Panduan multi-penyewaan untuk ISVs menjalankan database Amazon Neptunus

AWS Bimbingan Preskriptif

AWS Bimbingan Preskriptif: Panduan multi-penyewaan untuk ISVs menjalankan database Amazon Neptunus

Table of Contents

Panduan multi-penyewaan untuk ISVs menjalankan database Amazon Neptunus

Amazon Web Services (kontributor)

Agustus 2024 (sejarah dokumen)

Multi-tenancy adalah arsitektur sistem komputer di mana satu contoh aplikasi melayani banyak pelanggan. Setiap pelanggan disebut sebagai penyewa. Dalam arsitektur multi-tenant, contoh aplikasi ini beroperasi di lingkungan bersama di mana setiap penyewa secara fisik ditempatkan bersama pada infrastruktur yang sama tetapi secara logis terpisah.

Sebagai vendor perangkat lunak independen (ISV), Anda dapat menggunakan Amazon Neptunus untuk memberi daya pada aplikasi yang memerlukan navigasi di seluruh data yang sangat terhubung. Anda mungkin mengelola aplikasi perangkat lunak berbasis cloud sebagai layanan (SaaS) di akun Anda dan menyediakan langganan kepada penyewa. Penyewa kemudian dapat mengakses layanan melalui internet atau secara pribadi. AWS PrivateLink Ekonomi model ini bekerja untuk kedua belah pihak, karena penyewa mendapatkan akses ke perangkat lunak yang lebih murah daripada bagi mereka untuk membeli, membangun, dan memelihara. Sebagai ISV, Anda dapat mengenakan biaya lebih untuk berlangganan daripada biaya untuk membuat dan memelihara perangkat lunak. Pertanyaannya adalah bagaimana Anda menskalakan bisnis Anda ke beberapa penyewa.

Multi-tenancy memberikan manfaat ISVs ekonomis dan operasional yang penting. Arsitektur multi-tenant memberi organisasi Anda pengembalian investasi (ROI) yang lebih baik. Multi-tenancy juga menyederhanakan persyaratan operasional sehingga organisasi Anda dapat bergerak lebih cepat dan mengurangi biaya pengiriman perangkat lunak kepada penyewa Anda.

Dokumen ini memberikan panduan untuk menjalankan aplikasi ISV multi-tenant secara efektif menggunakan Amazon Neptunus. Panduan ini didasarkan pada praktik terbaik yang diperoleh selama bertahun-tahun untuk mendukung ISVs 'pengiriman solusi SaaS yang sukses kepada pelanggan mereka. Mengevaluasi panduan ini dalam konteks tujuan organisasi Anda dan prinsip-prinsip arsitektur akan membantu Anda menemukan cara untuk mengoptimalkan solusi Anda.

> **Note**
>
> Dokumen ini tidak memberikan daftar lengkap praktik terbaik. Ini melengkapi dokumen Menerapkan Kerangka Kerja AWS Well-Architected untuk Amazon Neptunus dengan

memberikan panduan khusus tambahan untuk beban kerja ISV multi-tenancy. Kami merekomendasikan untuk meninjau pertimbangan di kedua dokumen saat merancang solusi Anda.

Model partisi data SaaS

Salah satu tantangan bagi pengembang SaaS adalah merancang pola arsitektur untuk mewakili dan mengatur data dalam lingkungan multi-penyewa. Mekanisme dan pola penyimpanan multi-tenant ini biasanya disebut sebagai partisi data.

Dalam lingkungan SaaS multi-penyewa, penting untuk membedakan antara partisi data dan isolasi penyewa. Konsep-konsep ini, meskipun terkait, tidak identik. Partisi data mengacu pada metode penyimpanan data untuk setiap penyewa. Namun, partisi saja tidak menjamin isolasi penyewa. Langkah-langkah tambahan diperlukan untuk memastikan bahwa data satu penyewa tetap tidak dapat diakses oleh yang lain.

Tiga model partisi data umum dalam sistem SaaS multi-tenant adalah silo, pool, dan hybrid. Pilihan model apa pun tergantung pada faktor-faktor seperti berikut:

- Kepatuhan
- Tetangga yang berisik
- Strategi tiering
- Persyaratan operasional
- Kebutuhan isolasi penyewa

Selain itu, setiap jenis database yang tersedia AWS biasanya menawarkan koleksi unik partisi data dan model isolasi penyewa. Saat melihat bagaimana grafik penyewa dapat diatur untuk mendukung berbagai kebutuhan solusi Anda, pertimbangkan model yang disediakan Amazon Neptunus.

Banyak yang ISVs memulai desain mereka di Neptunus dengan salah satu pernyataan berikut:

- ISVSolusinya membutuhkan pemisahan fisik pelanggan di seluruh cluster terpisah.
- ISVSolusinya membutuhkan konstruksi seperti database bernama atau skema yang ditemukan dalam sistem manajemen database relasional tradisional.

Setelah dipertimbangkan, ISVs sadari bahwa pernyataan ini tidak benar karena, di bawah hampir semua beban kerja, setiap pelanggan mereka memiliki grafik yang terputus dalam database mereka. Menerapkan pemodelan data dan panduan akses yang dibahas dalam dokumen ini mencegah batas-batas data tersebut dilintasi dan menjaga privasi data pelanggan.

Panduan ini menjelaskan model silo dan model kolam renang, tetapi sebagian besar ISVs memilih model kolam untuk biaya dan efisiensi operasional. Panduan ini secara singkat membahas model hibrida yang menggabungkan aspek model silo dan kolam renang. Beberapa ISVs menggunakan model hibrida untuk pelanggan terbesar mereka untuk mengakomodasi persyaratan peraturan atau kepatuhan ukuran grafik.

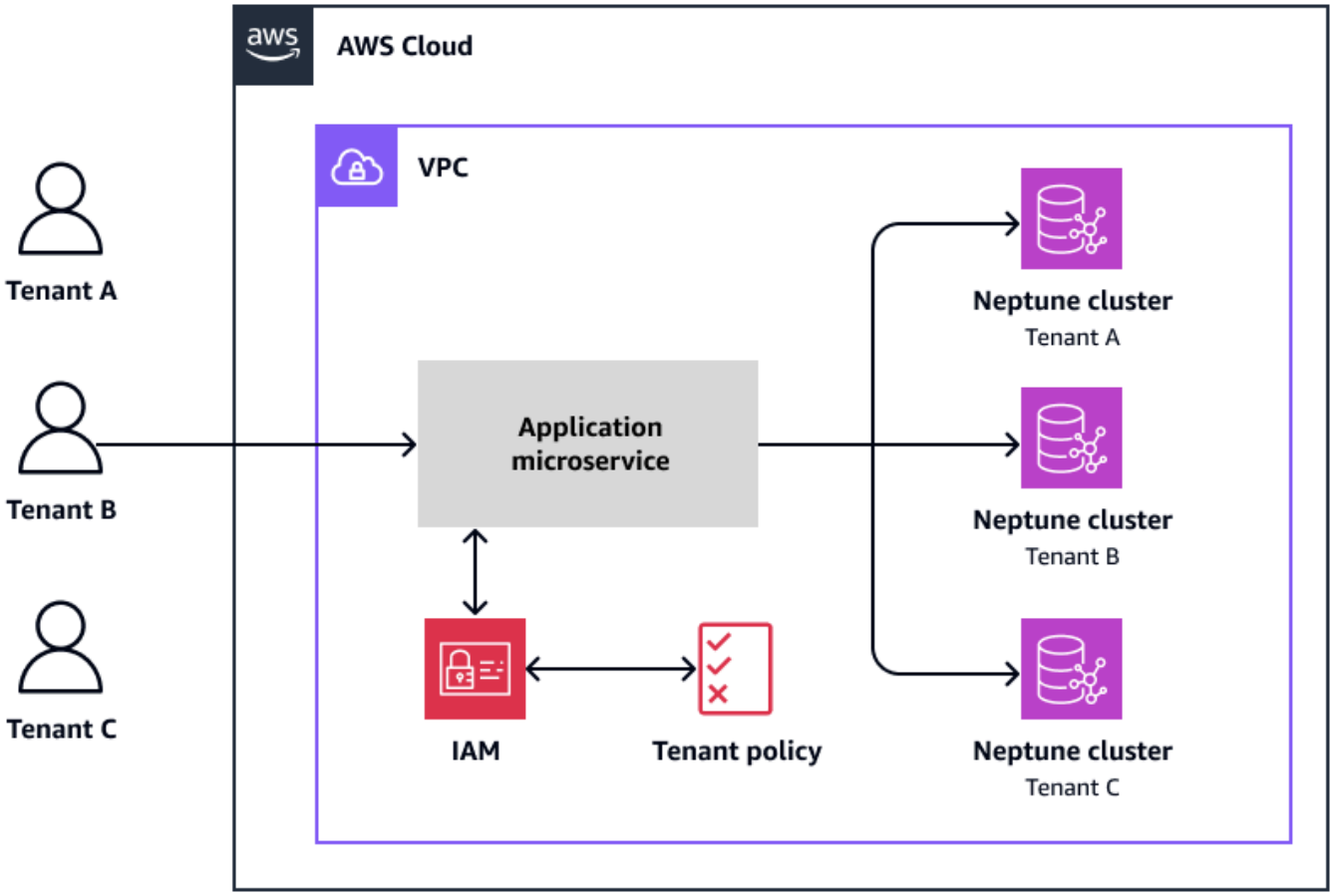
Multi-penyewaan model silo

Beberapa lingkungan SaaS multi-penyewa mungkin memerlukan data penyewa untuk digunakan pada sumber daya yang sepenuhnya terpisah karena kepatuhan dan persyaratan peraturan. Dalam beberapa kasus, pelanggan besar memerlukan cluster khusus untuk mengurangi dampak tetangga yang bising. Dalam situasi tersebut, Anda dapat menerapkan model silo.

