Cómo empezar a usar Terraform: orientación AWS CDK y expertos AWS CloudFormation

AWS Orientación prescriptiva

AWS Orientación prescriptiva: Cómo empezar a usar Terraform: orientación AWS CDK y expertos AWS CloudFormation

Table of Contents

Introducción a Terraform: orientación para expertos en AWS CDK y AWS CloudFormation

Steven Guggenheimer, Amazon Web Services (AWS)

Marzo de 2024 (historial del documento)

Si su experiencia con el aprovisionamiento de recursos en la nube se basa exclusivamente en el ámbito de AWS, es posible que tenga una experiencia limitada con las herramientas de infraestructura como código (IaC) más allá de lo convencional. AWS Cloud Development Kit (AWS CDK)AWS CloudFormation De hecho, es posible que no conozcas por completo herramientas similares, como Hashicorp Terraform. Sin embargo, cuanto más te adentres en tu viaje a la nube, más inevitable será que te encuentres con Terraform. No cabe duda de que será una ventaja para usted estar familiarizado con sus conceptos básicos.

Si bien Terraform AWS CDK, the y Terraform CloudFormation alcanzan objetivos similares y comparten muchos conceptos básicos, existen bastantes diferencias. Es posible que no estés preparado para estas diferencias si te acercas a Terraform por primera vez. Al fin AWS CDK y al cabo, todas las CloudFormation pilas se basan en uno mismo Cuentas de AWS, por lo que tienen una relación directa con la mayoría de los recursos que mantienen. Terraform no se basa en el entorno de un único proveedor de servicios en la nube. Esto le da la flexibilidad necesaria para dar soporte a varios proveedores diferentes, pero debe mantener los recursos desde una ubicación remota.

Esta guía ayuda a desmitificar los conceptos básicos de Terraform para ayudarle a afrontar cualquier desafío relacionado con la IaC que se le presente. Se centra en cómo Terraform utiliza conceptos, como proveedores, módulos y archivos de estado, para aprovisionar recursos. También contrasta los conceptos de Terraform con la forma en que AWS CDK CloudFormation realiza operaciones similares.

> **Note**
>
> AWS CDK Esto ayuda a los desarrolladores a implementar CloudFormation pilas mediante lenguajes de codificación programática. Después de ejecutar`cdk synth`, el código se convierte en CloudFormation plantillas. A partir de ese momento, el proceso es idéntico entre AWS CDK y CloudFormation. En aras de la brevedad, esta guía generalmente se refiere

al proceso AWS IaC en CloudFormation términos, pero las comparaciones son igualmente adecuadas para el. AWS CDK

CloudFormation y terminología de Terraform

Al comparar Terraform con AWS CDK y CloudFormation, conciliar los conceptos básicos de la IaC puede resultar difícil debido a la terminología incoherente que se utiliza para describirlos. Los siguientes son estos términos y cómo se referirá a ellos en esta guía:

- Pila : una pila es un iAC que se despliega en una CI/CD canalización y se puede rastrear como una sola unidad.Aunque este término es común en Terraform CloudFormation, en realidad, no lo usa. Una pila de Terraform es un módulo raíz desplegado con todos sus módulos secundarios. Sin embargo, para evitar confusiones con el término módulo, en esta guía se utiliza el término pila para describir una implementación única para ambas herramientas.
- Estado: el estado incluye todos los recursos rastreados actualmente y sus configuraciones actuales dentro de una pila de implementación de IaC. Como se describe en la Comprensión de los estados y los backends de Terraform sección, Terraform usa el término estado más que. CloudFormation Esto se debe a que el mantenimiento del estado es más visible en Terraform, pero el seguimiento y la actualización del estado son igualmente importantes para ellos. CloudFormation
- Archivo iAC: un archivo iAC es un archivo único que contiene la infraestructura como lenguaje de código (IaC). CloudFormation hace referencia a un único CloudFormation archivo como plantilla. Sin embargo, las plantillas y los archivos de plantillas en Terraform son algo completamente diferente. El equivalente a una CloudFormation plantilla en Terraform se denomina archivo de configuración. Para minimizar la confusión en esta guía, el término archivo o archivo IaC se utiliza para referirse tanto a las CloudFormation plantillas como a los archivos de configuración de Terraform.

En la siguiente tabla se compara la terminología utilizada para Terraform CloudFormation y Terraform. La intención de esta tabla es mostrar similitudes. No se trata de one-to-one comparaciones. Cada concepto difiere al menos ligeramente entre Terraform CloudFormation y Terraform. Los conceptos se explican en profundidad en las secciones pertinentes de esta guía.

CloudFormation término	Término de Terraform	Sección de esta guía
Interfaces CDK (como IBucket)	Origen de datos	Comprensión de las fuentes de datos de Terraform
Conjunto de cambios	Plan	Descripción de los módulos de Terraform
Funciones de condiciones	Expresiones condicionales	Comprensión de las funciones, expresiones y metaargumentos de Terraform
Atributo `DependsOn`	`depends_on` metaargumento	Comprensión de las funciones, expresiones y metaargumentos de Terraform
Funciones intrínsecas	Funciones	Comprensión de las funciones, expresiones y metaargumentos de Terraform
Módulos	Módulos	Descripción de los módulos de Terraform
Outputs	Valores de salida	Comprensión de las variables, los valores locales y los resultados de Terraform
Parameters	Variables	Comprensión de las variables, los valores locales y los resultados de Terraform
Registro	Proveedores	Entendiendo a los proveedores de Terraform
Plantilla	Archivo de configuración	Todos

Comprensión de los recursos de Terraform

La razón principal de la existencia de ambos AWS CloudFormation y Terraform es la creación y el mantenimiento de los recursos en la nube. Pero, ¿qué es exactamente un recurso en la nube? ¿Y CloudFormation los recursos y los recursos de Terraform son lo mismo? La respuesta es... sí y no. Para ilustrarlo, en esta guía se proporciona un ejemplo del uso CloudFormation y posterior creación de Terraform para crear un bucket de Amazon Simple Storage Service (Amazon S3).

