Menerapkan Kerangka Kerja AWS Well-Architected untuk Amazon Neptunus

AWS Bimbingan Preskriptif

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Bimbingan Preskriptif: Menerapkan Kerangka Kerja AWS Well-Architected untuk Amazon Neptunus

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Pengantar 1
Audiens yang dituju 1
Tujuan 2
Keunggulan operasional 3
Mengotomatiskan penerapan menggunakan pendekatan IAc 3
Buat perubahan yang sering, kecil, dan reversibel 4
Mengantisipasi kegagalan 4
Belajar dari semua kegagalan operasional 5
Gunakan kemampuan logging untuk memantau aktivitas yang tidak sah atau anomali 5
Keamanan 7
Menerapkan keamanan data 8
Amankan jaringan Anda 9
Menerapkan otentikasi dan otorisasi 9
Keandalan 11
Memahami kuota layanan Neptunus 11
Memahami pola penyebaran Neptunus 12
Kelola dan skala cluster Neptunus 12
Kelola pencadangan dan peristiwa failover 13
Efisiensi kinerja 15
Memahami pemodelan grafik 15
Optimalkan kueri 16
Cluster ukuran kanan 18
Optimalkan menulis 19
Optimalisasi biaya 21
Memahami pola penggunaan dan layanan yang dibutuhkan 21
Pilih sumber daya dengan memperhatikan biaya 22
Pilih konfigurasi instans Neptunus terbaik untuk beban kerja Anda 23
Penyimpanan dan transfer data ukuran kanan 25
Keberlanjutan 26
Pemilihan Wilayah AWS 26
Konsumsi berdasarkan pola perilaku pengguna 27
Optimalkan pengembangan perangkat lunak dan pola arsitektur 27
Sumber daya 29
Kontributor 30

Riwayat dokumen 31
Glosarium 32
..... 32
A 33
B 36
C 38
D 41
E 45
F 47
G 49
H 50
I 51
L 54
M 55
O 60
P 62
Q 65
R 66
D 69
T 73
U 74
V 75
W 75
Z 76
..... lxxviii

Menerapkan Kerangka Kerja AWS Well-Architected untuk Amazon Neptunus

Amazon Web Services (kontributor)

September 2023 (riwayat dokumen)

Anda dapat membuat solusi berbasis grafik di Amazon Web Services (AWS) dengan menggunakan Amazon Neptunus. Panduan ini memberikan panduan preskriptif untuk menerapkan prinsip AWS Well-Architected Framework saat Anda merencanakan penerapan Neptunus. Menerapkan AWS Kerangka Kerja Well-Architected untuk Amazon Neptunus Analytics mencakup topik yang sama untuk mesin analisis grafik Neptunus.

AWS Well-Architected Framework membantu Anda membangun infrastruktur yang aman, berkinerja tinggi, tangguh, dan efisien untuk berbagai aplikasi dan beban kerja. Ini juga memberikan pendekatan yang konsisten bagi Anda untuk mengevaluasi arsitektur dan menerapkan desain yang dapat diskalakan.

AWS Well-Architected Framework dibangun di sekitar enam pilar berikut:

- Keunggulan operasional
- Keamanan
- Keandalan
- Efisiensi kinerja
- Optimalisasi biaya
- Keberlanjutan

Panduan ini memberikan informasi dari pilar desain Well-Architected Framework dan praktik terbaik, dan pertimbangan yang perlu diingat saat menerapkan Neptunus. AWS

Audiens yang dituju

Panduan ini ditujukan untuk insinyur data, arsitek solusi, dan analis data yang merancang dan mengimplementasikan solusi yang menggunakan grafik. AWS

Tujuan

Panduan ini dapat membantu Anda dan organisasi Anda melakukan hal berikut:

- Pilih dari opsi penerapan dan bahasa kueri yang didukung, berdasarkan kasus penggunaan dan pola kueri Anda.
- Ikuti pola desain AWS Well-Architected yang akan membantu meningkatkan ketahanan dan keamanan.
- Rancang kueri Anda untuk kinerja optimal dan penghematan biaya.
- Pelajari cara menjadi efisien secara operasional saat mengelola klaster Neptunus Anda dalam produksi.

Pilar keunggulan operasional

Pilar keunggulan operasional AWS Well-Architected Framework berfokus pada menjalankan dan memantau sistem, serta terus meningkatkan proses dan prosedur. Ini mencakup kemampuan untuk mendukung pengembangan dan menjalankan beban kerja secara efektif, mendapatkan wawasan tentang operasi mereka, dan terus meningkatkan proses dan prosedur pendukung untuk memberikan nilai bisnis. Anda dapat mengurangi kompleksitas operasional melalui beban kerja penyembuhan diri, yang mendeteksi dan memperbaiki sebagian besar masalah tanpa campur tangan manusia. Anda dapat bekerja menuju tujuan ini dengan mengikuti praktik terbaik yang dijelaskan di bagian ini. Gunakan metrik APIs, dan mekanisme Amazon Neptunus untuk merespons dengan benar saat beban kerja Anda menyimpang dari perilaku yang diharapkan.

Diskusi pilar keunggulan operasional ini berfokus pada bidang-bidang utama berikut:

- Infrastruktur sebagai kode (IAc)
- Manajemen perubahan
- Strategi ketahanan
- Manajemen insiden
- Pelaporan audit untuk kepatuhan
- Pencatatan dan pemantauan

Mengotomatiskan penerapan menggunakan pendekatan IAc

Praktik terbaik untuk mengotomatiskan penerapan di Neptunus menggunakan IAc meliputi:

- Terapkan infrastruktur sebagai kode (IAc) untuk menyebarkan kluster Neptunus bila memungkinkan. Untuk konfigurasi lingkungan yang konsisten, gunakan AWS CloudFormationtemplat, AWS Cloud Development Kit (AWS CDK), atau HashiCorp Terraform untuk membuat semua sumber daya yang diperlukan untuk klaster Anda.
- Mengotomatiskan prosedur operasional Neptunus, seperti mengubah ukuran instance, menambahkan atau menghapus replika baca, atau melakukan failover manual pada tabel global, bila memungkinkan.
- Simpan string koneksi secara eksternal dari klien Anda. Gunakan proses ekstrak, transformasi, dan muat (ETL) untuk memfasilitasi strategi penyebaran biru/hijau, pemulihan bencana (DR), dan

migrasi downtime mendekati nol ke cluster baru. String koneksi dapat disimpan di AWS Secrets Manager, Amazon DynamoDB, atau lokasi mana pun di mana mereka dapat diubah secara dinamis.

- Gunakan tag untuk menambahkan metadata ke sumber daya Neptunus Anda, dan lacak penggunaan berdasarkan tag. Untuk informasi selengkapnya, lihat Menandai Sumber Daya Amazon Neptunus.

Buat perubahan yang sering, kecil, dan reversibel

Rekomendasi berikut berfokus pada perubahan kecil dan reversibel untuk meminimalkan kompleksitas dan mengurangi kemungkinan gangguan beban kerja:

- Simpan templat dan skrip IAC dalam layanan kontrol sumber, seperti GitHub atau. GitLab

  Penting: Jangan menyimpan AWS kredensil dalam kontrol sumber.

- Memerlukan penyebaran IAC untuk menggunakan layanan integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD), seperti atau. AWS CodeDeployAWS CodeBuild Layanan ini mengompilasi, menguji, dan menyebarkan kode di lingkungan non-produksi yang berisi klaster Neptunus sesaat sebelum memengaruhi kluster Amazon Neptunus produksi Anda.
- Uji infrastruktur dan kueri aplikasi di lingkungan yang lebih rendah sebelum Anda menerapkannya ke produksi. Ini akan meminimalkan kemungkinan gangguan dan membantu memastikan mereka bekerja dengan baik dengan beban kerja dan skala Anda.

Mengantisipasi kegagalan

Infrastruktur penyembuhan diri mencontohkan keunggulan operasional dengan mengantisipasi kegagalan dan berusaha menyelesaikan masalah apa pun tanpa intervensi. Rekomendasi berikut membantu Anda mencapai kedewasaan itu dengan Neptunus:

- Buat rencana pemantauan yang menggunakan CloudWatch metrik Amazon untuk memantau penggunaan CPU dan memori instans DB Anda, dan pahami pola penggunaannya. Buat CloudWatch dasbor dan alarm untuk metrik utama dan respons klien Neptunus yang ditemukan di log aplikasi Anda. Untuk informasi selengkapnya tentang indikator pemanfaatan CPU tinggi atau rendah, lihat Menggunakan CloudWatch untuk memantau kinerja instans DB di Neptunus dalam dokumentasi Neptunus.

Jika Anda sering mendapatkan out-of-memory pengecualian pada kueri saat `FreeableMemory` masih rendah, pertimbangkan untuk menggunakan instance dari keluarga X2.

- Atur notifikasi untuk memantau kesehatan cluster Neptunus. Misalnya, `BufferCacheHitRatio` harus selalu tinggi (lebih besar dari 99,9 persen), sedangkan `MainRequestQueuePendingRequests` harus selalu rendah (idealnya 0 tetapi tergantung pada persyaratan dan toleransi latensi Anda).
- Pertimbangkan untuk menggunakan replika baca untuk mencapai ketersediaan tinggi di Neptunus. Anda harus memiliki setidaknya dua replika baca di Availability Zone yang berbeda dari instance penulis untuk memastikan instance selalu tersedia untuk menyajikan kueri baca selama peristiwa failover.
- Secara otomatis menskalakan replika baca berdasarkan metrik pemanfaatan. Untuk informasi selengkapnya, lihat Penskalaan otomatis jumlah replika di kluster DB Amazon Neptunus.
- Tes failover untuk instans DB Anda untuk memahami berapa lama proses untuk kasus penggunaan Anda.
- Jika aplikasi Anda perlu bertahan dari Wilayah AWS pemadaman total, pertimbangkan untuk menggunakan database global sebagai bagian dari rencana DR Anda.

