Panduan AWS CDK lapisan

AWS Bimbingan Preskriptif

AWS Bimbingan Preskriptif: Panduan AWS CDK lapisan

Table of Contents

Panduan AWS CDK lapisan

Steven Guggenheimer, Amazon Web Services ()AWS

Desember 2023 (riwayat dokumen)

Salah satu konsep utama di balik AWS Cloud Development Kit (AWS CDK) ini sangat mirip dengan konsep di balik tetap hangat di hari yang dingin. Konsep itu disebut layering. Pada hari yang dingin Anda mengenakan kemeja, jaket, dan terkadang jaket yang lebih besar tergantung pada seberapa dinginnya. Kemudian jika Anda masuk ke dalam dan pemanas menyala, Anda dapat melepas satu atau kedua lapisan jaket sehingga Anda tidak terlalu panas. AWS CDK Menggunakan layering untuk memberikan tingkat abstraksi yang berbeda untuk menggunakan komponen cloud. Layering memastikan bahwa Anda tidak perlu menulis terlalu banyak kode atau memiliki terlalu sedikit akses ke properti sumber daya saat Anda menyebarkan infrastruktur sebagai tumpukan kode (IAC).

Jika Anda tidak menggunakan AWS CDK, Anda harus menulis AWS CloudFormationtemplate Anda dengan tangan; yaitu, Anda hanya memanfaatkan satu lapisan yang memaksa Anda untuk menulis kode jauh lebih banyak daripada yang biasanya diperlukan. Di sisi lain, jika AWS CDK mengabstraksikan segala sesuatu CloudFormation yang biasanya tidak perlu Anda tulis, Anda tidak akan dapat menangani kasus tepi apa pun.

Untuk mengatasi masalah ini, penyediaan sumber AWS CDK daya dibagi menjadi tiga lapisan terpisah dan berbeda:

- Layer 1 — CloudFormationLapisan: Lapisan paling dasar di mana CloudFormation sumber daya dan AWS CDK sumber daya hampir identik.
- Layer 2 - Lapisan yang dikurasi: Lapisan tempat CloudFormation sumber daya diabstraksikan ke dalam kelas terprogram yang merampingkan sebagian besar sintaks boilerplate di bawah CloudFormation tenda. Lapisan ini membentuk sebagian besar AWS CDK.
- Layer 3 - Lapisan pola: Lapisan paling abstrak di mana Anda dapat menggunakan blok bangunan yang disediakan oleh lapisan 1 dan 2 untuk menyesuaikan kode untuk kasus penggunaan spesifik Anda.

Setiap item dari setiap lapisan adalah instance dari AWS CDK kelas khusus yang disebut a`Construct`. Menurut AWS dokumentasi, konstruksi adalah “blok bangunan dasar AWS CDK aplikasi. Sebuah konstruksi mewakili 'komponen cloud' dan merangkum semua yang AWS CloudFormation dibutuhkan untuk membuat komponen. Konstruksi dalam lapisan ini dikenal sebagai

konstruksi L1, L2, dan L3 tergantung pada lapisan mana mereka berada. Dalam panduan ini kita akan melakukan tur melalui setiap AWS CDK lapisan untuk mencari tahu untuk apa mereka digunakan dan mengapa itu penting.

Panduan ini ditujukan untuk manajer teknis, prospek, dan pengembang yang tertarik untuk menggali lebih dalam konsep inti yang membuat AWS CDK pekerjaan. Ini AWS CDK adalah alat yang populer, tetapi sangat umum bagi tim untuk kehilangan sebagian besar dari apa yang ditawarkannya. Ketika Anda mulai memahami konsep yang dijelaskan dalam panduan ini, Anda dapat membuka dunia kemungkinan yang sama sekali baru dan mengoptimalkan proses penyediaan sumber daya tim Anda.

Dalam panduan ini:

Konstruksi lapisan 1

Konstruksi L1 adalah blok bangunan AWS CDK dan mudah dibedakan dari konstruksi lain dengan awalan. `Cfn` Misalnya, paket Amazon DynamoDB dalam berisi konstruksi AWS CDK , `Table` yang merupakan konstruksi L2. Konstruksi L1 yang sesuai dipanggil`CfnTable`, dan secara langsung mewakili CloudFormation DynamoDB. `Table` Tidak mungkin menggunakan AWS CDK tanpa mengakses lapisan pertama ini, meskipun AWS CDK aplikasi biasanya tidak pernah menggunakan konstruksi L1 secara langsung. Namun, dalam sebagian besar kasus, konstruksi L2 dan L3 yang biasa digunakan pengembang sangat bergantung pada konstruksi L1. Jadi Anda dapat menganggap konstruksi L1 sebagai jembatan antara CloudFormation dan. AWS CDK

Satu-satunya tujuan dari AWS CDK ini adalah untuk menghasilkan CloudFormation template dengan menggunakan bahasa pengkodean standar. Setelah Anda menjalankan perintah cdk synth CLI dan CloudFormation template yang dihasilkan dihasilkan, pekerjaan selesai. AWS CDK Perintah cdk deploy ada di sana untuk kenyamanan, tetapi apa yang Anda lakukan ketika Anda menjalankan perintah itu terjadi sepenuhnya di dalam. CloudFormation Bagian dari teka-teki yang menerjemahkan AWS CDK kode ke dalam format yang CloudFormation dipahami adalah konstruksi L1.

AWS CDK CloudFormation Siklus hidup untuk konstruksi L1

Proses untuk membuat dan menggunakan konstruksi L1 terdiri dari langkah-langkah berikut:

1. Proses AWS CDK build mengubah CloudFormation spesifikasi menjadi kode terprogram dalam bentuk konstruksi L1.
2. Pengembang menulis kode yang secara langsung atau tidak langsung mereferensikan konstruksi L1 sebagai bagian dari aplikasi. AWS CDK
3. Pengembang menjalankan perintah cdk synth untuk mengonversi kode terprogram kembali ke format yang ditentukan oleh CloudFormation spesifikasi (templat).
4. Pengembang menjalankan perintah cdk deploy untuk menyebarkan CloudFormation tumpukan dalam templat ini ke lingkungan akun. AWS

Mari kita lakukan sedikit latihan. Pergi ke repositori AWS CDK open source pada GitHub, pilih AWS layanan acak, dan kemudian pergi ke AWS CDK paket untuk layanan itu (terletak di folder`packages`,, `aws-cdk-libaws-<servicename>`,`lib`). Untuk contoh ini mari kita pilih Amazon S3, tetapi ini berfungsi untuk layanan apa pun. Jika Anda melihat file index.ts utama untuk paket itu, Anda akan melihat baris yang berbunyi:

```
export * from './s3.generated';
```

Namun, Anda tidak akan melihat `s3.generated` file di mana pun di direktori yang sesuai. Ini karena konstruksi L1 dibuat secara otomatis dari spesifikasi CloudFormation sumber daya selama proses pembuatan. AWS CDK Jadi Anda akan melihat `s3.generated` dalam paket hanya setelah Anda menjalankan perintah AWS CDK build untuk paket tersebut.

Spesifikasi AWS CloudFormation sumber daya

Spesifikasi AWS CloudFormation sumber daya mendefinisikan infrastruktur sebagai kode (IAC) untuk AWS dan menentukan bagaimana kode dalam CloudFormation template diubah menjadi sumber daya dalam akun AWS . Spesifikasi ini mendefinisikan AWS sumber daya dalam format JSON pada tingkat per wilayah. Setiap sumber daya diberi nama jenis sumber daya unik yang mengikuti format`provider::service::resource`. Misalnya, nama tipe sumber daya untuk bucket Amazon S3 adalah`AWS::S3::Bucket`, dan nama tipe sumber daya untuk jalur akses Amazon S3 adalah. `AWS::S3::AccessPoint` Jenis sumber daya ini dapat dirender dalam CloudFormation template dengan menggunakan sintaks yang ditentukan dalam spesifikasi AWS CloudFormation sumber daya. Saat proses AWS CDK build berjalan, setiap jenis sumber daya juga menjadi konstruksi L1.

Akibatnya, setiap konstruksi L1 adalah bayangan cermin terprogram dari sumber daya yang sesuai. CloudFormation Setiap properti yang akan Anda terapkan dalam CloudFormation template tersedia saat Anda membuat instance konstruksi L1, dan setiap CloudFormation properti wajib juga diperlukan sebagai argumen saat Anda membuat instance konstruksi L1 yang sesuai. Tabel berikut membandingkan bucket S3 seperti yang direpresentasikan dalam CloudFormation template dengan bucket S3 yang sama seperti yang didefinisikan sebagai AWS CDK konstruksi L1.

