Praktik terbaik untuk menggunakan Penyedia Terraform AWS

AWS Bimbingan Preskriptif

AWS Bimbingan Preskriptif: Praktik terbaik untuk menggunakan Penyedia Terraform AWS

Table of Contents

Praktik terbaik untuk menggunakan Penyedia Terraform AWS

Michael Begin, DevOps Konsultan Senior, Amazon Web Services (AWS)

Agustus 2025 (sejarah dokumen)

Mengelola infrastruktur sebagai kode (IAc) dengan Terraform on AWS menawarkan manfaat penting seperti peningkatan konsistensi, keamanan, dan kelincahan. Namun, karena konfigurasi Terraform Anda bertambah dalam ukuran dan kompleksitas, menjadi penting untuk mengikuti praktik terbaik untuk menghindari jebakan.

Panduan ini memberikan praktik terbaik yang direkomendasikan untuk menggunakan AWS Penyedia Terraform dari. HashiCorp Ini memandu Anda melalui versi yang tepat, kontrol keamanan, backend jarak jauh, struktur basis kode, dan penyedia komunitas untuk mengoptimalkan Terraform. AWS Setiap bagian menyelami detail lebih lanjut tentang spesifikasi penerapan praktik terbaik ini:

- Keamanan
- Backend
- Struktur dasar kode dan organisasi
- AWS Manajemen versi penyedia
- Modul komunitas

Tujuan

Panduan ini membantu Anda memperoleh pengetahuan operasional tentang AWS Penyedia Terraform dan membahas tujuan bisnis berikut yang dapat Anda capai dengan mengikuti praktik terbaik IAc seputar keamanan, keandalan, kepatuhan, dan produktivitas pengembang.

- Tingkatkan kualitas dan konsistensi kode infrastruktur di seluruh proyek Terraform.
- Mempercepat orientasi pengembang dan kemampuan untuk berkontribusi pada kode infrastruktur.
- Meningkatkan kelincahan bisnis melalui perubahan infrastruktur yang lebih cepat.
- Mengurangi kesalahan dan downtime yang terkait dengan perubahan infrastruktur.
- Optimalkan biaya infrastruktur dengan mengikuti praktik terbaik IAc.
- Perkuat postur keamanan Anda secara keseluruhan melalui implementasi praktik terbaik.

Target audiens

Target audiens untuk panduan ini mencakup prospek teknis dan manajer yang mengawasi tim yang menggunakan Terraform untuk IAc. AWS Pembaca potensial lainnya termasuk insinyur infrastruktur, DevOps insinyur, arsitek solusi, dan pengembang yang secara aktif menggunakan Terraform untuk mengelola infrastruktur. AWS

Mengikuti praktik terbaik ini akan menghemat waktu dan membantu membuka manfaat IAc untuk peran ini.

Gambaran Umum

Penyedia Terraform adalah plugin yang memungkinkan Terraform berinteraksi dengan API yang berbeda. AWS Penyedia Terraform adalah plugin resmi untuk mengelola AWS infrastruktur sebagai kode (IAc) dengan Terraform. Ini menerjemahkan sintaks Terraform ke dalam panggilan AWS API untuk membuat, membaca, memperbarui, dan menghapus sumber daya. AWS

AWS Penyedia menangani otentikasi, menerjemahkan sintaks Terraform ke panggilan AWS API, dan menyediakan sumber daya di. AWS Anda menggunakan blok `provider` kode Terraform untuk mengonfigurasi plugin penyedia yang digunakan Terraform untuk berinteraksi dengan API. AWS Anda dapat mengonfigurasi beberapa blok AWS Penyedia untuk mengelola sumber daya di berbagai wilayah Akun AWS dan wilayah.

Berikut adalah contoh konfigurasi Terraform yang menggunakan beberapa blok AWS Penyedia dengan alias untuk mengelola database Amazon Relational Database Service (Amazon RDS) yang memiliki replika di Wilayah dan akun yang berbeda. Penyedia primer dan sekunder mengambil peran AWS Identity and Access Management (IAM) yang berbeda:

```
# Configure the primary AWS Provider
provider "aws" {
  region = "us-west-1"
  alias  = "primary"
}

# Configure a secondary AWS Provider for the replica Region and account
provider "aws" {
  region      = "us-east-1"
  alias       = "replica"
  assume_role {
    role_arn     = "arn:aws:iam::<replica-account-id>:role/<role-name>"
    session_name = "terraform-session"
  }
}

# Primary Amazon RDS database
resource "aws_db_instance" "primary" {
  provider = aws.primary

  # ... RDS instance configuration
}
```

```
# Read replica in a different Region and account
resource "aws_db_instance" "read_replica" {
  provider = aws.replica

  # ... RDS read replica configuration
  replicate_source_db = aws_db_instance.primary.id
}
```

Dalam contoh ini:

- `provider`Blok pertama mengonfigurasi AWS Penyedia utama di `us-west-1` Wilayah dengan `primary` alias.
- `provider`Blok kedua mengonfigurasi AWS Penyedia sekunder di `us-east-1` Wilayah dengan `replica` alias. Penyedia ini digunakan untuk membuat replika baca database utama di Wilayah dan akun yang berbeda. `assume_role`Blok ini digunakan untuk mengambil peran IAM dalam akun replika. `role_arn`Menentukan Nama Sumber Daya Amazon (ARN) dari peran IAM yang akan diambil, `session_name` dan merupakan pengidentifikasi unik untuk sesi Terraform.
- `aws_db_instance.primary`Sumber daya membuat database Amazon RDS utama dengan menggunakan `primary` penyedia di `us-west-1` Wilayah.
- `aws_db_instance.read_replica`Sumber daya membuat replika baca database utama di `us-east-1` Wilayah dengan menggunakan `replica` penyedia. `replicate_source_db`Atribut mereferensikan ID `primary` database.

Praktik terbaik keamanan

Mengelola otentikasi, kontrol akses, dan keamanan dengan benar sangat penting untuk penggunaan Penyedia AWS Terraform yang aman. Bagian ini menguraikan praktik terbaik seputar:

- Peran dan izin IAM untuk akses hak istimewa paling sedikit
- Mengamankan kredensi untuk membantu mencegah akses tidak sah ke akun dan sumber daya AWS
- Enkripsi status jarak jauh untuk membantu melindungi data sensitif
- Infrastruktur dan pemindaian kode sumber untuk mengidentifikasi kesalahan konfigurasi
- Kontrol akses untuk penyimpanan status jarak jauh
- Penegakan kebijakan sentinel untuk menerapkan pagar pembatas tata kelola

Mengikuti praktik terbaik ini membantu memperkuat postur keamanan Anda saat Anda menggunakan Terraform untuk mengelola AWS infrastruktur.

Ikuti prinsip hak istimewa paling sedikit

Keistimewaan terkecil adalah prinsip keamanan mendasar yang mengacu pada pemberian hanya izin minimum yang diperlukan bagi pengguna, proses, atau sistem untuk menjalankan fungsi yang dimaksudkan. Ini adalah konsep inti dalam kontrol akses dan tindakan pencegahan terhadap akses yang tidak sah dan potensi pelanggaran data.

Prinsip hak istimewa paling sedikit ditekankan beberapa kali di bagian ini karena secara langsung berkaitan dengan bagaimana Terraform mengautentikasi dan menjalankan tindakan terhadap penyedia cloud seperti. AWS

Saat Anda menggunakan Terraform untuk menyediakan dan mengelola AWS sumber daya, ia bertindak atas nama entitas (pengguna atau peran) yang memerlukan izin yang sesuai untuk melakukan panggilan API. Tidak mengikuti sedikit hak istimewa membuka risiko keamanan utama:

- Jika Terraform memiliki izin berlebihan di luar yang diperlukan, kesalahan konfigurasi yang tidak diinginkan dapat membuat perubahan atau penghapusan yang tidak diinginkan.
- Hibah akses yang terlalu permisif meningkatkan cakupan dampak jika file atau kredensil status Terraform dikompromikan.

- Tidak mengikuti sedikit hak istimewa bertentangan dengan praktik terbaik keamanan dan persyaratan kepatuhan peraturan untuk memberikan akses minimal yang diperlukan.

Gunakan IAM role

Gunakan peran IAM alih-alih pengguna IAM sedapat mungkin untuk meningkatkan keamanan dengan Penyedia AWS Terraform. Peran IAM menyediakan kredensil keamanan sementara yang berputar secara otomatis, yang menghilangkan kebutuhan untuk mengelola kunci akses jangka panjang. Peran juga menawarkan kontrol akses yang tepat melalui kebijakan IAM.

Berikan akses hak istimewa paling sedikit dengan menggunakan kebijakan IAM

Buat kebijakan IAM dengan hati-hati untuk memastikan bahwa peran dan pengguna hanya memiliki set izin minimum yang diperlukan untuk beban kerja mereka. Mulailah dengan kebijakan kosong dan tambahkan layanan dan tindakan yang diizinkan secara berulang. Untuk mencapai ini:

- Aktifkan IAM Access Analyzer untuk mengevaluasi kebijakan dan menyorot izin yang tidak digunakan yang dapat dihapus.
- Tinjau kebijakan secara manual untuk menghapus kemampuan apa pun yang tidak penting untuk tanggung jawab peran yang dimaksudkan.
- Gunakan variabel dan tag kebijakan IAM untuk menyederhanakan manajemen izin.

Kebijakan yang dibangun dengan baik memberikan akses yang cukup untuk menyelesaikan tanggung jawab beban kerja dan tidak lebih. Tentukan tindakan di tingkat operasi, dan izinkan panggilan hanya diperlukan APIs pada sumber daya tertentu.

Mengikuti praktik terbaik ini mengurangi ruang lingkup dampak dan mengikuti prinsip-prinsip keamanan dasar pemisahan tugas dan akses hak istimewa yang paling sedikit. Mulai akses ketat dan buka secara bertahap sesuai kebutuhan, alih-alih mulai membuka dan mencoba membatasi akses nanti.

Asumsikan peran IAM untuk otentikasi lokal

Saat Anda menjalankan Terraform secara lokal, hindari mengonfigurasi kunci akses statis. Sebagai gantinya, gunakan peran IAM untuk memberikan akses istimewa sementara tanpa mengekspos kredensi jangka panjang.

Pertama, buat peran IAM dengan izin minimum yang diperlukan dan tambahkan hubungan kepercayaan yang memungkinkan peran IAM diasumsikan oleh akun pengguna Anda atau identitas gabungan. Ini mengizinkan penggunaan sementara peran tersebut.

Contoh kebijakan hubungan kepercayaan:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/terraform-execution"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Kemudian, jalankan AWS CLI perintah aws sts assume-role untuk mengambil kredensil berumur pendek untuk peran tersebut. Kredensi ini biasanya berlaku selama satu jam.