Dalam model silo, penyimpanan data penyewa sepenuhnya terisolasi dari data penyewa lainnya. Semua konstruksi yang digunakan untuk mewakili data penyewa dianggap unik secara fisik untuk klien tersebut, yang berarti bahwa setiap penyewa umumnya akan memiliki penyimpanan, pemantauan, dan manajemen yang berbeda. Setiap penyewa juga akan memiliki kunci AWS Key Management Service (AWS KMS) terpisah untuk enkripsi. Di Amazon Neptunus, silo adalah satu cluster per penyewa.

Cluster per penyewa

Anda dapat menerapkan model silo dengan Neptunus dengan memiliki satu penyewa per cluster. Diagram berikut menunjukkan tiga penyewa mengakses layanan mikro aplikasi di cloud pribadi virtual (VPC), dengan cluster terpisah untuk setiap penyewa.



Setiap cluster memiliki titik akhir masing-masing untuk membantu memastikan titik akses yang berbeda untuk interaksi dan manajemen data yang efisien. Dengan menempatkan setiap penyewa di klaster sendiri, Anda membuat batas yang terdefinisi dengan baik antara penyewa yang memastikan pelanggan bahwa data mereka berhasil diisolasi dari data penyewa lain. Isolasi ini juga menarik bagi solusi SaaS yang memiliki batasan peraturan dan keamanan yang ketat. Selain itu, ketika setiap penyewa memiliki cluster sendiri Anda tidak perlu khawatir tentang tetangga yang berisik, di mana satu penyewa memaksakan beban yang dapat mempengaruhi pengalaman penyewa lainnya.

Sementara model cluster-per-tenant silo memiliki kelebihan, ia juga memperkenalkan tantangan manajemen dan kelincahan. Sifat terdistribusi dari model ini membuat lebih sulit untuk mengumpulkan dan menilai aktivitas penyewa dan kesehatan operasional di semua penyewa. Penerapan juga menjadi lebih menantang karena menyiapkan penyewa baru sekarang memerlukan penyediaan cluster terpisah. Upgrade menjadi lebih menantang di lingkungan dengan lapisan klien bersama ketika upgrade klien dan versi digabungkan erat ke upgrade database.

Neptunus mendukung cluster tanpa server dan yang disediakan. Menilai apakah beban kerja aplikasi Anda ditangani dengan lebih baik oleh instans tanpa server atau yang disediakan. Secara umum, jika

beban kerja Anda memiliki tingkat permintaan yang konstan, instance yang disediakan akan lebih hemat biaya. Tanpa server dioptimalkan untuk beban kerja yang sangat bervariasi dan menuntut dengan penggunaan database yang berat untuk waktu yang singkat diikuti oleh periode aktivitas ringan yang lama atau tidak ada aktivitas.

Saat menggunakan cluster yang disediakan Neptunus per penyewa, Anda harus memilih ukuran instans yang mendekati beban maksimum permintaan penyewa Anda. Ketergantungan pada server ini juga memiliki dampak cascading pada efisiensi penskalaan dan biaya lingkungan SaaS Anda. Sementara tujuan SaaS adalah untuk mengukur secara dinamis berdasarkan beban penyewa yang sebenarnya, kluster yang disediakan Neptunus mengharuskan Anda melakukan penyediaan berlebihan untuk memperhitungkan periode penggunaan yang lebih berat dan lonjakan beban. Penyediaan berlebih meningkatkan biaya per penyewa. Selain itu, karena penggunaan penyewa berubah seiring waktu, penskalaan atau penskalaan cluster harus diterapkan secara terpisah untuk setiap penyewa.

Tim Neptunus umumnya menyarankan untuk tidak menggunakan model silo karena biaya yang lebih tinggi yang dikeluarkan oleh sumber daya yang menganggur dan kompleksitas operasional tambahan. Namun, untuk beban kerja yang sangat diatur atau sensitif memerlukan isolasi tambahan ini, pelanggan mungkin bersedia membayar biaya tambahan.

Panduan implementasi untuk model silo

Untuk menerapkan model cluster-per-tenant isolasi silo, buat kebijakan akses data AWS Identity and Access Management (IAM). Kebijakan ini mengontrol akses ke cluster Neptunus penyewa dengan memastikan bahwa penyewa hanya dapat mengakses cluster Neptunus yang berisi data mereka sendiri. Lampirkan kebijakan IAM untuk setiap penyewa ke peran IAM. Layanan mikro aplikasi kemudian menggunakan peran IAM untuk menghasilkan kredensil sementara berbutir halus menggunakan metode (). `AssumeRole` AWS Security Token Service AWS STS Kredensil ini, yang hanya memiliki akses ke cluster Neptunus untuk penyewa itu, digunakan untuk terhubung ke cluster Neptunus penyewa.