CloudFormation Template

```
"amzns3demobucket": {
    "Type": "AWS::S3::Bucket",
    "Properties": {
      "BucketName": "amzn-s3-demo-
bucket",
      "BucketEncryption": {
        "ServerSideEncryptionConfig
uration": [
          {
```

Konstruksi L1

```
new CfnBucket(this, "amzns3de
mobucket", {
  bucketName: "amzn-s3-demo-bucket",
  bucketEncryption: {
    serverSideEncryptionConfigu
ration: [
      {
        serverSideEncryptionByDefau
lt: {
          sseAlgorithm: "AES256"
```

```
            "ServerSideEncrypt
ionByDefault": {
              "SSEAlgorithm": "AES256"
            }
          }
        ]
      },
      "MetricsConfigurations": [
        {
          "Id": "myConfig"
        }
      ],
      "OwnershipControls": {
        "Rules": [
          {
            "ObjectOwnership":
 "BucketOwnerPreferred"
          }
        ]
      },
      "PublicAccessBlockConfigura
tion": {
        "BlockPublicAcls": true,
        "BlockPublicPolicy": true,
        "IgnorePublicAcls": true,
        "RestrictPublicBuckets": true
      },
      "VersioningConfiguration": {
        "Status": "Enabled"
      }
    }
  }
```

```
        }
      }
    ]
  },
  metricsConfigurations: [
    {
      id: "myConfig"
    }
  ],
  ownershipControls: {
    rules: [
      {
        objectOwnership: "BucketOw
nerPreferred"
      }
    ]
  },
  publicAccessBlockConfiguration: {
    blockPublicAcls: true,
    blockPublicPolicy: true,
    ignorePublicAcls: true,
    restrictPublicBuckets: true
  },
  versioningConfiguration: {
    status: "Enabled"
  }
});
```

Seperti yang Anda lihat, konstruksi L1 adalah manifestasi yang tepat dalam kode sumber daya. CloudFormation Tidak ada jalan pintas atau penyederhanaan, sehingga jumlah teks boilerplate yang harus ditulis kira-kira sama. Namun, salah satu keuntungan besar menggunakan AWS CDK ini adalah membantu menghilangkan banyak sintaks boilerplate CloudFormation itu. Jadi bagaimana itu bisa terjadi? Di situlah konstruksi L2 masuk.

Konstruksi lapisan 2

Repositori AWS CDK open source ditulis terutama dengan menggunakan bahasa TypeScriptpemrograman dan terdiri dari banyak paket dan modul. Pustaka paket utama, disebut`aws-cdk-lib`, secara kasar dibagi menjadi satu paket per AWS layanan, meskipun hal ini tidak selalu terjadi. Seperti yang telah dibahas sebelumnya, konstruksi L1 dihasilkan secara otomatis selama proses pembuatan, jadi apa semua kode yang Anda lihat ketika Anda melihat ke dalam repositori? Itu adalah konstruksi L2, yang merupakan abstraksi dari konstruksi L1.

Paket juga berisi kumpulan TypeScript jenis, enum, dan antarmuka serta kelas pembantu yang menambahkan lebih banyak fungsionalitas, tetapi semua item tersebut melayani konstruksi L2. Semua konstruksi L2 memanggil konstruksi L1 yang sesuai dalam konstruktornya setelah instantiasi, dan konstruksi L1 yang dihasilkan yang dibuat dapat diakses dari lapisan 2 seperti ini:

```
const role = new Bucket(this, "amzn-s3-demo-bucket", {/*...BucketProps*/});
const cfnBucket = role.node.defaultChild;
```

Konstruksi L2 mengambil properti default, metode kenyamanan, dan gula sintaksis lainnya dan menerapkannya pada konstruksi L1. Ini menghilangkan banyak pengulangan dan verbositas yang diperlukan untuk menyediakan sumber daya secara langsung. CloudFormation

Semua konstruksi L2 membangun konstruksi L1 yang sesuai di bawah tenda. Namun, konstruksi L2 sebenarnya tidak memperluas konstruksi L1. Baik konstruksi L1 dan L2 mewarisi kelas khusus yang disebut Construct. Dalam versi 1 `Construct` kelas dibangun ke dalam kit pengembangan, tetapi dalam versi 2 itu adalah paket mandiri yang terpisah. AWS CDK Ini agar paket lain seperti Cloud Development Kit for Terraform (CDKTF) dapat memasukkannya sebagai dependensi. Setiap kelas yang mewarisi `Construct` kelas adalah L1, L2, atau konstruksi L3. Konstruksi L2 memperluas kelas ini secara langsung sedangkan konstruksi L1 memperluas kelas yang disebut`CfnResource`, seperti yang ditunjukkan pada tabel berikut.

Pohon warisan L1	Pohon warisan L2
Konstruksi L1	Konstruksi L2
→ kelas CfnResource	→ kelas Membangun
→ → kelas abstrak CfnRefElement	

→ → → kelas abstrak CfnElement

→ → → → Konstruksi kelas

Jika konstruksi L1 dan L2 mewarisi `Construct` kelas, mengapa konstruksi L2 tidak memperpanjang L1 saja? Nah, kelas antara `Construct` kelas dan layer 1 mengunci konstruksi L1 di tempat sebagai bayangan cermin dari sumber daya. CloudFormation Mereka berisi metode abstrak (metode yang harus disertakan kelas hilir) seperti`_toCloudFormation`, yang memaksa konstruksi untuk secara langsung menampilkan CloudFormation sintaks. Konstruksi L2 melewati kelas-kelas tersebut dan memperluas `Construct` kelas secara langsung. Ini memberi mereka fleksibilitas untuk mengabstraksi banyak kode yang diperlukan untuk konstruksi L1 dengan membangunnya secara terpisah di dalam konstruktor mereka.

Bagian sebelumnya menampilkan side-by-side perbandingan bucket S3 dari CloudFormation template dan bucket S3 yang sama yang dirender sebagai konstruksi L1. Perbandingan itu menunjukkan bahwa properti dan sintaks hampir identik, dan konstruksi L1 hanya menyimpan tiga atau empat baris dibandingkan dengan konstruksi. CloudFormation Sekarang mari kita bandingkan konstruksi L1 dengan konstruksi L2 untuk bucket S3 yang sama:

Konstruksi L1 untuk ember S3

```
new CfnBucket(this, "amzns3de
mobucket", {
  bucketName: "amzn-s3-demo-bucket",
  bucketEncryption: {
    serverSideEncryptionConfigu
ration: [
      {
        serverSideEncryptionByDefau
lt: {
          sseAlgorithm: "AES256"
        }
      }
    ]
  },
  metricsConfigurations: [
    {
      id: "myConfig"
    }
```

Konstruksi L2 untuk ember S3

```
new Bucket(this, "amzns3demobucket",
 {
  bucketName: "amzn-s3-demo-bucket",
  encryption: BucketEncryption.S
3_MANAGED,
  metrics: [
    {
      id: "myConfig"
    },
  ],
  objectOwnership: ObjectOwnership.BU
CKET_OWNER_PREFERRED,
  blockPublicAccess: BlockPubl
icAccess.BLOCK_ALL,
  versioned: true
});
```

```
  ],
  ownershipControls: {
    rules: [
      {
        objectOwnership: "BucketOw
nerPreferred"
      }
    ]
  },
  publicAccessBlockConfiguration: {
    blockPublicAcls: true,
    blockPublicPolicy: true,
    ignorePublicAcls: true,
    restrictPublicBuckets: true
  },
  versioningConfiguration: {
    status: "Enabled"
  }
});
```

Seperti yang Anda lihat, konstruksi L2 kurang dari setengah ukuran konstruksi L1. Konstruksi L2 menggunakan banyak teknik untuk mencapai konsolidasi ini. Beberapa teknik ini berlaku untuk konstruksi L2 tunggal, tetapi yang lain dapat digunakan kembali di beberapa konstruksi sehingga mereka dipisahkan ke dalam kelas mereka sendiri untuk digunakan kembali. Konstruksi L2 mengkonsolidasikan CloudFormation sintaks dalam beberapa cara, seperti yang dibahas di bagian berikut.

Properti default

Cara termudah untuk mengkonsolidasikan kode untuk penyediaan sumber daya adalah dengan mengubah pengaturan properti yang paling umum menjadi default. Mereka AWS CDK memiliki akses ke bahasa pemrograman yang kuat dan CloudFormation tidak, jadi default ini sering bersifat kondisional. Terkadang beberapa baris CloudFormation konfigurasi dapat dihilangkan dari AWS CDK kode karena pengaturan tersebut dapat disimpulkan dari nilai properti lain yang diteruskan ke konstruksi.