Belajar dari semua kegagalan operasional

Infrastruktur penyembuhan diri adalah upaya jangka panjang yang berkembang dalam iterasi karena masalah langka terjadi atau respons tidak seefektif yang diinginkan. Mengadopsi praktik-praktik berikut mendorong fokus ke arah tujuan itu:

- Mendorong peningkatan dengan belajar dari semua kegagalan.
- Bagikan apa yang dipelajari di seluruh tim dan organisasi. Jika beberapa tim dalam organisasi menggunakan Neptunus, buat ruang obrolan umum atau grup pengguna untuk berbagi pembelajaran dan praktik terbaik.

Gunakan kemampuan logging untuk memantau aktivitas yang tidak sah atau anomali

Untuk mengamati pola kinerja dan aktivitas anomali, simpan log di Amazon CloudWatch Logs. Pertimbangkan praktik terbaik berikut:

- Aktifkan pencatatan kueri lambat. Tinjau log secara teratur dan diagnosa mengapa pertanyaan tertentu lambat. Gunakan titik akhir penjelasan dan profil Neptunus untuk Gremlin, SPARQL, atau OpenCypher untuk mendapatkan wawasan mengapa kueri ini lambat.
- Aktifkan log audit Neptunus, dan tinjau log secara teratur untuk akses atau anomali yang tidak sah.
- Jika Anda menggunakan pencatatan kueri lambat atau pencatatan audit, aktifkan penerbitan ke CloudWatch Log. Ini akan membantu Anda menghindari kehabisan ruang disk pada instance. Instans Neptunus memiliki kapasitas penyimpanan log terbatas dan akan menimpa file log lama ketika ruang log terlampaui. CloudWatch Log mendukung retensi log jangka panjang. Kemampuan pemantauan yang disempurnakan di CloudWatch Log akan meningkatkan kemampuan Anda untuk menanyakan log dan mendiagnosis masalah.
- Untuk memfasilitasi alat analisis yang lebih baik untuk log audit Anda, Anda dapat mengonfigurasi klaster DB Neptunus untuk mempublikasikan data log audit ke grup log di Log. CloudWatch Dengan CloudWatch Log, Anda dapat melakukan analisis real-time dari data log, digunakan CloudWatch untuk membuat alarm dan melihat metrik, dan menggunakan CloudWatch Log untuk menyimpan catatan log Anda dalam penyimpanan yang sangat tahan lama. Untuk informasi selengkapnya, lihat Menerbitkan log Neptunus ke Log Amazon. CloudWatch
- Neptunus mendukung pencatatan tindakan bidang kontrol menggunakan. AWS CloudTrail Untuk informasi selengkapnya, lihat Mencatat Panggilan API Amazon Neptunus dengan. AWS CloudTrail

Pilar keamanan

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menggambarkan ini sebagai keamanan cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang berjalan Layanan AWS di dalamnya AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas AWS keamanan sebagai bagian dari program AWS kepatuhan. Untuk mempelajari tentang program kepatuhan yang berlaku di Amazon Neptune, lihat Layanan AWS dalam Cakupan melalui Program Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh Layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor-faktor lain, termasuk sensitivitas data Anda, persyaratan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang privasi data, silakan lihat Pertanyaan Umum Privasi Data. Untuk informasi tentang perlindungan data di Eropa, lihat Model Tanggung Jawab AWS Bersama dan posting blog GDPR.

Pilar keamanan membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Neptunus. Topik berikut akan menunjukkan kepada Anda cara mengonfigurasi Neptune untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan Layanan AWS yang lain yang membantu Anda memantau dan mengamankan sumber daya Neptunus Anda.

Pilar keamanan mencakup area fokus utama berikut:

- Keamanan data
- Keamanan jaringan
- Autentikasi dan otorisasi

Menerapkan keamanan data

Kebocoran data dan pelanggaran menempatkan pelanggan Anda pada risiko dan dapat menyebabkan dampak negatif yang substansif pada perusahaan Anda. Praktik terbaik berikut membantu melindungi data pelanggan Anda dari paparan yang tidak disengaja dan berbahaya:

- Nama klaster, tag, grup parameter, peran AWS Identity and Access Management (IAM), dan metadata lainnya tidak boleh berisi informasi rahasia atau sensitif karena data tersebut mungkin muncul di log penagihan atau diagnostik.
- URIs atau tautan ke server eksternal yang disimpan sebagai data di Neptunus tidak boleh berisi informasi kredensyal untuk memvalidasi permintaan.
- Instans yang dienkripsi oleh Neptune memberikan lapisan perlindungan data tambahan dengan membantu mengamankan data Anda dari akses yang tidak sah ke penyimpanan yang mendasari. Anda dapat menggunakan enkripsi Neptune untuk meningkatkan perlindungan data aplikasi Anda yang disebarkan di cloud. Anda juga dapat enkripsi Neptunus untuk memenuhi persyaratan kepatuhan untuk data saat istirahat.

  Untuk mengaktifkan enkripsi untuk instans DB Neptunus baru, pilih Ya di bagian Aktifkan enkripsi pada konsol Neptunus (dipilih secara default), atau dengan menyetel properti di. AWS::Neptune::DBCluster::StorageEncrypted AWS CloudFormationJika enkripsi diaktifkan, Neptunus akan menggunakan kunci yang dikelola AWS Amazon Relational Database Service (Amazon RDS) AWS secara default, atau Anda dapat membuat kunci yang dikelola pelanggan. Untuk informasi tentang membuat instans DB Neptunus, lihat Membuat cluster DB Neptunus baru. Untuk detail selengkapnya, lihat Mengenkripsi Sumber Daya Neptunus saat Istirahat. Snapshot otomatis dan manual Anda menggunakan enkripsi yang sama dengan yang Anda pilih untuk cluster Neptunus Anda.
- Saat menggunakan bahasa SPARQL dan OpenCypher, praktikkan validasi input dan teknik parameterisasi yang tepat untuk mencegah injeksi SQL dan bentuk serangan lainnya. Hindari membuat kueri yang menggunakan penggabungan string dengan input yang disediakan pengguna. Gunakan kueri berparameter atau pernyataan yang disiapkan untuk meneruskan parameter input dengan aman ke database grafik. Untuk informasi lebih lanjut, lihat Contoh kueri parameter OpenCypher dan Pertahanan Injeksi SPARQL.
- Untuk bahasa Gremlin, gunakan Varian Bahasa Gremlin alih-alih langsung meneruskan Skrip Gremlin berbasis string untuk menghindari potensi masalah injeksi.

Amankan jaringan Anda

Cluster DB Amazon Neptunus hanya dapat dibuat di cloud pribadi virtual (VPC) aktif. AWS Titik akhir cluster Neptunus DB hanya dapat diakses dalam VPC itu, biasanya dari instance Amazon Elastic Compute Cloud (EC2Amazon) yang berjalan di VPC itu. Anda dapat mengamankan data Neptunus Anda dengan membatasi akses ke VPC tempat cluster DB Neptunus Anda berada. Untuk informasi selengkapnya, lihat Menghubungkan ke grafik Amazon Neptunus Anda.

Untuk melindungi data Anda saat transit, Neptunus menerapkan koneksi SSL melalui HTTPS ke instans atau titik akhir cluster apa pun menggunakan protokol dan cipher yang aman. Neptunus menyediakan sertifikat SSL untuk instans DB Neptunus Anda. Sertifikat SSL Neptunus hanya mendukung nama host titik akhir klaster, titik akhir pembaca, dan titik akhir instance. Jika Anda menggunakan penyeimbang beban atau server proxy (seperti HAProxy), Anda harus menggunakan penghentian SSL dan memiliki sertifikat SSL Anda sendiri di server proxy. Passthrough SSL tidak bekerja karena sertifikat SSL yang disediakan tidak cocok dengan nama host server proxy. Untuk informasi selengkapnya tentang menghubungkan ke titik akhir Neptunus dengan SSL, lihat Menggunakan titik akhir HTTP REST untuk terhubung ke instans DB Neptunus.

Menerapkan otentikasi dan otorisasi

Untuk mengontrol siapa yang dapat melakukan tindakan manajemen Neptunus pada cluster DB Neptunus dan instans DB, gunakan kredensyal IAM. Saat Anda terhubung AWS menggunakan kredensyal IAM, peran IAM Anda harus memiliki kebijakan IAM yang memberikan izin yang diperlukan untuk melakukan operasi manajemen Neptunus. Pastikan Anda mengikuti prinsip hak istimewa paling sedikit, hanya memberikan izin yang diperlukan untuk menyelesaikan tugas. Untuk informasi selengkapnya, lihat Menggunakan berbagai jenis kebijakan IAM untuk mengontrol akses ke Neptunus dan Otentikasi IAM Menggunakan Kredensyal Sementara.

Untuk mengontrol siapa yang dapat terhubung ke cluster Neptunus dan menanyakan data, Anda dapat menggunakan IAM untuk mengautentikasi instans DB Neptunus atau cluster DB Anda. Jika Anda mengaktifkan otentikasi IAM di cluster DB Neptunus, siapa pun yang mengakses cluster DB harus diautentikasi terlebih dahulu. Untuk informasi selengkapnya, lihat Mengaktifkan autentikasi database IAM di Neptunus untuk langkah-langkah mengaktifkan otentikasi IAM.

Ketika autentikasi database IAM diaktifkan, setiap permintaan harus ditandatangani menggunakan Tanda Tangan AWS Versi 4. Untuk memahami cara mengirim permintaan yang ditandatangani ke semua titik akhir Neptunus dengan autentikasi IAM diaktifkan, lihat Menghubungkan dan

Menandatangani dengan Versi Tanda Tangan 4. AWS Banyak perpustakaan dan alat, seperti awscurl, sudah mendukung AWS Signature Version 4.

Untuk berinteraksi dengan orang lain Layanan AWS, Amazon Neptunus menggunakan peran terkait layanan IAM. Peran tertaut layanan adalah tipe IAM role unik yang tertaut langsung ke Neptune. Peran terkait layanan telah ditentukan sebelumnya oleh Neptunus dan mencakup semua izin yang diperlukan layanan untuk memanggil orang lain atas nama Anda. Layanan AWS Untuk informasi selengkapnya, lihat Menggunakan Peran Tertaut Layanan untuk Neptunus.

Pilar keandalan

Pilar keandalan mencakup kemampuan beban kerja untuk menjalankan fungsi yang dimaksudkan dengan benar dan konsisten ketika diharapkan. Ini termasuk kemampuan untuk mengoperasikan dan menguji beban kerja di seluruh siklus hidupnya.