El siguiente ejemplo CloudFormation de código crea un bucket de Amazon S3 de muestra.

```
{
    "myS3Bucket": {
        "Type": "AWS::S3::Bucket",
        "Properties": {
            "BucketName": "my-s3-bucket",
            "BucketEncryption": {
                "ServerSideEncryptionConfiguration": [
                    {
                        "ServerSideEncryptionByDefault": {
                            "SSEAlgorithm": "AES256"
                        }
                    }
                ]
            },
            "PublicAccessBlockConfiguration": {
                "BlockPublicAcls": true,
                "BlockPublicPolicy": true,
                "IgnorePublicAcls": true,
                "RestrictPublicBuckets": true
            },
            "VersioningConfiguration": {
                "Status": "Enabled"
            }
        }
    }
}
```

El siguiente ejemplo de código de Terraform crea un bucket de Amazon S3 idéntico.

```
resource "aws_s3_bucket" "myS3Bucket" {
```

```
  bucket = "my-s3-bucket"
}

resource "aws_s3_bucket_server_side_encryption_configuration" "bucketencryption" {
  bucket = aws_s3_bucket.myS3Bucket.id
  rule {
    apply_server_side_encryption_by_default {
      sse_algorithm = "AES256"
    }
  }
}

resource "aws_s3_bucket_public_access_block" "publicaccess" {
  bucket                  = aws_s3_bucket.myS3Bucket.id
  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}

resource "aws_s3_bucket_versioning" "versioning" {
  bucket = aws_s3_bucket.myS3Bucket.id
  versioning_configuration {
    status = "Enabled"
  }
}
```

En el caso de Terraform, un proveedor define el recurso y, a continuación, los desarrolladores declaran y configuran esos recursos. Los proveedores son un concepto que se analiza en esta guía en la siguiente sección. El ejemplo de Terraform crea recursos completamente independientes para varios de los ajustes del bucket de S3. La creación de recursos separados para la configuración no es necesariamente algo típico de la forma en que el AWS proveedor de Terraform trata AWS los recursos. Sin embargo, este ejemplo muestra una distinción importante. Si bien un CloudFormation recurso está estrictamente definido por la especificación del CloudFormation recurso, Terraform no tiene ese requisito. En Terraform, el concepto de recurso es un poco más confuso.

Si bien las herramientas pueden diferir en cuanto a las barreras exactas que definen qué es un recurso único, en términos generales, un recurso en la nube es cualquier entidad concreta que existe en la nube y que se puede crear, actualizar o eliminar. Por lo tanto, independientemente de cuántos recursos estén involucrados, los dos ejemplos anteriores crean exactamente lo mismo con exactamente la misma configuración dentro de un. Cuenta de AWS

Entendiendo a los proveedores de Terraform

En Terraform, un proveedor es un complemento que interactúa con proveedores de nube, herramientas de terceros y otras API. Para usar Terraform AWS, utiliza el AWS proveedor, que interactúa con los recursos. AWS

Si nunca has utilizado el AWS CloudFormation registro para incorporar extensiones de terceros en tus paquetes de despliegue, es posible que los proveedores de Terraform tarden un poco en acostumbrarse. Como CloudFormation es nativo de AWS, el proveedor de AWS recursos ya está ahí de forma predeterminada. Terraform, por otro lado, no tiene un único proveedor predeterminado, por lo que no se puede suponer nada sobre el origen de un recurso determinado. Esto significa que lo primero que debe declararse en un archivo de configuración de Terraform es exactamente a dónde van los recursos y cómo van a llegar allí.

Esta distinción añade una capa adicional de complejidad a Terraform que no existe. CloudFormation Sin embargo, esa complejidad proporciona una mayor flexibilidad. Puede declarar varios proveedores en un único módulo de Terraform y, a continuación, los recursos subyacentes que se creen pueden interactuar entre sí como parte de la misma capa de despliegue.

Esto puede resultar útil de muchas maneras. Los proveedores no tienen por qué ser necesariamente proveedores de nube independientes. Los proveedores pueden representar cualquier fuente de recursos en la nube. Por ejemplo, tomemos Amazon Elastic Kubernetes Service (Amazon EKS). Al aprovisionar un clúster de Amazon EKS, es posible que desee utilizar los gráficos de Helm para gestionar extensiones de terceros y utilizar el propio Kubernetes para gestionar los recursos de los pods. Como AWSHelm y Kubernetes tienen sus propios proveedores de Terraform, puede aprovisionar e integrar todos estos recursos al mismo tiempo y, después, transferir valores entre ellos.

En el siguiente ejemplo de código para Terraform, el AWS proveedor crea un clúster de Amazon EKS y, a continuación, la información de configuración de Kubernetes resultante se transmite a los proveedores de Helm y Kubernetes.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 4.33.0"
    }
```

```
    helm = {
      source  = "hashicorp/helm"
      version = "2.12.1"
    }

    kubernetes = {
      source  = "hashicorp/kubernetes"
      version = "2.26.0"
    }
  }
  required_version = ">= 1.2.0"
}

provider "aws" {
  region = "us-west-2"
}

resource "aws_eks_cluster" "example_0" {
  name     = "example_0"
  role_arn = aws_iam_role.cluster_role.arn
  vpc_config {
    endpoint_private_access = true
    endpoint_public_access  = true
    subnet_ids              = var.subnet_ids
  }
}

locals {
  host        = aws_eks_cluster.example_0.endpoint
  certificate = base64decode(aws_eks_cluster.example_0.certificate_authority.data)
}

provider "helm" {
  kubernetes {
    host                   = local.host
    cluster_ca_certificate = local.certificate
    # exec allows for an authentication command to be run to obtain user
    # credentials rather than having them stored directly in the file
    exec {
      api_version = "client.authentication.k8s.io/v1beta1"
      args        = ["eks", "get-token", "--cluster-name",
 aws_eks_cluster.example_0.name]
      command     = "aws"
```

