangkap, sedetail mungkin, kondisi yang menyebabkan gangguan aplikasi. Laporan tersebut merinci serangkaian peristiwa yang dicap waktu dan menangkap data kuantitatif (seringkali metrik dan tangkapan layar dari dasbor pemantauan) yang menggambarkan keadaan aplikasi yang dapat diukur selama garis waktu. Laporan tersebut

harus menangkap proses pemikiran operator dan insinyur yang mengambil tindakan, dan informasi yang membawa mereka pada kesimpulan mereka. Laporan tersebut juga harus merinci kinerja indikator yang berbeda–misalnya, alarm mana yang dinaikkan, apakah alarm tersebut secara akurat mencerminkan keadaan aplikasi, jeda waktu antara peristiwa dan alarm yang dihasilkan, dan waktu untuk menyelesaikan insiden tersebut. Garis waktu juga menangkap runbook atau otomatisasi yang dimulai dan bagaimana mereka membantu aplikasi mendapatkan kembali status yang berguna. Elemen-elemen timeline ini membantu tim Anda memahami efektivitas respons otomatis dan operator, termasuk seberapa cepat mereka mengatasi masalah dan seberapa efektif mereka dalam mengurangi gangguan tersebut.

Gambaran rinci tentang peristiwa sejarah ini adalah alat pendidikan yang ampuh. Tim harus menyimpan laporan ini di repositori pusat yang tersedia untuk seluruh bisnis sehingga orang lain dapat meninjau peristiwa dan belajar dari mereka. Ini dapat meningkatkan intuisi tim Anda tentang apa yang bisa salah dalam produksi.

Repositori laporan insiden terperinci juga menjadi sumber materi pelatihan bagi operator. Tim dapat menggunakan laporan insiden untuk menginspirasi hari pertandingan atas meja atau langsung, di mana tim diberi informasi yang memutar kembali garis waktu yang diambil dalam laporan. Operator dapat berjalan melalui skenario dengan sebagian informasi dari timeline dan menjelaskan tindakan apa yang akan mereka ambil. Moderator untuk hari permainan kemudian dapat memberikan panduan tentang bagaimana aplikasi merespons berdasarkan tindakan operator. Ini mengembangkan keterampilan pemecahan masalah operator, sehingga mereka dapat lebih mudah mengantisipasi dan memecahkan masalah.

Tim terpusat yang bertanggung jawab atas keandalan aplikasi harus memelihara laporan ini di perpustakaan terpusat yang dapat diakses oleh seluruh organisasi. Tim ini juga harus bertanggung jawab untuk memelihara template laporan dan tim pelatihan tentang cara menyelesaikan laporan analisis insiden. Tim reliabilitas harus meninjau laporan secara berkala untuk mendeteksi tren di seluruh bisnis yang dapat diatasi melalui pustaka perangkat lunak, pola arsitektur, atau perubahan pada proses tim.

Melakukan tinjauan operasional

Seperti yang dibahas dalam Tahap 4: Beroperasi, tinjauan operasional adalah kesempatan untuk meninjau rilis fitur terbaru, insiden, dan metrik operasional. Tinjauan operasional juga merupakan kesempatan untuk berbagi pembelajaran dari rilis fitur dan insiden dengan komunitas teknik yang lebih luas di organisasi Anda. Selama tinjauan operasional, tim meninjau penyebaran fitur yang dibatalkan, insiden yang terjadi, dan bagaimana penanganannya. Ini memberi para insinyur

di seluruh organisasi kesempatan untuk belajar dari pengalaman orang lain dan mengajukan pertanyaan.

Buka ulasan operasional Anda ke komunitas teknik di perusahaan Anda sehingga mereka dapat mempelajari lebih lanjut tentang aplikasi TI yang menjalankan bisnis dan jenis masalah yang dapat mereka hadapi. Mereka akan membawa pengetahuan ini bersama mereka saat mereka merancang, mengimplementasikan, dan menyebarkan aplikasi lain untuk bisnis.