Cuplikan kode berikut menunjukkan contoh kebijakan IAM berbasis data:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "neptune-db:ReadDataViaQuery",
        "neptune-db:WriteDataViaQuery"
      ],
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:tenant-1-cluster/*",
      "Condition": {
        "ArnEquals": {
          "aws:PrincipalArn": "arn:aws:iam::123456789012:role/tenant-role-1"
        }
      }
    }
  ]
}
```

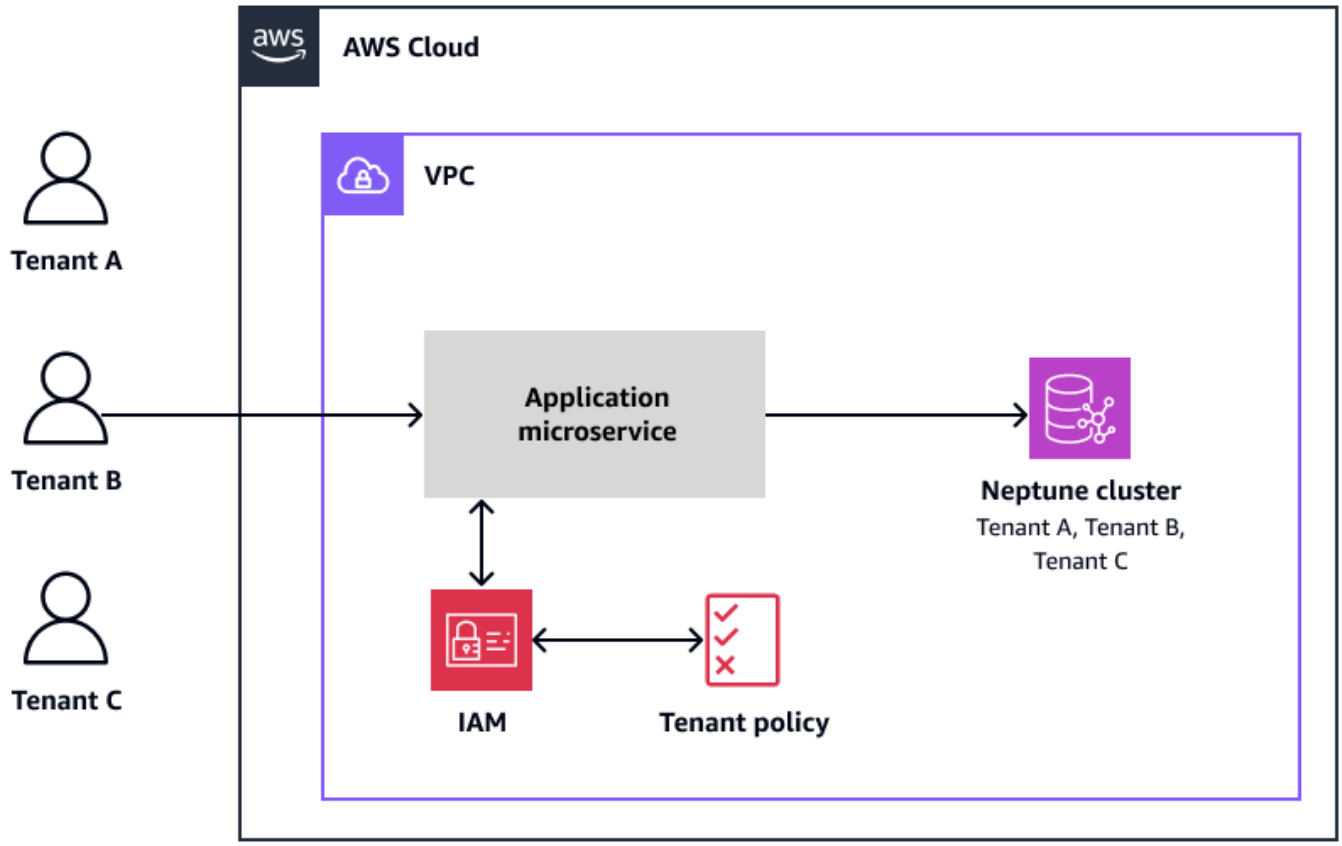
Kode ini menyediakan penyewa sampel,`tenant-1`, dengan akses kueri baca dan tulis ke cluster Neptunus masing-masing. `Condition`Elemen memastikan bahwa hanya entitas pemanggil (prinsipal), yang telah mengambil peran `tentant-1` IAM (`tenant-role-1`), diizinkan untuk mengakses cluster `tenant-1` Neptunus.

Multi-penyewaan model kolam

Terkadang tidak perlu atau layak untuk menerapkan model silo karena biaya atau overhead operasional:

- Anda mungkin tidak memiliki sumber daya untuk memelihara cluster individu per penyewa.
- Mungkin tidak perlu memisahkan data setiap penyewa secara fisik, dan pemisahan logis sudah cukup untuk memenuhi kebutuhan dan persyaratan kepatuhan mereka.

Diagram berikut menunjukkan model pool, dengan data penyewa ditempatkan dalam satu cluster Amazon Neptunus, dan semua penyewa berbagi database yang sama.



Model isolasi kolam ini mengurangi overhead manajemen dan dapat meningkatkan efisiensi operasional karena ada lebih sedikit cluster untuk dikelola. Selain itu, sumber daya komputasi dapat dibagikan ke beberapa pelanggan alih-alih tetap menganggur selama periode tidak aktif pelanggan.

Saat Anda menggunakan model pool, ada dua cara untuk memodelkan data. Pendekatan Anda tergantung pada apakah Anda sedang membangun grafik properti berlabel (LPG) atau grafik dengan Resource Description Framework (RDF).

Model kolam untuk grafik properti berlabel

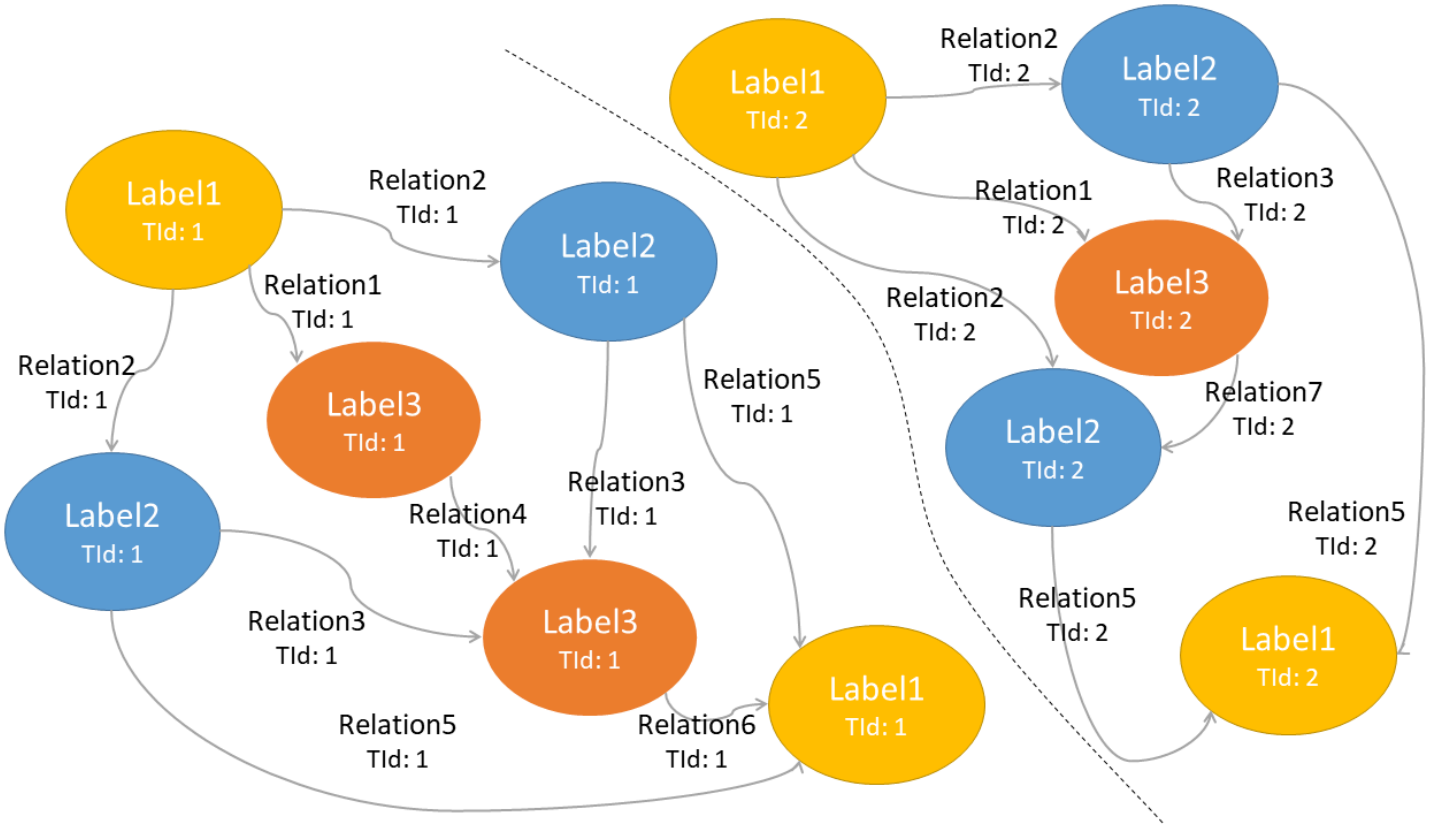
Ada tiga pendekatan berbeda untuk model pool LPGs di Amazon Neptunus:

- Strategi properti - Pilih strategi properti ketika Anda perlu memprioritaskan penggunaan konstruksi perpustakaan yang sudah mapan seperti kinerja berlebihan bahasa Apache TinkerPop Gremlin. PartitionStrategy
- Strategi label awalan - Kami merekomendasikan strategi label awalan untuk sebagian besar skenario berdasarkan kinerja dan membatasi efek tetangga yang bising.
- Strategi multi-label - Strategi multi-label memiliki kinerja yang lebih baik dari strategi label awalan. Ini juga mendukung menjalankan kueri yang menjangkau semua penyewa di klaster (misalnya, kueri ISV untuk pelaporan atau pemantauan di semua penyewa).

Strategi properti

Dengan LPGs, pengguna dapat menambahkan properti pasangan kunci-nilai ke node, atau simpul, dan tepi. Untuk mencapai pemisahan logis, sebagian besar pelanggan secara intuitif memodelkan ini sebagai properti unik di setiap node dan tepi dengan kunci properti penyewa umum. Kunci properti penyewa mewakili semua penyewa yang memiliki node. Pengidentifikasi penyewa adalah nilai unik yang mengidentifikasi penyewa individu.

Diagram berikut menunjukkan model ini. Dua subgraf yang terputus memiliki berbagai node dan tepi berlabel, dengan kunci properti penyewa diwakili oleh. `TId` Setiap simpul dan tepi dari satu subgraf memiliki `TId` nilai. `1` Dalam subgraf lainnya, setiap node dan edge memiliki `TId` nilai. `2`



Dalam grafik properti berlabel, ada dua cara untuk mengelola ini. Bahasa query Gremlin menawarkan perpustakaan PartitionStrategytraversal untuk membantu mengelola partisi data data. Kode dalam contoh berikut mengharapkan setiap node dan edge memiliki properti yang disebut`TId`:

```
strategy1 = new PartitionStrategy(partitionKey: "TId", writePartition: "1",
 readPartitions: ["1"])
strategy2 = new PartitionStrategy(partitionKey: "TId", writePartition: "2",
 readPartitions: ["2"])
```

Ketika node atau tepi baru ditulis, properti `"TId"` ditambahkan dengan nilai `"1"` atau`"2"`, tergantung pada apakah `strategy1` atau `strategy2` dipilih. Untuk pelanggan dengan `"TId""1"`, Anda gunakan`strategy1`. Contoh berikut menunjukkan penulisan data untuk pelanggan itu:

```
g.withStrategies(strategy1).addV("Label1").property("Value", "123456").property(id,
 "Item_1")
```

Untuk kueri baca, filter untuk `"TId == '1'"` atau `"TId == '2'"` ditambahkan ke setiap node atau edge traversal dengan menggunakan `strategy1` ataustrategy2, masing-masing. Strategi partisi ini menyederhanakan kode Anda, tetapi tidak diperlukan. Manfaat menggunakan strategi ini adalah

dapat disuntikkan pada tingkat otorisasi dan diteruskan ke kode tingkat bawah yang membentuk kueri. Ini memisahkan kode yang menentukan identifier pelanggan (`TId`) dari logika kueri.

Kode contoh berikut menunjukkan query Gremlin untuk membaca data:

```
g.withStrategies(strategy1).V().hasLabel("Label1")
```

Kode sebelumnya setara dengan contoh berikut:

```
g.V().hasLabel("Label1").has("TId", "1")
```

Demikian juga, saat menulis data dengan menggunakan Gremlin, Anda dapat menggunakan kueri berikut:

```
g.withStrategies(strategy1).addV("Label1").property("Value").property(id, "Item_1")
```

Kode sebelumnya setara dengan contoh berikut, yang tidak menggunakan strategi partisi dan oleh karena itu memerlukan `"TId"` properti untuk ditulis secara eksplisit:

```
g.addV("Label1").property("TId", "1").property("Value").property(id, "Item_1")
```

Di OpenCypher, pustaka ini tidak ada. Anda bertanggung jawab untuk menulis dan memodifikasi kueri Anda untuk menambahkan pengenal penyewa sebagai properti pada node dan tepi. Contoh:

```
CREATE (n:Item {`~id`: 'Item_1', Value: '123456', TId: '1'})
CREATE (n:Item {`~id`: 'Item_2', Value: '123456', TId: '2'})
```

Perhatikan kesamaan antara kode Gremlin tanpa strategi partisi. Anda kemudian dapat membaca simpul yang ditulis dari `CREATE` pernyataan pertama dengan menggunakan kode berikut:

```
MATCH (n:Item {TId: '1'})
RETURN n
--or
MATCH (n:Item)
WHERE n.TId == '1'
RETURN n
```

Anda dapat memilih strategi properti ketika Anda ingin menggunakan konstruksi TinkerPop Gremlin asli seperti. PartitionStrategy Namun, model ini memiliki kelemahan kinerja di Amazon Neptunus

dibandingkan dengan strategi label awalan. Untuk diskusi tentang kelemahan kinerja ini, lihat bagian Implikasi kinerja untuk model LPG.

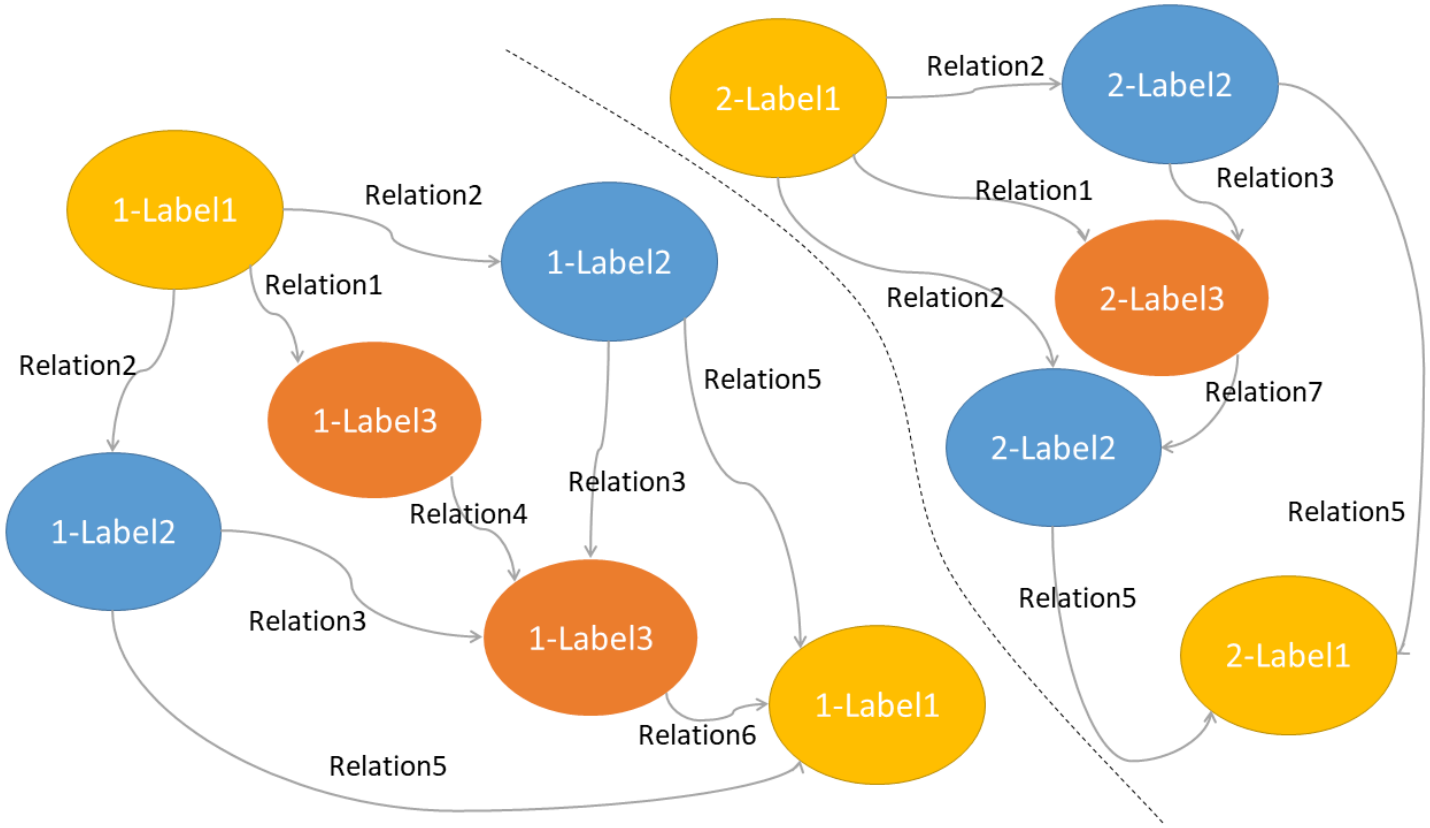
Jika kondisi berikut berlaku, pertimbangkan untuk memodelkan strategi properti hanya pada node, bukan di tepi:

- Grafik Anda memiliki tepi yang jauh lebih banyak daripada label.
- Setiap penyewa adalah grafik yang terputus.
- Anda mengakses grafik hanya dengan menggunakan node sebagai titik awal, bukan label.

Strategi label awalan

Jika kinerja menjadi perhatian utama, kami sangat menyarankan untuk mempertimbangkan strategi label awalan daripada strategi properti.

Dalam strategi label awalan, Anda memberi label pada setiap node dengan kombinasi pengenal penyewa dan label node. Misalnya, jika penyewa memiliki pengenal `"1"` dan label node`"Label1"`, Anda menentukan label node sebagai. `"1-Label1"` Diagram berikut menunjukkan dua subgraf terputus yang menggunakan model ini.



Saat menulis data di Gremlin, Anda dapat menambahkan nomor identifikasi ke label node apa pun:

```
g.addV("1-Label1")
g.addV("2-Label6")
```

Saat menanyakan grafik ini, Anda dapat memeriksa keberadaan awalan ini pada sebuah node:

```
g.V().hasLabel("1-Label1")
```

Di OpenCypher Anda dapat menulis data dengan menggunakan pernyataan: `CREATE`

```
CREATE (n:`1-Label1` {`~id`: 'Item_1', Value: 'XYZ123456'})
```

Untuk menanyakan data yang Anda tulis di OpenCypher, gunakan kode berikut:

```
MATCH n= (:`1-Label1`)
RETURN n
```

Strategi label awalan mengasumsikan bahwa semua node ditetapkan ke satu atau lebih penyewa dan bahwa izin tidak ditetapkan pada lingkup tepi. Hindari menggunakan strategi ini pada label tepi, karena itu akan menyebabkan sejumlah besar predikat dan akan berdampak negatif pada kinerja Neptunus.

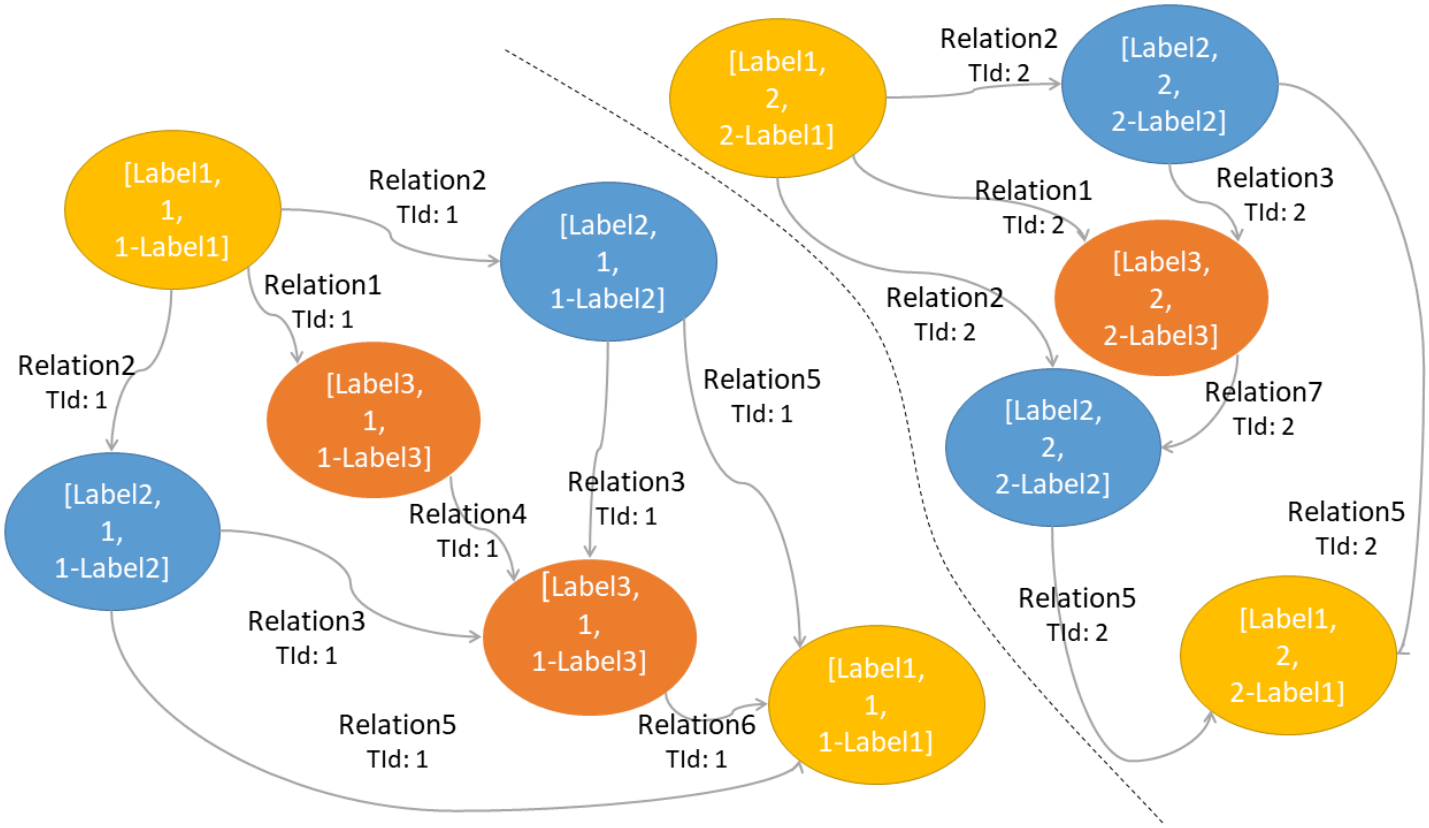
Ada dua kelemahan utama untuk pendekatan label awalan. Pertama, sulit untuk menjalankan kueri apa pun yang menjangkau penyewa. Contohnya adalah kueri yang menghitung semua node dari label yang diberikan untuk pelaporan atau pemantauan. Jika ini adalah kasus penggunaan Anda, pertimbangkan untuk menggabungkan strategi ini dengan strategi multi-label. Untuk informasi selengkapnya tentang menggabungkan strategi, lihat bagian Model hibrida.

Kedua, strategi label awalan memerlukan kontrol yang menerapkan penerapan awalan yang tepat untuk setiap kueri untuk mencegah kebocoran data. Namun, strategi ini adalah opsi paling efisien untuk beban kerja yang memerlukan kueri latensi rendah, dan kami sangat merekomendasikannya. Bagian Implikasi Kinerja untuk model LPG memberikan contoh mengapa ini adalah strategi yang paling efisien.

Strategi multi-label

Opsi ketiga adalah menggunakan strategi multi-label. Untuk pendekatan ini, Anda menambahkan label tambahan ke setiap node pada grafik. Misalnya, jika Anda perlu memfilter semua data untuk penyewa tertentu, tambahkan label ID penyewa. Jika Anda perlu memfilter semua data untuk label tertentu terlepas dari penyewa, tambahkan label itu. Diagram berikut menunjukkan strategi multi-label diterapkan dengan menggunakan tiga label untuk setiap node.

Anda sekarang dapat mengakses grafik dengan menggunakan tiga pola berbeda:



- Filter `Label1` untuk mengembalikan semua node dengan `Label1` seluruh penyewa.
- Filter 1 untuk mengembalikan semua node untuk penyewa 1.
- Filter `1-Label1` untuk mengembalikan semua node hanya untuk penyewa 1 dengan label`Label1`.

Sebab LPGs, ada dua cara untuk mengimplementasikan ini.

Di Gremlin, Anda dapat menggunakan strategi traversal yang dipanggil SubgraphStrategyuntuk membatasi ruang lingkup semua kueri hanya simpul dengan label tertentu, seperti: `"Label1"`