```
    }
  }
}

provider "kubernetes" {
  host                   = local.host
  cluster_ca_certificate = local.certificate
  exec {
    api_version = "client.authentication.k8s.io/v1beta1"
    args        = ["eks", "get-token", "--cluster-name",
 aws_eks_cluster.example_0.name]
    command     = "aws"
  }
}
```

En lo que respecta a los proveedores, las dos herramientas de IaC tienen que hacer concesiones. Terraform depende completamente de paquetes de proveedores ubicados externamente, que son el motor que impulsa sus despliegues. CloudFormation apoya internamente todos los procesos principales AWS . Con CloudFormation esto, solo debe preocuparse por los proveedores de terceros si desea incorporar una extensión de terceros. Cada enfoque tiene sus ventajas y desventajas. Cuál es el adecuado para usted va más allá del alcance de esta guía, pero es importante recordar la diferencia al evaluar ambas herramientas.

Uso de alias de Terraform

En Terraform, puede pasar configuraciones personalizadas a cada proveedor. Entonces, ¿qué pasa si desea usar configuraciones de varios proveedores dentro del mismo módulo? En ese caso, tendrías que usar un alias. Los alias le ayudan a seleccionar qué proveedor usar a nivel de recurso o módulo. Cuando tiene más de una instancia del mismo proveedor, utiliza un alias para definir las instancias no predeterminadas. Por ejemplo, la instancia de tu proveedor predeterminado puede ser específica Región de AWS, pero usas alias para definir regiones alternativas.

El siguiente ejemplo de Terraform muestra cómo usar un alias para aprovisionar depósitos en diferentes áreas. Regiones de AWS La región predeterminada para el proveedor es`us-west-2`, pero puedes usar el alias este para aprovisionar recursos. `us-east-2`

```
provider "aws" {
  region = "us-west-2"
}
```

```
provider "aws" {
  alias  = "east"
  region = "us-east-2"
}

resource "aws_s3_bucket" "myWestS3Bucket" {
  bucket = "my-west-s3-bucket"
}

resource "aws_s3_bucket" "myEastS3Bucket" {
  provider = aws.east
  bucket   = "my-east-s3-bucket"
}
```

Si utiliza un argumento `alias` junto con el `provider` metaargumento, como se muestra en el ejemplo anterior, puede especificar una configuración de proveedor diferente para recursos específicos. El aprovisionamiento de varios recursos Regiones de AWS en una sola pila es solo el principio. Asignar alias a los proveedores es increíblemente práctico en muchos sentidos.

Por ejemplo, es muy común aprovisionar varios clústeres de Kubernetes a la vez. Los alias pueden ayudarle a configurar otros proveedores de Helm y Kubernetes para que pueda utilizar estas herramientas de terceros de forma diferente para los distintos recursos de Amazon EKS. El siguiente ejemplo de código de Terraform ilustra cómo usar los alias para realizar esta tarea.

```
resource "aws_eks_cluster" "example_0" {
  name     = "example_0"
  role_arn = aws_iam_role.cluster_role.arn
  vpc_config {
    endpoint_private_access = true
    endpoint_public_access  = true
    subnet_ids              = var.subnet_ids[0]
  }
}

resource "aws_eks_cluster" "example_1" {
  name     = "example_1"
  role_arn = aws_iam_role.cluster_role.arn
  vpc_config {
    endpoint_private_access = true
    endpoint_public_access  = true
    subnet_ids              = var.subnet_ids[1]
  }
```

```
}

locals {
  host         = aws_eks_cluster.example_0.endpoint
  certificate  = base64decode(aws_eks_cluster.example_0.certificate_authority.data)
  host1        = aws_eks_cluster.example_1.endpoint
  certificate1 = base64decode(aws_eks_cluster.example_1.certificate_authority.data)
}

provider "helm" {
  kubernetes {
    host                   = local.host
    cluster_ca_certificate = local.certificate
    exec {
      api_version = "client.authentication.k8s.io/v1beta1"
      args        = ["eks", "get-token", "--cluster-name",
 aws_eks_cluster.example_0.name]
      command     = "aws"
    }
  }
}

provider "helm" {
  alias = "helm1"
  kubernetes {
    host                   = local.host1
    cluster_ca_certificate = local.certificate1
    exec {
      api_version = "client.authentication.k8s.io/v1beta1"
      args        = ["eks", "get-token", "--cluster-name",
 aws_eks_cluster.example_1.name]
      command     = "aws"
    }
  }
}

provider "kubernetes" {
  host                   = local.host
  cluster_ca_certificate = local.certificate
  exec {
    api_version = "client.authentication.k8s.io/v1beta1"
    args        = ["eks", "get-token", "--cluster-name",
 aws_eks_cluster.example_0.name]
```

```
    command     = "aws"
  }
}

provider "kubernetes" {
  alias                  = "kubernetes1"
  host                   = local.host1
  cluster_ca_certificate = local.certificate1
  exec {
    api_version = "client.authentication.k8s.io/v1beta1"
    args        = ["eks", "get-token", "--cluster-name",
 aws_eks_cluster.example_1.name]
    command     = "aws"
  }
}
```

Descripción de los módulos de Terraform

En el ámbito de la infraestructura como código (IaC), un módulo es un bloque de código autónomo que se aísla y se empaqueta para su reutilización. El concepto de módulos es un aspecto ineludible del desarrollo de Terraform. Para obtener más información, consulte los módulos en la documentación de Terraform. AWS CloudFormation también admite módulos. Para obtener más información, consulte Introducción a AWS CloudFormation los módulos en el blog sobre operaciones y migraciones en la AWS nube.

La principal diferencia entre los módulos de Terraform CloudFormation es que CloudFormation los módulos se importan mediante un tipo de recurso especial ()`AWS::CloudFormation::ModuleVersion`. En Terraform, cada configuración tiene al menos un módulo, conocido como módulo raíz. Los recursos de Terraform que se encuentran en el archivo.tf principal o los archivos de un archivo de configuración de Terraform se consideran en el módulo raíz. Luego, el módulo raíz puede llamar a otros módulos para incluirlos en la pila. El siguiente ejemplo muestra un módulo raíz que aprovisiona un clúster de Amazon Elastic Kubernetes Service (Amazon EKS) mediante el módulo eks de código abierto.