Meninjau kinerja alarm

Alarm, seperti yang dibahas dalam tahap operasi, dapat mengakibatkan peringatan dasbor, tiket dibuat, email dikirim, atau operator dihalaman. Sebuah aplikasi akan memiliki banyak alarm yang dikonfigurasi untuk memantau berbagai aspek operasinya. Seiring waktu, keakuratan dan efektivitas alarm ini harus ditinjau untuk meningkatkan presisi alarm, mengurangi positif palsu, dan mengkonsolidasikan peringatan duplikat.

Alarm presisi

Alarm harus sespesifik mungkin untuk mengurangi jumlah waktu yang harus Anda habiskan untuk menafsirkan atau mendiagnosis gangguan spesifik yang menyebabkan alarm. Ketika alarm dinyalakan sebagai respons terhadap gangguan aplikasi, operator yang menerima dan menanggapi alarm harus terlebih dahulu menafsirkan informasi yang disampaikan alarm. Informasi tersebut mungkin kode kesalahan sederhana yang memetakan ke tindakan seperti prosedur pemulihan, atau mungkin termasuk baris dari log aplikasi yang harus Anda tinjau untuk memahami mengapa alarm dinyalakan. Saat tim Anda belajar mengoperasikan aplikasi dengan lebih efektif, mereka harus memperbaiki alarm ini untuk membuatnya sejelas dan sesingkat mungkin.

Anda tidak dapat mengantisipasi semua kemungkinan gangguan pada suatu aplikasi, sehingga akan selalu ada alarm umum yang mengharuskan operator untuk menganalisis dan mendiagnosis. Tim Anda harus bekerja untuk mengurangi jumlah alarm umum untuk meningkatkan waktu respons dan mengurangi mean time to repair (MTTR). Idealnya, harus ada one-to-one hubungan antara alarm dan respons otomatis atau yang dilakukan manusia.

Positif palsu

Alarm yang tidak memerlukan tindakan dari operator tetapi menghasilkan peringatan karena email, halaman, atau tiket akan diabaikan oleh operator dari waktu ke waktu. Secara berkala, atau sebagai bagian dari analisis insiden, tinjau alarm untuk mengidentifikasi alarm yang sering diabaikan atau

tidak memerlukan tindakan dari operator (positif palsu). Anda harus bekerja untuk menghapus alarm, atau meningkatkan alarm sehingga mengeluarkan peringatan yang dapat ditindaklanjuti kepada operator.

Negatif palsu

Selama insiden, alarm yang dikonfigurasi untuk memperingatkan selama insiden mungkin gagal, mungkin karena peristiwa yang memengaruhi aplikasi dengan cara yang tidak terduga. Sebagai bagian dari analisis insiden, Anda harus meninjau alarm yang seharusnya dinaikkan tetapi tidak. Anda harus bekerja untuk meningkatkan alarm ini sehingga mereka lebih mencerminkan kondisi yang mungkin timbul dari suatu peristiwa. Atau, Anda mungkin harus membuat alarm tambahan yang memetakan ke gangguan yang sama tetapi dipicu oleh gejala gangguan yang berbeda.

Peringatan duplikatif

Gangguan yang mengganggu aplikasi Anda kemungkinan akan menyebabkan banyak gejala dan dapat mengakibatkan beberapa alarm. Secara berkala, atau sebagai bagian dari analisis insiden, Anda harus meninjau alarm dan peringatan yang dikeluarkan. Jika operator menerima peringatan duplikat, buat alarm agregat untuk mengkonsolidasikannya menjadi satu pesan peringatan.