Beban kerja yang andal dimulai dengan desain perangkat lunak dan infrastruktur yang diputuskan sejak awal. Pilihan arsitektur Anda akan memengaruhi perilaku beban kerja Anda di semua pilar Well-Architected. Untuk keandalan, terdapat beberapa pola tertentu yang harus diikuti.

Pilar keandalan berfokus pada bidang-bidang utama berikut:

- Arsitektur beban kerja, termasuk kuota layanan dan pola penerapan
- Manajemen perubahan
- Manajemen kegagalan

Memahami kuota layanan Neptunus

Volume cluster Neptunus dapat tumbuh hingga ukuran maksimum 128 tebibytes (TiB) di semua yang didukung Wilayah AWS kecuali Tiongkok GovCloud dan, di mana kuotanya 64 TiB.

Kuota 128 TiB cukup untuk menyimpan sekitar 200-400 miliar objek dalam grafik. Dalam grafik properti berlabel (LPG), objek adalah simpul, tepi, atau properti pada simpul atau tepi. Dalam grafik Resource Description Framework (RDF), sebuah objek adalah quad.

Untuk cluster Neptunus Tanpa Server apa pun, Anda menetapkan jumlah minimum dan maksimum Unit Kapasitas Neptunus (). NCUs Setiap NCU terdiri dari 2 gibibytes (GiB) memori dan vCPU dan jaringan terkait. Nilai NCU minimum dan maksimum berlaku untuk setiap instance tanpa server di cluster. Nilai NCU maksimum tertinggi yang dapat Anda atur adalah 128,0 NCUs, dan minimum terendah adalah 1,0. NCUs Optimalkan rentang NCU yang paling sesuai untuk aplikasi Anda dengan mengamati CloudWatch metrik Amazon `ServerlessDatabaseCapacity` dan `NCUUtilization` menangkap rentang yang biasa Anda jalankan dan mengkorelasikan perilaku atau biaya yang tidak diinginkan dalam rentang tersebut. Jika Anda menemukan bahwa beban kerja Anda tidak berskala cukup cepat, tingkatkan minimum NCUs untuk menyediakan pemrosesan yang cukup untuk lonjakan awal saat skala.

Masing-masing Akun AWS memiliki kuota untuk setiap Wilayah pada jumlah sumber daya database yang dapat Anda buat. Sumber daya ini termasuk instans DB dan klaster DB. Setelah batas sumber daya tercapai, panggilan tambahan untuk membuat sumber daya itu gagal dengan pengecualian. Beberapa kuota adalah kuota lunak yang dapat ditingkatkan berdasarkan permintaan. Untuk daftar kuota yang dibagikan antara Amazon Neptunus dan Amazon RDS, Amazon Aurora, dan Amazon DocumentDB (dengan kompatibilitas MongoDB), bersama dengan tautan untuk meminta peningkatan kuota bila tersedia, lihat Kuota di Amazon RDS.

Memahami pola penyebaran Neptunus

Dalam klaster DB Neptune, terdapat satu instans DB utama dan hingga 15 replika Neptune. Instans DB primer mendukung operasi baca dan tulis, dan melakukan semua modifikasi data pada volume cluster. Replika Neptunus terhubung ke volume penyimpanan yang sama dengan instans DB utama, dan mereka hanya mendukung operasi baca. Replika Neptune juga dapat memindahkan beban kerja baca dari instans DB utama.

Untuk mencapai ketersediaan tinggi, gunakan replika baca. Memiliki satu atau lebih instance replika baca yang tersedia di Availability Zone yang berbeda dapat meningkatkan ketersediaan karena replika baca berfungsi sebagai target failover untuk instance utama. Jika instance penulis gagal, Neptunus mempromosikan instance replika baca untuk menjadi contoh utama. Ketika ini terjadi, ada gangguan singkat (umumnya kurang dari 30 detik) saat instance yang dipromosikan di-boot ulang, di mana permintaan baca dan tulis yang dibuat ke instance utama gagal dengan pengecualian. Untuk keandalan tertinggi, pertimbangkan dua replika baca di Availability Zone yang berbeda. Jika instance utama di Availability Zone 1 offline, instance di Availability Zone 2 dipromosikan ke primer, tetapi tidak dapat menangani kueri saat itu terjadi. Jadi instance di Availability Zone 3 diperlukan untuk menangani kueri baca selama transisi.

Jika Anda menggunakan Neptunus Tanpa Server, instance pembaca dan penulis di semua Availability Zone akan naik dan turun, secara independen satu sama lain, tergantung pada pemuatan database mereka. Anda dapat mengatur tingkat promosi instance pembaca ke 0 atau 1 sehingga skala naik dan turun bersama dengan kapasitas instance penulis. Ini membuatnya siap untuk mengambil alih beban kerja saat ini kapan saja.

Kelola dan skala cluster Neptunus

Anda dapat menggunakan auto-scaling Neptunus untuk secara otomatis menyesuaikan jumlah replika Neptunus dalam cluster DB untuk memenuhi persyaratan konektivitas dan beban kerja Anda

berdasarkan ambang batas pemanfaatan CPU. Dengan auto-scaling, cluster DB Neptunus Anda dapat menangani peningkatan beban kerja yang tiba-tiba. Ketika beban kerja berkurang, auto-scaling menghapus replika yang tidak perlu sehingga Anda tidak membayar untuk kapasitas yang tidak digunakan. Ketahuilah bahwa startup instance baru dapat memakan waktu selama 15 menit, jadi auto-scaling saja bukanlah solusi yang cukup untuk perubahan permintaan yang cepat.

Anda dapat menggunakan auto-scaling hanya dengan cluster DB Neptunus yang sudah memiliki satu instance penulis utama dan setidaknya satu instance read-replica (lihat Amazon Neptunus DB Clusters and Instances). Selain itu, semua instance read-replica di cluster harus dalam keadaan tersedia. Jika ada replika baca dalam keadaan selain tersedia, auto-scaling Neptunus tidak melakukan apa pun sampai setiap replika baca di cluster tersedia.

Jika Anda mengalami perubahan permintaan yang cepat, pertimbangkan untuk menggunakan instance tanpa server. Instans tanpa server dapat menskalakan secara vertikal selama periode pendek sementara auto-scaling menskalakan secara horizontal selama periode yang lebih lama. Konfigurasi ini memberikan skalabilitas optimal karena instans tanpa server menskalakan secara vertikal sementara auto-scaling membuat instance replika baca baru untuk menangani beban kerja di luar kapasitas maksimum instans tanpa server tunggal. Untuk informasi selengkapnya tentang penskalaan kapasitas Amazon Neptunus Tanpa Server, lihat Penskalaan kapasitas di kluster DB Tanpa Server Neptunus.

Jika penskalaan Anda perlu berubah pada waktu yang dapat diprediksi, Anda dapat menjadwalkan perubahan pada instans minimum, instans maksimum, dan ambang batas untuk menangani kebutuhan pergeseran tersebut dengan lebih baik. Ingatlah untuk menjadwalkan acara penskalaan setidaknya 15 menit sebelumnya untuk memungkinkan contoh-contoh tersebut online saat diperlukan.

Anda mengelola konfigurasi database Anda di Amazon Neptunus dengan menggunakan parameter dalam grup parameter. Grup parameter bertindak sebagai wadah untuk nilai konfigurasi mesin yang diterapkan ke satu atau lebih instance DB. Saat memodifikasi parameter cluster dalam kelompok parameter, pahami perbedaan antara parameter statis dan dinamis, dan bagaimana dan kapan parameter tersebut diterapkan. Gunakan titik akhir status untuk melihat konfigurasi yang diterapkan saat ini.

Kelola pencadangan dan peristiwa failover

Neptunus mencadangkan volume cluster Anda secara otomatis, dan menyimpan data yang dicadangkan selama periode retensi cadangan. Cadangan Neptune bersifat terus-menerus dan

bertahap, sehingga Anda dapat dengan cepat memulihkan ke titik mana pun dalam periode penyimpanan cadangan. Anda dapat menentukan periode retensi cadangan 1-35 hari saat membuat atau memodifikasi klaster DB.

Untuk menyimpan cadangan di luar periode retensi cadangan, Anda juga dapat mengambil snapshot data dalam volume cluster Anda. Menyimpan snapshot menimbulkan biaya penyimpanan standar untuk Neptune.

Saat Anda membuat snapshot Amazon Neptunus dari kluster DB, Neptunus membuat snapshot volume penyimpanan cluster, mencadangkan semua datanya, bukan hanya instance individual. Anda nantinya dapat membuat cluster DB baru dengan memulihkan dari snapshot cluster DB ini. Saat Anda memulihkan cluster DB, Anda memberikan nama snapshot cluster DB untuk dipulihkan, dan kemudian Anda memberikan nama untuk cluster DB baru yang dibuat oleh pemulihan.

Uji bagaimana sistem Anda merespons peristiwa failover. Gunakan Neptunus API untuk memaksa peristiwa failover. Reboot dengan failover bermanfaat ketika Anda ingin mensimulasikan kegagalan instans DB untuk pengujian atau untuk memulihkan operasi ke Availability Zone asli setelah failover terjadi. Untuk informasi selengkapnya, lihat Mengonfigurasi dan mengelola penerapan Multi-AZ. Saat Anda me-reboot instance DB writer, itu gagal ke replika siaga. Mereboot replika Neptune tidak memulai failover.

Rancang klien Anda untuk keandalan. Uji perilaku mereka selama peristiwa failover. Terapkan logika coba lagi di klien Anda dengan logika backoff eksponensial. Contoh kode yang menerapkan logika ini dapat ditemukan dalam contoh AWS Lambda fungsi untuk Amazon Neptunus.

Pertimbangkan untuk menggunakan AWS Backupjika Anda memiliki serangkaian persyaratan cadangan umum yang Anda terapkan di beberapa mesin database.