Struct, tipe, dan antarmuka

Meskipun AWS CDK tersedia dalam beberapa bahasa pemrograman, itu ditulis secara native di TypeScript, sehingga sistem tipe bahasa digunakan untuk menentukan jenis yang membentuk konstruksi L2. Menyelam jauh ke dalam sistem tipe itu berada di luar cakupan panduan ini; lihat TypeScriptdokumentasi untuk detailnya. Untuk meringkas, a TypeScript `type` menjelaskan jenis data apa yang dimiliki variabel tertentu. Ini bisa berupa data dasar seperti`string`, atau data yang lebih kompleks seperti file`object`. A TypeScript `interface` adalah cara lain untuk mengekspresikan tipe TypeScript objek, dan a `struct` adalah nama lain untuk antarmuka.

TypeScript tidak menggunakan istilah struct, tetapi jika Anda melihat di Referensi AWS CDK API, Anda akan melihat bahwa struct sebenarnya hanyalah TypeScript antarmuka lain dalam kode. Referensi API juga mengacu pada antarmuka tertentu sebagai antarmuka juga. Jika struct dan antarmuka adalah hal yang sama, mengapa AWS CDK dokumentasi membuat perbedaan di antara mereka?

Apa yang AWS CDK disebut sebagai struct adalah antarmuka yang mewakili objek apa pun yang digunakan oleh konstruksi L2. Ini termasuk tipe objek untuk argumen properti yang diteruskan ke konstruksi L2 selama instantiasi, seperti `BucketProps` untuk konstruksi S3 Bucket dan `TableProps` untuk konstruksi DynamoDB Table, serta antarmuka lain yang digunakan dalam. TypeScript AWS CDK Singkatnya, jika itu adalah TypeScript antarmuka di dalam AWS CDK dan namanya tidak diawali dengan huruf`I`, itu AWS CDK menyebutnya struct.

Sebaliknya, AWS CDK penggunaan istilah antarmuka untuk mewakili elemen dasar objek biasa perlu dianggap sebagai representasi yang tepat dari konstruksi atau kelas pembantu tertentu. Artinya, antarmuka menjelaskan apa yang harus menjadi properti publik konstruksi L2. Semua nama AWS CDK antarmuka adalah nama konstruksi yang ada atau kelas pembantu yang diawali dengan huruf. `I` Semua konstruksi L2 memperluas `Construct` kelas, tetapi mereka juga mengimplementasikan antarmuka yang sesuai. Jadi konstruksi L2 `Bucket` mengimplementasikan antarmuka. `IBucket`

Metode statis

Setiap instance konstruksi L2 juga merupakan instance dari antarmuka yang sesuai, tetapi kebalikannya tidak benar. Ini penting ketika melihat melalui struct untuk melihat tipe data mana yang diperlukan. Jika struct memiliki properti yang disebut`bucket`, yang memerlukan tipe data`IBucket`, Anda bisa melewatkan objek yang berisi properti yang terdaftar di `IBucket` antarmuka atau instance `Bucket` L2. Salah satu akan bekerja. Namun, jika `bucket` properti itu memanggil L2`Bucket`, Anda hanya bisa melewatkan `Bucket` instance di bidang itu.

Perbedaan ini menjadi sangat penting ketika Anda mengimpor sumber daya yang sudah ada sebelumnya ke tumpukan Anda. Anda dapat membuat konstruksi L2 untuk sumber daya apa pun yang asli dari tumpukan Anda, tetapi jika Anda perlu mereferensikan sumber daya yang dibuat di luar tumpukan, Anda harus menggunakan antarmuka konstruksi L2 itu. Itu karena membuat konstruksi L2 menciptakan sumber daya baru jika belum ada di dalam tumpukan itu. Referensi ke sumber daya yang ada harus berupa objek biasa yang sesuai dengan antarmuka konstruksi L2 itu.

Untuk membuatnya lebih mudah dalam praktiknya, sebagian besar konstruksi L2 memiliki seperangkat metode statis yang terkait dengannya yang mengembalikan antarmuka konstruksi L2 itu. Metode statis ini biasanya dimulai dengan kata`from`. Dua argumen pertama yang diteruskan ke metode ini adalah sama `scope` dan `id` argumen yang diperlukan untuk konstruksi L2 standar. Namun, argumen ketiga tidak `props` hanya sebagian kecil dari properti (atau kadang-kadang hanya satu properti) yang mendefinisikan antarmuka. Untuk alasan ini, ketika Anda melewati konstruksi L2, dalam banyak kasus hanya elemen antarmuka yang diperlukan. Ini agar Anda dapat menggunakan sumber daya yang diimpor juga, jika memungkinkan.

```
// Example of referencing an external S3 bucket
const preExistingBucket = Bucket.fromBucketName(this, "external-bucket", "name-of-
bucket-that-already-exists");
```

Namun, Anda tidak harus terlalu bergantung pada antarmuka. Anda harus mengimpor sumber daya dan menggunakan antarmuka secara langsung hanya jika benar-benar diperlukan, karena antarmuka tidak menyediakan banyak properti — seperti metode pembantu — yang membuat konstruksi L2 begitu kuat.

Metode pembantu

Konstruksi L2 adalah kelas terprogram daripada objek sederhana, sehingga dapat mengekspos metode kelas yang memungkinkan Anda untuk memanipulasi konfigurasi sumber daya Anda setelah instantiasi telah terjadi. Contoh yang baik dari ini adalah konstruksi Peran L2 AWS Identity and Access Management (IAM). Cuplikan berikut menunjukkan dua cara untuk membuat peran IAM yang sama dengan menggunakan konstruksi L2. `Role`

Tanpa metode pembantu:

```
const role = new Role(this, "my-iam-role", {
   assumedBy: new FederatedPrincipal('my-identity-provider.com'),
   managedPolicies: [
```

```
      ManagedPolicy.fromAwsManagedPolicyName("ReadOnlyAccess")
   ],
   inlinePolicies: {
      lambdaPolicy: new PolicyDocument({
         statements: [
            new PolicyStatement({
               effect: Effect.ALLOW,
               actions: [ 'lambda:UpdateFunctionCode' ],
               resources: [ 'arn:aws:lambda:us-east-1:123456789012:function:my-
function' ]
            })
         ]
      })
   }
});
```

Dengan metode pembantu:

```
const role = new Role(this, "my-iam-role", {
   assumedBy: new FederatedPrincipal('my-identity-provider.com')
});

role.addManagedPolicy(ManagedPolicy.fromAwsManagedPolicyName("ReadOnlyAccess"));
role.attachInlinePolicy(new Policy(this, "lambda-policy", {
   policyName: "lambdaPolicy",
   statements: [
      new PolicyStatement({
         effect: Effect.ALLOW,
         actions: [ 'lambda:UpdateFunctionCode' ],
         resources: [ 'arn:aws:lambda:us-east-1:123456789012:function:my-function' ]
      })
   ]
 }));
```

Kemampuan untuk menggunakan metode instance untuk memanipulasi konfigurasi sumber daya setelah instantiation memberikan konstruksi L2 banyak fleksibilitas tambahan atas lapisan sebelumnya. Konstruksi L1 juga mewarisi beberapa metode sumber daya (sepertiaddPropertyOverride), tetapi tidak sampai lapisan dua Anda mendapatkan metode yang dirancang khusus untuk sumber daya itu dan propertinya.

Enum

CloudFormation sintaks sering mengharuskan Anda untuk menentukan banyak detail untuk menyediakan sumber daya dengan benar. Namun, sebagian besar kasus penggunaan sering kali hanya tercakup oleh segelintir konfigurasi. Mewakili konfigurasi tersebut dengan menggunakan serangkaian nilai yang disebutkan dapat sangat mengurangi jumlah kode yang dibutuhkan.

Misalnya, dalam contoh kode L2 bucket S3 dari sebelumnya di bagian ini, Anda harus menggunakan `bucketEncryption` properti CloudFormation template untuk memberikan semua detail, termasuk nama algoritma enkripsi yang akan digunakan. Sebagai gantinya, AWS CDK menyediakan `BucketEncryption` enum, yang mengambil lima bentuk enkripsi bucket yang paling umum dan memungkinkan Anda mengekspresikan masing-masing dengan menggunakan nama variabel tunggal.

Bagaimana dengan kasus tepi yang tidak tercakup oleh enum? Salah satu tujuan dari konstruksi L2 adalah untuk menyederhanakan tugas penyediaan sumber daya lapisan 1, sehingga kasus tepi tertentu yang kurang umum digunakan mungkin tidak didukung di lapisan 2. Untuk mendukung kasus tepi ini, AWS CDK memungkinkan Anda memanipulasi properti CloudFormation sumber daya yang mendasarinya secara langsung dengan menggunakan addPropertyOverridemetode ini. Untuk informasi selengkapnya tentang penggantian properti, lihat bagian Praktik terbaik dari panduan ini dan bagian Abstraksi dan lubang keluar dalam dokumentasi. AWS CDK

Kelas pembantu

Terkadang enum tidak dapat menyelesaikan logika terprogram yang diperlukan untuk mengonfigurasi sumber daya untuk kasus penggunaan tertentu. Dalam situasi ini, AWS CDK sering menawarkan kelas pembantu sebagai gantinya. Enum adalah objek sederhana yang menawarkan serangkaian pasangan kunci-nilai, sedangkan kelas pembantu menawarkan kemampuan penuh kelas. TypeScript Kelas pembantu masih dapat bertindak seperti enum dengan mengekspos properti statis, tetapi properti tersebut kemudian dapat menetapkan nilainya secara internal dengan logika bersyarat di konstruktor kelas pembantu atau dalam metode pembantu.