Melakukan tinjauan metrik

Tim Anda harus mengumpulkan metrik operasional tentang aplikasi Anda, seperti jumlah insiden berdasarkan tingkat keparahan per bulan, waktu untuk mendeteksi insiden, waktu untuk mengidentifikasi penyebabnya, waktu untuk memulihkan, dan jumlah tiket yang dibuat, peringatan yang dikirim, dan halaman yang diangkat. Tinjau metrik ini setidaknya setiap bulan untuk memahami beban staf operasional, signal-to-noise rasio yang mereka tangani (misalnya, peringatan informasi versus yang dapat ditindaklanjuti), dan apakah tim meningkatkan kemampuannya untuk mengoperasikan aplikasi di bawah kendali mereka. Gunakan ulasan ini untuk memahami tren dalam aspek terukur dari tim operasi. Mintalah ide dari tim tentang cara meningkatkan metrik ini.

Memberikan pelatihan dan pemberdayaan

Sulit untuk menangkap deskripsi rinci tentang aplikasi dan lingkungannya yang menyebabkan insiden atau perilaku tak terduga. Selain itu, memodelkan ketahanan aplikasi Anda untuk mengantisipasi skenario seperti itu tidak selalu mudah. Organisasi Anda harus berinvestasi dalam materi pelatihan dan pemberdayaan untuk tim operasi dan pengembang Anda untuk berpartisipasi dalam kegiatan

seperti pemodelan ketahanan, analisis insiden, hari permainan, dan eksperimen rekayasa kekacauan. Ini akan meningkatkan kesetiaan laporan yang dihasilkan tim Anda dan pengetahuan yang mereka tangkap. Tim juga akan menjadi lebih siap untuk mengantisipasi kegagalan tanpa bergantung pada kelompok insinyur yang lebih kecil dan lebih berpengalaman yang harus memberikan wawasan mereka melalui tinjauan terjadwal.

Menciptakan basis pengetahuan insiden

Laporan insiden adalah output standar dari analisis insiden. Anda harus menggunakan laporan yang sama atau serupa untuk mendokumentasikan skenario di mana Anda mendeteksi perilaku aplikasi anomali, meskipun aplikasi tidak mengalami gangguan. Gunakan struktur laporan standar yang sama untuk menangkap hasil eksperimen chaos dan hari permainan. Laporan tersebut mewakili snapshot dari aplikasi dan lingkungannya yang menyebabkan insiden atau perilaku yang tidak terduga. Anda harus menyimpan laporan standar ini di repositori pusat yang dapat diakses oleh semua insinyur dalam bisnis.

Tim operasi dan pengembang kemudian dapat mencari basis pengetahuan ini untuk memahami apa yang telah mengganggu aplikasi di masa lalu, jenis skenario apa yang dapat menyebabkan gangguan, dan apa yang mencegah kerusakan aplikasi. Basis pengetahuan ini menjadi akselerator untuk meningkatkan keterampilan tim operasi Anda dan pengembang Anda, dan memungkinkan mereka untuk berbagi pengetahuan dan pengalaman mereka. Selain itu, Anda dapat menggunakan laporan sebagai materi pelatihan atau skenario untuk hari permainan atau eksperimen kekacauan untuk meningkatkan intuisi dan kemampuan tim operasional untuk memecahkan masalah gangguan.

> **Note**
>
> Format laporan standar juga memberi pembaca rasa keakraban dan membantu mereka menemukan informasi yang mereka cari dengan lebih cepat.

Menerapkan ketahanan secara mendalam

Seperti dibahas sebelumnya, organisasi tingkat lanjut akan menerapkan beberapa tanggapan terhadap alarm. Tidak ada jaminan bahwa respons akan efektif, sehingga dengan melapisi tanggapan aplikasi akan lebih siap untuk gagal dengan anggun. Kami menyarankan Anda menerapkan setidaknya dua tanggapan untuk setiap indikator untuk memastikan bahwa respons individu tidak menjadi satu titik kegagalan yang mungkin mengarah pada skenario DR. Lapisan

ini harus dibuat dalam urutan serial, sehingga respons berturut-turut dilakukan hanya jika respons sebelumnya tidak efektif. Anda tidak boleh menjalankan beberapa respons berlapis ke satu alarm. Sebagai gantinya, gunakan alarm yang menunjukkan apakah respons tidak berhasil, dan, jika demikian, memulai respons berlapis berikutnya.