Pilar efisiensi performa

Pilar efisiensi kinerja dari AWS Well-Architected Framework berfokus pada cara mengoptimalkan kinerja sambil menelan atau menanyakan data. Optimalisasi kinerja adalah proses tambahan dan berkelanjutan dari hal-hal berikut:

- Mengonfirmasi persyaratan bisnis
- Mengukur kinerja beban kerja
- Mengidentifikasi komponen yang berkinerja buruk
- Menyetel komponen untuk memenuhi kebutuhan bisnis Anda

Pilar efisiensi kinerja menyediakan pedoman khusus kasus penggunaan yang dapat membantu mengidentifikasi model data grafik dan bahasa kueri yang tepat untuk digunakan. Ini juga mencakup praktik terbaik untuk diikuti saat menelan data ke dalam dan mengkonsumsi data dari Amazon Neptunus.

Pilar efisiensi kinerja berfokus pada bidang-bidang utama berikut:

- Pemodelan grafik
- Pengoptimalan kueri
- Ukuran kanan cluster
- Tulis optimasi

Memahami pemodelan grafik

Memahami perbedaan antara model Grafik Properti Berlabel (LPG) dan Resource Description Framework (RDF). Dalam kebanyakan kasus, ini adalah masalah preferensi. Namun, ada beberapa kasus penggunaan, di mana satu model lebih cocok daripada yang lain. Jika Anda memerlukan pengetahuan tentang jalur yang menghubungkan dua node dalam grafik Anda, pilih LPG. Jika Anda ingin menggabungkan data di seluruh cluster Neptunus atau penyimpanan tiga grafik lainnya, pilih RDF.

Jika Anda membangun aplikasi perangkat lunak sebagai layanan (SaaS) atau aplikasi yang membutuhkan multi-tenancy, pertimbangkan untuk memasukkan pemisahan logis penyewa dalam model data Anda alih-alih memiliki satu penyewa untuk setiap cluster. Untuk mencapai jenis desain

tersebut, Anda dapat menggunakan grafik bernama SPARQL dan strategi pelabelan, seperti menambahkan pengidentifikasi pelanggan ke label atau menambahkan pasangan nilai kunci properti yang mewakili pengidentifikasi penyewa. Pastikan layer klien Anda menyuntikkan nilai-nilai ini untuk menjaga pemisahan logis itu.

Kinerja kueri Anda tergantung pada jumlah objek grafik (node, tepi, properti) yang perlu dievaluasi dalam pemrosesan kueri Anda. Dengan demikian, model grafik dapat berdampak signifikan pada kinerja aplikasi Anda. Gunakan label granular bila memungkinkan, dan simpan hanya properti yang Anda butuhkan untuk mencapai penentuan jalur atau penyaringan. Untuk mencapai kinerja yang lebih tinggi, pertimbangkan untuk menghitung bagian grafik Anda, seperti membuat simpul ringkasan atau lebih banyak tepi langsung yang menghubungkan jalur umum.

Cobalah untuk menghindari navigasi melintasi node yang memiliki jumlah tepi yang sangat tinggi dengan label yang sama. Node seperti itu sering memiliki ribuan tepi (di mana sebagian besar node memiliki jumlah tepi dalam puluhan). Hasilnya adalah komputasi dan kompleksitas data yang jauh lebih tinggi. Node ini mungkin tidak bermasalah dalam beberapa pola kueri, tetapi kami menyarankan pemodelan data Anda secara berbeda untuk menghindarinya, terutama jika Anda akan menavigasi di seluruh node sebagai langkah perantara. Anda dapat menggunakan log kueri lambat untuk membantu mengidentifikasi kueri yang menavigasi di seluruh node ini. Anda mungkin akan mengamati latensi dan metrik akses data yang jauh lebih tinggi daripada pola kueri rata-rata Anda, terutama jika Anda menggunakan mode debug.

Gunakan simpul deterministik IDs untuk node dan tepi jika kasus penggunaan Anda mendukungnya alih-alih menggunakan Neptunus untuk menetapkan nilai GUID acak. IDs Mengakses node dengan ID adalah metode yang paling efisien.

Optimalkan kueri

Bahasa OpenCypher dan Gremlin dapat digunakan secara bergantian pada model LPG. Jika kinerja menjadi perhatian utama, pertimbangkan untuk menggunakan dua bahasa secara bergantian karena yang satu mungkin berkinerja lebih baik daripada yang lain untuk pola kueri tertentu.

Neptunus sedang dalam proses konversi ke mesin kueri alternatifnya (DFE). OpenCypher berjalan pada DFE saja, tetapi kueri Gremlin dan SPARQL dapat diatur secara opsional untuk berjalan pada DFE dengan menggunakan anotasi kueri. Pertimbangkan untuk menguji kueri Anda dengan DFE diaktifkan dan membandingkan kinerja pola kueri Anda saat tidak menggunakan DFE.

Neptunus dioptimalkan untuk kueri tipe transaksional yang dimulai pada satu node atau set node dan menyebar dari sana, bukan kueri analitis yang mengevaluasi seluruh grafik. Untuk beban kerja

kueri analitis Anda, pertimbangkan untuk menggunakan AWS SDK for Pandas atau menggunakan neptune-export yang digabungkan dengan atau Amazon EMR. AWS Glue

Untuk mengidentifikasi inefisiensi dan kemacetan dalam model dan kueri Anda, gunakan dan `explain` APIs untuk setiap bahasa kueri untuk mendapatkan penjelasan rinci tentang rencana kueri `profile` dan metrik kueri. Untuk informasi lebih lanjut, lihat profil Gremlin, OpenCypher menjelaskan, dan SPARQL menjelaskan.

Pahami pola kueri Anda. Jika jumlah tepi yang berbeda dalam grafik menjadi besar, strategi akses Neptunus default dapat menjadi tidak efisien. Pertanyaan berikut mungkin menjadi sangat tidak efisien:

- Kueri yang menavigasi mundur melintasi tepi saat tidak ada label tepi yang diberikan.
- Klausa yang menggunakan pola yang sama ini secara internal, seperti `.both()` di Gremlin, atau klausa yang menjatuhkan node dalam bahasa apa pun (yang mengharuskan menjatuhkan tepi masuk tanpa sepengetahuan label).
- Kueri yang mengakses nilai properti tanpa menentukan label properti. Pertanyaan ini mungkin menjadi sangat tidak efisien. Jika ini cocok dengan pola penggunaan Anda, pertimbangkan untuk mengaktifkan indeks OSGP (objek, subjek, grafik, predikat).

Gunakan pencatatan kueri lambat untuk mengidentifikasi kueri lambat. Kueri yang lambat dapat disebabkan oleh rencana kueri yang tidak dioptimalkan atau sejumlah besar pencarian indeks yang tidak perlu, yang dapat meningkatkan biaya I/O. Neptunus menjelaskan dan membuat profil titik akhir untuk Gremlin, SPARQL, atau OpenCypher dapat membantu Anda memahami mengapa kueri ini lambat. Penyebabnya mungkin termasuk yang berikut:

- Node dengan jumlah tepi yang sangat tinggi dibandingkan dengan simpul rata-rata dalam grafik (misalnya, ribuan dibandingkan dengan puluhan) dapat menambah kompleksitas komputasi dan karenanya latensi yang lebih lama dan konsumsi sumber daya yang lebih besar. Tentukan apakah node ini dimodelkan dengan benar, atau apakah pola akses dapat ditingkatkan untuk mengurangi jumlah tepi yang harus dilalui.
- Kueri yang tidak dioptimalkan akan berisi peringatan bahwa langkah-langkah tertentu tidak dioptimalkan. Menulis ulang kueri ini untuk menggunakan langkah-langkah yang dioptimalkan dapat meningkatkan kinerja.
- Filter redundan dapat menyebabkan pencarian indeks yang tidak perlu. Demikian juga, pola redundan dapat menyebabkan pencarian indeks duplikat yang dapat dioptimalkan dengan meningkatkan kueri (lihat `Index Operations - Duplication ratio` di output profil).

- Beberapa bahasa seperti Gremlin tidak memiliki nilai numerik yang diketik dengan kuat, dan mereka menggunakan promosi tipe sebagai gantinya. Misalnya, jika nilainya 55, Neptunus mencari nilai yang merupakan bilangan bulat, panjang, float, dan tipe numerik lainnya yang setara dengan 55. Ini menghasilkan operasi tambahan. Jika Anda tahu jenis Anda cocok sebelumnya, Anda dapat menghindari ini dengan menggunakan petunjuk kueri.
- Model grafik Anda dapat sangat memengaruhi kinerja. Pertimbangkan untuk mengurangi jumlah objek yang perlu dievaluasi dengan menggunakan label yang lebih terperinci atau dengan menghitung pintasan ke jalur linier multi-hop.

Jika optimasi kueri saja tidak memungkinkan Anda untuk mencapai persyaratan kinerja Anda, pertimbangkan untuk menggunakan berbagai teknik caching dengan Neptunus untuk mencapai persyaratan tersebut.

Cluster ukuran kanan

Ukur ukuran cluster Anda untuk persyaratan konkurensi dan throughput Anda. Jumlah query bersamaan yang dapat ditangani oleh setiap instance di cluster sama dengan dua kali jumlah virtual CPUs (vCPUs) pada instance itu. Kueri tambahan yang muncul saat semua thread pekerja ditempati dimasukkan ke dalam antrean sisi server. Kueri tersebut ditangani berdasarkan first-in-first-out (FIFO) saat thread pekerja tersedia. CloudWatch Metrik `MainRequestQueuePendingRequests` Amazon menunjukkan kedalaman antrian saat ini untuk setiap instance. Jika nilai ini sering di atas nol, pertimbangkan untuk memilih instance dengan lebih banyak vCPUs. Jika kedalaman antrian melebihi 8.192, Neptunus akan mengembalikan kesalahan. `ThrottlingException`

Sekitar 65 persen RAM untuk setiap instance dicadangkan untuk cache buffer. Cache buffer menyimpan kumpulan data yang berfungsi (bukan seluruh grafik; hanya data yang sedang ditanyakan). Untuk menentukan persentase data yang diambil dari cache buffer alih-alih penyimpanan, pantau metrik. CloudWatch `BufferCacheHitRatio` Jika metrik ini sering turun di bawah 99,9 persen, pertimbangkan untuk mencoba instance dengan lebih banyak memori untuk menentukan apakah itu mengurangi latensi dan biaya I/O Anda.

Replika baca tidak harus berukuran sama dengan instance penulis Anda. Namun, beban kerja tulis yang berat dapat menyebabkan replika yang lebih kecil tertinggal dan reboot karena mereka tidak dapat mengikuti replikasi. Oleh karena itu, kami merekomendasikan membuat replika sama dengan atau lebih besar dari contoh penulis.