AWS CLI contoh perintah:

```
aws sts assume-role --role-arn arn:aws:iam::111122223333:role/terraform-execution --
role-session-name terraform-session-example
```

Output dari perintah berisi kunci akses, kunci rahasia, dan token sesi yang dapat Anda gunakan untuk mengautentikasi ke AWS:

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:terraform-session-example",
        "Arn": "arn:aws:sts::111122223333:assumed-role/terraform-execution/terraform-
session-example"
    },
    "Credentials": {
        "SecretAccessKey": " wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "SessionToken": " AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
```

```
        "Expiration": "2024-03-15T00:05:07Z",
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE"
    }
}
```

AWS Penyedia juga dapat secara otomatis menangani asumsi peran.

Contoh konfigurasi penyedia untuk mengasumsikan peran IAM:

```
provider "aws" {
  assume_role {
    role_arn     = "arn:aws:iam::111122223333:role/terraform-execution"
    session_name = "terraform-session-example"
  }
}
```

Ini memberikan hak istimewa yang tinggi secara ketat untuk durasi sesi Terraform. Kunci sementara tidak dapat bocor karena akan kedaluwarsa secara otomatis setelah durasi maksimum sesi.

Manfaat utama dari praktik terbaik ini termasuk peningkatan keamanan dibandingkan dengan kunci akses yang berumur panjang, kontrol akses halus pada peran untuk hak istimewa paling sedikit, dan kemampuan untuk dengan mudah mencabut akses dengan memodifikasi izin peran. Dengan menggunakan peran IAM, Anda juga menghindari keharusan menyimpan rahasia secara langsung secara lokal dalam skrip atau disk, yang membantu Anda berbagi konfigurasi Terraform dengan aman di seluruh tim.

Menggunakan peran IAM untuk otentikasi Amazon EC2

Saat Anda menjalankan instans Terraform dari Amazon Elastic Compute Cloud (Amazon EC2), hindari menyimpan kredensil jangka panjang secara lokal. Sebagai gantinya, gunakan peran IAM dan profil instance untuk memberikan izin hak istimewa paling sedikit secara otomatis.

Pertama, buat peran IAM dengan izin minimum dan tetapkan peran ke profil instance. Profil instans memungkinkan instans EC2 mewarisi izin yang ditentukan dalam peran. Kemudian, luncurkan instance dengan menentukan profil instance itu. Instance akan mengautentikasi melalui peran terlampir.

Sebelum Anda menjalankan operasi Terraform apa pun, verifikasi bahwa peran tersebut ada dalam metadata instance untuk mengonfirmasi bahwa kredensil berhasil diwarisi.

```
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600")

curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/meta-data/
iam/security-credentials/
```

Pendekatan ini menghindari hardcoding AWS kunci permanen ke dalam skrip atau konfigurasi Terraform dalam instance. Kredensi sementara tersedia untuk Terraform secara transparan melalui peran dan profil instans.

Manfaat utama dari praktik terbaik ini termasuk peningkatan keamanan atas kredensil jangka panjang, pengurangan overhead manajemen kredensi, dan konsistensi antara pengembangan, pengujian, dan lingkungan produksi. Otentikasi peran IAM menyederhanakan Terraform berjalan dari instans EC2 sambil menegakkan akses hak istimewa paling sedikit.

Gunakan kredensil dinamis untuk ruang kerja HCP Terraform

HCP Terraform adalah layanan terkelola yang disediakan oleh HashiCorp yang membantu tim menggunakan Terraform untuk menyediakan dan mengelola infrastruktur di berbagai proyek dan lingkungan. Saat Anda menjalankan Terraform di HCP Terraform, gunakan kredensil dinamis untuk menyederhanakan dan mengamankan otentikasi. AWS Terraform secara otomatis menukar kredensil sementara pada setiap proses tanpa memerlukan asumsi peran IAM.

Manfaatnya termasuk rotasi rahasia yang lebih mudah, manajemen kredensi terpusat di seluruh ruang kerja, izin hak istimewa paling sedikit, dan menghilangkan kunci hardcode. Mengandalkan kunci fana yang di-hash meningkatkan keamanan dibandingkan dengan kunci akses yang berumur panjang.

Gunakan peran IAM di AWS CodeBuild

Di AWS CodeBuild, jalankan build Anda dengan menggunakan peran IAM yang ditetapkan ke proyek. CodeBuild Hal ini memungkinkan setiap build untuk secara otomatis mewarisi kredensi sementara dari peran alih-alih menggunakan kunci jangka panjang.

Jalankan GitHub Tindakan dari jarak jauh di HCP Terraform

Konfigurasikan alur kerja GitHub Tindakan untuk menjalankan Terraform dari jarak jauh di ruang kerja HCP Terraform. Andalkan kredensil dinamis dan penguncian status jarak jauh alih-alih manajemen GitHub rahasia.

Gunakan GitHub Tindakan dengan OIDC dan konfigurasikan tindakan Credentials AWS

Gunakan standar OpenID Connect (OIDC) untuk menggabungkan identitas Actions melalui IAM. GitHub Gunakan tindakan Configure AWS Credentials untuk menukar GitHub token dengan AWS kredensial sementara tanpa memerlukan kunci akses jangka panjang.

Gunakan GitLab dengan OIDC dan AWS CLI

Gunakan standar OIDC untuk menyatukan GitLab identitas melalui IAM untuk akses sementara. Dengan mengandalkan OIDC, Anda menghindari keharusan langsung mengelola kunci AWS akses jangka panjang di dalamnya. GitLab Kredensil dipertukarkan just-in-time, yang meningkatkan keamanan. Pengguna juga mendapatkan akses hak istimewa paling sedikit sesuai dengan izin dalam peran IAM.

Gunakan pengguna IAM unik dengan alat otomatisasi lama

Jika Anda memiliki alat otomatisasi dan skrip yang tidak memiliki dukungan asli untuk menggunakan peran IAM, Anda dapat membuat pengguna IAM individual untuk memberikan akses terprogram. Prinsip hak istimewa paling sedikit masih berlaku. Minimalkan izin kebijakan dan andalkan peran terpisah untuk setiap pipeline atau skrip. Saat Anda bermigrasi ke alat atau skrip yang lebih modern, mulailah mendukung peran secara native dan bertahap bertransisi ke sana.

> **Warning**
>
> Pengguna IAM memiliki kredensi jangka panjang, yang menghadirkan risiko keamanan. Untuk membantu mengurangi risiko ini, kami menyarankan agar Anda memberikan pengguna ini hanya izin yang mereka perlukan untuk melakukan tugas dan menghapus pengguna ini ketika mereka tidak lagi diperlukan.

Gunakan plugin Jenkins AWS Credentials

Gunakan plugin AWS Credentials di Jenkins untuk mengonfigurasi dan menyuntikkan kredensil secara terpusat ke dalam build secara dinamis AWS . Ini menghindari memeriksa rahasia ke dalam kontrol sumber.

Terus memantau, memvalidasi, dan mengoptimalkan hak istimewa paling sedikit

Seiring waktu, izin tambahan mungkin diberikan yang dapat melebihi kebijakan minimum yang diperlukan. Terus menganalisis akses untuk mengidentifikasi dan menghapus hak yang tidak perlu.

Terus memantau penggunaan kunci akses

Jika Anda tidak dapat menghindari penggunaan kunci akses, gunakan laporan kredensi IAM untuk menemukan kunci akses yang tidak digunakan yang lebih tua dari 90 hari, dan cabut kunci tidak aktif di kedua akun pengguna dan peran mesin. Peringatkan administrator untuk secara manual mengonfirmasi penghapusan kunci untuk karyawan dan sistem aktif.

Memantau penggunaan kunci membantu Anda mengoptimalkan izin karena Anda dapat mengidentifikasi dan menghapus hak yang tidak digunakan. Ketika Anda mengikuti praktik terbaik ini dengan rotasi kunci akses, ini membatasi masa pakai kredensi dan menerapkan akses hak istimewa paling sedikit.

AWS menyediakan beberapa layanan dan fitur yang dapat Anda gunakan untuk mengatur peringatan dan pemberitahuan untuk administrator. Berikut adalah beberapa opsi:

- AWS Config: Anda dapat menggunakan AWS Config aturan untuk mengevaluasi pengaturan konfigurasi AWS sumber daya Anda, termasuk kunci akses IAM. Anda dapat membuat aturan khusus untuk memeriksa kondisi tertentu, seperti kunci akses yang tidak digunakan yang lebih lama dari jumlah hari tertentu. Ketika aturan dilanggar, AWS Config dapat memulai evaluasi untuk perbaikan atau mengirim pemberitahuan ke topik Simple Notification Service Amazon (Amazon SNS).
- AWS Security Hub CSPMSecurity Hub CSPM memberikan pandangan komprehensif tentang postur keamanan AWS akun Anda dan dapat membantu mendeteksi dan memberi tahu Anda tentang potensi masalah keamanan, termasuk kunci akses IAM yang tidak digunakan atau tidak aktif. Security Hub CSPM dapat diintegrasikan dengan Amazon EventBridge dan Amazon SNS atau Amazon Q Developer dalam aplikasi obrolan untuk mengirim pemberitahuan ke administrator.
- AWS Lambda: Fungsi Lambda dapat dipanggil oleh berbagai acara, termasuk CloudWatch Acara Amazon atau AWS Config aturan. Anda dapat menulis fungsi Lambda khusus untuk mengevaluasi penggunaan kunci akses IAM, melakukan pemeriksaan tambahan, dan mengirim pemberitahuan dengan menggunakan layanan seperti Amazon SNS atau Amazon Q Developer dalam aplikasi obrolan.

Terus memvalidasi kebijakan IAM

Gunakan IAM Access Analyzer untuk mengevaluasi kebijakan yang melekat pada peran dan mengidentifikasi layanan yang tidak digunakan atau tindakan berlebih yang diberikan. Menerapkan tinjauan akses berkala untuk memverifikasi secara manual bahwa kebijakan sesuai dengan persyaratan saat ini.