Kesimpulan dan sumber daya

Panduan ini menyajikan siklus hidup yang membantu Anda terus meningkatkan ketahanan aplikasi Anda dengan menerapkan praktik terbaik di lima tahap: Tetapkan tujuan, Desain dan implementasikan, Evaluasi dan uji, Operasikan, dan Tanggapi dan pelajari.

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

AWS layanan:

- AWS Backup
- AWS Elastic Disaster Recovery
- AWS Fault Injection Service (AWS FIS)
- AWS Resilience Hub
- Pengontrol Pemulihan Aplikasi Amazon Route 53
- AWS X-Ray

Posting blog dan artikel:

- Ketersediaan dan Selanjutnya: Memahami dan Meningkatkan Ketahanan Sistem Terdistribusi AWS
- AWS Batas Isolasi Kesalahan
- AWS Dasar-dasar Multi-Wilayah
- Rekayasa Kekacauan di awan
- Terus menilai ketahanan aplikasi dengan dan AWS Resilience HubAWS CodePipeline
- Pemulihan Bencana Aplikasi Lokal ke AWS
- Pilar Keandalan - Kerangka AWS Well-Architected
- Kerangka analisis ketahanan

Kontributor

Kontributor untuk panduan ini meliputi:

- Bruno Emer, Arsitek Solusi Utama, AWS
- Clark Richey, Arsitek Solusi Utama, AWS
- Elaine Harvey, Manajer Umum, Layanan Keandalan, AWS
- Jason Barto, Arsitek Solusi Utama, AWS
- John Formento, Arsitek Solusi Utama, AWS
- Lisi Lewis, Sr. Manajer Pemasaran Produk, AWS
- Michael Haken, Arsitek Solusi Utama, AWS
- Neeraj Kumar, Arsitek Solusi Utama, AWS
- Wangechi Doble, Arsitek Solusi Utama, 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
Publikasi awal	—	Oktober 6, 2023

AWS Glosarium Panduan Preskriptif

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

Nomor

7 Rs

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

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

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

A

ABAC

Lihat kontrol akses berbasis atribut.

layanan abstrak

Lihat layanan terkelola.

ASAM

Lihat atomisitas, konsistensi, isolasi, daya tahan.

migrasi aktif-aktif

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

migrasi aktif-pasif

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

fungsi agregat

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

AI

Lihat kecerdasan buatan.

AIOps

Lihat operasi kecerdasan buatan.

anonimisasi

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

anti-pola

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

kontrol aplikasi

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

portofolio aplikasi

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

kecerdasan buatan (AI)

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

operasi kecerdasan buatan (AIOps)

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

enkripsi asimetris

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

atomisitas, konsistensi, isolasi, daya tahan (ACID)

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

kontrol akses berbasis atribut (ABAC)

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

sumber data otoritatif

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

Zona Ketersediaan

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

AWS Kerangka Adopsi Cloud (AWS CAF)

Kerangka pedoman dan praktik terbaik AWS untuk membantu organisasi mengembangkan rencana yang efisien dan efektif untuk bergerak dengan sukses ke cloud. AWS CAF mengatur panduan ke dalam enam area fokus yang disebut perspektif: bisnis, orang, tata kelola, platform, keamanan, dan operasi. Perspektif bisnis, orang, dan tata kelola fokus pada keterampilan dan proses bisnis; perspektif platform, keamanan, dan operasi fokus pada keterampilan dan proses teknis. Misalnya, perspektif masyarakat menargetkan pemangku kepentingan yang menangani sumber daya manusia (SDM), fungsi kepegawaian, dan manajemen orang. Untuk perspektif ini, AWS CAF memberikan panduan untuk pengembangan, pelatihan, dan komunikasi orang untuk membantu mempersiapkan organisasi untuk adopsi cloud yang sukses. Untuk informasi lebih lanjut, lihat situs web AWS CAF dan whitepaper AWS CAF.