```
terraform {
  required_providers {
    helm = {
      source  = "hashicorp/helm"
      version = "2.12.1"
    }
  }
  required_version = ">= 1.2.0"
}

module "eks" {
  source  = "terraform-aws-modules/eks/aws"
  version = "20.2.1"
  vpc_id  = var.vpc_id
}

provider "helm" {
  kubernetes {
    host                   = module.eks.cluster_endpoint
    cluster_ca_certificate =
 base64decode(module.eks.cluster_certificate_authority_data)
  }
```

```
}
```

Es posible que haya observado que el archivo de configuración anterior no incluye al proveedor. AWS Esto se debe a que los módulos son autónomos y pueden incluir sus propios proveedores. Como los proveedores de Terraform son globales, los proveedores de un módulo secundario se pueden usar en el módulo raíz. Sin embargo, esto no es cierto para todos los valores de los módulos. De forma predeterminada, otros valores internos de un módulo se limitan únicamente a ese módulo y deben declararse como salidas para que se pueda acceder a ellos en el módulo raíz. Puede aprovechar los módulos de código abierto para simplificar la creación de recursos dentro de su pila. Por ejemplo, el módulo eks hace más que aprovisionar un clúster de EKS: aprovisiona un entorno de Kubernetes en pleno funcionamiento. Su uso puede ahorrarle tener que escribir docenas de líneas de código adicionales, siempre que la configuración del módulo eks se adapte a sus necesidades.

Módulos de llamadas

Dos de los principales comandos CLI de Terraform que se ejecutan durante la implementación de Terraform son terraform init y terraform apply. Uno de los pasos predeterminados que realiza el `terraform init` comando es localizar todos los módulos secundarios e importarlos como dependencias al directorio. `.terraform/modules` Durante el desarrollo, siempre que añada un nuevo módulo de origen externo, debe volver a inicializarlo antes de utilizar el comando. `apply` Cuando escuchas una referencia a un módulo de Terraform, se refiere a los paquetes de este directorio. Estrictamente hablando, el módulo que declaras en tu código es el módulo de llamada, por lo que, en la práctica, la palabra clave module llama al módulo real, que se almacena como una dependencia.

De esta manera, el módulo de llamada sirve como una representación más sucinta del módulo completo que se reemplazará cuando se lleve a cabo el despliegue. Puedes aprovechar esta idea creando tus propios módulos dentro de tus pilas para reforzar la separación lógica de los recursos utilizando los criterios que desees. Recuerda que el objetivo final de hacer esto debería ser reducir la complejidad de tus pilas. Dado que compartir datos entre módulos requiere generar esos datos desde dentro del módulo, a veces confiar demasiado en los módulos puede complicar demasiado las cosas.

El módulo raíz

Como cada configuración de Terraform tiene al menos un módulo, puede ser útil examinar las propiedades del módulo con el que más se va a tratar: el módulo raíz. Siempre que trabajes en un

proyecto de Terraform, el módulo raíz se compone de todos los `.tf` (o `.tf.json`) archivos de tu directorio de nivel superior. Cuando ejecutas `terraform apply` en ese directorio de nivel superior, Terraform intenta ejecutar todos los `.tf` archivos que encuentra allí. Todos los archivos de los subdirectorios se ignoran a menos que se invoquen en uno de estos archivos de configuración de nivel superior.

Esto proporciona cierta flexibilidad a la hora de estructurar el código. También es la razón por la que es más preciso referirse a su implementación de Terraform como un módulo que como un archivo, ya que varios archivos podrían estar involucrados en un solo proceso. Existe una estructura de módulos estándar que Terraform recomienda como prácticas recomendadas. Sin embargo, si colocara cualquier `.tf` archivo en su directorio de nivel superior, se ejecutaría junto con el resto de los archivos. De hecho, todos los `.tf` archivos de nivel superior de un módulo se despliegan cuando se ejecuta. `terraform apply` Entonces, ¿qué archivo ejecuta Terraform primero? La respuesta a esa pregunta es muy importante.

Hay una serie de pasos que Terraform lleva a cabo después de la inicialización y antes del despliegue de la pila. En primer lugar, se analizan las configuraciones existentes y, a continuación, se crea un gráfico de dependencias. El gráfico de dependencia determina qué recursos se requieren y en qué orden deben abordarse. Los recursos que contienen propiedades a las que se hace referencia en otros recursos, por ejemplo, se gestionarían antes que sus recursos dependientes. Del mismo modo, los recursos que declaran la dependencia de forma explícita mediante el `depends_on` parámetro se gestionarán después de los recursos que especifiquen. Cuando sea posible, Terraform puede implementar el paralelismo y gestionar recursos no dependientes simultáneamente. Puede ver el gráfico de dependencias antes de implementarlo mediante el comando terraform graph.

Una vez creado el gráfico de dependencias, Terraform determina lo que hay que hacer durante la implementación. Compara el gráfico de dependencias con el archivo de estado más reciente. El resultado de este proceso se denomina plan y es muy parecido a un conjunto de CloudFormation cambios. Puede ver el plan actual mediante el comando terraform plan.

Como práctica recomendada, se recomienda permanecer lo más cerca posible de la estructura de módulos estándar. En los casos en los que los archivos de configuración sean demasiado largos para gestionarlos de forma eficiente y las separaciones lógicas puedan simplificar la administración, puede distribuir el código en varios archivos. Ten en cuenta cómo funcionan el gráfico de dependencias y el proceso de planificación para que tus pilas funcionen de la forma más eficiente posible.

Comprensión de los estados y los backends de Terraform

Uno de los conceptos más importantes de la infraestructura como código (IaC) es el concepto de estado. Los servicios de IaC mantienen el estado, lo que le permite declarar un recurso en un archivo de IaC sin tener que volver a crearlo cada vez que se implementa. Los archivos IaC documentan el estado de todos los recursos al final de una implementación para poder comparar ese estado con el estado objetivo, tal como se declara en la siguiente implementación. Por lo tanto, si el estado actual contiene un depósito de Amazon Simple Storage Service (Amazon S3) `my-s3-bucket` denominado y los cambios entrantes también contienen ese mismo depósito, el nuevo proceso aplicará todos los cambios encontrados en el depósito existente en lugar de intentar crear un depósito completamente nuevo.

En la siguiente tabla, se proporcionan ejemplos del proceso general de estado de la IaC.

Estado actual	Estado objetivo	Acción
No se ha nombrado ningún bucket de S3 `my-s3-bucket`	Nombre del bucket de S3 `my-s3-bucket`	Cree un depósito de S3 con el nombre `my-s3-bucket`
`my-s3-bucket` sin configurar el control de versiones del bucket	`my-s3-bucket` sin configurar el control de versiones de bucket	Sin acciones
`my-s3-bucket` sin configurar el control de versiones de bucket	`my-s3-bucket` con el control de versiones de bucket configurado	`my-s3-bucket` Configúrelo para tener el control de versiones en bucket
`my-s3-bucket` con el control de versiones de bucket configurado	No se ha nombrado ningún bucket de S3 `my-s3-bucket`	Intento de borrarlo `my-s3-bucket`

Para entender las diferentes formas en que AWS CloudFormation Terraform rastrea el estado, es importante recordar la primera diferencia básica entre las dos herramientas: CloudFormation está alojada dentro de una y Terraform es esencialmente remota. Nube de AWS Este hecho permite CloudFormation mantener el estado internamente. Puedes ir a la CloudFormation consola y ver el

historial de eventos de una pila determinada, pero el propio CloudFormation servicio hace cumplir las reglas estatales por ti.

Los tres modos en los que CloudFormation funciona un recurso determinado son `Create`, `Update`, y`Delete`. El modo actual se determina en función de lo que ocurrió en la última implementación y no se puede influir en él de otro modo. Quizás puedas actualizar CloudFormation los recursos manualmente para determinar el modo que se determine, pero no puedes pasarle un comando CloudFormation que diga «Para este recurso, opera en `Create` modo».

Como Terraform no está alojado en el Nube de AWS, el proceso de mantenimiento del estado debe ser más configurable. Por esta razón, el estado de Terraform se mantiene dentro de un archivo de estado generado automáticamente. Un desarrollador de Terraform tiene que tratar con el estado de forma mucho más directa de lo que lo haría con él. CloudFormation Lo importante es recordar que el seguimiento del estado es igual de importante para ambas herramientas.

De forma predeterminada, el archivo de estado de Terraform se almacena localmente en el nivel superior del directorio principal en el que se ejecuta la pila de Terraform. Si ejecuta el `terraform apply` comando desde su entorno de desarrollo local, verá cómo Terraform genera el archivo terraform.tfstate que utiliza para mantener el estado en tiempo real. Para bien o para mal, esto te da mucho más control sobre el estado en Terraform del que tienes en él. CloudFormation Si bien nunca debe actualizar el archivo de estado directamente, hay varios comandos CLI de Terraform que puede ejecutar para actualizar el estado entre las implementaciones. Por ejemplo, la importación de terraform te permite añadir recursos creados fuera de Terraform a tu pila de despliegues. Por el contrario, puedes eliminar un recurso del estado ejecutando terraform state rm.

El hecho de que Terraform necesite almacenar su estado en algún lugar lleva a otro concepto que no se aplica CloudFormation: el backend. Un backend de Terraform es el lugar en el que una pila de Terraform almacena su archivo de estado tras su implementación. También es donde espera encontrar el archivo de estado cuando comience una nueva implementación. Al ejecutar la pila de forma local, como se ha descrito anteriormente, puede guardar una copia del estado de Terraform en el directorio local de nivel superior. Esto se conoce como backend local.

Cuando se desarrolla para un entorno de integración e implementación continuas (CI/CD), el archivo de estado local generalmente se incluye en el archivo.gitignore para mantenerlo fuera del control de versiones. Por lo tanto, no hay ningún archivo estatal local en la canalización. Para que funcione correctamente, esa fase de la canalización debe encontrar el archivo de estado correcto en alguna parte.Esta es la razón por la que los archivos de configuración de Terraform suelen contener un

bloque de backend. El bloque de fondo indica a la pila de Terraform que necesita buscar el archivo de estado en otro lugar que no sea su propio directorio de nivel superior.

Un backend de Terraform se puede ubicar prácticamente en cualquier lugar: un bucket de Amazon S3, un punto final de API o incluso un espacio de trabajo remoto de Terraform. El siguiente es un ejemplo de un backend de Terraform almacenado en un bucket de Amazon S3.