```
g.withStrategies(
    new SubgraphStrategy(
        vertices=hasLabel("Label1")
    )
)
```

Tidak seperti PartitionStrategy, SubgraphStrategy berdampak hanya membaca data, bukan menulis data. Untuk menulis data, tetapkan label secara manual di setiap kueri:

```
g.addV("Label1").property("Value","XYZ123456")
```

```
.addV("Label2").property("Value","XYZ123456")
```

Saat membaca data, Anda dapat menggunakan SubgraphStrategy untuk menanyakan semua node dengan`"Label1"`:

```
g.withStrategies(
new SubgraphStrategy(vertices=.hasLabel("Label1"))
).
V().has("Value","XYZ123456")
```

Neptunus hanya mengembalikan rekor pertama, yang `"Label1"` memiliki dan nilai. `"XYZ123456"` Ini setara dengan kueri berikut, yang tidak menggunakan SubgraphStrategy:

```
g.V().hasLabel("Label1").hasValue("XYZ123456")
```

Dalam kueri dasar ini, tampaknya SubgraphStrategy lebih kompleks untuk digunakan. Perlu diingat bahwa perpustakaan Anda dapat menyediakan contoh g dengan strategi yang sudah ditentukan. Pengembang tidak perlu memastikan bahwa filter yang tepat diterapkan:

```
def getGraphTraversal():
return g.withStrategies(new SubgraphStrategy(vertices=.hasLabel("Label1"))

getGraphTraversal().has("Value","XYZ123456")
```

Pustaka OpenCypher tidak memiliki konstruksi ini, jadi Anda harus membuat beberapa label untuk setiap node:

```
CREATE (n:`1`:`Label1`:`1-Label1` {`~id`: 'Item_1', Value: '12345'})
```

Saat Anda menggunakan label ini untuk memfilter subgraf, Anda dapat mengembalikan node yang memiliki label pelanggan yang Anda cari atau yang berbagi hubungan dengan node lain yang memiliki label itu:

```
MATCH n=(:Label1:`1`)
// or
MATCH n=(:`1-Label1`)
```

Strategi multi-label memberi Anda fleksibilitas paling besar untuk menanyakan node berdasarkan type (`Label1`) atau tenant (1), atau menggunakan strategi label awalan yang lebih efisien saat kinerja paling penting (). `1-Label1`

Kelemahan utama dari strategi ini adalah bahwa setiap label adalah objek tambahan yang disimpan dalam grafik Anda. Objek adalah simpul, tepi, atau properti pada simpul atau tepi di LPGs. Kecepatan konsumsi diukur dan diikat oleh objek per detik, dan biaya penyimpanan tergantung pada jumlah gigabyte yang dikonsumsi. Ini berarti bahwa objek tambahan mungkin memiliki dampak yang dapat diukur dalam skala besar.

Implikasi kinerja untuk model LPG

Kursus Pemodelan Data AWS Skill Builder untuk Amazon Neptunus menjelaskan secara mendalam internal model data Neptunus dan implikasi pemodelan, tetapi kami akan merangkum pertimbangan penting untuk desain ini di sini. Pertimbangkan untuk memiliki tiga penyewa (T1, T2, T3) pada satu cluster Neptunus. Penyewa ini memiliki atribut berikut:

- Tenant 1 (T1) memiliki total 100 juta node, dan 10 juta adalah tipe Item.
- Tenant 2 (T2) memiliki total 10 juta node, dan 1 juta adalah tipe Item.
- Tenant 3 (T3) memiliki total 100 juta node, dan 1 juta adalah tipe Item.

Jalankan kueri yang akan mengambil item untuk Tenant 3 dengan menggunakan strategi properti. Neptunus memeriksa statistik untuk dua panggilan indeks:

- Dimana `tenant property key=T3` memiliki 100 juta hasil
- Dimana `label = Item` memiliki 12 juta hasil (10 juta dari T1 + 1 juta dari T2 + 1 juta dari T3)

Pengoptimal kueri Neptunus menentukan bahwa kueri terakhir paling baik diterapkan terlebih dahulu (12 juta hasil) dan kemudian memeriksa setiap item. `tenant property key=T3` Anda mengambil 12 juta item untuk menemukan 1 juta hasil.

Perhatikan dampak tetangga yang bising dari kueri ini. Jika Anda memiliki 100 juta node Item per penyewa, kueri pertama akan memiliki 300 juta hasil, bukan 12 juta (Ini terlalu disederhanakan untuk tujuan ilustrasi. Pengoptimal Neptunus mungkin telah menerapkan urutan operasi yang berbeda).

Selanjutnya, pertimbangkan strategi label awalan. Buat panggilan indeks tunggal di mana`label=T3-Item`, yang mengembalikan 1 juta hasil. Ini mencapai hasil yang sama dengan strategi properti,

tetapi mengambil 11 juta catatan lebih sedikit. Selain itu, Anda tidak lagi memiliki masalah tetangga yang berisik karena label tidak tumpang tindih dalam indeks.

Strategi multi-label tidak memberikan peningkatan kinerja kueri atas strategi properti secara langsung. Pemfilteran berdasarkan nilai properti sebanding dengan pemfilteran berdasarkan nilai label saat ruang pencarian juga sebanding. Sebaliknya, strategi multi-label mendukung lebih banyak fleksibilitas. Strategi multi-label memberikan kinerja yang setara dengan strategi label awalan untuk atau label. `label=T3` `T3-Item` Strategi multi-label memberikan kinerja yang setara dengan strategi properti untuk. `label=Item` Manfaatnya adalah mendukung berbagai pola akses.

Model kolam renang untuk RDF

Resource Description Framework (RDF) memiliki konsep grafik bernama, yang menyediakan cara logis untuk memisahkan data. Di Amazon Neptunus, Anda memiliki grafik bernama default dan grafik bernama yang ditentukan pengguna. Anda dapat membuat grafik bernama sebanyak yang Anda inginkan. Secara kolektif, mereka disebut dataset RDF. Semua grafik bernama, default atau yang ditentukan pengguna, didefinisikan oleh Internationalized Resource Identifier (IRI) dalam dataset RDF. Di Neptunus, kecuali pengguna mendeklarasikan grafik bernama saat menulis data, semua tripel dianggap sebagai bagian dari grafik bernama default.

Ada beberapa kasus penggunaan untuk grafik bernama:

- Partisi data dan isolasi data
- Asal data
- Penentuan versi
- Inferensi

Panduan ini berfokus pada kasus penggunaan partisi data. Sebaiknya buat satu grafik bernama yang ditentukan pengguna untuk setiap penyewa.

Opsi kueri SPARQL menggunakan Protokol HTTP Graph Store

Contoh query berikut menggunakan SPARQL Protocol dan RDF Query Language (SPARQL) dan Graph Store HTTP Protocol untuk query atau membuat grafik bernama untuk penyewa.

- `HTTP GET`Untuk mengambil grafik tertentu dari penyewa:

```
curl --request GET 'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//
www.example.com/named/tenant1'
```

- `HTTP PUT`Untuk membuat atau mengganti grafik bernama tertentu dengan muatan yang ditentukan dalam permintaan:

```
curl --request PUT -H "Content-Type: text/turtle" \  --data-raw "@prefix ex: http://
example.com/ . ex:subject ex:predicate ex:object ." \
'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//www.example.com/named/
tenant1'
```

  Dalam RDF, sebuah objek adalah rangkap tiga.

- `HTTP POST`Untuk membuat grafik bernama baru jika tidak ada, atau bergabung dengan grafik yang ada:

```
curl --request POST -H "Content-Type: text/turtle" \
--data-raw "@prefix ex: http://example.com/ . ex:subject ex:predicate ex:object ." \

'https://your-neptune-endpoint:port/sparql/gsp/?graph=http%3A//www.example.com/named/
tenant1'
```

Isolasi penyewa untuk RDF

Untuk isolasi logis data dengan pagar pembatas yang diperlukan di lapisan aplikasi, buat pemetaan antara grafik bernama penyewa dan yang ditentukan pengguna. Saat Anda merancang multi-tenancy untuk kumpulan data RDF, perhatikan aspek-aspek RDF dan SPARQL berikut:

- Di Neptunus, ketika Anda menanyakan tanpa menentukan grafik bernama, ia mengambil semua tiga kali lipat yang cocok dengan pola di semua grafik bernama dalam database.
- Dalam RDF, tidak ada kendala di sekitar koneksi antara node dari grafik bernama yang berbeda. Misalnya, pada diagram sebelumnya, simpul di `:G1` dapat dihubungkan ke simpul di: `G2` melalui tepi.

Misalnya, jika pengguna akhir penyewa tertentu mengirimkan kueri ke API, API harus memvalidasi persyaratan berikut sebelum mengirimkan kueri ke database Neptunus:

- Setiap kueri yang dicakup pada penyewa tunggal harus menentukan grafik bernama. Jika tidak, Anda berisiko membocorkan data di seluruh penyewa.
- Perbarui atau Hapus kueri harus selalu menentukan grafik bernama.
- Node di kedua sisi tepi atau hubungan harus selalu milik grafik bernama yang benar.

Untuk informasi tambahan tentang praktik terbaik, lihat dokumentasi Neptunus.

Bersiaplah untuk pertumbuhan

Saat Anda berhasil menggunakan model kumpulan, Anda akhirnya melebihi ukuran satu cluster Neptunus. Penyewa bertambah, atau jumlah penyewa bertambah, dan tingkat konsumsi data yang dibutuhkan di semua pelanggan Anda melebihi kemampuan cluster. Ketika ini terjadi, Anda perlu membagi pelanggan Anda di beberapa cluster. Desain untuk konfigurasi ini di muka alih-alih mencoba retrofit untuk itu nanti. Bahkan jika skala awal Anda hanya menggunakan satu cluster, tiruan komponen yang Anda perlukan untuk merutekan penyewa melintasi beberapa cluster di masa mendatang ketika Anda mencapai skala itu.

Jika solusi Anda membutuhkan lebih banyak sumber daya berdasarkan ukuran penyewa Anda, bersiaplah untuk pertumbuhannya juga. Jika beberapa pelanggan pada satu cluster tumbuh secara signifikan, cluster itu mungkin tidak lagi mendukung kebutuhan Anda. Rancang strategi untuk memindahkan penyewa ke cluster lain atau membagi cluster yang ada menjadi dua dengan menggunakan fitur kloning Amazon Neptunus DB.

Biasakan diri dengan Protokol Copy-on-Write Neptunus, yang dapat menghemat uang Anda saat Anda menerapkan kloning DB., Jika Anda membagi cluster karena kemacetan konsumsi, mungkin lebih efisien untuk tidak menghapus data dari cluster, asalkan kebijakan Anda mengizinkannya. Kedua cluster akan berbagi halaman data jika tidak berubah tetapi tidak jika halaman data dimodifikasi (karena beberapa data di dalamnya telah dihapus).

> **Note**
>
> Panduan ini berlaku untuk versi Neptunus terbaru pada saat penulisan ini, yaitu Neptunus versi 1.3.1. Panduan ini mungkin berubah di versi masa depan saat lapisan penyimpanan Neptunus berevolusi.

Keterbatasan untuk skenario multi-tenancy

Ketahuilah bahwa beberapa fitur Neptunus tidak dibuat untuk skenario multi-tenancy. Penyewa tidak boleh diberikan akses langsung ke titik akhir Neptunus dalam model kumpulan karena strategi multi-tenancy ini tidak diberlakukan pada tingkat database. Selalu simpan semacam proxy antara pelanggan Anda dan titik akhir Neptunus yang memberlakukan desain yang dijelaskan dalam dokumen ini. Contoh proxy tersebut meliputi yang berikut:

- Menambahkan filter label di lapisan klien Anda
- Memiliki API yang memetakan token otentikasi ke ID penyewa dan menyuntikkan filter ini ke dalam kueri

Panduan ini juga berlaku untuk memberi pelanggan akses langsung ke fitur-fitur seperti notebook grafik Neptunus, penjelajah grafik Neptunus, atau AliranNeptunus.

Multi-penyewa model Hybrid

Solusi SaaS sering menggunakan campuran model silo dan kolam renang. Berbagai faktor mempengaruhi keputusan kapan dan bagaimana menggunakan model silo dan kolam dalam lingkungan yang sama.

Salah satu faktor tersebut adalah tiering, di mana solusi SaaS menawarkan pengalaman unik untuk setiap tingkat penyewa. Misalnya, jika tingkatan Anda Gratis, Standar, dan Premium, data penyewa tingkat Gratis Anda dapat disimpan di kluster Neptunus bersama menggunakan model kumpulan. Untuk penyewa tingkat Standar dan Premium, Anda dapat menggunakan model cluster-per-tenant silo.

Selain itu, beberapa penyedia SaaS memiliki kemampuan untuk membangun solusi kumpulan mereka di cluster Amazon Neptunus bersama sebagai fondasi mereka. Selanjutnya, mereka dapat membuat cluster Neptunus terpisah untuk penyewa yang membutuhkan penyimpanan silo, seringkali karena kepatuhan dan mandat peraturan.

Meskipun ini dapat menambah tingkat kerumitan pada lapisan akses data dan profil manajemen Anda, ini juga dapat menawarkan bisnis Anda cara untuk meningkatkan penawaran Anda untuk memenuhi kebutuhan pelanggan.

Praktik terbaik operasional untuk ISVs

Banyak pedoman di bagian ini adalah praktik terbaik untuk semua pelanggan, tetapi mereka telah menambahkan signifikansi untuk ISVs.

Perbarui cluster Neptunus Anda dengan versi terbaru

Dalam catatan rilis Amazon Neptunus, Anda dapat melihat bahwa setiap versi membawa sejumlah perbaikan bug, peningkatan kinerja, dan fitur baru. Simpan cluster Neptunus Anda pada versi terbaru sebanyak mungkin.

Jika Anda menemukan bug yang sebelumnya belum ditemukan di beban kerja Anda dan klaster Anda ada di versi terbaru, teknisi Neptunus dapat membuat tambalan pribadi untuk klaster Anda (jika diperlukan dan Anda menginginkannya). Patch dapat menjembatani hingga rilis berikutnya ketika perbaikan itu akan tersedia secara umum. Untuk membantu memperbarui cluster Anda ke versi terbaru, gunakan solusi Neptunus Biru/Hijau.

Gunakan delta alih-alih hapus dan ganti untuk konsumsi data

Anda dapat menggunakan beberapa teknik untuk menelan, atau menulis, data ke Neptunus. Banyak pelanggan mencoba menyederhanakan konsumsi data mereka dengan menghapus dan memasukkan kembali grafik mereka setiap kali perubahan diterima di umpan. Mereka mungkin menambahkan `last-modified` properti ke setiap node dan secara berkala memindai node yang belum dimodifikasi sejak beberapa tanggal tertentu dan menghapusnya. Meskipun teknik ini menyederhanakan proses konsumsi data, mereka memiliki implikasi kesehatan dan skalabilitas jangka panjang untuk cluster Neptunus Anda.

Pertama, Neptunus menggunakan pengkodean string kamus. Kecuali Anda secara eksplisit menentukan node dan tepi, Neptunus menghasilkan GUID yang direpresentasikan sebagai string untuk ID dan menyimpan string itu dalam kamus. IDs Jika Anda terus-menerus menghapus dan menambahkan objek, yang dihasilkan secara otomatis IDs akan menyebabkan kembung di kamus.

Kedua, skala Neptunus hingga menelan sekitar 120 K objek per detik secara maksimal. Jika Anda terus menghapus dan menambahkan objek, Anda mengkonsumsi banyak bandwidth pada objek yang pada dasarnya tidak berubah. Ini membatasi jumlah penyewa yang dapat Anda host di cluster, membutuhkan instance penulis yang lebih besar di cluster, dan membutuhkan lebih banyak operasi. I/O Semua faktor ini meningkatkan biaya Anda.

Kami sangat menyarankan Anda mengembangkan cara untuk menghitung delta sebenarnya dari apa yang telah berubah alih-alih menggunakan metode hapus dan tambahkan. Namun, beberapa sumber data tidak kondusif untuk ini (misalnya, panggilan API yang mengembalikan status saat ini, atau peristiwa yang tidak melacak persis apa yang berubah). Jika sumber data mentah Anda tidak kondusif untuk mengidentifikasi perubahan, gunakan proses ekstrak, transformasi, dan muat (ETL) Anda untuk menghitung delta. Misalnya, Anda dapat mempertahankan snapshot dari setiap pengambilan data sebelumnya dalam format Parket, gunakan AWS Glue untuk menghitung perbedaan antara snapshot tersebut, dan hanya mendorong perbedaan ke Neptunus.

Model bagaimana biaya Neptunus akan berkembang dengan penyewa Anda

Baik Anda menggunakan model silo, kolam renang, atau hibrida, biaya cloud Anda akan disesuaikan dengan ukuran penyewa Anda. Penyewa yang membutuhkan lebih banyak koneksi bersamaan membutuhkan instance yang lebih besar atau lebih banyak replika baca daripada yang memiliki koneksi bersamaan yang lebih sedikit. Hal yang sama berlaku untuk penyewa yang membutuhkan konsumsi data lebih cepat.

Tiga komponen biaya cluster Neptunus adalah ukuran instans (dan jumlah), ukuran data (GB-bulan), I/O dan operasi (per juta). Meskipun biaya-biaya ini umumnya spesifik untuk beban kerja, mereka menskalakan dengan ukuran dan volume data, mereka dapat diukur dengan menggunakan AWS alat. Lacak dan pahami skala ekonomi terhadap indikator utama ukuran penyewa Anda, termasuk bagaimana ukurannya bervariasi dari waktu ke waktu. Jika I/O biaya yang tidak dapat diprediksi memengaruhi margin Anda, pertimbangkan untuk memilih penyimpanan Neptunus I/O-Optimized untuk biaya yang lebih dapat diprediksi.

Skala cluster Anda untuk permintaan pelanggan

Tidak ada formula yang dicoba atau benar untuk mengukur ukuran instans Neptunus Anda dengan benar. Dokumentasi Neptunus memberikan panduan, tetapi ada terlalu banyak variabel untuk merekomendasikan pemetaan langsung. Variabel-variabel ini termasuk tetapi tidak terbatas pada yang berikut:

- Model Data
- Bentuk data
- Konkurensi kueri

- Kompleksitas kueri.

Rencanakan pengujian untuk menentukan ukuran optimal untuk beban kerja dan profil penyewa Anda. Secara umum, kami merekomendasikan penggunaan instans yang disediakan untuk efisiensi biaya dan prediktabilitas. Jika sasaran pengalaman pelanggan Anda memprioritaskan penskalaan optimal daripada biaya, pertimbangkan untuk menggunakan instans Neptunus Tanpa Server untuk memastikan pengalaman yang lebih konsisten terlepas dari fluktuasi beban kerja.

Jika beban kerja baca penyewa Anda memiliki variabilitas yang signifikan di puncak dan palungnya, gabungkan instance Neptunus Tanpa Server dengan auto-scaling Neptunus. Biasanya dibutuhkan waktu 10-15 menit untuk replika baca baru untuk online setelah diinisialisasi. Ini berarti bahwa auto-scaling saja dapat menangani perubahan lalu lintas yang berkepanjangan, tetapi itu tidak cukup untuk mengubah lonjakan aktivitas dengan cepat. Dengan menggabungkan Neptunus Tanpa Server dan auto-scaling Neptunus, Anda dapat menskalakan instance naik atau turun dan menskalakan jumlah replika baca masuk dan keluar.

Jika penyewa Anda memiliki profil beban kerja atau perjanjian tingkat layanan (SLAs) yang berbeda secara signifikan, pertimbangkan untuk menggunakan titik akhir khusus dan replika baca khusus untuk mengarahkan lalu lintas ke instans yang dioptimalkan untuk lalu lintas tersebut. Optimasi dapat mencakup ukuran instance yang berbeda, pola kueri tertentu, atau pra-pemanasan cache buffer.

Langkah selanjutnya

Jika Anda baru memulai perjalanan Anda menerapkan Amazon Neptunus untuk aplikasi ISV multi-tenancy Anda, berikan pemikiran ekstra ke dalam model yang Anda inginkan. Mengubah model akan lebih mahal di kemudian hari dalam perjalanan Anda.

Jika Anda berada di awal perjalanan Anda, verifikasi bahwa Anda menggunakan model terbaik untuk kebutuhan Anda dan bahwa Anda mengikuti panduan untuk model itu.

Rencana ke depan. Saat Anda berada di awal perjalanan, tergoda untuk menunda pekerjaan sharding pelanggan di seluruh cluster atau mengoptimalkan ETL proses Anda untuk memberikan delta perubahan alih-alih menghapus dan menambahkan kembali simpul dan tepi. Saat Anda menskalakan, keputusan tersebut dapat berdampak negatif pada kinerja dan biaya.

Akhirnya, jika Anda sudah memasuki perjalanan Anda, panduan ini mungkin meyakinkan Anda bahwa arsitektur Anda optimal, atau mungkin memberikan perubahan untuk meningkatkan arsitektur Anda.