Bandingkan kebijakan yang ada dengan kebijakan yang dihasilkan oleh IAM Access Analyzer dan hapus izin yang tidak perlu. Anda juga harus memberikan laporan kepada pengguna dan secara otomatis mencabut izin yang tidak digunakan setelah masa tenggang. Ini membantu memastikan bahwa kebijakan minimal tetap berlaku.

Secara proaktif dan sering mencabut akses usang meminimalkan kredensil yang mungkin berisiko selama pelanggaran. Otomasi menyediakan kebersihan kredenal jangka panjang dan pengoptimalan izin yang berkelanjutan. Mengikuti praktik terbaik ini membatasi ruang lingkup dampak dengan secara proaktif menegakkan hak istimewa terkecil di seluruh AWS identitas dan sumber daya.

Penyimpanan status jarak jauh yang aman

Penyimpanan status jarak jauh mengacu pada penyimpanan file status Terraform dari jarak jauh alih-alih secara lokal di mesin tempat Terraform berjalan. File status sangat penting karena melacak sumber daya yang disediakan oleh Terraform dan metadatanya.

Kegagalan untuk mengamankan status jarak jauh dapat menyebabkan masalah serius seperti hilangnya data negara, ketidakmampuan untuk mengelola infrastruktur, penghapusan sumber daya yang tidak disengaja, dan paparan informasi sensitif yang mungkin ada dalam file status. Untuk alasan ini, mengamankan penyimpanan status jarak jauh sangat penting untuk penggunaan Terraform tingkat produksi.

Aktifkan enkripsi dan kontrol akses

Gunakan Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) enkripsi sisi server (SSE) untuk mengenkripsi status jarak jauh saat istirahat.

Batasi akses langsung ke alur kerja kolaboratif

- Struktur alur kerja kolaborasi di HCP Terraform atau dalam pipeline di dalam repositori Git Anda untuk membatasi akses status langsung. CI/CD

- Andalkan permintaan tarik, jalankan persetujuan, pemeriksaan kebijakan, dan pemberitahuan untuk mengoordinasikan perubahan.

Mengikuti panduan ini membantu mengamankan atribut sumber daya yang sensitif dan menghindari konflik dengan perubahan anggota tim. Enkripsi dan perlindungan akses yang ketat membantu mengurangi permukaan serangan, dan alur kerja kolaborasi memungkinkan produktivitas.

Gunakan AWS Secrets Manager

Ada banyak sumber daya dan sumber data di Terraform yang menyimpan nilai rahasia dalam teks biasa di file status. Hindari menyimpan rahasia di negara bagian–gunakan AWS Secrets Managersebagai gantinya.

Alih-alih mencoba mengenkripsi nilai sensitif secara manual, andalkan dukungan bawaan Terraform untuk manajemen status sensitif. Saat mengekspor nilai sensitif ke output, pastikan nilainya ditandai sebagai sensitif.

Terus memindai infrastruktur dan kode sumber

Secara proaktif memindai infrastruktur dan kode sumber secara terus menerus untuk risiko seperti kredensi yang terpapar atau kesalahan konfigurasi untuk mengeraskan postur keamanan Anda. Atasi temuan segera dengan mengkonfigurasi ulang atau menambal sumber daya.

Gunakan AWS layanan untuk pemindaian dinamis

Gunakan alat AWS bawaan seperti Amazon Inspector, AWS Security Hub CSPMAmazon Detective, dan Amazon GuardDuty untuk memantau infrastruktur yang disediakan di seluruh akun dan Wilayah. Jadwalkan pemindaian berulang di Security Hub CSPM untuk melacak penyebaran dan penyimpangan konfigurasi. Pindai instans EC2, fungsi Lambda, wadah, bucket S3, dan sumber daya lainnya.

Lakukan analisis statis

Sematkan penganalisis statis seperti Checkov langsung ke CI/CD saluran pipa untuk memindai kode konfigurasi Terraform (HCL) dan mengidentifikasi risiko terlebih dahulu sebelum penerapan. Ini memindahkan pemeriksaan keamanan ke titik sebelumnya dalam proses pengembangan (disebut sebagai pergeseran ke kiri) dan mencegah infrastruktur yang salah konfigurasi.

Pastikan remediasi yang cepat

Untuk semua temuan pemindaian, pastikan perbaikan segera dengan memperbarui konfigurasi Terraform, menerapkan tambalan, atau mengonfigurasi ulang sumber daya secara manual sebagaimana mestinya. Menurunkan tingkat risiko dengan mengatasi akar penyebabnya.

Menggunakan pemindaian infrastruktur dan pemindaian kode memberikan wawasan berlapis di seluruh konfigurasi Terraform, sumber daya yang disediakan, dan kode aplikasi. Ini memaksimalkan cakupan risiko dan kepatuhan melalui kontrol preventif, detektif, dan reaktif sambil menyematkan keamanan lebih awal ke dalam siklus hidup pengembangan perangkat lunak (SDLC).

Menegakkan pemeriksaan kebijakan

Gunakan kerangka kerja kode seperti kebijakan HashiCorp Sentinel untuk menyediakan pagar pembatas tata kelola dan templat standar untuk penyediaan infrastruktur dengan Terraform.

Kebijakan sentinel dapat menentukan persyaratan atau pembatasan konfigurasi Terraform agar selaras dengan standar organisasi dan praktik terbaik. Misalnya, Anda dapat menggunakan kebijakan Sentinel untuk:

- Memerlukan tag pada semua sumber daya.
- Batasi jenis instans ke daftar yang disetujui.
- Menegakkan variabel wajib.
- Mencegah penghancuran sumber daya produksi.

Menyematkan pemeriksaan kebijakan ke dalam siklus hidup konfigurasi Terraform memungkinkan penegakan standar dan pedoman arsitektur secara proaktif. Sentinel menyediakan logika kebijakan bersama yang membantu mempercepat pengembangan sekaligus mencegah praktik yang tidak disetujui.

Praktik terbaik backend

Menggunakan backend jarak jauh yang tepat untuk menyimpan file status Anda sangat penting untuk memungkinkan kolaborasi, memastikan integritas file status melalui penguncian, menyediakan pencadangan dan pemulihan yang andal, mengintegrasikan dengan CI/CD alur kerja, dan memanfaatkan fitur keamanan, tata kelola, dan manajemen tingkat lanjut yang ditawarkan oleh layanan terkelola seperti HCP Terraform.

Terraform mendukung berbagai jenis backend seperti Kubernetes, Consul, dan HTTP. HashiCorp Namun, panduan ini berfokus pada Amazon S3, yang merupakan solusi backend optimal untuk sebagian besar pengguna. AWS

Sebagai layanan penyimpanan objek yang dikelola sepenuhnya yang menawarkan daya tahan dan ketersediaan tinggi, Amazon S3 menyediakan backend yang aman, dapat diskalakan, dan berbiaya rendah untuk mengelola status Terraform. AWS Jejak global dan ketahanan Amazon S3 melebihi apa yang dapat dicapai sebagian besar tim dengan mengelola sendiri penyimpanan status. Selain itu, terintegrasi secara native dengan kontrol AWS akses, opsi enkripsi, kemampuan pembuatan versi, dan layanan lainnya menjadikan Amazon S3 pilihan backend yang nyaman.

Panduan ini tidak memberikan panduan backend untuk solusi lain seperti Kubernetes atau Konsul karena target audiens utama adalah pelanggan. AWS Untuk tim yang sepenuhnya berada di dalamnya AWS Cloud, Amazon S3 biasanya merupakan pilihan ideal daripada klaster Kubernetes atau Konsul. HashiCorp Kesederhanaan, ketahanan, dan AWS integrasi yang ketat dari penyimpanan status Amazon S3 memberikan fondasi optimal bagi sebagian besar pengguna yang AWS mengikuti praktik terbaik. Tim dapat memanfaatkan daya tahan, perlindungan cadangan, dan ketersediaan AWS layanan untuk menjaga status Terraform jarak jauh sangat tangguh.

Mengikuti rekomendasi backend di bagian ini akan menghasilkan basis kode Terraform yang lebih kolaboratif sambil membatasi dampak kesalahan atau modifikasi yang tidak sah. Dengan menerapkan backend jarak jauh yang dirancang dengan baik, tim dapat mengoptimalkan alur kerja Terraform.

Praktik terbaik:

- Gunakan Amazon S3 untuk penyimpanan jarak jauh
- Memfasilitasi kolaborasi tim
- Pisahkan backend untuk setiap lingkungan
- Secara aktif memonitor aktivitas status jarak jauh

Gunakan Amazon S3 untuk penyimpanan jarak jauh

Menyimpan status Terraform dari jarak jauh di Amazon S3 dan menerapkan penguncian status dan pemeriksaan konsistensi dengan menggunakan Amazon DynamoDB memberikan manfaat besar dibandingkan penyimpanan file lokal. Status jarak jauh memungkinkan kolaborasi tim, pelacakan perubahan, perlindungan cadangan, dan penguncian jarak jauh untuk meningkatkan keamanan.

Menggunakan Amazon S3 dengan kelas penyimpanan Standar S3 (default) alih-alih penyimpanan lokal sementara atau solusi yang dikelola sendiri memberikan daya tahan 99,999999999% dan perlindungan ketersediaan 99,99% untuk mencegah kehilangan data status yang tidak disengaja. AWS layanan terkelola seperti Amazon S3 dan DynamoDB menyediakan perjanjian tingkat layanan (SLAs) yang melebihi apa yang dapat dicapai sebagian besar organisasi ketika mereka mengelola penyimpanan sendiri. Andalkan perlindungan ini untuk menjaga backend jarak jauh dapat diakses.

Aktifkan penguncian status jarak jauh

Penguncian status membatasi akses untuk mencegah operasi penulisan bersamaan dan mengurangi kesalahan dari modifikasi simultan oleh beberapa pengguna. Terraform mendukung dua mekanisme penguncian untuk backend Amazon S3:

- Penguncian status asli Amazon S3 (disarankan): Tersedia sejak Terraform 1.10.0; menggunakan kemampuan penguncian asli di Amazon S3
- Penguncian status DynamoDB (tidak digunakan lagi): Pendekatan lama yang akan dihapus di versi Terraform mendatang