```
terraform {
  backend "s3" {
    bucket = "my-s3-bucket"
    key    = "state-file-folder"
    region = "us-east-1"
  }
}
```

Para evitar almacenar información confidencial en los archivos de configuración de Terraform, los backends también admiten configuraciones parciales. En el ejemplo anterior, las credenciales necesarias para acceder al depósito no están presentes en la configuración. Las credenciales se pueden obtener a partir de variables de entorno o mediante otros medios, como AWS Secrets Manager. Para obtener más información, consulte Proteger los datos confidenciales mediante AWS Secrets Manager HashiCorp Terraform.

Un escenario de backend común es un backend local que se utiliza en su entorno local con fines de prueba. El archivo terraform.tfstate se incluye en el archivo.gitignore para que no se envíe al repositorio remoto. Luego, cada entorno de la canalización de CI/CD mantendría su propio backend. En este escenario, es posible que varios desarrolladores tengan acceso a este estado remoto, por lo que conviene proteger la integridad del archivo de estado. Si varias implementaciones se están ejecutando y actualizando el estado al mismo tiempo, el archivo de estado podría dañarse. Por este motivo, en situaciones con backends no locales, el archivo de estado suele estar bloqueado durante la implementación.

Comprensión de las fuentes de datos de Terraform

Es muy común que las pilas de despliegues se basen en datos de recursos previamente existentes. La mayoría de las herramientas de IaC permiten importar recursos que se crearon mediante algún otro proceso. Estos recursos importados suelen ser de solo lectura (aunque las funciones de IAM son una notable excepción) y se utilizan para acceder a los datos que necesitan los recursos de la pila. AWS CloudFormation permite importar recursos, pero esta idea puede explicarse mejor consultando la AWS Cloud Development Kit (AWS CDK).

AWS CDK Esto ayuda a los desarrolladores a utilizar los lenguajes de programación existentes para generar CloudFormation plantillas. El resultado final de una AWS CDK operación es un recurso importado en CloudFormation. Sin embargo, la sintaxis utilizada con el AWS CDK facilita la comparación con Terraform. A continuación, se muestra un ejemplo de cómo importar un recurso mediante. AWS CDK