AWS Kerangka Kualifikasi Beban Kerja (AWS WQF)

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

B

bot buruk

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

BCP

Lihat perencanaan kontinuitas bisnis.

grafik perilaku

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

sistem big-endian

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

klasifikasi biner

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

filter mekar

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

deployment biru/hijau

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

bot

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

botnet

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

cabang

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

akses break-glass

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

strategi brownfield

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

cache penyangga

Area memori tempat data yang paling sering diakses disimpan.

kemampuan bisnis

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

perencanaan kelangsungan bisnis (BCP)

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

C

KAFE

Lihat Kerangka Adopsi AWS Cloud.

penyebaran kenari

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

CCoE

Lihat Cloud Center of Excellence.

CDC

Lihat mengubah pengambilan data.

ubah pengambilan data (CDC)

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

rekayasa kekacauan

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

CI/CD

Lihat integrasi berkelanjutan dan pengiriman berkelanjutan.

klasifikasi

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

Enkripsi sisi klien

Enkripsi data secara lokal, sebelum target AWS layanan menerimanya.

Cloud Center of Excellence (CCoE)

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

komputasi cloud

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

model operasi cloud

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

tahap adopsi cloud

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

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

Tahapan ini didefinisikan oleh Stephen Orban dalam posting blog The Journey Toward Cloud-First & the Stages of Adoption di blog Strategi 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. AWS CodeCommit 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, AWS Panorama menawarkan perangkat yang menambahkan CV ke jaringan kamera lokal, dan Amazon SageMaker menyediakan algoritme 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 umumnya 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 AWS Cloud dapat mengurangi risiko privasi, biaya, dan jejak karbon analitik Anda.

perimeter data

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

prapemrosesan data

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

asal data

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

subjek data

Individu yang datanya dikumpulkan dan diproses.

gudang data

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

bahasa definisi database (DDL)

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

bahasa manipulasi basis data (DHTML)

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

DDL

Lihat bahasa definisi database.

ansambel yang dalam

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

pembelajaran mendalam

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

defense-in-depth

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

administrator yang didelegasikan

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

deployment

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

lingkungan pengembangan

Lihat lingkungan.

kontrol detektif

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

pemetaan aliran nilai pengembangan (DVSM)

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

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

kembar digital

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

tabel dimensi

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

musibah

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

pemulihan bencana (DR)

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

DML~

Lihat bahasa manipulasi database.

desain berbasis domain

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

DR

Lihat pemulihan bencana.

deteksi drift

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

DVSM

Lihat pemetaan aliran nilai pengembangan.

E

EDA

Lihat analisis data eksplorasi.

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.

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 pipa CI/CD, 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.

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.

G

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.

strategi greenfield

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

pagar pembatas

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

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

migrasi inkremental

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

Industri 4.0

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

infrastruktur

Semua sumber daya dan aset yang terkandung dalam lingkungan aplikasi.

infrastruktur sebagai kode (IAc)

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

Internet of Things industri (IIoT)

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

inspeksi VPC

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

Internet of Things (IoT)

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

interpretabilitas

Karakteristik model pembelajaran mesin yang menggambarkan sejauh mana manusia dapat memahami bagaimana prediksi model bergantung pada inputnya. Untuk informasi selengkapnya, 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.

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.