```
terraform {
  backend "s3" {
    bucket       = "myorg-terraform-states"
    key          = "myapp/production/tfstate"
    region       = "us-east-1"
    use_lockfile = true
  }
}
```

Untuk kompatibilitas mundur selama migrasi, Anda dapat mengonfigurasi penguncian Amazon S3 dan DynamoDB secara bersamaan. Namun, kami menyarankan Anda bermigrasi ke penguncian asli Amazon S3 karena penguncian berbasis DynamoDB tidak digunakan lagi.

Aktifkan pembuatan versi dan pencadangan otomatis

Untuk pengamanan tambahan, aktifkan pembuatan versi dan pencadangan otomatis dengan menggunakan backend Amazon S3. AWS Backup Pembuatan versi mempertahankan semua versi status sebelumnya setiap kali perubahan dilakukan. Ini juga memungkinkan Anda memulihkan snapshot status kerja sebelumnya jika diperlukan untuk mengembalikan perubahan yang tidak diinginkan atau pulih dari kecelakaan.

Kembalikan versi sebelumnya jika diperlukan

Bucket status Amazon S3 berversi memudahkan untuk mengembalikan perubahan dengan memulihkan snapshot status baik yang diketahui sebelumnya. Ini membantu melindungi dari perubahan yang tidak disengaja dan menyediakan kemampuan cadangan tambahan.

Gunakan HCP Terraform

HCP Terraform menyediakan alternatif backend yang dikelola sepenuhnya untuk mengonfigurasi penyimpanan status Anda sendiri. HCP Terraform secara otomatis menangani penyimpanan status dan enkripsi yang aman sambil membuka fitur tambahan.

Saat Anda menggunakan HCP Terraform, status disimpan dari jarak jauh secara default, yang memungkinkan berbagi status dan penguncian di seluruh organisasi Anda. Kontrol kebijakan terperinci membantu Anda membatasi akses dan perubahan status.

Kemampuan tambahan termasuk integrasi kontrol versi, pagar pembatas kebijakan, otomatisasi alur kerja, manajemen variabel, dan integrasi masuk tunggal dengan SAMP. Anda juga dapat menggunakan kebijakan Sentinel sebagai kode untuk menerapkan kontrol tata kelola.

Meskipun HCP Terraform memerlukan penggunaan platform perangkat lunak sebagai layanan (SaaS), bagi banyak tim manfaat seputar keamanan, kontrol akses, pemeriksaan kebijakan otomatis, dan fitur kolaborasi menjadikannya pilihan optimal daripada penyimpanan status pengelolaan mandiri dengan Amazon S3 atau DynamoDB.

Integrasi yang mudah dengan layanan seperti GitHub dan GitLab dengan konfigurasi kecil juga menarik bagi pengguna yang sepenuhnya merangkul alat cloud dan SaaS untuk alur kerja tim yang lebih baik.

Memfasilitasi kolaborasi tim

Gunakan backend jarak jauh untuk berbagi data status di semua anggota tim Terraform Anda. Ini memfasilitasi kolaborasi karena memberikan visibilitas seluruh tim ke dalam perubahan infrastruktur. Protokol backend bersama yang dikombinasikan dengan transparansi riwayat negara menyederhanakan manajemen perubahan internal. Semua perubahan infrastruktur melalui jalur pipa yang mapan, yang meningkatkan kelincahan bisnis di seluruh perusahaan.

Meningkatkan akuntabilitas dengan menggunakan AWS CloudTrail

Integrasikan AWS CloudTrail dengan bucket Amazon S3 untuk menangkap panggilan API yang dilakukan ke bucket status. Filter CloudTrail acara untuk dilacak`PutObject`, `DeleteObject`, dan panggilan relevan lainnya.

CloudTrail log menunjukkan AWS identitas prinsipal yang membuat setiap panggilan API untuk perubahan status. Identitas pengguna dapat dicocokkan dengan akun mesin atau anggota tim yang berinteraksi dengan penyimpanan backend.

Gabungkan CloudTrail log dengan versi status Amazon S3 untuk mengikat perubahan infrastruktur dengan prinsipal yang menerapkannya. Dengan menganalisis beberapa revisi, Anda dapat mengaitkan pembaruan apa pun ke akun mesin atau anggota tim yang bertanggung jawab.

Jika terjadi perubahan yang tidak diinginkan atau mengganggu, versi status menyediakan kemampuan rollback. CloudTrail melacak perubahan ke pengguna sehingga Anda dapat mendiskusikan perbaikan pencegahan.

Kami juga menyarankan agar Anda menerapkan izin IAM untuk membatasi akses bucket status. Secara keseluruhan, CloudTrail Pemantauan dan Pemantauan Versi S3 mendukung audit di seluruh perubahan infrastruktur. Tim mendapatkan peningkatan akuntabilitas, transparansi, dan kemampuan audit ke dalam sejarah negara bagian Terraform.

Pisahkan backend untuk setiap lingkungan

Gunakan backend Terraform yang berbeda untuk setiap lingkungan aplikasi. Backend terpisah mengisolasi status antara pengembangan, pengujian, dan produksi.

Mengurangi ruang lingkup dampak

Mengisolasi status membantu memastikan bahwa perubahan di lingkungan yang lebih rendah tidak memengaruhi infrastruktur produksi. Kecelakaan atau eksperimen dalam lingkungan pengembangan dan pengujian memiliki dampak terbatas.

Batasi akses produksi

Kunci izin untuk backend status produksi ke akses hanya-baca untuk sebagian besar pengguna. Batasi siapa yang dapat memodifikasi infrastruktur produksi ke peran CI/CD pipa dan memecahkan kaca.

Sederhanakan kontrol akses

Mengelola izin di tingkat backend menyederhanakan kontrol akses antar lingkungan. Menggunakan bucket S3 yang berbeda untuk setiap aplikasi dan lingkungan berarti bahwa izin baca atau tulis yang luas dapat diberikan di seluruh bucket backend.

Hindari ruang kerja bersama

Meskipun Anda dapat menggunakan ruang kerja Terraform untuk memisahkan status antar lingkungan, backend yang berbeda memberikan isolasi yang lebih kuat. Jika Anda memiliki ruang kerja bersama, kecelakaan masih dapat memengaruhi beberapa lingkungan.

Menjaga backend lingkungan sepenuhnya terisolasi meminimalkan dampak dari setiap kegagalan atau pelanggaran tunggal. Backend terpisah juga menyelaraskan kontrol akses ke tingkat sensitivitas lingkungan. Misalnya, Anda dapat memberikan perlindungan tulis untuk lingkungan produksi dan akses yang lebih luas untuk lingkungan pengembangan dan pengujian.

Secara aktif memonitor aktivitas status jarak jauh

Terus memantau aktivitas keadaan jarak jauh sangat penting untuk mendeteksi potensi masalah sejak dini. Cari pembukaan kunci, perubahan, atau upaya akses anomali.

Dapatkan peringatan tentang pembukaan kunci yang mencurigakan

Sebagian besar perubahan status harus dijalankan melalui CI/CD jaringan pipa. Hasilkan peringatan jika pembukaan status terjadi secara langsung melalui workstation pengembang, yang dapat menandakan perubahan yang tidak sah atau belum diuji.

Pantau upaya akses

Kegagalan otentikasi pada bucket status mungkin menunjukkan aktivitas pengintaian. Perhatikan jika beberapa akun mencoba mengakses status, atau alamat IP yang tidak biasa muncul, yang menandakan kredensi yang dikompromikan.

Praktik terbaik untuk struktur dasar kode dan organisasi

Struktur dan organisasi basis kode yang tepat sangat penting karena penggunaan Terraform tumbuh di seluruh tim dan perusahaan besar. Basis kode yang dirancang dengan baik memungkinkan kolaborasi dalam skala besar sekaligus meningkatkan pemeliharaan.

Bagian ini memberikan rekomendasi tentang modularitas Terraform, konvensi penamaan, dokumentasi, dan standar pengkodean yang mendukung kualitas dan konsistensi.

Panduan termasuk memecah konfigurasi menjadi modul yang dapat digunakan kembali berdasarkan lingkungan dan komponen, menetapkan konvensi penamaan dengan menggunakan awalan dan sufiks, mendokumentasikan modul dan menjelaskan input dan output dengan jelas, dan menerapkan aturan pemformatan yang konsisten dengan menggunakan pemeriksaan gaya otomatis.

Praktik terbaik tambahan mencakup pengorganisasian modul dan sumber daya secara logis dalam hierarki terstruktur, membuat katalog modul publik dan pribadi dalam dokumentasi, dan mengabstraksi detail implementasi yang tidak perlu dalam modul untuk menyederhanakan penggunaan.

Dengan menerapkan pedoman struktur dasar kode seputar modularitas, dokumentasi, standar, dan organisasi logis, Anda dapat mendukung kolaborasi luas di seluruh tim sambil menjaga Terraform tetap dapat dipertahankan saat penggunaan menyebar di seluruh organisasi. Dengan menegakkan konvensi dan standar, Anda dapat menghindari kompleksitas basis kode yang terfragmentasi.

Praktik terbaik:

- Menerapkan struktur repositori standar
- Struktur untuk modularitas
- Ikuti konvensi penamaan
- Gunakan sumber daya lampiran
- Gunakan tag default
- Memenuhi persyaratan registri Terraform
- Gunakan sumber modul yang direkomendasikan
- Ikuti standar pengkodean

Menerapkan struktur repositori standar

Kami menyarankan Anda menerapkan tata letak repositori berikut. Standarisasi praktik konsistensi ini di seluruh modul meningkatkan kemampuan ditemukan, transparansi, organisasi, dan keandalan sambil memungkinkan penggunaan kembali di banyak konfigurasi Terraform.

- Modul atau direktori root: Ini harus menjadi titik masuk utama untuk root Terraform dan modul yang dapat digunakan kembali dan diharapkan unik. Jika Anda memiliki arsitektur yang lebih kompleks, Anda dapat menggunakan modul bersarang untuk membuat abstraksi ringan. Ini membantu Anda menggambarkan infrastruktur dalam hal arsitekturnya, bukan secara langsung, dalam hal objek fisik.
- README: Modul root dan modul bersarang apa pun harus memiliki file README. File ini harus diberi nama`README.md`. Ini harus berisi deskripsi modul dan untuk apa modul itu harus digunakan. Jika Anda ingin menyertakan contoh menggunakan modul ini dengan sumber daya lain, letakkan di `examples` direktori. Pertimbangkan untuk menyertakan diagram yang menggambarkan sumber daya infrastruktur yang mungkin dibuat modul dan hubungannya. Gunakan terraform-docs untuk menghasilkan input atau output modul secara otomatis.
- main.tf: Ini adalah titik masuk utama. Untuk modul sederhana, semua sumber daya dapat dibuat dalam file ini. Untuk modul yang kompleks, pembuatan sumber daya mungkin tersebar di beberapa file, tetapi panggilan modul bersarang apa pun harus ada di `main.tf` file.
- variables.tf dan outputs.tf: File-file ini berisi deklarasi untuk variabel dan output. Semua variabel dan output harus memiliki deskripsi satu kalimat atau dua kalimat yang menjelaskan tujuannya. Deskripsi ini digunakan untuk dokumentasi. Untuk informasi selengkapnya, lihat HashiCorp dokumentasi untuk konfigurasi variabel dan konfigurasi keluaran.
  - Semua variabel harus memiliki tipe yang ditentukan.
  - Deklarasi variabel juga dapat menyertakan argumen default. Jika deklarasi menyertakan argumen default, variabel dianggap opsional, dan nilai default digunakan jika Anda tidak menetapkan nilai saat Anda memanggil modul atau menjalankan Terraform. Argumen default membutuhkan nilai literal dan tidak dapat mereferensikan objek lain dalam konfigurasi. Untuk membuat variabel diperlukan, hilangkan default dalam deklarasi variabel dan pertimbangkan apakah pengaturan masuk `nullable = false` akal.
  - Untuk variabel yang memiliki nilai yang tidak bergantung pada lingkungan (sepertidisk_size), berikan nilai default.
  - Untuk variabel yang memiliki nilai khusus lingkungan (sepertiproject_id), jangan berikan nilai default. Dalam hal ini, modul panggilan harus memberikan nilai yang berarti.

  - Gunakan default kosong untuk variabel seperti string kosong atau daftar hanya ketika membiarkan variabel kosong adalah preferensi valid yang mendasarinya APIs tidak menolak.
  - Bersikaplah bijaksana dalam penggunaan variabel. Parameterisasi nilai hanya jika mereka harus bervariasi untuk setiap instance atau lingkungan. Ketika Anda memutuskan apakah akan mengekspos variabel, pastikan bahwa Anda memiliki kasus penggunaan konkret untuk mengubah variabel itu. Jika hanya ada kemungkinan kecil bahwa variabel mungkin diperlukan, jangan mengeksposnya.
    - Menambahkan variabel dengan nilai default kompatibel ke belakang.
    - Menghapus variabel tidak kompatibel ke belakang.
    - Dalam kasus di mana literal digunakan kembali di banyak tempat, Anda harus menggunakan nilai lokal tanpa mengeksposnya sebagai variabel.
  - Jangan melewatkan output secara langsung melalui variabel input, karena hal itu mencegahnya ditambahkan dengan benar ke grafik ketergantungan. Untuk memastikan bahwa dependensi implisit dibuat, pastikan bahwa output referensi atribut dari sumber daya. Alih-alih mereferensikan variabel input untuk sebuah instance secara langsung, berikan atribut.
- locals.tf: File ini berisi nilai-nilai lokal yang menetapkan nama ke ekspresi, sehingga nama dapat digunakan beberapa kali dalam modul alih-alih mengulangi ekspresi. Nilai lokal seperti variabel lokal sementara fungsi. Ekspresi dalam nilai lokal tidak terbatas pada konstanta literal; mereka juga dapat mereferensikan nilai lain dalam modul, termasuk variabel, atribut sumber daya, atau nilai lokal lainnya, untuk menggabungkannya.
- providers.tf: File ini berisi blok terraform dan blok penyedia. `provider`blok harus dideklarasikan hanya dalam modul root oleh konsumen modul.

  Jika Anda menggunakan HCP Terraform, tambahkan juga blok cloud kosong. `cloud`Blok harus dikonfigurasi seluruhnya melalui variabel lingkungan dan kredenal variabel lingkungan sebagai bagian dari pipeline. CI/CD
- versions.tf: File ini berisi blok required_providers. Semua modul Terraform harus mendeklarasikan penyedia mana yang diperlukan sehingga Terraform dapat menginstal dan menggunakan penyedia ini.
- data.tf: Untuk konfigurasi sederhana, letakkan sumber data di sebelah sumber daya yang mereferensikannya. Misalnya, jika Anda mengambil gambar yang akan digunakan dalam meluncurkan instance, letakkan di samping instance alih-alih mengumpulkan sumber daya data dalam file mereka sendiri. Jika jumlah sumber data menjadi terlalu besar, pertimbangkan untuk memindahkannya ke `data.tf` file khusus.

- .tfvars file: Untuk modul root, Anda dapat menyediakan variabel non-sensitif dengan menggunakan file. `.tfvars` Untuk konsistensi, beri nama file variabel`terraform.tfvars`. Tempatkan nilai umum di root repositori, dan nilai khusus lingkungan di dalam folder. `envs/`
- Modul bersarang: Modul bersarang harus ada di bawah subdirektori. `modules/` Setiap modul bersarang yang memiliki a `README.md` dianggap dapat digunakan oleh pengguna eksternal. Jika `README.md` tidak ada, modul dipertimbangkan untuk penggunaan internal saja. Modul bersarang harus digunakan untuk membagi perilaku kompleks menjadi beberapa modul kecil yang dapat dipilih dan dipilih oleh pengguna dengan hati-hati.

  Jika modul root menyertakan panggilan ke modul bersarang, panggilan ini harus menggunakan jalur relatif seperti `./modules/sample-module` sehingga Terraform akan menganggapnya sebagai bagian dari repositori atau paket yang sama alih-alih mengunduhnya lagi secara terpisah.

  Jika repositori atau paket berisi beberapa modul bersarang, mereka idealnya dapat dikomposisi oleh pemanggil alih-alih langsung memanggil satu sama lain dan membuat pohon modul yang sangat bersarang.
- Contoh: Contoh penggunaan modul yang dapat digunakan kembali harus ada di bawah `examples/` subdirektori di root repositori. Untuk setiap contoh, Anda dapat menambahkan README untuk menjelaskan tujuan dan penggunaan contoh. Contoh untuk submodul juga harus ditempatkan di `examples/` direktori root.

  Karena contoh sering disalin ke repositori lain untuk penyesuaian, blok modul harus mengatur sumbernya ke alamat yang akan digunakan pemanggil eksternal, bukan ke jalur relatif.
- File bernama layanan: Pengguna sering ingin memisahkan sumber daya Terraform berdasarkan layanan dalam beberapa file. Praktik ini harus berkecil hati sebanyak mungkin, dan sumber daya harus didefinisikan sebagai gantinya. `main.tf` Namun, jika kumpulan sumber daya (misalnya, peran dan kebijakan IAM) melebihi 150 baris, masuk akal untuk memecahnya menjadi file-nya sendiri, seperti`iam.tf`. Jika tidak, semua kode sumber daya harus didefinisikan dalam file`main.tf`.
- Skrip khusus: Gunakan skrip hanya jika diperlukan. Terraform tidak memperhitungkan, atau mengelola, status sumber daya yang dibuat melalui skrip. Gunakan skrip khusus hanya jika sumber daya Terraform tidak mendukung perilaku yang diinginkan. Tempatkan skrip khusus yang dipanggil oleh Terraform di direktori. `scripts/`
- Skrip pembantu: Atur skrip pembantu yang tidak dipanggil oleh Terraform di direktori. `helpers/` Dokumen skrip pembantu dalam `README.md` file dengan penjelasan dan contoh pemanggilan. Jika skrip pembantu menerima argumen, berikan pemeriksaan argumen dan `--help` output.

- File statis: File statis yang direferensikan Terraform tetapi tidak dijalankan (seperti skrip startup yang dimuat ke instance EC2) harus diatur ke dalam direktori. `files/` Tempatkan dokumen panjang dalam file eksternal, terpisah dari HCL mereka. Referensikan mereka dengan fungsi file ().
- Template: Untuk file yang dibaca fungsi Templatefile Terraform, gunakan ekstensi file. `.tftpl` Template harus ditempatkan di `templates/` direktori.

Struktur modul root

Terraform selalu berjalan dalam konteks modul root tunggal. Konfigurasi Terraform lengkap terdiri dari modul root dan pohon modul anak (yang mencakup modul yang dipanggil oleh modul root, modul apa pun yang disebut oleh modul tersebut, dan sebagainya).

Contoh dasar tata letak modul root Terraform:

```
.
### data.tf
### envs
#   ### dev
#   #   ### terraform.tfvars
#   ### prod
#   #   ### terraform.tfvars
#   ### test
#       ### terraform.tfvars
### locals.tf
### main.tf
### outputs.tf
### providers.tf
### README.md
### terraform.tfvars
### variables.tf
### versions.tf
```

Struktur modul yang dapat digunakan kembali

Modul yang dapat digunakan kembali mengikuti konsep yang sama dengan modul root. Untuk mendefinisikan modul, buat direktori baru untuk itu dan tempatkan `.tf` file di dalamnya, sama seperti Anda akan mendefinisikan modul root. Terraform dapat memuat modul baik dari jalur relatif lokal atau dari repositori jarak jauh. Jika Anda mengharapkan modul digunakan kembali oleh banyak konfigurasi, letakkan di repositori kontrol versinya sendiri. Sangat penting untuk menjaga pohon

modul relatif datar untuk membuatnya lebih mudah untuk menggunakan kembali modul dalam kombinasi yang berbeda.

Contoh dasar tata letak modul Terraform yang dapat digunakan kembali:

```
.
### data.tf
### examples
#   ### multi-az-new-vpc
#   #   ### data.tf
#   #   ### locals.tf
#   #   ### main.tf
#   #   ### outputs.tf
#   #   ### providers.tf
#   #   ### README.md
#   #   ### terraform.tfvars
#   #   ### variables.tf
#   #   ### versions.tf
#   #   ### vpc.tf
#   ### single-az-existing-vpc
#   #   ### data.tf
#   #   ### locals.tf
#   #   ### main.tf
#   #   ### outputs.tf
#   #   ### providers.tf
#   #   ### README.md
#   #   ### terraform.tfvars
#   #   ### variables.tf
#   #   ### versions.tf
### iam.tf
### locals.tf
### main.tf
### outputs.tf
### README.md
### variables.tf
### versions.tf
```

Struktur untuk modularitas

Pada prinsipnya, Anda dapat menggabungkan sumber daya apa pun dan konstruksi lain ke dalam modul, tetapi terlalu sering menggunakan modul bersarang dan dapat digunakan kembali dapat

membuat konfigurasi Terraform Anda secara keseluruhan lebih sulit untuk dipahami dan dipelihara, jadi gunakan modul ini dalam jumlah sedang.

Jika masuk akal, pisahkan konfigurasi Anda menjadi modul yang dapat digunakan kembali yang meningkatkan tingkat abstraksi dengan menjelaskan konsep baru dalam arsitektur Anda yang dibangun dari tipe sumber daya.

Saat Anda memodulasi infrastruktur Anda menjadi definisi yang dapat digunakan kembali, arahkan kumpulan sumber daya yang logis alih-alih komponen individual atau koleksi yang terlalu kompleks.

Jangan membungkus sumber daya tunggal

Anda tidak boleh membuat modul yang merupakan pembungkus tipis di sekitar jenis sumber daya tunggal lainnya. Jika Anda kesulitan menemukan nama untuk modul Anda yang berbeda dari nama tipe sumber daya utama di dalamnya, modul Anda mungkin tidak membuat abstraksi baru–itu menambahkan kompleksitas yang tidak perlu. Sebagai gantinya, gunakan jenis sumber daya secara langsung di modul panggilan.

Merangkum hubungan logis

Kumpulan kumpulan sumber daya terkait seperti fondasi jaringan, tingkatan data, kontrol keamanan, dan aplikasi. Modul yang dapat digunakan kembali harus merangkum bagian infrastruktur yang bekerja sama untuk memungkinkan suatu kemampuan.

Jaga agar warisan tetap rata

Saat Anda membuat modul sarang di subdirektori, hindari kedalaman lebih dari satu atau dua level. Struktur pewarisan yang sangat bersarang memperumit konfigurasi dan pemecahan masalah. Modul harus dibangun di atas modul lain–bukan membangun terowongan melaluinya.

Dengan memfokuskan modul pada pengelompokan sumber daya logis yang mewakili pola arsitektur, tim dapat dengan cepat mengkonfigurasi fondasi infrastruktur yang andal. Seimbangkan abstraksi tanpa rekayasa berlebihan atau penyederhanaan berlebihan.

Sumber referensi dalam output

Untuk setiap sumber daya yang didefinisikan dalam modul yang dapat digunakan kembali, sertakan setidaknya satu output yang mereferensikan sumber daya. Variabel dan output memungkinkan Anda menyimpulkan dependensi antara modul dan sumber daya. Tanpa output apa pun, pengguna tidak dapat memesan modul Anda dengan benar sehubungan dengan konfigurasi Terraform mereka.

Modul terstruktur dengan baik yang memberikan konsistensi lingkungan, pengelompokan yang digerakkan oleh tujuan, dan referensi sumber daya yang diekspor memungkinkan kolaborasi Terraform di seluruh organisasi dalam skala besar. Tim dapat merakit infrastruktur dari blok bangunan yang dapat digunakan kembali.

Jangan mengkonfigurasi penyedia

Meskipun modul bersama mewarisi penyedia dari modul panggilan, modul tidak boleh mengonfigurasi pengaturan penyedia sendiri. Hindari menentukan blok konfigurasi penyedia dalam modul. Konfigurasi ini seharusnya hanya dideklarasikan sekali secara global.

Deklarasikan penyedia yang diperlukan

Meskipun konfigurasi penyedia dibagi antar modul, modul bersama juga harus mendeklarasikan persyaratan penyedia mereka sendiri. Praktik ini memungkinkan Terraform untuk memastikan bahwa ada satu versi penyedia yang kompatibel dengan semua modul dalam konfigurasi, dan untuk menentukan alamat sumber yang berfungsi sebagai pengidentifikasi global (modul-agnostik) untuk penyedia. Namun, persyaratan penyedia khusus modul tidak menentukan pengaturan konfigurasi apa pun yang menentukan titik akhir jarak jauh apa yang akan diakses penyedia, seperti. Wilayah AWS

Dengan mendeklarasikan persyaratan versi dan menghindari konfigurasi penyedia hardcode, modul menyediakan portabilitas dan penggunaan kembali di seluruh konfigurasi Terraform menggunakan penyedia bersama.

Untuk modul bersama, tentukan versi penyedia minimum yang diperlukan di blok required_providers di. `versions.tf`

Untuk menyatakan bahwa modul memerlukan versi AWS penyedia tertentu, gunakan `required_providers` blok di dalam blok: `terraform`

```
terraform {
  required_version = ">= 1.0.0"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 4.0.0"
    }
  }
```

```
}
```

Jika modul bersama hanya mendukung versi AWS penyedia tertentu, gunakan operator kendala pesimis (~>), yang hanya mengizinkan komponen versi paling kanan untuk bertambah:

```
terraform {
  required_version = ">= 1.0.0"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}
```

Dalam contoh ini, `~> 4.0` memungkinkan instalasi `4.57.1` dan `4.67.0` tetapi tidak`5.0.0`. Untuk informasi selengkapnya, lihat Sintaks Batasan Versi dalam dokumentasi. HashiCorp

Ikuti konvensi penamaan

Nama yang jelas dan deskriptif menyederhanakan pemahaman Anda tentang hubungan antara sumber daya dalam modul dan tujuan nilai konfigurasi. Konsistensi dengan pedoman gaya meningkatkan keterbacaan bagi pengguna modul dan pengelola.

Ikuti panduan untuk penamaan sumber daya

- Gunakan snake_case (di mana istilah huruf kecil dipisahkan oleh garis bawah) untuk semua nama sumber daya agar sesuai dengan standar gaya Terraform. Praktik ini memastikan konsistensi dengan konvensi penamaan untuk tipe sumber daya, tipe sumber data, dan nilai standar lainnya. Konvensi ini tidak berlaku untuk argumen nama.
- Untuk menyederhanakan referensi ke sumber daya yang merupakan satu-satunya dari jenisnya (misalnya, penyeimbang beban tunggal untuk seluruh modul), beri nama sumber daya `main` atau `this` untuk kejelasan.
- Gunakan nama bermakna yang menggambarkan tujuan dan konteks sumber daya, dan yang membantu membedakan antara sumber daya yang serupa (misalnya, `primary` untuk database utama dan `read_replica` untuk replika baca database).
- Gunakan nama tunggal, bukan jamak.

- Jangan ulangi jenis sumber daya dalam nama sumber daya.

Ikuti panduan untuk penamaan variabel

- Tambahkan unit ke nama input, variabel lokal, dan output yang mewakili nilai numerik seperti ukuran disk atau ukuran RAM (misalnya, `ram_size_gb` untuk ukuran RAM dalam gigabyte). Praktek ini membuat unit input yang diharapkan jelas untuk pengelola konfigurasi.
- Gunakan unit biner seperti MiB dan GiB untuk ukuran penyimpanan, dan unit desimal seperti MB atau GB untuk metrik lainnya.
- Berikan variabel Boolean nama-nama positif seperti`enable_external_access`.

Gunakan sumber daya lampiran

Beberapa sumber daya memiliki sumber daya semu yang disematkan sebagai atribut di dalamnya. Jika memungkinkan, Anda harus menghindari penggunaan atribut sumber daya yang disematkan ini dan menggunakan sumber daya unik untuk melampirkan sumber daya semu tersebut. Hubungan sumber daya ini dapat menyebabkan cause-and-effect masalah yang unik untuk setiap sumber daya.

Menggunakan atribut tertanam (hindari pola ini):