```
const importedBucket: IBucket = Bucket.fromBucketAttributes(
    scope,
    "imported-bucket",
    {
        bucketName: "My_S3_Bucket"
    }
);
```

Por lo general, un recurso importado se crea llamando a un método estático de la misma clase que se utiliza para crear un nuevo recurso del mismo tipo. La llamada `new Bucket(...` crearía un recurso nuevo y la llamada `Bucket.fromBucketAttributes(...` importaría uno existente. Pasas un subconjunto de las propiedades del depósito a la función para que AWS CDK pueda encontrar el depósito correcto. Sin embargo, otra diferencia es que al crear un depósito nuevo se obtiene una instancia completa de la `Bucket` clase, con todas las propiedades y métodos disponibles en ella. Al importar el recurso`IBucket`, se devuelve un, que es un tipo que contiene solo las propiedades que `Bucket` debe tener. Aunque puede importar un recurso desde una pila externa, las opciones de lo que puede hacer con él son limitadas.

En Terraform, se logra un objetivo similar mediante el uso de fuentes de datos. La mayoría de los recursos definidos de Terraform tienen una fuente de datos adjunta disponible junto con ellos. El siguiente es un ejemplo de un recurso de bucket de Terraform S3 seguido de su fuente de datos correspondiente.

```
# S3 Bucket resource:
resource "aws_s3_bucket" "My_S3_Bucket" {
  bucket = "My_S3_Bucket"
}

# S3 Bucket data source:
data "aws_s3_bucket" "My_S3_Bucket" {
  bucket = "My_S3_Bucket"
}
```

La única diferencia entre estos dos elementos es el prefijo del nombre. Como se muestra en la documentación de una fuente de datos, hay menos parámetros disponibles que pueda pasar a una fuente de datos que a un recurso. Esto se debe a que el recurso usa esos parámetros para declarar todas las propiedades de un nuevo bucket de S3, mientras que la fuente de datos solo necesita la información suficiente para identificar e importar de forma exclusiva los datos de un recurso existente.

La similitud entre la sintaxis de un recurso de Terraform y una fuente de datos puede resultar práctica, pero también problemática. Es habitual que los desarrolladores novatos de Terraform utilicen accidentalmente una fuente de datos en lugar de un recurso en su configuración. Las fuentes de datos de Terraform siempre son de solo lectura. Puede utilizarlas en lugar del recurso correspondiente para las acciones de lectura (como proporcionar un nombre de identificación a otro recurso). Sin embargo, no se pueden usar para acciones de escritura, ya que modifican de manera fundamental algún aspecto del recurso subyacente. Por esta razón, puede pensar en una fuente de datos de Terraform como una versión clonada del recurso subyacente.

Al igual que en el ejemplo anterior de iBucket de AWS CDK, las fuentes de datos son útiles para escenarios de solo lectura. Si necesita obtener datos de un recurso existente pero no necesita mantener ese recurso dentro de su pila, utilice una fuente de datos. Un buen ejemplo de ello es cuando se crea una instancia de Amazon EC2 que utiliza la VPC predeterminada de la cuenta. Como esa VPC ya existe, todo lo que necesita hacer es extraer sus datos. En el siguiente ejemplo de código, se muestra cómo utilizar los datos para identificar la VPC de destino.

```
data "aws_vpc" "default" {
  default = true
}

resource "aws_instance" "instance1" {
  ami           = "ami-123456"
```

```
  instance_type = "t2.micro"
  subnet_id     = data.aws_vpc.default.main_route_table_id
}
```

Comprensión de las variables, los valores locales y los resultados de Terraform

Las variables mejoran la flexibilidad del código al permitir colocar marcadores de posición dentro de los bloques de código. Las variables pueden representar valores diferentes siempre que se reutilice el código. Terraform distingue sus tipos de variables por su alcance modular. Las variables de entrada son valores externos que se pueden inyectar en un módulo, los valores de salida son valores internos que se pueden compartir externamente y los valores locales siempre permanecen dentro de su ámbito original.

Variables

AWS CloudFormation usa parámetros para representar valores personalizados que se pueden configurar y restablecer de una implementación de pila a la siguiente. Del mismo modo, Terraform usa variables o variables de entrada. Las variables se pueden declarar en cualquier parte de un archivo de configuración de Terraform y, por lo general, se declaran con el tipo de datos requerido o el valor predeterminado. Las tres expresiones siguientes son declaraciones de variables de Terraform válidas.

```
variable "thing_i_made_up" {
  type = string
}

variable "random_number" {
  default = 5
}

variable "dogs" {
  type = list(object({
    name  = string
    breed = string
  }))

  default = [
    {
      name  = "Sparky",
      breed = "poodle"
    }
```

```
  ]
}
```

Para acceder a la raza de Sparky dentro de la configuración, debes usar la variable. `var.dogs[0].breed` Si una variable no tiene valores predeterminados y no está clasificada como anulable, se debe establecer el valor de la variable para cada implementación. De lo contrario, es opcional establecer un nuevo valor para la variable. En un módulo raíz, puede establecer los valores de las variables actuales en la línea de comandos, como variables de entorno o en el archivo terraform.tfvars. El siguiente ejemplo muestra cómo introducir valores de variables en el archivo terraform.tfvars, que se almacena en el directorio de nivel superior del módulo.