Jadi meskipun `BucketEncryption` enum dapat mengurangi jumlah kode yang diperlukan untuk mengatur algoritme enkripsi pada bucket S3, strategi yang sama tidak akan berfungsi untuk mengatur durasi waktu karena ada terlalu banyak nilai yang mungkin untuk dipilih. Membuat enum untuk setiap nilai akan jauh lebih merepotkan daripada nilainya. Untuk alasan ini, kelas helper digunakan untuk pengaturan konfigurasi Kunci Objek S3 bucket default S3, seperti yang diwakili oleh kelas. ObjectLockRetention `ObjectLockRetention`berisi dua metode statis: satu untuk retensi kepatuhan

dan yang lainnya untuk retensi tata kelola. Kedua metode mengambil contoh dari kelas pembantu Durasi sebagai argumen untuk menyatakan jumlah waktu yang harus dikonfigurasi kunci.

Contoh lain adalah Runtime kelas AWS Lambda pembantu. Pada pandangan pertama, mungkin tampak bahwa properti statis yang terkait dengan kelas ini dapat ditangani oleh enum. Namun, di bawah tenda, setiap nilai properti mewakili instance dari `Runtime` kelas itu sendiri, sehingga logika yang dilakukan dalam konstruktor kelas tidak dapat dicapai dalam enum.

Konstruksi lapisan 3

Jika konstruksi L1 melakukan terjemahan literal CloudFormation sumber daya ke dalam kode terprogram, dan konstruksi L2 menggantikan banyak CloudFormation sintaks verbose dengan metode pembantu dan logika khusus, apa yang dilakukan konstruksi L3? Jawabannya hanya dibatasi oleh imajinasi Anda. Anda dapat membuat layer 3 agar sesuai dengan kasus penggunaan tertentu. Jika proyek Anda membutuhkan sumber daya yang memiliki subset properti tertentu, Anda dapat membuat konstruksi L3 yang dapat digunakan kembali untuk memenuhi kebutuhan tersebut.

Konstruksi L3 disebut pola di dalam. AWS CDK Pola adalah objek apa pun yang memperluas `Construct` kelas di AWS CDK (atau memperluas kelas yang memperluas `Construct` kelas) untuk melakukan logika abstrak apa pun di luar lapisan 2. Ketika Anda menggunakan AWS CDK CLI untuk menjalankan cdk init untuk memulai AWS CDK proyek baru, Anda harus memilih dari tiga jenis AWS CDK aplikasi:`app`,, `lib` dan. `sample-app`

```
Available templates:
* app: Template for a CDK Application
   └ cdk init app --language=[csharp|fsharp|go|java|javascript|python|typescript]
* lib: Template for a CDK Construct Library
   └ cdk init lib --language=typescript
* sample-app: Example CDK Application with some constructs
   └ cdk init sample-app --language=[csharp|fsharp|go|java|javascript|python|typescript]
```

`app`dan `sample-app` keduanya mewakili AWS CDK aplikasi klasik tempat Anda membangun dan menerapkan CloudFormation tumpukan ke lingkungan AWS. Ketika Anda memilih`lib`, Anda memilih untuk membangun konstruksi L3 baru. `app`dan `sample-app` memungkinkan Anda untuk memilih bahasa apa pun yang AWS CDK mendukung, tetapi Anda hanya dapat memilih TypeScript dengan`lib`. Ini karena ditulis AWS CDK secara native TypeScript dan menggunakan sistem open source yang dipanggil JSiiuntuk menerjemahkan kode asli ke bahasa lain yang didukung. Ketika Anda memilih `lib` untuk memulai proyek Anda, Anda memilih untuk membangun ekstensi ke. AWS CDK

Setiap kelas yang memperluas `Construct` kelas dapat berupa konstruksi L3, tetapi kasus penggunaan yang paling umum untuk lapisan 3 adalah interaksi sumber daya, ekstensi sumber daya, dan sumber daya khusus. Sebagian besar konstruksi L3 menggunakan satu atau lebih dari tiga kasus ini untuk memperluas AWS CDK fungsionalitas.

Interaksi sumber daya

Solusi biasanya menggunakan beberapa layanan AWS yang bekerja sama. Misalnya, CloudFront distribusi Amazon sering menggunakan bucket S3 sebagai asalnya dan AWS WAF untuk perlindungan terhadap eksploitasi umum. AWS AppSync dan Amazon API Gateway sering menggunakan tabel Amazon DynamoDB sebagai sumber data untuk tabel tersebut. APIs Pipeline AWS CodePipeline sering menggunakan Amazon S3 sebagai sumbernya dan AWS CodeBuild untuk tahap pembuatannya. Dalam kasus ini, seringkali berguna untuk membuat konstruksi L3 tunggal yang menangani penyediaan dua atau lebih konstruksi L2 yang saling berhubungan.

Berikut adalah contoh konstruksi L3 yang menyediakan CloudFront distribusi bersama dengan asal S3-nya, AWS WAF untuk diletakkan di depannya, catatan Amazon Route 53, dan sertifikat AWS Certificate Manager (ACM) untuk menambahkan titik akhir khusus dengan enkripsi dalam transit—semuanya dalam satu konstruksi yang dapat digunakan kembali:

```
// Define the properties passed to the L3 construct
export interface CloudFrontWebsiteProps {
     distributionProps: DistributionProps
     bucketProps: BucketProps
     wafProps: CfnWebAclProps
     zone: IHostedZone
}

// Define the L3 construct
export class CloudFrontWebsite extends Construct {
     public distribution: Distribution

     constructor(
        scope: Construct,
        id: string,
        props: CloudFrontWebsiteProps
     ) {
         super(scope, id);

         const certificate = new Certificate(this, "Certificate", {
            domainName: props.zone.zoneName,
            validation: CertificateValidation.fromDns(props.zone)
         });
         const defaultBehavior = {
            origin: new S3Origin(new Bucket(this, "bucket", props.bucketProps))
         }
```

```
        const waf = new CfnWebACL(this, "waf", props.wafProps);
        this.distribution = new Distribution(this, id, {
           ...props.distributionProps,
           defaultBehavior,
           certificate,
           domainNames: [this.domainName],
           webAclId: waf.attrArn,
         });
    }
}
```

Perhatikan bahwa CloudFront, Amazon S3, Route 53, dan ACM semuanya menggunakan konstruksi L2, tetapi ACL web (yang mendefinisikan aturan untuk menangani permintaan web) menggunakan konstruksi L1. Ini karena ini AWS CDK adalah paket open source yang berkembang yang belum sepenuhnya lengkap, dan belum ada konstruksi L2. `WebAcl` Namun, siapa pun dapat berkontribusi AWS CDK dengan membuat konstruksi L2 baru. Jadi sampai AWS CDK menawarkan konstruksi L2 untuk`WebAcl`, Anda harus menggunakan konstruksi L1. Untuk membuat situs web baru dengan menggunakan konstruksi L3`CloudFrontWebsite`, Anda menggunakan kode berikut:

```
const siteADotCom = new CloudFrontWebsite(stack, "siteA", siteAProps);
const siteBDotCom = new CloudFrontWebsite(stack, "siteB", siteBProps);
const siteCDotCom = new CloudFrontWebsite(stack, "siteC", siteCProps);
```

Dalam contoh ini, konstruksi CloudFront `Distribution` L2 diekspos sebagai milik umum dari konstruksi L3. Masih akan ada kasus di mana Anda perlu mengekspos properti L3 seperti ini, seperlunya. Bahkan kita akan melihat `Distribution` lagi nanti, di bagian Custom resources.

AWS CDK Termasuk beberapa contoh pola interaksi sumber daya seperti ini. Selain `aws-ecs` paket yang berisi konstruksi L2 untuk Amazon Elastic Container Service (Amazon ECS), AWS CDK paket tersebut memiliki paket yang disebut. aws-ecs-patterns Paket ini berisi beberapa konstruksi L3 yang menggabungkan Amazon ECS dengan Application Load Balancers, Network Load Balancers, dan grup target sambil menawarkan versi berbeda yang telah ditetapkan untuk Amazon Elastic Compute Cloud (Amazon) dan. EC2 AWS Fargate Karena banyak aplikasi tanpa server menggunakan Amazon ECS hanya dengan Fargate, konstruksi L3 ini memberikan kenyamanan yang dapat menghemat waktu pengembang dan uang pelanggan.