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

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

arsitektur microservices

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

Program Percepatan Migrasi (MAP)

AWS Program yang menyediakan dukungan konsultasi, pelatihan, dan layanan untuk membantu organisasi membangun fondasi operasional yang kuat untuk pindah ke cloud, dan untuk membantu mengimbangi biaya awal migrasi. MAP mencakup metodologi migrasi untuk mengeksekusi migrasi lama dengan cara metodis dan seperangkat alat untuk mengotomatisasi dan mempercepat skenario migrasi umum.

migrasi dalam skala

Proses memindahkan sebagian besar portofolio aplikasi ke cloud dalam gelombang, dengan lebih banyak aplikasi bergerak pada tingkat yang lebih cepat di setiap gelombang. Fase

ini menggunakan praktik terbaik dan pelajaran yang dipetik dari fase sebelumnya untuk mengimplementasikan pabrik migrasi tim, alat, dan proses untuk merampingkan migrasi beban kerja melalui otomatisasi dan pengiriman tangkas. Ini adalah fase ketiga dari strategi AWS migrasi.

pabrik migrasi

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

P

batas izin

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

Informasi Identifikasi Pribadi (PII)

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

PII

Lihat informasi yang dapat diidentifikasi secara pribadi.

buku pedoman

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

PLC

Lihat pengontrol logika yang dapat diprogram.

PLM

Lihat manajemen siklus hidup produk.

kebijakan

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

persistensi poliglot

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

penilaian portofolio

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

predikat

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

predikat pushdown

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

kontrol preventif

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

principal

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

Privasi oleh Desain

Pendekatan dalam rekayasa sistem yang memperhitungkan privasi di seluruh proses rekayasa.

zona host pribadi

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

kontrol proaktif

Kontrol keamanan yang dirancang untuk mencegah penyebaran sumber daya yang tidak sesuai. Kontrol ini memindai sumber daya sebelum disediakan. Jika sumber daya tidak sesuai dengan kontrol, maka itu tidak disediakan. Untuk informasi selengkapnya, lihat panduan referensi Kontrol dalam AWS Control Tower dokumentasi dan lihat Kontrol proaktif dalam Menerapkan kontrol keamanan pada AWS.

manajemen siklus hidup produk (PLM)

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

lingkungan produksi

Lihat lingkungan.

pengontrol logika yang dapat diprogram (PLC)

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

pseudonimisasi

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

terbitkan/berlangganan (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).

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 hilangnya 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.

Wilayah

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

regresi

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

rehost

Lihat 7 Rs.

melepaskan

Dalam proses penyebaran, tindakan mempromosikan perubahan pada lingkungan produksi.

memindahkan

Lihat 7 Rs.

memplatform ulang

Lihat 7 Rs.

pembelian kembali

Lihat 7 Rs.

ketahanan

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

kebijakan berbasis sumber daya

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

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

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

kontrol responsif

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

melestarikan

Lihat 7 Rs.

pensiun

Lihat 7 Rs.

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 AWS Management Console 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.

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 AWS layanan yang menerimanya.

kebijakan kontrol layanan (SCP)

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

titik akhir layanan

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

T

tag

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

variabel target

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

daftar tugas

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

lingkungan uji

Lihat lingkungan.

pelatihan

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

gerbang transit

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

alur kerja berbasis batang

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

akses tepercaya

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

penyetelan

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

tim dua pizza

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

U

waswas

Sebuah konsep yang mengacu pada informasi yang tidak tepat, tidak lengkap, atau tidak diketahui yang dapat merusak keandalan model ML prediktif. Ada dua jenis ketidakpastian: ketidakpastian epistemik disebabkan oleh data yang terbatas dan tidak lengkap, sedangkan ketidakpastian aleatorik disebabkan oleh kebisingan dan keacakan yang melekat dalam data. Untuk informasi lebih lanjut, lihat panduan Mengukur ketidakpastian dalam sistem pembelajaran mendalam.

tugas yang tidak terdiferensiasi

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

lingkungan atas

Lihat lingkungan.

V

menyedot debu

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

kendali versi

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

Peering VPC

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

kerentanan

Kelemahan perangkat lunak atau perangkat keras yang membahayakan keamanan sistem.

W

cache hangat

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

data hangat

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

fungsi jendela

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

beban kerja

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

aliran kerja

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

CACING

Lihat menulis sekali, baca banyak.

WQF

Lihat Kerangka Kualifikasi Beban Kerja AWS.

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.

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.