```
# terraform.tfvars
dogs = [
   {
      name  = "Sparky",
      breed = "poodle"
   },
   {
      name  = "Fluffy",
      breed = "chihuahua"
   }
]

random_number = 7

thing_i_made_up = "Kabibble"
```

El valor del archivo terraform.tfvars de este ejemplo anularía el valor predeterminado de la declaración de variables. `dogs` Si declaras variables dentro de un módulo secundario, puedes establecer los valores de las variables directamente dentro del bloque de declaración del módulo, como se muestra en el siguiente ejemplo.

```
module "my_custom_module" {
  source        = "modulesource/custom"
  version       = "0.0.1"
  random_number = 8
}
```

Algunos de los otros argumentos que puedes usar al declarar una variable son:

- `sensitive`— Configurarlo para `true` evitar que el valor de la variable quede expuesto en las salidas del proceso de Terraform.
- `nullable`— Si se configura de esta manera, se `true` permite que la variable no tenga ningún valor. Esto resulta práctico para las variables en las que no se establece un valor predeterminado.
- `description`— Añadir una descripción de la variable a los metadatos de la pila.
- `validation`— Establecer reglas de validación para la variable.

Uno de los aspectos más convenientes de las variables de Terraform es la posibilidad de añadir uno o más objetos de validación dentro de la declaración de variables. Puede usar los objetos de validación para agregar una condición que la variable deba cumplir o, de lo contrario, el despliegue fallará. También puede configurar un mensaje de error personalizado para que se muestre cada vez que se infrinja la condición.

Por ejemplo, estás configurando un archivo de configuración de Terraform que ejecutarán los miembros de tu equipo. Antes de implementar las pilas, un miembro del equipo debe crear un archivo terraform.tfvars para establecer un valor de configuración importante. Para recordárselo, puedes hacer algo como lo siguiente.

```
variable "important_config_setting" {
  type = string

  validation {
    condition     = length(var.important_config_setting) > 0
    error_message = "Don't forget to create the terraform.tfvars file!"
  }

  validation {
    condition     = substr(var.important_config_setting, 0, 7) == "prefix-"
    error_message = "Remember that the value always needs to start with 'prefix-'"
  }
}
```

Como se muestra en este ejemplo, puede establecer varias condiciones dentro de una sola variable. Terraform solo muestra los mensajes de error de las condiciones fallidas. De esta forma, puede hacer cumplir todo tipo de reglas sobre los valores de las variables. Si un valor variable provoca un fallo en la canalización, sabrá exactamente por qué.

Valores locales

Si hay algún valor dentro de un módulo al que desee asignar un alias, utilice la `locals` palabra clave en lugar de declarar una variable predeterminada que nunca se actualizará. Como su nombre indica, un `locals` bloque contiene términos que se refieren internamente a ese módulo específico. Si quieres transformar un valor de cadena, por ejemplo, añadiendo un prefijo a un valor de variable para usarlo en el nombre de un recurso, usar un valor local puede ser una buena solución. Un solo `locals` bloque puede declarar todos los valores locales del módulo, como se muestra en el siguiente ejemplo.

```
locals {
  moduleName    = "My Module"
  localConfigId = concat("prefix-", var.important_config_setting)
}
```

Recuerda que cuando accedes al valor, la `locals` palabra clave pasa a ser singular, por ejemplo`local.LocalConfigId`.

Valores de salida

Si las variables de entrada de Terraform son como CloudFormation parámetros, entonces se podría decir que los valores de salida de Terraform son como CloudFormation salidas. Ambos se utilizan para exponer los valores de una pila de despliegue. Sin embargo, dado que el módulo Terraform está más integrado en la estructura de la herramienta, los valores de salida de Terraform también se utilizan para exponer los valores de un módulo a un módulo principal o a otros módulos secundarios, incluso si esos módulos están todos dentro de la misma pila de despliegue. Si estás creando dos módulos personalizados y el primer módulo necesita acceder al valor de ID del segundo módulo, tendrás que añadir el siguiente `output` bloque al segundo módulo.

```
output "module_id" {
  value = local.module_id
}
Then in the first module you could use it like this:
module "first_module" {
  source = "path/to/first/module"
}

resource "example_resource" "example_resource_name" {
```

```
  module_id = module.first_module.module_id
}
```

Como los valores de salida de Terraform se pueden usar en la misma pila, también puedes usar el `sensitive` atributo en un `output` bloque para evitar que el valor se muestre en la salida de la pila. Además, un `output` bloque puede usar `precondition` bloques de la misma manera que las variables usan `validation` bloques: para garantizar que las variables sigan un conjunto determinado de reglas. Esto ayuda a garantizar que todos los valores de un módulo existan según lo esperado antes de continuar con la implementación.