Saat menggunakan auto-scaling untuk replika baca Anda, ingatlah bahwa mungkin perlu waktu hingga 15 menit untuk menghadirkan replika baca baru secara online. Ketika lalu lintas klien meningkat dengan cepat tetapi dapat diprediksi, pertimbangkan untuk menggunakan penskalaan terjadwal untuk menetapkan jumlah minimum replika baca yang lebih tinggi untuk memperhitungkan waktu inisialisasi tersebut.

Instans tanpa server mendukung beberapa kasus penggunaan dan beban kerja yang berbeda. Pertimbangkan tanpa server atas instance yang disediakan untuk skenario berikut:

- Beban kerja Anda sering berfluktuasi sepanjang hari.
- Anda membuat aplikasi baru dan Anda tidak yakin berapa ukuran beban kerja nantinya.
- Anda sedang melakukan pengembangan dan pengujian.

Penting untuk dicatat bahwa instans tanpa server lebih mahal daripada instance yang disediakan setara dengan satu dolar per GB basis RAM. Setiap instance tanpa server terdiri dari 2 GB RAM bersama dengan vCPU dan jaringan terkait. Lakukan analisis biaya di antara opsi Anda untuk menghindari tagihan kejutan. Secara umum, Anda akan mencapai penghematan biaya dengan tanpa server hanya ketika beban kerja Anda sangat berat hanya beberapa jam setiap hari dan hampir nol sepanjang hari atau jika beban kerja Anda berfluktuasi secara signifikan sepanjang hari.

Optimalkan menulis

Untuk mengoptimalkan penulisan, pertimbangkan hal berikut:

- Pemuat Massal Neptunus adalah cara optimal untuk memuat database Anda atau menambahkan ke data yang ada. Pemuat Neptunus tidak transaksional dan tidak dapat menghapus data, jadi jangan gunakan jika ini adalah persyaratan Anda.
- Pembaruan transaksional dapat dilakukan dengan menggunakan bahasa kueri yang didukung. Untuk mengoptimalkan operasi tulis I/O, tulis data dalam batch 50-100 objek per komit. Objek adalah node, edge, atau properti pada node atau edge di LPG, atau triple store atau quad di RDF..
- Semua operasi penulisan Neptunus adalah ulir tunggal untuk setiap koneksi. Saat mengirim sejumlah besar data ke Neptunus, pertimbangkan untuk memiliki beberapa koneksi paralel yang masing-masing merupakan data penulisan. Saat Anda memilih instance yang disediakan Neptunus, ukuran instance dikaitkan dengan sejumlah v. CPUs Neptunus membuat dua utas database untuk setiap vCPU pada instance, jadi mulailah dengan dua kali jumlah CPUs v saat

menguji paralelisasi optimal. Instans tanpa server menskalakan jumlah v dengan CPUs kecepatan kira-kira satu untuk setiap 4. NCUs

- Rencanakan dan tangani secara efisien ConcurrentModificationExceptionsselama semua proses penulisan, bahkan jika hanya satu koneksi yang menulis data kapan saja. Rancang klien Anda untuk keandalan saat `ConcurrentModificationExceptions` terjadi.
- Jika Anda ingin menghapus semua data Anda, pertimbangkan untuk menggunakan API reset cepat alih-alih mengeluarkan kueri penghapusan bersamaan. Yang terakhir akan memakan waktu lebih lama dan menimbulkan biaya I/O yang besar dibandingkan dengan yang pertama.
- Jika Anda ingin menghapus sebagian besar data Anda, pertimbangkan untuk mengekspor data yang ingin Anda simpan dengan menggunakan neptune-export untuk memuat data ke dalam klaster baru. Kemudian hapus cluster asli.

Pilar optimasi biaya

Pilar optimasi biaya dari AWS Well-Architected Framework berfokus pada menghindari biaya yang tidak perlu. Rekomendasi berikut dapat membantu Anda memenuhi prinsip desain pengoptimalan biaya dan praktik terbaik arsitektur untuk Amazon Neptunus.

Pilar optimasi biaya berfokus pada bidang-bidang utama berikut:

- Memahami pengeluaran dari waktu ke waktu dan mengendalikan alokasi dana
- Memilih sumber daya dari jenis dan kuantitas yang tepat
- Penskalaan untuk memenuhi kebutuhan bisnis tanpa pengeluaran berlebihan

Memahami pola penggunaan dan layanan yang dibutuhkan

Neptunus cocok untuk beban kerja Anda jika model data Anda memiliki struktur grafik yang dapat dilihat, dan kueri Anda perlu mengeksplorasi hubungan dan melintasi beberapa lompatan. Database grafik tidak cocok untuk pola berikut:

- Terutama kueri single-hop (pertimbangkan apakah data Anda mungkin lebih baik direpresentasikan sebagai atribut objek)
- Data JSON atau BLOB disimpan sebagai properti
- Kueri yang digabungkan di seluruh kumpulan data, seperti menghitung jumlah properti numerik di sejumlah besar node

Pertimbangkan apakah menggunakan beberapa database yang dibuat khusus bersama-sama untuk pola akses tertentu dapat memenuhi semua kebutuhan Anda. Sebagai contoh:

- API yang membutuhkan navigasi grafik kompleks yang lebih jarang di samping pengambilan properti yang sangat bersamaan untuk satu node mungkin paling baik disajikan dengan menggunakan satu atau lebih Neptune, DynamoDB, atau Amazon DocumentDB.
- Database relasional dapat hidup berdampingan dengan Neptunus untuk mempertahankan fungsionalitas Anda yang ada, tetapi gunakan Neptunus hanya untuk traversal multiple-hop yang tidak berkinerja dan skala yang baik dalam database relasional.

Memahami biaya yang terkait dengan layanan yang berinteraksi dengan dan melengkapi Neptunus, termasuk yang berikut:

- Biaya penyimpanan Amazon Simple Storage Service (Amazon S3) untuk file data yang dimuat secara massal ke Neptunus
- Fungsi Lambda yang digunakan untuk menyisipkan atau meningkatkan kueri, membaca kueri, dan pemrosesan aliran Neptunus
- Lapisan API yang dibangun di Neptunus untuk berinteraksi dengan aplikasi klien (alih-alih memiliki koneksi langsung ke database) di atau AWS AppSync
- AWS Glue pekerjaan yang digunakan untuk mentransfer data ke dan dari Neptunus
- Amazon Kinesis atau Amazon Managed Streaming for Apache Kafka (Amazon MSK) instans menerima data streaming untuk konsumsi hampir real-time ke Neptunus.
- AWS Database Migration Service untuk migrasi data relasional ke Neptunus
- Biaya Amazon SageMaker Runtime untuk notebook Jupyter dan model pembelajaran mesin perpustakaan grafik mendalam

Pilih sumber daya dengan memperhatikan biaya

Harga Neptunus didasarkan pada biaya instans per jam (atau Unit Komputasi Neptunus yang digunakan untuk tanpa server), I/O data, dan penggunaan penyimpanan. Contoh merupakan, rata-rata, 85 persen dari keseluruhan biaya, sehingga ukuran yang tepat dapat memiliki implikasi biaya yang signifikan. Cara terbaik untuk mengukur instans yang tepat adalah dengan menguji kinerja aplikasi pada berbagai instance dan membandingkan faktor-faktor berikut:

- Apakah `MainRequestQueuePendingRequests` CloudWatch metrik tetap pada angka rendah secara konsisten mendekati nol?
- Apakah `BufferCacheHitRatio` CloudWatch metrik tetap pada atau di atas 99,9 persen sebagian besar waktu?
- Berapa kurva biaya dan kinerja untuk biaya misalnya dan untuk biaya I/O data terkait? Biaya pembacaan data mungkin meningkat secara signifikan dengan instance berukuran kecil yang memerlukan pertukaran cache buffer yang sering dengan penyimpanan. `BufferCacheHitRatio`akan sering jatuh dalam skenario ini.

Biaya instans diskalakan secara linier dengan ukuran dalam keluarga instance yang sama. Biaya per jam `db.r6i.2xlarge` instance adalah dua kali lipat dari `db.r6i.xlarge` instance dan juga memiliki alokasi sumber daya dua kali lipat. `db.r6i.24xlarge`Instans adalah 24 kali biaya per jam dari `db.r6i.xlarge` instans.

Perkirakan jumlah kueri bersamaan yang harus Anda dukung. Anda dapat memiliki antara nol dan lima belas replika baca untuk memproses kueri hanya-baca. Jika persyaratan Anda bervariasi menurut waktu hari, minggu, atau bulan, Anda dapat menggunakan beberapa instance yang lebih kecil untuk menskalakan jadwal. Setiap vCPU pada sebuah instance menyediakan dua thread untuk menangani kueri bersamaan. Tiga replika `db.r6i.xlarge` baca, dengan masing-masing 4 vCPU, dapat menangani 24 kueri bersamaan..

Jika volume lalu lintas Anda diukur dalam kueri per detik (QPS), Anda harus bereksperimen untuk menentukan latensi rata-rata kueri Anda. Jumlah kueri per detik yang dapat didukung oleh cluster Neptunus sama dengan. `vCPU × 2 × (1 second/average query latency)` Misalnya, jika Anda memiliki 4 vCPU dan latensi kueri 100 milidetik (0,1 detik),. `QPS = 4 × 2 × (1s/0.1s) = 80 queries per second`

Instans yang disediakan lebih murah daripada tanpa server untuk beban kerja yang berkelanjutan, stabil, dan dapat diprediksi. Tanpa server memberikan peluang untuk mengoptimalkan biaya ketika Anda memiliki beban kerja yang membutuhkan penggunaan sangat tinggi hanya beberapa jam per hari (misalnya,`db.r6i.4xlarge`) dan kemudian hampir tidak ada lalu lintas untuk sisa hari itu (misalnya, 1 Unit Komputasi Neptunus). Instans tanpa server yang ditingkatkan selama beberapa jam dan kemudian mundur akan lebih murah daripada menggunakan instance yang disediakan sepanjang `db.r6i.4xlarge` hari.