Jika Anda memiliki pertanyaan tentang panduan ini atau memerlukan bantuan lebih lanjut, silakan hubungi Akun AWS tim Anda dan mintalah sesi dengan spesialis Neptunus.

Sumber daya

- Dokumentasi Amazon Neptunus
- Pemodelan Data untuk Amazon Neptunus (kursus)
- Menerapkan Kerangka Kerja AWS Well-Architected untuk Amazon Neptunus
- Kerangka Kerja Well-Architected Lensa SaaS
- Panduan untuk Arsitektur Multi-Tenant pada AWS
- Strategi Isolasi Penyewa SaaS: Mengisolasi Sumber Daya di Lingkungan Multi-Penyewa
- Dokumentasi Apache TinkerPop
- SPARQL

Kontributor

Kontributor untuk panduan ini meliputi:

- Brian O'Keefe, Kepala Sekolah Perang Dunia Neptunus, SSA AWS
- Veeresham Gande, Manajer Akun Teknis Senior, AWS
- Dana Owens, Arsitek Solusi Startup, AWS
- Nima Seifi, Arsitek Solusi Startup, AWS

Riwayat dokumen

Tabel berikut menjelaskan perubahan signifikan pada panduan ini. Jika Anda ingin diberi tahu tentang pembaruan masa depan, Anda dapat berlangganan RSSumpan.

Perubahan	Deskripsi	Tanggal
Publikasi awal	—	3 September 2024

AWS Glosarium Panduan Preskriptif

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

Nomor

7 Rs

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

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

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

A

ABAC

Lihat kontrol akses berbasis atribut.

layanan abstrak

Lihat layanan terkelola.

ASAM

Lihat atomisitas, konsistensi, isolasi, daya tahan.

migrasi aktif-aktif

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

migrasi aktif-pasif

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

fungsi agregat

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

AI

Lihat kecerdasan buatan.

AIOps

Lihat operasi kecerdasan buatan.

anonimisasi

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

anti-pola

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

kontrol aplikasi

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

portofolio aplikasi

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

kecerdasan buatan (AI)

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

operasi kecerdasan buatan (AIOps)

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

enkripsi asimetris

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

atomisitas, konsistensi, isolasi, daya tahan (ACID)

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

kontrol akses berbasis atribut (ABAC)

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

sumber data otoritatif

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

Zona Ketersediaan

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

AWS Kerangka Adopsi Cloud (AWS CAF)

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

AWS Kerangka Kualifikasi Beban Kerja (AWS WQF)

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

B

bot buruk

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

BCP

Lihat perencanaan kontinuitas bisnis.

grafik perilaku

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

sistem big-endian

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

klasifikasi biner

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

filter mekar

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

deployment biru/hijau

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

bot

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

botnet

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

cabang

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

akses break-glass

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

strategi brownfield

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

cache penyangga

Area memori tempat data yang paling sering diakses disimpan.

kemampuan bisnis

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

perencanaan kelangsungan bisnis (BCP)

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

C

KAFE

Lihat Kerangka Adopsi AWS Cloud.

penyebaran kenari

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

CCoE

Lihat Cloud Center of Excellence.

CDC

Lihat mengubah pengambilan data.

ubah pengambilan data (CDC)

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

rekayasa kekacauan

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

CI/CD

Lihat integrasi berkelanjutan dan pengiriman berkelanjutan.

klasifikasi

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

Enkripsi sisi klien

Enkripsi data secara lokal, sebelum target Layanan AWS menerimanya.

Pusat Keunggulan Cloud (CCoE)

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

komputasi cloud

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

model operasi cloud

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

tahap adopsi cloud

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

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

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

CMDB

Lihat database manajemen konfigurasi.

repositori kode

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

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

cache dingin

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

data dingin

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

visi komputer (CV)

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

konfigurasi drift

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

database manajemen konfigurasi (CMDB)

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

paket kesesuaian

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

integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD)

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

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

CV

Lihat visi komputer.

D

data saat istirahat

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

klasifikasi data

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

penyimpangan data

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

data dalam transit

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

jala data

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

minimalisasi data

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

perimeter data

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

prapemrosesan data

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

asal data

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

subjek data

Individu yang datanya dikumpulkan dan diproses.

gudang data

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

bahasa definisi database (DDL)

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

bahasa manipulasi basis data (DHTML)

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

DDL

Lihat bahasa definisi database.

ansambel yang dalam

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

pembelajaran mendalam

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

defense-in-depth

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

administrator yang didelegasikan

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

deployment

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

lingkungan pengembangan

Lihat lingkungan.

kontrol detektif

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

pemetaan aliran nilai pengembangan (DVSM)

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

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

kembar digital

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

tabel dimensi

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

musibah

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

pemulihan bencana (DR)

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

DML~

Lihat bahasa manipulasi basis data.

desain berbasis domain

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

DR

Lihat pemulihan bencana.

deteksi drift

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

DVSM

Lihat pemetaan aliran nilai pengembangan.

E

EDA

Lihat analisis data eksplorasi.

EDI

Lihat pertukaran data elektronik.

komputasi tepi

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

pertukaran data elektronik (EDI)

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

enkripsi

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

kunci enkripsi

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

endianness

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

titik akhir

Lihat titik akhir layanan.

layanan endpoint

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

perencanaan sumber daya perusahaan (ERP)

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

enkripsi amplop

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

lingkungan

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

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

- lingkungan atas — Semua lingkungan yang dapat diakses oleh pengguna selain tim pengembangan inti. Ini dapat mencakup lingkungan produksi, lingkungan praproduksi, dan lingkungan untuk pengujian penerimaan pengguna.

epik

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

ERP

Lihat perencanaan sumber daya perusahaan.

analisis data eksplorasi (EDA)

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

F

tabel fakta

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

gagal cepat

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

batas isolasi kesalahan

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

cabang fitur

Lihat cabang.

fitur

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

pentingnya fitur

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

transformasi fitur

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

beberapa tembakan mendorong

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

FGAC

Lihat kontrol akses berbutir halus.

kontrol akses berbutir halus (FGAC)

Penggunaan beberapa kondisi untuk mengizinkan atau menolak permintaan akses.

migrasi flash-cut

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

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

FM

Lihat model pondasi.

model pondasi (FM)

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

G

AI generatif

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

pemblokiran geografis

Lihat pembatasan geografis.

pembatasan geografis (pemblokiran geografis)

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

Alur kerja Gitflow

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

gambar emas

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

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

strategi greenfield

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

pagar pembatas

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

H

HA

Lihat ketersediaan tinggi.

migrasi database heterogen

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

ketersediaan tinggi (HA)

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

modernisasi sejarawan

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

data penahanan

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

migrasi database homogen

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

data panas

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

perbaikan terbaru

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

periode hypercare

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

I

IAc

Lihat infrastruktur sebagai kode.

kebijakan berbasis identitas

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

aplikasi idle

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

IIoT

Lihat Internet of Things industri.

infrastruktur yang tidak dapat diubah

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

masuk (masuknya) VPC

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

migrasi inkremental

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

Industri 4.0

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

infrastruktur

Semua sumber daya dan aset yang terkandung dalam lingkungan aplikasi.

infrastruktur sebagai kode (IAc)

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

Internet of Things industri (IIoT)

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

inspeksi VPC

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