```
resource "aws_security_group" "allow_tls" {
  ...
  ingress {
    description      = "TLS from VPC"
    from_port        = 443
    to_port          = 443
    protocol         = "tcp"
    cidr_blocks      = [aws_vpc.main.cidr_block]
    ipv6_cidr_blocks = [aws_vpc.main.ipv6_cidr_block]
  }

  egress {
    from_port        = 0
    to_port          = 0
    protocol         = "-1"
    cidr_blocks      = ["0.0.0.0/0"]
    ipv6_cidr_blocks = ["::/0"]
  }
}
```

Menggunakan sumber daya lampiran (lebih disukai):

```
resource "aws_security_group" "allow_tls" {
  ...
}

resource "aws_security_group_rule" "example" {
  type              = "ingress"
  description      = "TLS from VPC"
  from_port        = 443
  to_port          = 443
  protocol         = "tcp"
  cidr_blocks      = [aws_vpc.main.cidr_block]
  ipv6_cidr_blocks = [aws_vpc.main.ipv6_cidr_block]
  security_group_id = aws_security_group.allow_tls.id
}
```

Gunakan tag default

Tetapkan tag ke semua sumber daya yang dapat menerima tag. AWS Penyedia Terraform memiliki sumber data aws_default_tags yang harus Anda gunakan di dalam modul root.

Pertimbangkan untuk menambahkan tag yang diperlukan ke semua sumber daya yang dibuat oleh modul Terraform. Berikut daftar kemungkinan tag untuk dilampirkan:

- Nama: Nama sumber daya yang dapat dibaca manusia
- AppId: ID untuk aplikasi yang menggunakan sumber daya
- AppRole: Fungsi teknis sumber daya; misalnya, “server web” atau “database”
- AppPurpose: Tujuan bisnis sumber daya; misalnya, “frontend ui” atau “prosesor pembayaran”
- Lingkungan: Lingkungan perangkat lunak, seperti dev, test, atau prod
- Proyek: Proyek yang menggunakan sumber daya
- CostCenter: Siapa yang harus ditagih untuk penggunaan sumber daya

Memenuhi persyaratan registri Terraform

Repositori modul harus memenuhi semua persyaratan berikut sehingga dapat dipublikasikan ke registri Terraform.

Anda harus selalu mengikuti persyaratan ini bahkan jika Anda tidak berencana untuk mempublikasikan modul ke registri dalam jangka pendek. Dengan demikian, Anda dapat mempublikasikan modul ke registri nanti tanpa harus mengubah konfigurasi dan struktur repositori.

- Nama repositori: Untuk repositori modul, gunakan nama tiga bagian`terraform-aws-<NAME>`, yang `<NAME>` mencerminkan jenis infrastruktur yang dikelola modul. `<NAME>`Segmen dapat berisi tanda hubung tambahan (misalnya,`terraform-aws-iam-terraform-roles`).
- Struktur modul standar: Modul harus mematuhi struktur repositori standar. Ini memungkinkan registri untuk memeriksa modul Anda dan menghasilkan dokumentasi, melacak penggunaan sumber daya, dan banyak lagi.
  - Setelah Anda membuat repositori Git, salin file modul ke root repositori. Kami menyarankan Anda menempatkan setiap modul yang dimaksudkan untuk dapat digunakan kembali di root repositorinya sendiri, tetapi Anda juga dapat mereferensikan modul dari subdirektori.
  - Jika Anda menggunakan HCP Terraform, publikasikan modul yang dimaksudkan untuk dibagikan ke registri organisasi Anda. Registri menangani unduhan dan mengontrol akses dengan token API HCP Terraform, sehingga konsumen tidak memerlukan akses ke repositori sumber modul bahkan ketika mereka menjalankan Terraform dari baris perintah.
- Lokasi dan izin: Repositori harus berada di salah satu penyedia sistem kontrol versi (VCS) yang dikonfigurasi, dan akun pengguna HCP Terraform VCS harus memiliki akses administrator ke repositori. Registri memerlukan akses administrator untuk membuat webhook untuk mengimpor versi modul baru.
- tag x.y.z untuk rilis: Setidaknya satu tag rilis harus ada bagi Anda untuk menerbitkan modul. Registri menggunakan tag rilis untuk mengidentifikasi versi modul. Nama tag rilis harus menggunakan versi semantik, yang dapat Anda awalan secara opsional dengan `v` (misalnya, dan). `v1.1.0` `1.1.0` Registri mengabaikan tag yang tidak terlihat seperti nomor versi. Untuk informasi selengkapnya tentang modul penerbitan, lihat dokumentasi Terraform.

Untuk informasi selengkapnya, lihat Mempersiapkan Repositori Modul dalam dokumentasi Terraform.

Gunakan sumber modul yang direkomendasikan

Terraform menggunakan `source` argumen dalam blok modul untuk menemukan dan mengunduh kode sumber untuk modul anak.

Kami menyarankan Anda menggunakan jalur lokal untuk modul terkait erat yang memiliki tujuan utama untuk memfaktorkan elemen kode berulang, dan menggunakan registri modul Terraform asli atau penyedia VCS untuk modul yang dimaksudkan untuk dibagikan oleh beberapa konfigurasi.

Contoh berikut menggambarkan jenis sumber yang paling umum dan direkomendasikan untuk berbagi modul. Modul registri mendukung pembuatan versi. Anda harus selalu memberikan versi tertentu, seperti yang ditunjukkan pada contoh berikut.

Registri

Registri Terraform:

```
module "lambda" {
  source = "github.com/terraform-aws-modules/terraform-aws-lambda.git?
ref=e78cdf1f82944897ca6e30d6489f43cf24539374" #--> v4.18.0

  ...

}
```

Dengan menyematkan hash komit, Anda dapat menghindari penyimpangan dari pendaftar publik yang rentan terhadap serangan rantai pasokan.

Terraform HCP:

```
module "eks_karpenter" {
  source = "app.terraform.io/my-org/eks/aws"
  version = "1.1.0"

  ...

  enable_karpenter = true
}
```

Perusahaan Terraform:

```
module "eks_karpenter" {
  source = "terraform.mydomain.com/my-org/eks/aws"
  version = "1.1.0"

  ...
```

```
  enable_karpenter = true
}
```

Penyedia VCS

Penyedia VCS mendukung `ref` argumen untuk memilih revisi tertentu, seperti yang ditunjukkan pada contoh berikut.

GitHub (HTTPS):

```
module "eks_karpenter" {
  source = "github.com/my-org/terraform-aws-eks.git?ref=v1.1.0"

  ...

  enable_karpenter = true
}
```

Repositori Git Generik (HTTPS):

```
module "eks_karpenter" {
  source = "git::https://example.com/terraform-aws-eks.git?ref=v1.1.0"

  ...

  enable_karpenter = true
}
```

Repositori Git Generik (SSH):

> **Warning**
>
> Anda perlu mengonfigurasi kredensil untuk mengakses repositori pribadi.

```
module "eks_karpenter" {
  source = "git::ssh://username@example.com/terraform-aws-eks.git?ref=v1.1.0"