Pilih konfigurasi instans Neptunus terbaik untuk beban kerja Anda

Untuk eksperimen tingkat pemula dengan Neptunus, Anda dapat menggunakan AWS Tingkat Gratis. 750 jam gratis `db.t3.medium` dan penggunaan `db.t4g.medium` instance sudah cukup bagi Anda untuk mendapatkan pemahaman yang baik tentang Neptunus dalam skala rendah. Cluster Anda akan tetap ada setelah periode uji coba gratis berakhir, meskipun Anda akan dikenakan biaya untuk penggunaan selanjutnya sejak saat itu.

`db.t4g.medium`Instance `db.t3.medium` dan baik untuk lingkungan pengembangan berbiaya rendah, tetapi perlu diketahui bahwa mereka memiliki rasio RAM terhadap vCPU yang lebih kecil (2:1) daripada instance keluarga R (8:1) atau instance keluarga X (16:1). Profil kinerja mungkin

berbeda dari kelas-kelas tersebut, terutama mengenai `OutOfMemoryExceptions` dan ketika kueri menavigasi sebagian besar grafik. Untuk menentukan apakah kondisi terakhir mungkin terpengaruh, periksa `BufferCacheHitRatio` CloudWatch metrik.

Kami sangat menyarankan untuk tidak melakukan pengujian kinerja atau beban apa pun dengan instance keluarga T karena Anda mungkin mengalami hasil yang tidak konsisten yang tidak menunjukkan lingkungan produksi.

Instans yang disediakan memberi Anda kombinasi biaya dan kinerja terbaik ketika beban kerja Anda cukup stabil dan dapat diprediksi. Pilih ukuran instance berdasarkan konkurensi permintaan yang diperlukan dan kompleksitas kueri. Konkurensi yang lebih tinggi membutuhkan lebih banyak vCPUs. Kompleksitas kueri yang lebih tinggi membutuhkan lebih banyak RAM. Gunakan `MainRequestQueuePendingRequests` CloudWatch metrik untuk menentukan dampak dari yang pertama (lebih besar dari nol mewakili lebih banyak permintaan bersamaan daripada yang dapat ditangani). Gunakan `BufferCacheHitRatio` CloudWatch metrik untuk menentukan dampak yang terakhir. Rasio yang sering turun lebih rendah dari 99,9 persen menunjukkan bahwa tidak ada cukup RAM untuk menampung bagian kerja grafik yang sedang dievaluasi, yang menghasilkan pertukaran cache yang lebih sering. Jika keluarga R instance memberikan konkurensi yang cukup tetapi tidak cukup RAM, pertimbangkan untuk mencoba keluarga instance X.

Kasus penggunaan ideal untuk instance tanpa server dijelaskan dalam dokumentasi Neptunus. Jika Anda tidak yakin apakah provisioned atau serverless adalah yang terbaik untuk Anda, dan biaya adalah perhatian utama Anda, uji beban kerja Anda di tanpa server untuk menentukan jumlah yang NCUs digunakan dan bandingkan biaya provisioned () dengan serverless (). `N hours × hourly provisioned cost` `sum of NCUs × hourly cost per NCU` Jika Anda tidak yakin tentang instance penyediaan berukuran setara, satu NCU setara dengan sekitar 2 GB RAM dan vCPU dan jaringan terkait. Jika instans yang Anda sediakan berasal dari keluarga R6i, rasionya adalah 1 vCPU per 8 GB RAM, atau 4 NCUs, bersama dengan jaringan terkait.

Saat menggunakan tanpa server untuk instance primer dan replika, ingatlah bahwa replika baca di tingkat promosi 0 dan 1 akan menskalakan sesuai dengan instance penulis sehingga diskalakan dengan benar jika peristiwa failover terjadi. NCUs Tetapkan batas NCU Anda untuk instans ini berdasarkan instans Anda—penulis atau pembaca—yang menerima lalu lintas terbanyak.

Di lingkungan di mana cluster tidak diperlukan 24 jam per hari, 7 hari seminggu, pertimbangkan untuk menulis skrip yang akan mematikan instance Neptunus saat tidak digunakan dan memulainya lagi sebelum digunakan. Instans Neptunus akan dimulai ulang secara otomatis setiap 7 hari untuk memastikan pembaruan pemeliharaan yang diperlukan diterapkan. Jika Anda berniat untuk

meninggalkan instance untuk jangka waktu yang lama, gunakan skrip mingguan untuk mematikannya lagi.

Penyimpanan dan transfer data ukuran kanan

Kueri yang lebih efisien (misalnya, kueri yang perlu menyentuh lebih sedikit node, tepi, dan properti dalam grafik) memerlukan lebih sedikit transfer I/O dan berpotensi dapat menggunakan instance yang lebih kecil karena lebih sedikit cache buffer yang diperlukan. Gunakan profil atau jelaskan titik akhir untuk bahasa kueri Anda untuk mengoptimalkan kueri Anda, dan pertimbangkan untuk mengoptimalkan model grafik Anda untuk kinerja kueri Anda.

Neptunus menggunakan pengkodean kamus pada string besar, dan kamus itu dioptimalkan untuk kinerja, bukan efisiensi. Jika Anda memiliki string besar BLOBs, JSON, atau sering berubah untuk properti, pertimbangkan untuk menyimpannya di luar Neptunus di Amazon S3, Amazon DynamoDB, atau Amazon DocumentDB, dan simpan hanya referensi dalam node Neptunus.

Dalam beberapa kasus, memilih ukuran instans yang lebih besar bisa lebih murah. Jika biaya I/O Anda sangat tinggi karena rendah`BufferCacheHitRatio`, mungkin cache buffer yang lebih besar akan secara signifikan mengurangi biaya itu. Itu karena semua data akan masuk ke dalam cache alih-alih sering ditukar dari penyimpanan dan menimbulkan kecepatan transfer I/O.

Neptunus menggunakan kloning copy-on-write. Saat mengkloning untuk membagi grafik menjadi beberapa pecahan, mungkin lebih efisien untuk tidak menghapus data yang tidak diinginkan pada kloning kloning karena itu akan melibatkan pembuatan halaman data baru, yang mengakibatkan peningkatan biaya penyimpanan. Data yang tidak berubah dari sebelum peristiwa kloning akan ada dalam satu halaman data yang dibagikan di dua cluster dan akan dikenakan biaya hanya untuk salinan tunggal itu.

Jangan aktifkan indeks OSGP atau gunakan instans R5d kecuali Anda telah menguji untuk mengonfirmasi bahwa indeks tersebut membuat perbedaan besar dalam beban kerja Anda. Keduanya dirancang untuk skenario yang jarang terjadi, dan mereka dapat meningkatkan biaya Anda untuk keuntungan minimal atau tanpa keuntungan.

Pilar keberlanjutan

Pilar keberlanjutan berfokus pada meminimalkan dampak lingkungan dari menjalankan beban kerja cloud. Topik utama mencakup model tanggung jawab bersama untuk keberlanjutan, memahami dampak, dan memaksimalkan penggunaan untuk meminimalkan sumber daya yang dibutuhkan dan mengurangi dampak hilir.

Pilar keberlanjutan berisi area fokus utama berikut:

- Dampak Anda
- Tujuan keberlanjutan
- Penggunaan yang dimaksimalkan
- Mengantisipasi dan mengadopsi penawaran perangkat keras dan perangkat lunak baru yang lebih efisien
- Penggunaan layanan terkelola
- Pengurangan dampak hilir

Panduan ini berfokus pada dampak Anda. Untuk informasi selengkapnya tentang prinsip desain keberlanjutan lainnya, lihat AWS Well-Architected Framework.

Pilihan dan persyaratan Anda berdampak pada lingkungan. Jika Anda dapat memilih Wilayah AWS yang memiliki intensitas karbon lebih rendah, dan jika kebutuhan Anda mencerminkan kebutuhan beban kerja yang sebenarnya, bukan hanya memaksimalkan waktu kerja dan daya tahan, keberlanjutan beban kerja meningkat. Bagian selanjutnya membahas praktik terbaik dan pertimbangan bijaksana yang akan memiliki dampak lingkungan positif jika diadopsi dalam desain beban kerja Anda dan operasi yang sedang berlangsung.

Pemilihan Wilayah AWS

Beberapa Wilayah AWS berada di dekat proyek energi terbarukan Amazon atau terletak di mana jaringan memiliki intensitas karbon yang diterbitkan yang lebih rendah daripada yang lain. Pertimbangkan dampak keberlanjutan untuk Wilayah yang mungkin layak untuk beban kerja Anda, dan rujuk silang daftar Anda dengan Wilayah tempat Neptunus tersedia.

Konsumsi berdasarkan pola perilaku pengguna

Mengukur konsumsi Anda dengan benar agar sesuai dengan lalu lintas dan perilaku pengguna Anda membantu AWS meminimalkan dampak layanan terhadap lingkungan. Pertimbangkan praktik terbaik berikut saat merancang solusi Anda:

- Pantau CloudWatch metrik Amazon seperti`CPUUtilization`,`MainRequestQueuePendingRequests`, dan `TotalRequestsPerSec` untuk menentukan kapan permintaan Anda tertinggi dan terendah, dan pastikan bahwa sumber daya klaster Anda berukuran tepat selama waktu tersebut.
- Otomatiskan penghentian lingkungan non-produksi selama jam-jam ketika tidak digunakan. Untuk informasi selengkapnya, lihat posting blog Mengotomatiskan penghentian dan memulai sumber daya lingkungan Amazon Neptunus menggunakan tag sumber daya.
- Jika pola lalu lintas Anda sering bervariasi dan tidak terduga, pertimbangkan untuk menggunakan instance Neptunus Tanpa Server yang akan meningkat dan turun dengan permintaan alih-alih menggunakan instance yang disediakan untuk lalu lintas puncak.
- Pertimbangkan untuk menyelaraskan perjanjian tingkat layanan Anda dengan tujuan keberlanjutan selain tujuan kelangsungan bisnis. Persyaratan pelonggaran seperti pemulihan bencana multi-wilayah, ketersediaan tinggi, atau retensi cadangan jangka panjang, terutama untuk lingkungan non-produksi atau beban kerja kritis non-misi, dapat mengurangi jumlah sumber daya yang dibutuhkan untuk memenuhi tujuan tersebut.