Internet of Things (IoT)

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

interpretabilitas

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

IoT

Lihat Internet of Things.

Perpustakaan informasi TI (ITIL)

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

Manajemen layanan TI (ITSM)

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

ITIL

Lihat perpustakaan informasi TI.

ITSM

Lihat manajemen layanan TI.

L

kontrol akses berbasis label (LBAC)

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

landing zone

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

model bahasa besar (LLM)

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

migrasi besar

Migrasi 300 atau lebih server.

LBAC

Lihat kontrol akses berbasis label.

hak istimewa paling sedikit

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

angkat dan geser

Lihat 7 Rs.

sistem endian kecil

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

LLM

Lihat model bahasa besar.

lingkungan yang lebih rendah

Lihat lingkungan.

M

pembelajaran mesin (ML)

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

cabang utama

Lihat cabang.

malware

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

layanan terkelola

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

sistem eksekusi manufaktur (MES)

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

PETA

Lihat Program Percepatan Migrasi.

mekanisme

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

akun anggota

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

MES

Lihat sistem eksekusi manufaktur.

Transportasi Telemetri Antrian Pesan (MQTT)

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

layanan mikro

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

arsitektur microservices

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

Program Percepatan Migrasi (MAP)

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

migrasi dalam skala

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

pabrik migrasi

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

metadata migrasi

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

pola migrasi

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

Penilaian Portofolio Migrasi (MPA)

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

Penilaian Kesiapan Migrasi (MRA)

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

strategi migrasi

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

ML

Lihat pembelajaran mesin.

modernisasi

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

penilaian kesiapan modernisasi

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

aplikasi monolitik (monolit)

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

MPA

Lihat Penilaian Portofolio Migrasi.

MQTT

Lihat Transportasi Telemetri Antrian Pesan.

klasifikasi multiclass

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

infrastruktur yang bisa berubah

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

O

OAC

Lihat kontrol akses asal.

OAI

Lihat identitas akses asal.

OCM

Lihat manajemen perubahan organisasi.

migrasi offline

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

OI

Lihat integrasi operasi.

OLA

Lihat perjanjian tingkat operasional.

migrasi online

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

OPC-UA

Lihat Komunikasi Proses Terbuka - Arsitektur Terpadu.

Komunikasi Proses Terbuka - Arsitektur Terpadu (OPC-UA)

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

perjanjian tingkat operasional (OLA)

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

Tinjauan Kesiapan Operasional (ORR)

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

teknologi operasional (OT)

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

integrasi operasi (OI)

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

jejak organisasi

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

manajemen perubahan organisasi (OCM)

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

kontrol akses asal (OAC)

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

identitas akses asal (OAI)

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

ORR

Lihat tinjauan kesiapan operasional.

OT

Lihat teknologi operasional.

keluar (jalan keluar) VPC

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

P

batas izin

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

Informasi Identifikasi Pribadi (PII)

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

PII

Lihat informasi yang dapat diidentifikasi secara pribadi.

buku pedoman

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

PLC

Lihat pengontrol logika yang dapat diprogram.

PLM

Lihat manajemen siklus hidup produk.

kebijakan

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

ketekunan poliglot

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

penilaian portofolio

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

predikat

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

predikat pushdown

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

kontrol preventif

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

principal

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

privasi berdasarkan desain

Pendekatan rekayasa sistem yang memperhitungkan privasi melalui seluruh proses pengembangan.

zona yang dihosting pribadi

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

kontrol proaktif

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

manajemen siklus hidup produk (PLM)

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

lingkungan produksi

Lihat lingkungan.

pengontrol logika yang dapat diprogram (PLC)

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

rantai cepat

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

pseudonimisasi

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

publish/subscribe (pub/sub)

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

Q

rencana kueri

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

regresi rencana kueri

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

R

Matriks RACI

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

LAP

Lihat Retrieval Augmented Generation.

ransomware

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

Matriks RASCI

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

RCAC

Lihat kontrol akses baris dan kolom.

replika baca

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

arsitek ulang

Lihat 7 Rs.

tujuan titik pemulihan (RPO)

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

tujuan waktu pemulihan (RTO)

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

refactor

Lihat 7 Rs.

Region

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

regresi

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

rehost

Lihat 7 Rs.

melepaskan

Dalam proses penyebaran, tindakan mempromosikan perubahan pada lingkungan produksi.

memindahkan

Lihat 7 Rs.

memplatform ulang

Lihat 7 Rs.

pembelian kembali

Lihat 7 Rs.

ketahanan

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

kebijakan berbasis sumber daya

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

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

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

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

kontrol responsif

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

melestarikan

Lihat 7 Rs.

pensiun

Lihat 7 Rs.

Retrieval Augmented Generation (RAG)

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

rotasi

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

kontrol akses baris dan kolom (RCAC)

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

RPO

Lihat tujuan titik pemulihan.

RTO

Lihat tujuan waktu pemulihan.

buku runbook

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

D

SAML 2.0

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

SCADA

Lihat kontrol pengawasan dan akuisisi data.

SCP

Lihat kebijakan kontrol layanan.

Rahasia

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

keamanan dengan desain

Pendekatan rekayasa sistem yang memperhitungkan keamanan melalui seluruh proses pengembangan.

kontrol keamanan

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

pengerasan keamanan

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

sistem informasi keamanan dan manajemen acara (SIEM)

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

otomatisasi respons keamanan

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

enkripsi sisi server

Enkripsi data di tujuannya, oleh Layanan AWS yang menerimanya.

kebijakan kontrol layanan (SCP)

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

titik akhir layanan

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

perjanjian tingkat layanan (SLA)

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

indikator tingkat layanan (SLI)

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

tujuan tingkat layanan (SLO)

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

model tanggung jawab bersama

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

SIEM

Lihat informasi keamanan dan sistem manajemen acara.

titik kegagalan tunggal (SPOF)

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

SLA

Lihat perjanjian tingkat layanan.

SLI

Lihat indikator tingkat layanan.

SLO

Lihat tujuan tingkat layanan.

split-and-seed model

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

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

SPOF

Lihat satu titik kegagalan.

skema bintang

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

pola ara pencekik

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

subnet

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

kontrol pengawasan dan akuisisi data (SCADA)

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

enkripsi simetris

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

pengujian sintetis

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

sistem prompt

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

T

tag

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

variabel target

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

daftar tugas

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

lingkungan uji

Lihat lingkungan.

pelatihan

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

gerbang transit

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

alur kerja berbasis batang

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

akses tepercaya

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

penyetelan

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

tim dua pizza

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

U

waswas

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

tugas yang tidak terdiferensiasi

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

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

lingkungan atas

Lihat lingkungan.

V

menyedot debu

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

kendali versi

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

Peering VPC

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

kerentanan

Kelemahan perangkat lunak atau perangkat keras yang membahayakan keamanan sistem.

W

cache hangat

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

data hangat

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

fungsi jendela

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

beban kerja

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

aliran kerja

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

CACING

Lihat menulis sekali, baca banyak.

WQF

Lihat AWS Kerangka Kualifikasi Beban Kerja.

tulis sekali, baca banyak (WORM)

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

Z

eksploitasi zero-day

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

kerentanan zero-day

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

bisikan zero-shot

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

aplikasi zombie

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

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