  ...
```

```
  enable_karpenter = true
}
```

Ikuti standar pengkodean

Terapkan aturan dan gaya pemformatan Terraform yang konsisten di semua file konfigurasi. Menegakkan standar dengan menggunakan pemeriksaan gaya otomatis di CI/CD jaringan pipa. Saat Anda menyematkan praktik terbaik pengkodean ke dalam alur kerja tim, konfigurasi tetap dapat dibaca, dipelihara, dan kolaboratif karena penggunaan menyebar luas di seluruh organisasi.

Ikuti pedoman gaya

- Format semua file Terraform (`.tf`file) dengan perintah terraform fmt agar sesuai dengan standar gaya. HashiCorp
- Gunakan perintah terraform validate untuk memverifikasi sintaks dan struktur konfigurasi Anda.
- Analisis kualitas kode secara statis dengan menggunakan TFLint. Linter ini memeriksa praktik terbaik Terraform lebih dari sekadar memformat dan gagal membangun saat menemukan kesalahan.

Konfigurasikan kait pra-komit

Konfigurasikan kait pra-komit sisi klien yang menjalankan`terraform fmt`,, `tflint`, `checkov`, dan pemindaian kode lainnya serta pemeriksaan gaya sebelum Anda mengizinkan komit. Praktik ini membantu Anda memvalidasi kesesuaian standar sebelumnya dalam alur kerja pengembang.

Gunakan kerangka kerja pra-komit seperti pra-komit untuk menambahkan linting Terraform, pemformatan, dan pemindaian kode sebagai kait pada mesin lokal Anda. Hooks berjalan pada setiap komit Git dan gagal komit jika pemeriksaan tidak lolos.

Memindahkan gaya dan pemeriksaan kualitas ke kait pra-komit lokal memberikan umpan balik cepat kepada pengembang sebelum perubahan diperkenalkan. Standar menjadi bagian dari alur kerja pengkodean.

Praktik terbaik untuk manajemen versi AWS Penyedia

Mengelola versi AWS Penyedia dan modul Terraform terkait dengan hati-hati sangat penting untuk stabilitas. Bagian ini menguraikan praktik terbaik seputar kendala dan peningkatan versi.

Praktik terbaik:

- Tambahkan pemeriksaan versi otomatis
- Pantau rilis baru
- Berkontribusi pada penyedia

Tambahkan pemeriksaan versi otomatis

Tambahkan pemeriksaan versi untuk penyedia Terraform di pipeline CI/CD Anda untuk memvalidasi penyematan versi, dan gagal membangun jika versinya tidak ditentukan.

- Tambahkan pemeriksaan TFlint di pipeline CI/CD untuk memindai versi penyedia yang tidak memiliki batasan versi mayor/minor yang disematkan. Gunakan plugin TFLint ruleset untuk AWS Penyedia Terraform, yang menyediakan aturan untuk mendeteksi kemungkinan kesalahan dan memeriksa praktik terbaik tentang sumber daya. AWS
- Fail CI berjalan yang mendeteksi versi penyedia yang tidak disematkan untuk mencegah peningkatan implisit mencapai produksi.

Pantau rilis baru

- Pantau catatan rilis penyedia dan umpan changelog. Dapatkan pemberitahuan tentang rilis mayor/minor baru.
- Menilai catatan rilis untuk perubahan yang berpotensi melanggar dan mengevaluasi dampaknya terhadap infrastruktur yang ada.
- Tingkatkan versi minor di lingkungan non-produksi terlebih dahulu untuk memvalidasinya sebelum memperbarui lingkungan produksi.

Dengan mengotomatiskan pemeriksaan versi di pipeline dan memantau rilis baru, Anda dapat menangkap peningkatan yang tidak didukung lebih awal dan memberi waktu kepada tim Anda untuk mengevaluasi dampak rilis mayor/minor baru sebelum memperbarui lingkungan produksi.

Berkontribusi pada penyedia

Berkontribusi secara aktif kepada HashiCorp AWS Penyedia dengan melaporkan cacat atau meminta fitur dalam GitHub masalah:

- Buka masalah yang terdokumentasi dengan baik di repositori AWS Penyedia untuk merinci bug yang Anda temui atau fungsionalitas yang hilang. Berikan langkah-langkah yang dapat direproduksi.
- Meminta dan memberikan suara pada perangkat tambahan untuk memperluas kemampuan AWS Penyedia untuk mengelola layanan baru.
- Referensi permintaan tarik yang dikeluarkan saat Anda menyumbangkan perbaikan yang diusulkan untuk cacat atau penyempurnaan penyedia. Tautan ke masalah terkait.
- Ikuti panduan kontribusi dalam repositori untuk konvensi pengkodean, standar pengujian, dan dokumentasi.

Dengan memberikan kembali kepada penyedia yang Anda gunakan, Anda dapat memberikan masukan langsung ke peta jalan mereka dan membantu meningkatkan kualitas dan kemampuan mereka untuk semua pengguna.

Praktik terbaik untuk modul komunitas

Menggunakan modul secara efektif adalah kunci untuk mengelola konfigurasi Terraform yang kompleks dan mempromosikan penggunaan kembali. Bagian ini memberikan praktik terbaik seputar modul komunitas, dependensi, sumber, abstraksi, dan kontribusi.

Praktik terbaik:

- Temukan modul komunitas
- Memahami dependensi
- Gunakan sumber tepercaya
- Berkontribusi pada modul komunitas

Temukan modul komunitas

Cari Terraform Registry, GitHub, dan sumber lain untuk AWS modul yang ada yang mungkin menyelesaikan kasus penggunaan Anda sebelum Anda membuat modul baru. Cari opsi populer yang memiliki pembaruan terbaru dan dipertahankan secara aktif.

Gunakan variabel untuk kustomisasi

Saat Anda menggunakan modul komunitas, berikan input melalui variabel alih-alih melakukan forking atau langsung memodifikasi kode sumber. Ganti default jika diperlukan alih-alih mengubah internal modul.

Forking harus dibatasi untuk berkontribusi perbaikan atau fitur ke modul asli untuk memberi manfaat bagi komunitas yang lebih luas.

Memahami dependensi

Sebelum Anda menggunakan modul, tinjau kode sumber dan dokumentasinya untuk mengidentifikasi dependensi:

- Penyedia yang diperlukan: Perhatikan versi AWS, Kubernetes, atau penyedia lain yang dibutuhkan modul.
- Modul bersarang: Periksa modul lain yang digunakan secara internal yang memperkenalkan dependensi cascading.

- Sumber data eksternal: Perhatikan API, plugin kustom, atau dependensi infrastruktur yang diandalkan modul.

Dengan memetakan pohon penuh dependensi langsung dan tidak langsung, Anda dapat menghindari kejutan saat menggunakan modul.

Gunakan sumber tepercaya

Sumber modul Terraform dari penerbit yang tidak diverifikasi atau tidak dikenal menimbulkan risiko yang signifikan. Gunakan modul hanya dari sumber tepercaya.

- Mendukung modul bersertifikat dari Terraform Registry yang diterbitkan oleh pembuat terverifikasi seperti AWS atau HashiCorp mitra.
- Untuk modul khusus, tinjau riwayat penerbit, tingkat dukungan, dan reputasi penggunaan, meskipun modul tersebut berasal dari organisasi Anda sendiri.

Dengan tidak mengizinkan modul dari sumber yang tidak dikenal atau tidak diperiksa, Anda dapat mengurangi risiko menyuntikkan kerentanan atau masalah pemeliharaan ke dalam kode Anda.

Berlangganan notifikasi

Berlangganan notifikasi untuk rilis modul baru dari penerbit tepercaya:

- Tonton repositori GitHub modul untuk mendapatkan peringatan tentang versi baru modul.
- Pantau blog penerbit dan changelog untuk pembaruan.
- Dapatkan notifikasi proaktif untuk versi baru dari sumber terverifikasi dan berperingkat tinggi alih-alih secara implisit menarik pembaruan.

Mengkonsumsi modul hanya dari sumber tepercaya dan perubahan pemantauan memberikan stabilitas dan keamanan. Modul yang diperiksa meningkatkan produktivitas sekaligus meminimalkan risiko rantai pasokan.

Berkontribusi pada modul komunitas

Kirimkan perbaikan dan penyempurnaan untuk modul komunitas yang di-host di: GitHub

- Buka permintaan tarik pada modul untuk mengatasi cacat atau batasan yang Anda temui dalam penggunaan Anda.
- Minta konfigurasi praktik terbaik baru untuk ditambahkan ke modul OSS yang ada dengan membuat masalah.

Berkontribusi pada modul komunitas meningkatkan pola yang dapat digunakan kembali dan dikodifikasi untuk semua praktisi Terraform.

Pertanyaan yang Sering Diajukan

T. Mengapa fokus pada AWS Provider?

A. AWS Penyedia adalah salah satu penyedia yang paling banyak digunakan dan kompleks untuk penyediaan infrastruktur di Terraform. Mengikuti praktik terbaik ini membantu pengguna mengoptimalkan penggunaan penyedia mereka untuk AWS lingkungan.

T. Saya baru mengenal Terraform. Bisakah saya menggunakan panduan ini?

A. Panduan ini untuk orang-orang yang baru mengenal Terraform serta praktisi yang lebih maju yang ingin meningkatkan keterampilan mereka. Praktik meningkatkan alur kerja bagi pengguna pada setiap tahap pembelajaran.

T. Apa sajakah praktik terbaik utama yang dibahas?

A. Praktik terbaik utama termasuk menggunakan peran IAM melalui kunci akses, menyematkan versi, menggabungkan pengujian otomatis, penguncian status jarak jauh, rotasi kredensi, berkontribusi kembali ke penyedia, dan mengatur basis kode secara logis.

T. Di mana saya bisa mempelajari lebih lanjut tentang Terraform?

A. Bagian Sumber Daya mencakup tautan ke dokumentasi HashiCorp Terraform resmi dan forum komunitas. Gunakan tautan untuk mempelajari lebih lanjut tentang alur kerja Terraform tingkat lanjut.

Langkah selanjutnya

Berikut adalah beberapa langkah potensial selanjutnya setelah membaca panduan ini:

- Jika Anda memiliki basis kode Terraform yang ada, tinjau konfigurasi Anda dan identifikasi area yang dapat ditingkatkan berdasarkan rekomendasi yang disediakan dalam panduan ini. Misalnya, tinjau praktik terbaik untuk menerapkan backend jarak jauh, memisahkan kode menjadi modul, menggunakan penyematan versi, dan sebagainya, dan validasi ini dalam konfigurasi Anda.
- Jika Anda tidak memiliki basis kode Terraform yang ada, gunakan praktik terbaik ini saat Anda menyusun konfigurasi baru Anda. Ikuti saran seputar manajemen negara, otentikasi, struktur kode, dan sebagainya dari awal.
- Coba gunakan beberapa modul HashiCorp komunitas yang dirujuk dalam panduan ini untuk melihat apakah mereka menyederhanakan pola arsitektur Anda. Modul memungkinkan tingkat abstraksi yang lebih tinggi, jadi Anda tidak perlu menulis ulang sumber daya umum.
- Aktifkan linting, pemindaian keamanan, pemeriksaan kebijakan, dan alat pengujian otomatis untuk memperkuat beberapa praktik terbaik seputar keamanan, kepatuhan, dan kualitas kode. Alat seperti TFlint, tfsec, dan Checkov dapat membantu.
- Tinjau dokumentasi AWS Penyedia terbaru untuk melihat apakah ada sumber daya atau fungsionalitas baru yang dapat membantu mengoptimalkan penggunaan Terraform Anda. Tetap up to date pada versi baru AWS Penyedia.
- Untuk panduan tambahan, lihat dokumentasi Terraform, panduan praktik terbaik, dan panduan gaya di situs web. HashiCorp

Sumber daya

Referensi

Tautan berikut menyediakan bahan bacaan tambahan untuk AWS Penyedia Terraform dan menggunakan Terraform untuk IAc aktif. AWS

- AWS Penyedia Terraform (dokumentasi) HashiCorp
- Modul Terraform untuk AWS layanan (Terraform Registry)
- HashiCorp Kemitraan AWS dan Kemitraan (posting HashiCorp blog)
- Kredensi Dinamis dengan AWS Penyedia (dokumentasi HCP Terraform)
- DynamoDB State Locking (dokumentasi Terraform)
- Menegakkan Kebijakan dengan Sentinel (dokumentasi Terraform)

Alat

Alat berikut membantu meningkatkan kualitas kode dan otomatisasi konfigurasi Terraform AWS, seperti yang direkomendasikan dalam panduan praktik terbaik ini.

Kualitas kode:

- Checkov: Memindai kode Terraform untuk mengidentifikasi kesalahan konfigurasi sebelum penerapan.
- TFLint: Mengidentifikasi kemungkinan kesalahan, sintaks usang, dan deklarasi yang tidak digunakan. Linter ini juga dapat menegakkan praktik AWS terbaik dan konvensi penamaan.
- terraform-docs: Menghasilkan dokumentasi dari modul Terraform dalam berbagai format keluaran.

Alat otomatisasi:

- HCP Terraform: Membantu tim membuat versi, berkolaborasi, dan membangun alur kerja Terraform dengan pemeriksaan kebijakan dan gerbang persetujuan.
- Atlantis: Alat otomatisasi permintaan tarik Terraform open source untuk memvalidasi perubahan kode.

- CDK untuk Terraform: Kerangka kerja yang memungkinkan Anda menggunakan bahasa yang sudah dikenal seperti, TypeScript Python, Java, C #, dan Go alih-alih Bahasa HashiCorp Konfigurasi (HCL) untuk menentukan, menyediakan, dan menguji infrastruktur Terraform Anda sebagai kode.

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
Pembaruan penguncian status jarak jauh	Memperbarui bagian praktik terbaik Backend untuk mencerminkan penguncia n status asli Amazon S3, yang sekarang merupakan pendekatan yang disarankan.	Agustus 26, 2025
Publikasi awal	—	28 Mei 2024

AWS Glosarium Panduan Preskriptif

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

Nomor

7 Rs

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

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

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

A

ABAC

Lihat kontrol akses berbasis atribut.

layanan abstrak

Lihat layanan terkelola.

ASAM

Lihat atomisitas, konsistensi, isolasi, daya tahan.

migrasi aktif-aktif

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

migrasi aktif-pasif

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

fungsi agregat

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

AI

Lihat kecerdasan buatan.

AIOps

Lihat operasi kecerdasan buatan.

anonimisasi

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

anti-pola

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

kontrol aplikasi

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

portofolio aplikasi

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

kecerdasan buatan (AI)

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

operasi kecerdasan buatan (AIOps)

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

enkripsi asimetris

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

atomisitas, konsistensi, isolasi, daya tahan (ACID)

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

kontrol akses berbasis atribut (ABAC)

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

sumber data otoritatif

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

Zona Ketersediaan

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

AWS Kerangka Adopsi Cloud (AWS CAF)

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

AWS Kerangka Kualifikasi Beban Kerja (AWS WQF)

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

B

bot buruk

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

BCP

Lihat perencanaan kontinuitas bisnis.

grafik perilaku

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

sistem big-endian

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

klasifikasi biner

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

filter mekar

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

deployment biru/hijau

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

bot

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

botnet

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

cabang

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

akses break-glass

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

strategi brownfield

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

cache penyangga

Area memori tempat data yang paling sering diakses disimpan.

kemampuan bisnis

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

perencanaan kelangsungan bisnis (BCP)

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

C

KAFE

Lihat Kerangka Adopsi AWS Cloud.

penyebaran kenari

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

CCoE

Lihat Cloud Center of Excellence.

CDC

Lihat mengubah pengambilan data.

ubah pengambilan data (CDC)

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

rekayasa kekacauan

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

CI/CD

Lihat integrasi berkelanjutan dan pengiriman berkelanjutan.

klasifikasi

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

Enkripsi sisi klien

Enkripsi data secara lokal, sebelum target Layanan AWS menerimanya.

Pusat Keunggulan Cloud (CCoE)

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

komputasi cloud

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

model operasi cloud

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

tahap adopsi cloud

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

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

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

CMDB

Lihat database manajemen konfigurasi.

repositori kode

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

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

cache dingin

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

data dingin

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

visi komputer (CV)

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

konfigurasi drift

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

database manajemen konfigurasi (CMDB)

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

paket kesesuaian

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

integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD)

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

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

CV

Lihat visi komputer.

D

data saat istirahat

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

klasifikasi data

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

penyimpangan data

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

data dalam transit

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

jala data

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

minimalisasi data

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

perimeter data

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

prapemrosesan data

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

asal data

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

subjek data

Individu yang datanya dikumpulkan dan diproses.

gudang data

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

bahasa definisi database (DDL)

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