Ekstensi sumber daya

Beberapa kasus penggunaan memerlukan sumber daya untuk memiliki pengaturan default tertentu yang bukan asli dari konstruksi L2. Pada tingkat tumpukan, ini dapat ditangani dengan menggunakan aspek, tetapi cara lain yang mudah untuk memberikan konstruksi L2 default baru adalah dengan memperluas lapisan 2. Karena konstruksi adalah kelas apa pun yang mewarisi `Construct` kelas, dan konstruksi L2 memperluas kelas itu, Anda juga dapat membuat konstruksi L3 dengan langsung memperluas konstruksi L2.

Hal ini dapat sangat berguna untuk logika bisnis kustom yang mendukung kebutuhan tunggal pelanggan. Misalkan sebuah perusahaan memiliki repositori yang menyimpan semua kode AWS Lambda fungsinya dalam satu direktori yang dipanggil `src/lambda` dan sebagian besar fungsi Lambda menggunakan kembali runtime dan nama handler yang sama setiap kali. Alih-alih mengonfigurasi jalur kode setiap kali Anda mengonfigurasi fungsi Lambda baru, Anda dapat membuat konstruksi L3 baru:

```
export class MyCompanyLambdaFunction extends Function {
      constructor(
        scope: Construct,
        id: string,
        props: Partial<FunctionProps> = {}
     ) {
         super(scope, id, {
            handler: 'index.handler',
            runtime: Runtime.NODEJS_LATEST,
            code: Code.fromAsset(`src/lambda/${props.functionName || id}`),
            ...props
     });
}
```

Anda kemudian dapat mengganti `Function` konstruksi L2 di mana-mana di repositori sebagai berikut:

```
new MyCompanyLambdaFunction(this, "MyFunction");
new MyCompanyLambdaFunction(this, "MyOtherFunction");
new MyCompanyLambdaFunction(this, "MyThirdFunction", {
   runtime: Runtime.PYTHON_3_11
});
```

Defaultnya memungkinkan Anda membuat fungsi Lambda baru pada satu baris, dan konstruksi L3 diatur sehingga Anda masih dapat mengganti properti default jika diperlukan.

Memperluas konstruksi L2 secara langsung berfungsi paling baik ketika Anda hanya ingin menambahkan default baru ke konstruksi L2 yang ada. Jika Anda membutuhkan logika khusus lainnya juga, lebih baik untuk memperluas `Construct` kelas. Alasan untuk ini berasal dari `super` metode, yang disebut dalam konstruktor. Dalam kelas yang memperluas kelas lain, `super` metode ini digunakan untuk memanggil konstruktor kelas induk, dan ini harus menjadi hal pertama yang terjadi dalam konstruktor Anda. Ini berarti bahwa setiap manipulasi argumen yang diteruskan atau logika khusus lainnya dapat terjadi hanya setelah konstruksi L2 asli dibuat. Jika Anda perlu melakukan logika kustom ini sebelum membuat instance konstruksi L2 Anda, lebih baik mengikuti pola yang diuraikan sebelumnya di bagian Interaksi sumber daya.

Sumber daya khusus

Sumber daya khusus adalah fitur canggih CloudFormation yang memungkinkan Anda menjalankan logika khusus dari fungsi Lambda yang diaktifkan selama penerapan tumpukan. Kapan pun Anda memerlukan proses apa pun selama penerapan yang tidak didukung secara langsung CloudFormation, Anda dapat menggunakan sumber daya khusus untuk mewujudkannya. AWS CDK Menawarkan kelas yang memungkinkan Anda membuat sumber daya khusus secara terprogram juga. Dengan menggunakan sumber daya khusus dalam konstruktor L3, Anda dapat membuat konstruksi dari hampir semua hal.

Salah satu keuntungan menggunakan Amazon CloudFront adalah kemampuan caching globalnya yang kuat. Jika Anda ingin mengatur ulang cache secara manual sehingga situs web Anda segera mencerminkan perubahan baru yang dibuat pada asal Anda, Anda dapat menggunakan CloudFront pembatalan. Namun, pembatalan adalah proses yang berjalan pada CloudFrontdistribusi alih-alih menjadi properti distribusi. CloudFront Mereka dapat dibuat dan diterapkan ke distribusi yang ada kapan saja, sehingga mereka bukan bagian asli dari proses penyediaan dan penerapan.

Dalam skenario ini, Anda mungkin ingin membuat dan menjalankan pembatalan setelah setiap pembaruan ke asal distribusi. Karena sumber daya khusus, Anda dapat membuat konstruksi L3 yang terlihat seperti ini:

```
export interface CloudFrontInvalidationProps {
     distribution: Distribution
     region?: string
     paths?: string[]
}
```

```
export class CloudFrontInvalidation extends Construct {
      constructor(
        scope: Construct,
        id: string,
        props: CloudFrontInvalidationProps
     ) {
          super(scope, id);
          const policy = AwsCustomResourcePolicy.fromSdkCalls({
               resources:AwsCustomResourcePolicy.ANY_RESOURCE
          });
          new AwsCustomResource(scope, `${id}Invalidation`, {
             policy,
             onUpdate: {
                  service: 'CloudFront',
                  action: 'createInvalidation',
                  region: props.region || 'us-east-1',
                  physicalResourceId:
 PhysicalResourceId.fromResponse('Invalidation.Id'),
                  parameters: {
                     DistributionId: props.distribution.distributionId,
                     InvalidationBatch: {
                         Paths: {
                            Quantity: props.paths?.length || 1,
                            Items: props.paths || ['/*']
                         },
                         CallerReference: crypto.randomBytes(5).toString('hex')
                     }
                  }
             }
          }
     }
}
```

Menggunakan distribusi yang kami buat sebelumnya di konstruksi `CloudFrontWebsite` L3, Anda dapat melakukan ini dengan sangat mudah:

```
new CloudFrontInvalidation(this, 'MyInvalidation', {
       distribution: siteADotCom.distribution
});
```

Konstruksi L3 ini menggunakan konstruksi AWS CDK L3 yang dipanggil AwsCustomResourceuntuk membuat sumber daya khusus yang melakukan logika kustom. `AwsCustomResource`sangat

nyaman ketika Anda perlu membuat tepat satu panggilan AWS SDK, karena memungkinkan Anda melakukannya tanpa harus menulis kode Lambda apa pun. Jika Anda memiliki persyaratan yang lebih kompleks dan ingin menerapkan logika Anda sendiri, Anda dapat menggunakan CustomResourcekelas dasar secara langsung.

Contoh bagus lainnya dari AWS CDK penggunaan konstruksi L3 sumber daya khusus adalah penerapan bucket S3. Fungsi Lambda yang dibuat oleh sumber daya khusus dalam konstruktor konstruksi L3 ini menambahkan fungsionalitas yang CloudFormation tidak dapat ditangani sebaliknya: ia menambahkan dan memperbarui objek dalam ember S3. Tanpa penerapan bucket S3, Anda tidak akan dapat memasukkan konten ke dalam bucket S3 yang baru saja Anda buat sebagai bagian dari tumpukan Anda, yang akan sangat merepotkan.

Contoh terbaik dari AWS CDK menghilangkan kebutuhan untuk menulis rim CloudFormation sintaks adalah dasar ini: `S3BucketDeployment`

```
new BucketDeployment(this, 'BucketObjects', {
  sources: [Source.asset('./path/to/amzn-s3-demo-bucket')],
  destinationBucket: amzn-s3-demo-bucket
});
```