Optimalkan pengembangan perangkat lunak dan pola arsitektur

Untuk mencegah pemborosan, optimalkan model dan kueri, dan bagikan sumber daya komputasi sehingga Anda menggunakan semua sumber daya yang tersedia di instans dan cluster Neptunus. Praktik terbaik khusus meliputi:

- Mintalah pengembang berbagi instance Neptunus dan instance aplikasi Jupyter Notebook alih-alih masing-masing membuat sendiri. Berikan masing-masing pengembang partisi logisnya sendiri dalam satu cluster Neptunus melalui penggunaan strategi partisi multi-tenancy, dan buat folder notebook terpisah untuk setiap pengembang pada satu instance Jupyter.
- Menerapkan pola yang memaksimalkan penggunaan sumber daya dan meminimalkan waktu idle, seperti thread paralel untuk memuat data dan mengumpulkan catatan bersama-sama ke dalam transaksi yang lebih besar.

- Optimalkan kueri dan model grafik Anda untuk meminimalkan sumber daya yang diperlukan untuk menghitung hasil.
- Untuk hasil kueri Gremlin, gunakan fitur cache hasil untuk meminimalkan sumber daya yang dihabiskan untuk menghitung ulang kueri yang dipaginasi atau sering berulang.
- Perbarui lingkungan Neptunus Anda. Versi terbaru Neptunus mendukung contoh EC2 terbaru, seperti Graviton, yang lebih efisien. Mereka juga memiliki peningkatan optimasi kueri dan perbaikan bug yang mengurangi jumlah sumber daya yang dibutuhkan untuk menghitung kueri Anda.

Sumber daya

Referensi

- AWS Well-Architected
- AWS Dokumentasi Kerangka Well-Architected
- Menerapkan Kerangka Kerja AWS Well-Architected untuk Amazon Neptune Analytics
- Pembaruan terbaru Amazon Neptunus
- Praktik terbaik: mendapatkan hasil maksimal dari Neptunus

Posting blog

- Pengujian otomatis akses data Amazon Neptunus dengan Apache Gremlin TinkerPop
- Otomatiskan penghentian dan dimulainya sumber daya lingkungan Amazon Neptunus menggunakan tag sumber daya
- Kontrol Akses Berbutir Halus untuk tindakan bidang data Amazon Neptunus
- Gunakan penalaran semantik untuk menyimpulkan fakta baru dari grafik RDF Anda dengan mengintegrasikan dengan Amazon Neptunus RDFox
- Membangun solusi deteksi penipuan real-time menggunakan Amazon Neptunus ML

Kursus AWS Skill Builder gratis

- Memulai dengan Amazon Neptunus
- Membangun Aplikasi di Amazon Neptunus
- Pemodelan Data untuk Amazon Neptunus

Kontributor

Kontributor untuk panduan ini meliputi:

- Brian O'Keefe, Arsitek Solusi Utama Neptunus, AWS
- Abhishek Mishra, Arsitek Solusi Neptunus Senior, AWS
- Ganesh Sawhney, Ketua Tim - Arsitek Solusi Sukses Mitra Strategis, AWS
- Michael Havey, Arsitek Solusi Neptunus Senior, AWS
- Kevin Phillips, Arsitek Solusi Neptunus, AWS
- Melissa Kwok, Arsitek Solusi Neptunus, AWS
- Sakti Mishra, Arsitek Solusi Utama AWS
- Javed Ali, Arsitek Solusi Senior, 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	—	27 September 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: Memigrasikan database Oracle lokal Anda ke Amazon Relational Database Service (Amazon RDS) untuk Oracle di. AWS Cloud
- Pembelian kembali (drop and shop) - Beralih ke produk yang berbeda, biasanya dengan beralih dari lisensi tradisional ke model SaaS. Contoh: Migrasikan sistem manajemen hubungan pelanggan (CRM) Anda ke Salesforce.com.
- Rehost (lift dan shift) — Pindahkan aplikasi ke cloud tanpa membuat perubahan apa pun untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan database Oracle lokal Anda ke Oracle pada instance EC2 di. AWS Cloud
- Relokasi (hypervisor-level lift and shift) — Pindahkan infrastruktur ke cloud tanpa membeli perangkat keras baru, menulis ulang aplikasi, atau memodifikasi operasi yang ada. Anda memigrasikan server dari platform lokal ke layanan cloud untuk platform yang sama. Contoh: Migrasikan Microsoft Hyper-V aplikasi ke AWS.
- Pertahankan (kunjungi kembali) - Simpan aplikasi di lingkungan sumber Anda. Ini mungkin termasuk aplikasi yang memerlukan refactoring besar, dan Anda ingin menunda pekerjaan itu sampai nanti, dan aplikasi lama yang ingin Anda pertahankan, karena tidak ada pembenaran bisnis untuk memigrasikannya.

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

A

ABAC

Lihat kontrol akses berbasis atribut.

layanan abstrak

Lihat layanan terkelola.

ASAM

Lihat atomisitas, konsistensi, isolasi, daya tahan.

migrasi aktif-aktif

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

migrasi aktif-pasif

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

fungsi agregat

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

AI

Lihat kecerdasan buatan.

AIOps

Lihat operasi kecerdasan buatan.

anonimisasi

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

anti-pola

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

kontrol aplikasi

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

portofolio aplikasi

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

kecerdasan buatan (AI)

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

operasi kecerdasan buatan (AIOps)

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

enkripsi asimetris

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

atomisitas, konsistensi, isolasi, daya tahan (ACID)

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

kontrol akses berbasis atribut (ABAC)

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

sumber data otoritatif

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

Zona Ketersediaan

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

AWS Kerangka Adopsi Cloud (AWS CAF)

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

AWS Kerangka Kualifikasi Beban Kerja (AWS WQF)

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

B

bot buruk

Bot yang dimaksudkan untuk mengganggu atau membahayakan 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

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

CI/CD

Lihat integrasi berkelanjutan dan pengiriman berkelanjutan.

klasifikasi

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

Enkripsi sisi klien

Enkripsi data secara lokal, sebelum target Layanan AWS menerimanya.

Pusat Keunggulan Cloud (CCoE)

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

komputasi cloud

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

model operasi cloud

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

tahap adopsi cloud

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

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

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

CMDB

Lihat database manajemen konfigurasi.

repositori kode

Lokasi di mana kode sumber dan aset lainnya, seperti dokumentasi, sampel, dan skrip, disimpan dan diperbarui melalui proses kontrol versi. Repositori cloud umum termasuk GitHub atau. Bitbucket Cloud Setiap versi kode disebut cabang. Dalam struktur layanan mikro, setiap repositori

dikhususkan untuk satu bagian fungsionalitas. Pipa CI/CD tunggal dapat menggunakan beberapa repositori.

cache dingin

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

data dingin

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

visi komputer (CV)

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

konfigurasi drift

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

database manajemen konfigurasi (CMDB)

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

paket kesesuaian

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

integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD)

Proses mengotomatiskan sumber, membangun, menguji, pementasan, dan tahap produksi dari proses rilis perangkat lunak. CI/CD is commonly described as a pipeline. CI/CDdapat membantu

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

CV

Lihat visi komputer.

D

data saat istirahat

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

klasifikasi data

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

penyimpangan data

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

data dalam transit

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

jala data

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

minimalisasi data

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

perimeter data

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

prapemrosesan data

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

asal data

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

subjek data

Individu yang datanya dikumpulkan dan diproses.

gudang data

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

bahasa definisi database (DDL)

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

bahasa manipulasi basis data (DHTML)

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

DDL

Lihat bahasa definisi database.

ansambel yang dalam

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

pembelajaran mendalam

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

defense-in-depth

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

administrator yang didelegasikan

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

deployment

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

lingkungan pengembangan

Lihat lingkungan.

kontrol detektif

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

pemetaan aliran nilai pengembangan (DVSM)

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

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

kembar digital

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

tabel dimensi

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

musibah

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

pemulihan bencana (DR)

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

DML~

Lihat bahasa manipulasi basis data.

desain berbasis domain

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

DR

Lihat pemulihan bencana.

deteksi drift

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

DVSM

Lihat pemetaan aliran nilai pengembangan.

E

EDA

Lihat analisis data eksplorasi.

EDI

Lihat pertukaran data elektronik.

komputasi tepi

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

pertukaran data elektronik (EDI)

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

enkripsi

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

kunci enkripsi

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

endianness

Urutan byte disimpan dalam memori komputer. Sistem big-endian menyimpan byte paling signifikan terlebih dahulu. Sistem little-endian menyimpan byte paling tidak signifikan terlebih dahulu.

titik akhir

Lihat titik akhir layanan.

layanan endpoint

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

perencanaan sumber daya perusahaan (ERP)

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

enkripsi amplop

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

lingkungan

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

- Development Environment — Sebuah contoh dari aplikasi yang berjalan yang hanya tersedia untuk tim inti yang bertanggung jawab untuk memelihara aplikasi. Lingkungan pengembangan digunakan untuk menguji perubahan sebelum mempromosikannya ke lingkungan atas. Jenis lingkungan ini kadang-kadang disebut sebagai lingkungan pengujian.
- lingkungan yang lebih rendah — Semua lingkungan pengembangan untuk aplikasi, seperti yang digunakan untuk build awal dan pengujian.
- lingkungan produksi — Sebuah contoh dari aplikasi yang berjalan yang pengguna akhir dapat mengakses. Dalam 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.

beberapa tembakan mendorong

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

FGAC

Lihat kontrol akses berbutir halus.

kontrol akses berbutir halus (FGAC)

Penggunaan beberapa kondisi untuk mengizinkan atau menolak permintaan akses.

migrasi flash-cut

Metode migrasi database yang menggunakan replikasi data berkelanjutan melalui pengambilan data perubahan untuk memigrasikan data dalam waktu sesingkat mungkin, alih-alih

menggunakan pendekatan bertahap. Tujuannya adalah untuk menjaga downtime seminimal mungkin.