bahasa manipulasi basis data (DHTML)

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

DDL

Lihat bahasa definisi database.

ansambel yang dalam

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

pembelajaran mendalam

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

defense-in-depth

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

administrator yang didelegasikan

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

deployment

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

lingkungan pengembangan

Lihat lingkungan.

kontrol detektif

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

pemetaan aliran nilai pengembangan (DVSM)

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

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

kembar digital

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

tabel dimensi

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

musibah

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

pemulihan bencana (DR)

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

DML~

Lihat bahasa manipulasi basis data.

desain berbasis domain

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

DR

Lihat pemulihan bencana.

deteksi drift

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

DVSM

Lihat pemetaan aliran nilai pengembangan.

E

EDA

Lihat analisis data eksplorasi.

EDI

Lihat pertukaran data elektronik.

komputasi tepi

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

pertukaran data elektronik (EDI)

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

enkripsi

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

kunci enkripsi

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

endianness

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

titik akhir

Lihat titik akhir layanan.

layanan endpoint

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

perencanaan sumber daya perusahaan (ERP)

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

enkripsi amplop

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

lingkungan

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

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

epik

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

ERP

Lihat perencanaan sumber daya perusahaan.

analisis data eksplorasi (EDA)

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

F

tabel fakta

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

gagal cepat

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

batas isolasi kesalahan

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

cabang fitur

Lihat cabang.

fitur

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

pentingnya fitur

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

transformasi fitur

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

beberapa tembakan mendorong

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

FGAC

Lihat kontrol akses berbutir halus.

kontrol akses berbutir halus (FGAC)

Penggunaan beberapa kondisi untuk mengizinkan atau menolak permintaan akses.

migrasi flash-cut

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

FM

Lihat model pondasi.

model pondasi (FM)

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

G

AI generatif

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

pemblokiran geografis

Lihat pembatasan geografis.

pembatasan geografis (pemblokiran geografis)

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

Alur kerja Gitflow

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

gambar emas

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

strategi greenfield

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

pagar pembatas

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

H

HA

Lihat ketersediaan tinggi.

migrasi database heterogen

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

ketersediaan tinggi (HA)

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

modernisasi sejarawan

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

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

data penahanan

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

migrasi database homogen

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

data panas

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

perbaikan terbaru

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

periode hypercare

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

I

IAc

Lihat infrastruktur sebagai kode.

kebijakan berbasis identitas

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

aplikasi idle

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

IIoT

Lihat Internet of Things industri.

infrastruktur yang tidak dapat diubah

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

masuk (masuknya) VPC

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

migrasi inkremental

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

Industri 4.0

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

infrastruktur

Semua sumber daya dan aset yang terkandung dalam lingkungan aplikasi.

infrastruktur sebagai kode (IAc)

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

Internet of Things industri (IIoT)

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

inspeksi VPC

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

Internet of Things (IoT)

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

interpretasi

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

IoT

Lihat Internet of Things.

Perpustakaan informasi TI (ITIL)

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

Manajemen layanan TI (ITSM)

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

ITIL

Lihat perpustakaan informasi TI.

ITSM

Lihat manajemen layanan TI.

L

kontrol akses berbasis label (LBAC)

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

landing zone

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

model bahasa besar (LLM)

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

migrasi besar

Migrasi 300 atau lebih server.

LBAC

Lihat kontrol akses berbasis label.

hak istimewa paling sedikit

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

angkat dan geser

Lihat 7 Rs.

sistem endian kecil

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

LLM

Lihat model bahasa besar.

lingkungan yang lebih rendah

Lihat lingkungan.

M

pembelajaran mesin (ML)

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

cabang utama

Lihat cabang.

malware

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

layanan terkelola

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

sistem eksekusi manufaktur (MES)

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

PETA

Lihat Program Percepatan Migrasi.

mekanisme

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

akun anggota

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

MES

Lihat sistem eksekusi manufaktur.

Transportasi Telemetri Antrian Pesan (MQTT)

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

layanan mikro

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

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

arsitektur microservices

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

Program Percepatan Migrasi (MAP)

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

migrasi dalam skala

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

pabrik migrasi

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

metadata migrasi

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

pola migrasi

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

Penilaian Portofolio Migrasi (MPA)

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

Penilaian Kesiapan Migrasi (MRA)

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

strategi migrasi

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

ML

Lihat pembelajaran mesin.

modernisasi

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

penilaian kesiapan modernisasi

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

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

aplikasi monolitik (monolit)

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

MPA

Lihat Penilaian Portofolio Migrasi.

MQTT

Lihat Transportasi Telemetri Antrian Pesan.

klasifikasi multiclass

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

infrastruktur yang bisa berubah

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

O

OAC

Lihat kontrol akses asal.

OAI

Lihat identitas akses asal.

OCM

Lihat manajemen perubahan organisasi.

migrasi offline

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

OI

Lihat integrasi operasi.

OLA

Lihat perjanjian tingkat operasional.

migrasi online

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

OPC-UA

Lihat Komunikasi Proses Terbuka - Arsitektur Terpadu.

Komunikasi Proses Terbuka - Arsitektur Terpadu (OPC-UA)

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

perjanjian tingkat operasional (OLA)

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

Tinjauan Kesiapan Operasional (ORR)

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

teknologi operasional (OT)

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

integrasi operasi (OI)

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

jejak organisasi

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

manajemen perubahan organisasi (OCM)

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

kontrol akses asal (OAC)

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

identitas akses asal (OAI)

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

ORR

Lihat tinjauan kesiapan operasional.

OT

Lihat teknologi operasional.

keluar (jalan keluar) VPC

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

P

batas izin

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

Informasi Identifikasi Pribadi (PII)

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

PII

Lihat informasi yang dapat diidentifikasi secara pribadi.

buku pedoman

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

PLC

Lihat pengontrol logika yang dapat diprogram.

PLM

Lihat manajemen siklus hidup produk.

kebijakan

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

persistensi poliglot

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

penilaian portofolio

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

predikat

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

predikat pushdown

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

kontrol preventif

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

principal

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

privasi berdasarkan desain

Pendekatan rekayasa sistem yang memperhitungkan privasi melalui seluruh proses pengembangan.

zona host pribadi

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

kontrol proaktif

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

manajemen siklus hidup produk (PLM)

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

lingkungan produksi

Lihat lingkungan.

pengontrol logika yang dapat diprogram (PLC)

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

rantai cepat

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

pseudonimisasi

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

publish/subscribe (pub/sub)

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

Q

rencana kueri

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

regresi rencana kueri

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

R

Matriks RACI

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

LAP

Lihat Retrieval Augmented Generation.

ransomware

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

Matriks RASCI

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

RCAC

Lihat kontrol akses baris dan kolom.

replika baca

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

arsitek ulang

Lihat 7 Rs.

tujuan titik pemulihan (RPO)

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

tujuan waktu pemulihan (RTO)

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

refactor

Lihat 7 Rs.

Region

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

regresi

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

rehost

Lihat 7 Rs.

melepaskan

Dalam proses penyebaran, tindakan mempromosikan perubahan pada lingkungan produksi.

memindahkan

Lihat 7 Rs.

memplatform ulang

Lihat 7 Rs.

pembelian kembali

Lihat 7 Rs.

ketahanan

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

kebijakan berbasis sumber daya

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

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

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

kontrol responsif

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

melestarikan

Lihat 7 Rs.

pensiun

Lihat 7 Rs.

Retrieval Augmented Generation (RAG)

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

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

rotasi

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

kontrol akses baris dan kolom (RCAC)

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

RPO

Lihat tujuan titik pemulihan.

RTO

Lihat tujuan waktu pemulihan.

buku runbook

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

D

SAML 2.0

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

PENIPUAN

Lihat kontrol pengawasan dan akuisisi data.

SCP

Lihat kebijakan kontrol layanan.

Rahasia

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

keamanan dengan desain

Pendekatan rekayasa sistem yang memperhitungkan keamanan melalui seluruh proses pengembangan.

kontrol keamanan

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

pengerasan keamanan

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

sistem informasi keamanan dan manajemen acara (SIEM)

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

otomatisasi respons keamanan

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

enkripsi sisi server

Enkripsi data di tujuannya, oleh Layanan AWS yang menerimanya.

kebijakan kontrol layanan (SCP)

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

titik akhir layanan

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

perjanjian tingkat layanan (SLA)

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

indikator tingkat layanan (SLI)

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

tujuan tingkat layanan (SLO)

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

model tanggung jawab bersama

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

SIEM

Lihat informasi keamanan dan sistem manajemen acara.

titik kegagalan tunggal (SPOF)

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

SLA

Lihat perjanjian tingkat layanan.

SLI

Lihat indikator tingkat layanan.

SLO

Lihat tujuan tingkat layanan.

split-and-seed model

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

SPOF

Lihat satu titik kegagalan.

skema bintang

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

pola ara pencekik

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

subnet

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

kontrol pengawasan dan akuisisi data (SCADA)

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

enkripsi simetris

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

pengujian sintetis

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

sistem prompt

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

T

tag

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

variabel target

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

daftar tugas

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

lingkungan uji

Lihat lingkungan.

pelatihan

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

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

gerbang transit

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

alur kerja berbasis batang

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

akses tepercaya

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

penyetelan

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

tim dua pizza

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

U

waswas

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

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

tugas yang tidak terdiferensiasi

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

lingkungan atas

Lihat lingkungan.

V

menyedot debu

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

kendali versi

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

Peering VPC

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

kerentanan

Kelemahan perangkat lunak atau perangkat keras yang membahayakan keamanan sistem.

W

cache hangat

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

data hangat

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

fungsi jendela

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

beban kerja

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

aliran kerja

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

CACING

Lihat menulis sekali, baca banyak.

WQF

Lihat AWS Kerangka Kualifikasi Beban Kerja.

tulis sekali, baca banyak (WORM)

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

Z

eksploitasi zero-day

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

kerentanan zero-day

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

bisikan zero-shot

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

aplikasi zombie

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

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