Bandingkan dengan CloudFormation kode yang harus Anda tulis untuk mencapai hal yang sama:

```
"lambdapolicyA5E98E09": {
   "Type": "AWS::IAM::Policy",
   "Properties": {
    "PolicyDocument": {
     "Statement": [
      {
       "Action": "lambda:UpdateFunctionCode",
       "Effect": "Allow",
       "Resource": "arn:aws:lambda:us-east-1:123456789012:function:my-function"
      }
     ],
     "Version": "2012-10-17"
    },
    "PolicyName": "lambdaPolicy",
    "Roles": [
     {
      "Ref": "myiamroleF09C7974"
     }
    ]
   },
```

```
   "Metadata": {
    "aws:cdk:path": "CdkScratchStack/lambda-policy/Resource"
   }
  },
  "BucketObjectsAwsCliLayer8C081206": {
   "Type": "AWS::Lambda::LayerVersion",
   "Properties": {
    "Content": {
     "S3Bucket": {
      "Fn::Sub": "cdk-hnb659fds-assets-${AWS::AccountId}-${AWS::Region}"
     },
     "S3Key": "e2277687077a2abf9ae1af1cc9565e6715e2ebb62f79ec53aa75a1af9298f642.zip"
    },
    "Description": "/opt/awscli/aws"
   },
   "Metadata": {
    "aws:cdk:path": "CdkScratchStack/BucketObjects/AwsCliLayer/Resource",
    "aws:asset:path":
"asset.e2277687077a2abf9ae1af1cc9565e6715e2ebb62f79ec53aa75a1af9298f642.zip",
    "aws:asset:is-bundled": false,
    "aws:asset:property": "Content"
   }
  },
  "BucketObjectsCustomResourceB12E6837": {
   "Type": "Custom::CDKBucketDeployment",
   "Properties": {
    "ServiceToken": {
     "Fn::GetAtt": [
      "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756C81C01536",
      "Arn"
     ]
    },
    "SourceBucketNames": [
     {
      "Fn::Sub": "cdk-hnb659fds-assets-${AWS::AccountId}-${AWS::Region}"
     }
    ],
    "SourceObjectKeys": [
     "f888a9d977f0b5bdbc04a1f8f07520ede6e00d4051b9a6a250860a1700924f26.zip"
    ],
    "DestinationBucketName": {
     "Ref": "amzn-s3-demo-bucket77F80CC0"
    },
    "Prune": true
```

```
    },
    "UpdateReplacePolicy": "Delete",
    "DeletionPolicy": "Delete",
    "Metadata": {
     "aws:cdk:path": "CdkScratchStack/BucketObjects/CustomResource/Default"
    }
   },
   "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRole89A01265": {
    "Type": "AWS::IAM::Role",
    "Properties": {
     "AssumeRolePolicyDocument": {
      "Statement": [
       {
        "Action": "sts:AssumeRole",
        "Effect": "Allow",
        "Principal": {
         "Service": "lambda.amazonaws.com"
        }
       }
      ],
      "Version": "2012-10-17"
     },
     "ManagedPolicyArns": [
      {
       "Fn::Join": [
        "",
        [
         "arn:",
         {
          "Ref": "AWS::Partition"
         },
         ":iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
        ]
       ]
      }
     ]
    },
    "Metadata": {
     "aws:cdk:path": "CdkScratchStack/
Custom::CDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756C/ServiceRole/Resource"
    }
   },
```

```
"CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRoleDefaultPolicy88902FDF":
{
  "Type": "AWS::IAM::Policy",
  "Properties": {
   "PolicyDocument": {
    "Statement": [
     {
      "Action": [
       "s3:GetBucket*",
       "s3:GetObject*",
       "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": [
       {
        "Fn::Join": [
         "",
         [
          "arn:",
          {
           "Ref": "AWS::Partition"
          },
          ":s3:::",
          {
           "Fn::Sub": "cdk-hnb659fds-assets-${AWS::AccountId}-${AWS::Region}"
          },
          "/*"
         ]
        ]
       },
       {
        "Fn::Join": [
         "",
         [
          "arn:",
          {
           "Ref": "AWS::Partition"
          },
          ":s3:::",
          {
           "Fn::Sub": "cdk-hnb659fds-assets-${AWS::AccountId}-${AWS::Region}"
          }
         ]
```

```
        ]
       }
      ]
     },
     {
      "Action": [
       "s3:Abort*",
       "s3:DeleteObject*",
       "s3:GetBucket*",
       "s3:GetObject*",
       "s3:List*",
       "s3:PutObject",
       "s3:PutObjectLegalHold",
       "s3:PutObjectRetention",
       "s3:PutObjectTagging",
       "s3:PutObjectVersionTagging"
      ],
      "Effect": "Allow",
      "Resource": [
       {
        "Fn::GetAtt": [
         "amzns3demobucket77F80CC0",
         "Arn"
        ]
       },
       {
        "Fn::Join": [
         "",
         [
          {
           "Fn::GetAtt": [
            "amzns3demobucket77F80CC0",
            "Arn"
           ]
          },
          "/*"
         ]
        ]
       }
      ]
     }
    ],
    "Version": "2012-10-17"
   },
```

```
    "PolicyName":
 "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRoleDefaultPolicy88902FDF",
    "Roles": [
     {
      "Ref":
 "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRole89A01265"
     }
    ]
   },
   "Metadata": {
    "aws:cdk:path": "CdkScratchStack/
Custom::CDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756C/ServiceRole/DefaultPolicy/
Resource"
   }
  },
  "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756C81C01536": {
   "Type": "AWS::Lambda::Function",
   "Properties": {
    "Code": {
     "S3Bucket": {
      "Fn::Sub": "cdk-hnb659fds-assets-${AWS::AccountId}-${AWS::Region}"
     },
     "S3Key": "9eb41a5505d37607ac419321497a4f8c21cf0ee1f9b4a6b29aa04301aea5c7fd.zip"
    },
    "Role": {
     "Fn::GetAtt": [
      "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRole89A01265",
      "Arn"
     ]
    },
    "Environment": {
     "Variables": {
      "AWS_CA_BUNDLE": "/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem"
     }
    },
    "Handler": "index.handler",
    "Layers": [
     {
      "Ref": "BucketObjectsAwsCliLayer8C081206"
     }
    ],
    "Runtime": "python3.9",
    "Timeout": 900
   },
```

```
    "DependsOn": [

  "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRoleDefaultPolicy88902FDF",
     "CustomCDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756CServiceRole89A01265"
    ],
    "Metadata": {
     "aws:cdk:path": "CdkScratchStack/
Custom::CDKBucketDeployment8693BB64968944B69AAFB0CC9EB8756C/Resource",
     "aws:asset:path":
  "asset.9eb41a5505d37607ac419321497a4f8c21cf0ee1f9b4a6b29aa04301aea5c7fd",
     "aws:asset:is-bundled": false,
     "aws:asset:property": "Code"
    }
   }
```

4 baris versus 241 baris adalah perbedaan besar! Dan ini hanyalah salah satu contoh dari apa yang mungkin ketika Anda memanfaatkan lapisan 3 untuk menyesuaikan tumpukan Anda.

Praktik terbaik

Konstruksi L1

- Anda tidak selalu dapat menghindari penggunaan konstruksi L1 secara langsung, tetapi Anda harus menghindarinya bila memungkinkan. Jika konstruksi L2 tertentu tidak mendukung kasus tepi Anda, Anda dapat menjelajahi dua opsi ini alih-alih menggunakan konstruksi L1 secara langsung:
  - Aksesdefaultchild: Jika CloudFormation properti yang Anda butuhkan tidak tersedia dalam konstruksi L2, Anda dapat mengakses konstruksi L1 yang mendasarinya dengan menggunakan. `L2Construct.node.defaultChild` Anda dapat memperbarui properti publik apa pun dari konstruksi L1 dengan mengaksesnya melalui properti ini alih-alih mengalami kesulitan membuat konstruksi L1 sendiri.
  - Gunakan penggantian properti: Bagaimana jika properti yang ingin Anda perbarui tidak bersifat publik? Palka escape utama yang memungkinkan AWS CDK untuk melakukan apa pun yang dapat dilakukan CloudFormation template adalah dengan menggunakan metode yang tersedia di setiap konstruksi L1:. addPropertyOverride Anda dapat memanipulasi tumpukan Anda di tingkat CloudFormation template dengan meneruskan nama CloudFormation properti dan nilai langsung ke metode ini.

Konstruksi L2

- Ingatlah untuk memanfaatkan metode pembantu yang sering ditawarkan oleh konstruksi L2. Dengan layer 2, Anda tidak harus melewatkan setiap properti saat instantiation. Metode pembantu L2 dapat membuat penyediaan sumber daya secara eksponensial lebih nyaman, terutama ketika logika bersyarat diperlukan. Salah satu metode pembantu yang paling nyaman berasal dari kelas Grant. Kelas ini tidak digunakan secara langsung, tetapi banyak konstruksi L2 menggunakannya untuk menyediakan metode pembantu yang membuat izin lebih mudah diimplementasikan. Misalnya, jika Anda ingin memberikan izin ke fungsi L2 Lambda untuk mengakses bucket L2 S3, Anda dapat `s3Bucket.grantReadWrite(lambdaFunction)` memanggil alih-alih membuat peran dan kebijakan baru.