FM

Lihat model pondasi.

model pondasi (FM)

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

G

AI generatif

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

pemblokiran geografis

Lihat pembatasan geografis.

pembatasan geografis (pemblokiran geografis)

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

Alur kerja Gitflow

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

gambar emas

Sebuah snapshot dari sistem atau perangkat lunak yang digunakan sebagai template untuk menyebarkan instance baru dari sistem atau perangkat lunak itu. Misalnya, di bidang manufaktur,

gambar emas dapat digunakan untuk menyediakan perangkat lunak pada beberapa perangkat dan membantu meningkatkan kecepatan, skalabilitas, dan produktivitas dalam operasi manufaktur perangkat.

strategi greenfield

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

pagar pembatas

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

H

HA

Lihat ketersediaan tinggi.

migrasi database heterogen

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

ketersediaan tinggi (HA)

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

modernisasi sejarawan

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

data penahanan

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

migrasi database homogen

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

data panas

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

perbaikan terbaru

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

periode hypercare

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

I

IAc

Lihat infrastruktur sebagai kode.

kebijakan berbasis identitas

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

aplikasi idle

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

IIoT

Lihat Internet of Things industri.

infrastruktur yang tidak dapat diubah

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

masuk (masuknya) VPC

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

migrasi inkremental

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

Industri 4.0

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

infrastruktur

Semua sumber daya dan aset yang terkandung dalam lingkungan aplikasi.

infrastruktur sebagai kode (IAc)

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

Internet of Things industri (IIoT)

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

inspeksi VPC

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

Internet of Things (IoT)

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

interpretabilitas

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

IoT

Lihat Internet of Things.

Perpustakaan informasi TI (ITIL)

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

Manajemen layanan TI (ITSM)

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

ITIL

Lihat perpustakaan informasi TI.

ITSM

Lihat manajemen layanan TI.

L

kontrol akses berbasis label (LBAC)

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

landing zone

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

model bahasa besar (LLM)

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

migrasi besar

Migrasi 300 atau lebih server.

LBAC

Lihat kontrol akses berbasis label.

hak istimewa paling sedikit

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

angkat dan geser

Lihat 7 Rs.

sistem endian kecil

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

LLM

Lihat model bahasa besar.

lingkungan yang lebih rendah

Lihat lingkungan.

M

pembelajaran mesin (ML)

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

cabang utama

Lihat cabang.

malware

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

layanan terkelola

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

sistem eksekusi manufaktur (MES)

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

PETA

Lihat Program Percepatan Migrasi.

mekanisme

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

akun anggota

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

MES

Lihat sistem eksekusi manufaktur.

Transportasi Telemetri Antrian Pesan (MQTT)

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

layanan mikro

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

arsitektur microservices

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

Program Percepatan Migrasi (MAP)

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

migrasi dalam skala

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

pabrik migrasi

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

metadata migrasi

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

pola migrasi

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

Penilaian Portofolio Migrasi (MPA)

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

Penilaian Kesiapan Migrasi (MRA)

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

strategi migrasi

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

ML

Lihat pembelajaran mesin.

modernisasi

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

penilaian kesiapan modernisasi

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

aplikasi monolitik (monolit)

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

MPA

Lihat Penilaian Portofolio Migrasi.

MQTT

Lihat Transportasi Telemetri Antrian Pesan.

klasifikasi multiclass

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

infrastruktur yang bisa berubah

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

O

OAC

Lihat kontrol akses asal.

OAI

Lihat identitas akses asal.

OCM

Lihat manajemen perubahan organisasi.

migrasi offline

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

OI

Lihat integrasi operasi.

OLA

Lihat perjanjian tingkat operasional.

migrasi online

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

OPC-UA

Lihat Komunikasi Proses Terbuka - Arsitektur Terpadu.

Komunikasi Proses Terbuka - Arsitektur Terpadu (OPC-UA)

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

perjanjian tingkat operasional (OLA)

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

Tinjauan Kesiapan Operasional (ORR)

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

teknologi operasional (OT)

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

integrasi operasi (OI)

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

jejak organisasi

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

manajemen perubahan organisasi (OCM)

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

kontrol akses asal (OAC)

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

identitas akses asal (OAI)

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

ORR

Lihat tinjauan kesiapan operasional.

OT

Lihat teknologi operasional.

keluar (jalan keluar) VPC

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

P

batas izin

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

Informasi Identifikasi Pribadi (PII)

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

PII

Lihat informasi yang dapat diidentifikasi secara pribadi.

buku pedoman

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

PLC

Lihat pengontrol logika yang dapat diprogram.

PLM

Lihat manajemen siklus hidup produk.

kebijakan

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

ketekunan poliglot

Secara independen memilih teknologi penyimpanan data microservice berdasarkan pola akses data dan persyaratan lainnya. Jika layanan mikro Anda memiliki teknologi penyimpanan data yang sama, mereka dapat menghadapi tantangan implementasi atau mengalami kinerja yang buruk. Layanan mikro lebih mudah diimplementasikan dan mencapai kinerja dan skalabilitas yang lebih baik jika mereka menggunakan penyimpanan data yang paling sesuai dengan kebutuhan mereka. 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 optimasi kueri database yang menyaring data dalam kueri sebelum transfer. Ini mengurangi jumlah data yang harus diambil dan diproses dari database relasional, dan meningkatkan kinerja kueri.

kontrol preventif

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

principal

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

privasi berdasarkan desain

Pendekatan rekayasa sistem yang memperhitungkan privasi melalui seluruh proses pengembangan.

zona yang dihosting pribadi

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

kontrol proaktif

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

manajemen siklus hidup produk (PLM)

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

lingkungan produksi

Lihat lingkungan.

pengontrol logika yang dapat diprogram (PLC)

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

rantai cepat

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

pseudonimisasi

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

publish/subscribe (pub/sub)

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

Q

rencana kueri

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

regresi rencana kueri

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

R

Matriks RACI

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

LAP

Lihat Retrieval Augmented Generation.

ransomware

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

Matriks RASCI

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

RCAC

Lihat kontrol akses baris dan kolom.

replika baca

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

arsitek ulang

Lihat 7 Rs.

tujuan titik pemulihan (RPO)

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

tujuan waktu pemulihan (RTO)

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

refactor

Lihat 7 Rs.

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

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

Matriks yang mendefinisikan peran dan tanggung jawab untuk semua pihak yang terlibat dalam kegiatan migrasi dan operasi cloud. Nama matriks berasal dari jenis tanggung jawab yang

didefinisikan dalam matriks: bertanggung jawab (R), akuntabel (A), dikonsultasikan (C), dan diinformasikan (I). Tipe dukungan (S) adalah opsional. Jika Anda menyertakan dukungan, matriks disebut matriks RASCI, dan jika Anda mengecualikannya, itu disebut matriks RACI.

kontrol responsif

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

melestarikan

Lihat 7 Rs.

pensiun

Lihat 7 Rs.

Retrieval Augmented Generation (RAG)

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

rotasi

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

kontrol akses baris dan kolom (RCAC)

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

RPO

Lihat tujuan titik pemulihan.

RTO

Lihat tujuan waktu pemulihan.

buku runbook

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

D

SAML 2.0

Standar terbuka yang digunakan oleh banyak penyedia identitas (IdPs). Fitur ini memungkinkan sistem masuk tunggal gabungan (SSO), sehingga pengguna dapat masuk ke 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.

SCADA

Lihat kontrol pengawasan dan akuisisi data.

SCP

Lihat kebijakan kontrol layanan.

Rahasia

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

keamanan dengan desain

Pendekatan rekayasa sistem yang memperhitungkan keamanan melalui seluruh proses pengembangan.

kontrol keamanan

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

pengerasan keamanan

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

sistem informasi keamanan dan manajemen acara (SIEM)

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

otomatisasi respons keamanan

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

enkripsi sisi server

Enkripsi data di tujuannya, oleh Layanan AWS yang menerimanya.

kebijakan kontrol layanan (SCP)

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

titik akhir layanan

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

perjanjian tingkat layanan (SLA)

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

indikator tingkat layanan (SLI)

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

tujuan tingkat layanan (SLO)

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

model tanggung jawab bersama

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

SIEM

Lihat informasi keamanan dan sistem manajemen acara.

titik kegagalan tunggal (SPOF)

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

SLA

Lihat perjanjian tingkat layanan.

SLI

Lihat indikator tingkat layanan.

SLO

Lihat tujuan tingkat layanan.

split-and-seed model

Pola untuk menskalakan dan mempercepat proyek modernisasi. Ketika fitur baru dan rilis produk didefinisikan, tim inti berpisah untuk membuat tim produk baru. Ini membantu meningkatkan kemampuan dan layanan organisasi Anda, meningkatkan produktivitas pengembang, dan

mendukung inovasi yang cepat. Untuk informasi lebih lanjut, lihat Pendekatan bertahap untuk memodernisasi aplikasi di. AWS Cloud

SPOF

Lihat satu titik kegagalan.

skema bintang

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

pola ara pencekik

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

subnet

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

kontrol pengawasan dan akuisisi data (SCADA)

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

enkripsi simetris

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

pengujian sintetis

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

sistem prompt

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

T

tag

Pasangan nilai kunci yang bertindak sebagai metadata untuk mengatur sumber daya Anda. AWS Tanda 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 jaringan Anda VPCs dan lokal. Untuk informasi selengkapnya, lihat Apa itu gateway transit dalam AWS Transit Gateway dokumentasi.

alur kerja berbasis batang

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

akses tepercaya

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

penyetelan

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

tim dua pizza

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

U

waswas

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

tugas yang tidak terdiferensiasi

Juga dikenal sebagai angkat berat, pekerjaan yang diperlukan untuk membuat dan mengoperasikan aplikasi tetapi itu tidak memberikan nilai langsung kepada pengguna akhir atau

memberikan keunggulan kompetitif. Contoh tugas yang tidak terdiferensiasi termasuk pengadaan, pemeliharaan, dan perencanaan kapasitas.

lingkungan atas

Lihat lingkungan.

V

menyedot debu

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

kendali versi

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

Peering VPC

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

kerentanan

Kelemahan perangkat lunak atau perangkat keras yang membahayakan keamanan sistem.

W

cache hangat

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

data hangat

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

fungsi jendela

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

beban kerja

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

aliran kerja

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

CACING

Lihat menulis sekali, baca banyak.

WQF

Lihat AWS Kerangka Kualifikasi Beban Kerja.

tulis sekali, baca banyak (WORM)

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

Z

eksploitasi zero-day

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

kerentanan zero-day

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

bisikan zero-shot

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

aplikasi zombie

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

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