Konstruksi L3

- Meskipun konstruksi L3 bisa sangat nyaman ketika Anda ingin membuat tumpukan Anda lebih dapat digunakan kembali dan dapat disesuaikan, kami sarankan Anda menggunakannya dengan

hati-hati. Pertimbangkan jenis konstruksi L3 yang Anda butuhkan atau apakah Anda memerlukan konstruksi L3 sama sekali:

- Jika Anda tidak berinteraksi dengan AWS sumber daya secara langsung, seringkali lebih tepat untuk membuat kelas pembantu daripada memperluas kelas. `Construct` Ini karena `Construct` kelas melakukan banyak tindakan secara default yang diperlukan hanya jika Anda berinteraksi langsung dengan AWS sumber daya. Jadi, jika Anda tidak membutuhkan tindakan itu dilakukan, lebih efisien untuk menghindarinya.
- Jika Anda menentukan bahwa membuat konstruksi L3 baru sesuai, dalam banyak kasus Anda ingin memperluas `Construct` kelas secara langsung. Perluas konstruksi L2 lainnya hanya ketika Anda ingin memperbarui properti default konstruksi. Jika konstruksi L2 atau logika kustom lainnya terlibat, perluas `Construct` secara langsung dan buat instance semua sumber daya dalam konstruktor.

Pertanyaan yang Sering Diajukan

Tidak bisakah saya menggunakan lapisan AWS CDK tanpa pemahaman?

Anda benar-benar bisa. Tetapi seperti halnya alat yang paling kuat, AWS CDK semakin kuat semakin Anda mengetahuinya. Mempelajari bagaimana lapisan berinteraksi membuka tingkat pemahaman baru yang membantu menyederhanakan penerapan tumpukan Anda jauh melampaui apa yang dapat Anda lakukan AWS CDK hanya dengan pengetahuan dasar. AWS CDK

Bisakah saya membuat konstruksi L2 dari L1 dengan cara yang sama seperti saya membuat konstruksi L3 dari L2?

Jika sumber daya sudah memiliki konstruksi L2, kami sarankan Anda menggunakan konstruksi itu dan membuat penyesuaian Anda di lapisan 3. Ini karena banyak penelitian telah dilakukan untuk mencari tahu cara terbaik untuk mengkonfigurasi konstruksi L2 yang ada untuk sumber daya tertentu. Namun, ada beberapa konstruksi L1 yang konstruksi L2-nya belum ada. Dalam kasus tersebut, kami mendorong Anda untuk membuat konstruksi L2 Anda sendiri dan membagikannya dengan orang lain dengan menjadi kontributor perpustakaan AWS CDK open source. Anda dapat menemukan semua yang Anda butuhkan untuk memulai dalam pedoman kontribusi untuk AWS CDK.

AWS Sumber daya mana yang belum memiliki konstruksi L2 resmi?

Jumlah AWS sumber daya yang tidak memiliki konstruksi L2 semakin rendah dari hari ke hari, tetapi jika Anda tertarik untuk membantu membuat konstruksi L2 untuk salah satu sumber daya ini, kunjungi Referensi API.AWS CDK Lihatlah daftar sumber daya di panel kiri. Sumber daya yang memiliki superskrip 1 di sebelah namanya tidak memiliki konstruksi L2 resmi.

Bisakah saya membuat konstruksi L2 atau L3 dalam bahasa apa pun yang didukung? AWS CDK

Ini AWS CDK mendukung beberapa bahasa pemrograman, termasuk TypeScript, JavaScript, Python, Java, C #, dan Go. Anda dapat membuat konstruksi L3 pribadi Anda dengan menggunakan AWS

CDK kode yang dikompilasi ke dalam bahasa yang relevan. Namun, jika Anda ingin berkontribusi pada AWS CDK atau membuat AWS CDK konstruksi asli, Anda harus menggunakannya TypeScript. Ini karena TypeScript merupakan satu-satunya bahasa yang asli dari AWS CDK. AWS CDK Versi untuk bahasa lain dibangun dari TypeScript kode asli dengan menggunakan AWS pustaka yang disebut JSii.

Di mana saya dapat menemukan konstruksi L3 yang ada di luar? AWS CDK

Ada terlalu banyak lokasi untuk dibagikan di sini, tetapi Anda dapat menemukan banyak konstruksi paling populer di situs web Konstruksi AWS Solusi dan di AWS CDK bagian Construct Hub.

Sumber daya

- AWS CDK Referensi API
- AWS CloudFormation spesifikasi sumber daya
- AWS CDK membangun dokumentasi
- AWS CDK abstraksi dan lubang pelarian
- Manfaatkan konstruksi L2 untuk mengurangi kompleksitas AWS CDK aplikasi Anda (AWS posting blog)
- AWS CloudFormation sumber daya khusus
- AWS Solusi Konstruksi
- Membangun Hub
- AWS CDK Contoh (GitHub repositori)

Riwayat dokumen

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

Perubahan	Deskripsi	Tanggal
Publikasi awal	—	Desember 4, 2023

AWS Glosarium Panduan Preskriptif

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

Nomor

7 Rs

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

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

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

A

ABAC

Lihat kontrol akses berbasis atribut.

layanan abstrak

Lihat layanan terkelola.

ASAM

Lihat atomisitas, konsistensi, isolasi, daya tahan.

migrasi aktif-aktif

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

migrasi aktif-pasif

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

fungsi agregat

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

AI

Lihat kecerdasan buatan.

AIOps

Lihat operasi kecerdasan buatan.

anonimisasi

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

anti-pola

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

kontrol aplikasi

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

portofolio aplikasi

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

kecerdasan buatan (AI)

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

operasi kecerdasan buatan (AIOps)

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

enkripsi asimetris

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

atomisitas, konsistensi, isolasi, daya tahan (ACID)

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

kontrol akses berbasis atribut (ABAC)

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

sumber data otoritatif

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

Zona Ketersediaan

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

integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD)

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

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

CV

Lihat visi komputer.

D

data saat istirahat

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

klasifikasi data

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

penyimpangan data

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

data dalam transit

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

jala data

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

minimalisasi data

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

perimeter data

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

prapemrosesan data

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

asal data

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

subjek data

Individu yang datanya dikumpulkan dan diproses.

gudang data

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

bahasa definisi database (DDL)

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

bahasa manipulasi basis data (DHTML)

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

DDL

Lihat bahasa definisi database.

ansambel yang dalam

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

pembelajaran mendalam

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

defense-in-depth

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

administrator yang didelegasikan

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

deployment

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

lingkungan pengembangan

Lihat lingkungan.

kontrol detektif

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

pemetaan aliran nilai pengembangan (DVSM)

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

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

kembar digital

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

tabel dimensi

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

musibah

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

pemulihan bencana (DR)

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

DML~

Lihat bahasa manipulasi basis data.

desain berbasis domain

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

DR

Lihat pemulihan bencana.

deteksi drift

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

DVSM

Lihat pemetaan aliran nilai pengembangan.

E

EDA

Lihat analisis data eksplorasi.

EDI

Lihat pertukaran data elektronik.

komputasi tepi

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

pertukaran data elektronik (EDI)

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

enkripsi

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

kunci enkripsi

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

endianness

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

titik akhir

Lihat titik akhir layanan.

layanan endpoint

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

perencanaan sumber daya perusahaan (ERP)

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

enkripsi amplop

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

lingkungan

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

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

epik

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

ERP

Lihat perencanaan sumber daya perusahaan.

analisis data eksplorasi (EDA)

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

F

tabel fakta

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

gagal cepat

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

batas isolasi kesalahan

Dalam AWS Cloud, batas seperti Availability Zone, 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 zero-shot prompting.

FGAC

Lihat kontrol akses berbutir halus.

kontrol akses berbutir halus (FGAC)

Penggunaan beberapa kondisi untuk mengizinkan atau menolak permintaan akses.

migrasi flash-cut

Metode migrasi database yang menggunakan replikasi data berkelanjutan melalui pengambilan data perubahan untuk memigrasikan data dalam waktu sesingkat mungkin, alih-alih menggunakan pendekatan bertahap. Tujuannya adalah untuk menjaga downtime seminimal mungkin.

FM

Lihat model pondasi.

model pondasi (FM)

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

G

AI generatif

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

pemblokiran geografis

Lihat pembatasan geografis.

pembatasan geografis (pemblokiran geografis)

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

Alur kerja Gitflow

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

gambar emas

Sebuah snapshot dari sistem atau perangkat lunak yang digunakan sebagai template untuk menyebarkan instance baru dari sistem atau perangkat lunak itu. Misalnya, di bidang manufaktur, gambar emas dapat digunakan untuk menyediakan perangkat lunak pada beberapa perangkat dan membantu meningkatkan kecepatan, skalabilitas, dan produktivitas dalam operasi manufaktur perangkat.

strategi greenfield

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

pagar pembatas

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

H

HA

Lihat ketersediaan tinggi.

migrasi database heterogen

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

ketersediaan tinggi (HA)

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

modernisasi sejarawan

Pendekatan yang digunakan untuk memodernisasi dan meningkatkan sistem teknologi operasional (OT) untuk melayani kebutuhan industri manufaktur dengan lebih baik. Sejarawan

adalah jenis database yang digunakan untuk mengumpulkan dan menyimpan data dari berbagai sumber di pabrik.

data penahanan

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

migrasi database homogen

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

data panas

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

perbaikan terbaru

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

periode hypercare

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

I

IAc

Lihat infrastruktur sebagai kode.

kebijakan berbasis identitas

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

aplikasi idle

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

IIoT

Lihat Internet of Things industri.

infrastruktur yang tidak dapat diubah

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

masuk (masuknya) VPC

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

migrasi inkremental

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

Industri 4.0

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

infrastruktur

Semua sumber daya dan aset yang terkandung dalam lingkungan aplikasi.

infrastruktur sebagai kode (IAc)

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

Internet of Things industri (IIoT)

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

inspeksi VPC

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

Internet of Things (IoT)

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

interpretasi

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

IoT

Lihat Internet of Things.

Perpustakaan informasi TI (ITIL)

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

Manajemen layanan TI (ITSM)

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

ITIL

Lihat perpustakaan informasi TI.

ITSM

Lihat manajemen layanan TI.

L

kontrol akses berbasis label (LBAC)

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

landing zone

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

model bahasa besar (LLM)

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

migrasi besar

Migrasi 300 atau lebih server.

LBAC

Lihat kontrol akses berbasis label.

hak istimewa paling sedikit

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

angkat dan geser

Lihat 7 Rs.

sistem endian kecil

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

LLM

Lihat model bahasa besar.

lingkungan yang lebih rendah

Lihat lingkungan.

M

pembelajaran mesin (ML)

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

cabang utama

Lihat cabang.

malware

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

layanan terkelola

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

sistem eksekusi manufaktur (MES)

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

PETA

Lihat Program Percepatan Migrasi.

mekanisme

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

akun anggota

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

MES

Lihat sistem eksekusi manufaktur.

Transportasi Telemetri Antrian Pesan (MQTT)

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

layanan mikro

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

informasi selengkapnya, lihat Mengintegrasikan layanan mikro dengan menggunakan layanan tanpa AWS server.

arsitektur microservices

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

Program Percepatan Migrasi (MAP)

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

migrasi dalam skala

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

pabrik migrasi

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

metadata migrasi

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

pola migrasi

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

Penilaian Portofolio Migrasi (MPA)

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

Penilaian Kesiapan Migrasi (MRA)

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

strategi migrasi

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

ML

Lihat pembelajaran mesin.

modernisasi

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

penilaian kesiapan modernisasi

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

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

aplikasi monolitik (monolit)

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

MPA

Lihat Penilaian Portofolio Migrasi.

MQTT

Lihat Transportasi Telemetri Antrian Pesan.

klasifikasi multiclass

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

infrastruktur yang bisa berubah

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

O

OAC

Lihat kontrol akses asal.

OAI

Lihat identitas akses asal.

OCM

Lihat manajemen perubahan organisasi.

migrasi offline

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

OI

Lihat integrasi operasi.

OLA

Lihat perjanjian tingkat operasional.

migrasi online

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

OPC-UA

Lihat Komunikasi Proses Terbuka - Arsitektur Terpadu.

Komunikasi Proses Terbuka - Arsitektur Terpadu (OPC-UA)

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

perjanjian tingkat operasional (OLA)

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

Tinjauan Kesiapan Operasional (ORR)

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

teknologi operasional (OT)

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

integrasi operasi (OI)

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

jejak organisasi

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

manajemen perubahan organisasi (OCM)

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

kontrol akses asal (OAC)

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

identitas akses asal (OAI)

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

ORR

Lihat tinjauan kesiapan operasional.

OT

Lihat teknologi operasional.

keluar (jalan keluar) VPC

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

P

batas izin

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

Informasi Identifikasi Pribadi (PII)

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

PII

Lihat informasi yang dapat diidentifikasi secara pribadi.

buku pedoman

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

PLC

Lihat pengontrol logika yang dapat diprogram.

PLM

Lihat manajemen siklus hidup produk.

kebijakan

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

persistensi poliglot

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

penilaian portofolio

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

predikat

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

predikat pushdown

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

kontrol preventif

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

principal

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

privasi berdasarkan desain

Pendekatan rekayasa sistem yang memperhitungkan privasi melalui seluruh proses pengembangan.

zona host pribadi

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

kontrol proaktif

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

manajemen siklus hidup produk (PLM)

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

lingkungan produksi

Lihat lingkungan.

pengontrol logika yang dapat diprogram (PLC)

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

rantai cepat

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

pseudonimisasi

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

publish/subscribe (pub/sub)

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

Q

rencana kueri

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

regresi rencana kueri

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

R

Matriks RACI

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

LAP

Lihat Retrieval Augmented Generation.

ransomware

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

Matriks RASCI

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

RCAC

Lihat kontrol akses baris dan kolom.

replika baca

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

arsitek ulang

Lihat 7 Rs.

tujuan titik pemulihan (RPO)

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

tujuan waktu pemulihan (RTO)

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

refactor

Lihat 7 Rs.

Region

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

regresi

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

rehost

Lihat 7 Rs.

melepaskan

Dalam proses penyebaran, tindakan mempromosikan perubahan pada lingkungan produksi.

memindahkan

Lihat 7 Rs.

memplatform ulang

Lihat 7 Rs.

pembelian kembali

Lihat 7 Rs.

ketahanan

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

kebijakan berbasis sumber daya

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

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

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

kontrol responsif

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

melestarikan

Lihat 7 Rs.

pensiun

Lihat 7 Rs.

Retrieval Augmented Generation (RAG)

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

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

rotasi

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

kontrol akses baris dan kolom (RCAC)

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

RPO

Lihat tujuan titik pemulihan.

RTO

Lihat tujuan waktu pemulihan.

buku runbook

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

D

SAML 2.0

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

PENIPUAN

Lihat kontrol pengawasan dan akuisisi data.

SCP

Lihat kebijakan kontrol layanan.

Rahasia

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

keamanan dengan desain

Pendekatan rekayasa sistem yang memperhitungkan keamanan melalui seluruh proses pengembangan.

kontrol keamanan

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

pengerasan keamanan

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

sistem informasi keamanan dan manajemen acara (SIEM)

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

otomatisasi respons keamanan

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

enkripsi sisi server

Enkripsi data di tujuannya, oleh Layanan AWS yang menerimanya.

kebijakan kontrol layanan (SCP)

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

titik akhir layanan

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

perjanjian tingkat layanan (SLA)

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

indikator tingkat layanan (SLI)

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

tujuan tingkat layanan (SLO)

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

model tanggung jawab bersama

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

SIEM

Lihat informasi keamanan dan sistem manajemen acara.

titik kegagalan tunggal (SPOF)

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

SLA

Lihat perjanjian tingkat layanan.

SLI

Lihat indikator tingkat layanan.

SLO

Lihat tujuan tingkat layanan.

split-and-seed model

Pola untuk menskalakan dan mempercepat proyek modernisasi. Ketika fitur baru dan rilis produk didefinisikan, tim inti berpisah untuk membuat tim produk baru. Ini membantu meningkatkan kemampuan dan layanan organisasi Anda, meningkatkan produktivitas pengembang, dan mendukung inovasi yang cepat. Untuk informasi lebih lanjut, lihat Pendekatan bertahap untuk memodernisasi aplikasi di. AWS Cloud

SPOF

Lihat satu titik kegagalan.

skema bintang

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

pola ara pencekik

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

subnet

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

kontrol pengawasan dan akuisisi data (SCADA)

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

enkripsi simetris

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

pengujian sintetis

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

sistem prompt

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

T

tag

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

variabel target

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

daftar tugas

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

lingkungan uji

Lihat lingkungan.

pelatihan

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

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

gerbang transit

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

alur kerja berbasis batang

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

akses tepercaya

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

penyetelan

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

tim dua pizza

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

U

waswas

Sebuah konsep yang mengacu pada informasi yang tidak tepat, tidak lengkap, atau tidak diketahui yang dapat merusak keandalan model ML prediktif. Ada dua jenis ketidakpastian:

ketidakpastian epistemik disebabkan oleh data yang terbatas dan tidak lengkap, sedangkan ketidakpastian aleatorik disebabkan oleh kebisingan dan keacakan yang melekat dalam data.

tugas yang tidak terdiferensiasi

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

lingkungan atas

Lihat lingkungan.

V

menyedot debu

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

kendali versi

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

Peering VPC

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

kerentanan

Kelemahan perangkat lunak atau perangkat keras yang membahayakan keamanan sistem.

W

cache hangat

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

data hangat

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

fungsi jendela

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

beban kerja

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

aliran kerja

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

CACING

Lihat menulis sekali, baca banyak.

WQF

Lihat AWS Kerangka Kualifikasi Beban Kerja.

tulis sekali, baca banyak (WORM)

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

Z

eksploitasi zero-day

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

kerentanan zero-day

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

bisikan zero-shot

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

aplikasi zombie

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

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