```
output "important_config_setting" {
  value = var.important_config_setting

  precondition {
    condition     = length(var.important_config_setting) > 0
    error_message = "You forgot to create the terraform.tfvars file again."
  }
}
```

Comprensión de las funciones, expresiones y metaargumentos de Terraform

Una crítica a las herramientas de IaC que utilizan archivos de configuración declarativos en lugar de lenguajes de programación comunes es que dificultan la implementación de una lógica programática personalizada. En las configuraciones de Terraform, este problema se resuelve mediante funciones, expresiones y metaargumentos.

Funciones

Una de las grandes ventajas de usar código para aprovisionar la infraestructura es la capacidad de almacenar flujos de trabajo comunes y reutilizarlos una y otra vez, con frecuencia utilizando argumentos diferentes cada vez. Las funciones de Terraform son similares a las funciones AWS CloudFormation intrínsecas, aunque su sintaxis es más parecida a la forma en que se denominan las funciones en los lenguajes de programación. Es posible que ya hayas visto algunas funciones de Terraform, como substr, concat, length y base64decode, en los ejemplos de esta guía. Al igual que CloudFormation las funciones intrínsecas, Terraform tiene una serie de funciones integradas que están disponibles para su uso en sus configuraciones. Por ejemplo, si un atributo de recurso concreto ocupa un objeto JSON muy grande y sería ineficiente pegarlo directamente en el archivo, podrías poner el objeto en un archivo.json y usar las funciones de Terraform para acceder a él. En el siguiente ejemplo, la `file` función devuelve el contenido del archivo en forma de cadena y, a continuación, lo convierte en un tipo de objeto. `jsondecode`

```
resource "example_resource" "example_resource_name" {
  json_object = jsondecode(file("/path/to/file.json"))
}
```

Expressions

Terraform también admite expresiones condicionales, que son similares a CloudFormation `condition` las funciones, excepto que utilizan la sintaxis de operador ternario más tradicional. En el siguiente ejemplo, las dos expresiones devuelven exactamente el mismo resultado. El segundo ejemplo es lo que Terraform denomina expresión splat. El asterisco hace que Terraform recorra la lista y cree una nueva lista utilizando solo la `id` propiedad de cada elemento.

```
resource "example_resource" "example_resource_name" {
  boolean_value  = var.value ? true : false
  numeric_value  = var.value > 0 ? 1 : 0
  string_value   = var.value == "change_me" ? "New value" : var.value
  string_value_2 = var.value != "change_me" ? var.value : "New value"
}
There are two ways to express for loops in a Terraform configuration:
resource "example_resource" "example_resource_name" {
  list_value   = [for object in var.ids : object.id]
  list_value_2 = var.ids[*].id
}
```

Metaargumentos

En el ejemplo de código anterior, *`list_value`* y se *`list_value_2`* denominan argumentos. Es posible que ya estés familiarizado con algunos de estos metaargumentos. Terraform también tiene algunos metaargumentos, que actúan como argumentos pero con alguna funcionalidad adicional:

- El metaargumento depends_on es muy similar al atributo. CloudFormation DependsOn
- El metaargumento del proveedor le permite usar múltiples configuraciones de proveedores a la vez.
- El metaargumento del ciclo de vida permite personalizar la configuración de los recursos, de forma similar a como se hace en las políticas de eliminación y eliminación. CloudFormation

Otros metaargumentos permiten añadir funciones y expresiones directamente a un recurso. Por ejemplo, el metaargumento count es un mecanismo útil para crear varios recursos similares al mismo tiempo. En el siguiente ejemplo, se muestra cómo crear dos clústeres de Amazon Elastic Container Service (Amazon EKS) sin utilizar el `count` metaargumento.

```
resource "aws_eks_cluster" "example_0" {
  name     = "example_0"
  role_arn = aws_iam_role.cluster_role.arn
  vpc_config {
    endpoint_private_access = true
    endpoint_public_access  = true
    subnet_ids              = var.subnet_ids[0]
  }
}
```

```
resource "aws_eks_cluster" "example_1" {
  name     = "example_1"
  role_arn = aws_iam_role.cluster_role.arn
  vpc_config {
    endpoint_private_access = true
    endpoint_public_access  = true
    subnet_ids              = var.subnet_ids[1]
  }
}
```

El siguiente ejemplo muestra cómo utilizar el `count` metaargumento para crear dos clústeres de Amazon EKS.

```
resource "aws_eks_cluster" "clusters" {
  count    = 2
  name     = "cluster_${count.index}"
  role_arn = aws_iam_role.cluster_role.arn
  vpc_config {
    endpoint_private_access = true
    endpoint_public_access  = true
    subnet_ids              = var.subnet_ids[count.index]
  }
}
```

Para asignar a cada uno un nombre de unidad, puede acceder al índice de la lista dentro del bloque de recursos en. `count.index` Pero, ¿qué pasa si quieres crear varios recursos similares que sean un poco más complejos? Ahí es donde entra en juego el metaargumento for_each. El `for_each` metaargumento es muy similar `acount`, excepto que se pasa una lista o un objeto en lugar de un número. Terraform crea un nuevo recurso para cada miembro de la lista u objeto. Es similar a si lo configura`count = length(list)`, excepto que puedes acceder al contenido de la lista en lugar del índice del bucle.

Esto funciona tanto para una lista de elementos como para un solo objeto. El siguiente ejemplo crearía dos recursos que tienen `id-0` y `id-1` como sus IDs.

```
variable "ids" {
  default = [
    { id = "id-0" },
    { id = "id-1" },
  ]
```

```
}

resource "example_resource" "example_resource_name" {
  # If your list fails, you might have to call "toset" on it to convert it to a set
  for_each = toset(var.ids)
  id       = each.value
}
```

En el ejemplo siguiente se crearían también dos recursos, uno para Sparky, el caniche, y otro para Fluffy, el chihuahua.

```
variable "dogs" {
  default = {
    poodle    = "Sparky"
    chihuahua = "Fluffy"
  }
}

resource "example_resource" "example_resource_name" {
  for_each = var.dogs
  breed    = each.key
  name     = each.value
}
```

Del mismo modo que puedes acceder al índice de bucles en count mediante count.index, puedes acceder a la clave y al valor de cada elemento de un bucle for_each utilizando cada objeto. Como for_each recorre listas y objetos, puede resultar un poco confuso realizar un seguimiento de cada clave y valor. En la siguiente tabla se muestran las distintas formas en que se puede utilizar el metaargumento for_each y cómo se puede hacer referencia a los valores en cada iteración.

Ejemplo	Tipo de **`for_each`**	Primera iteración	Segunda iteración

A	`["poodle", "chihuahua"]`	`each.key = "poodle"`
		
		`each.value = null`
			`each.key = "chihuahua"`
			
			`each.value = null`

B	`[`	`each.key = {`	`each.key = {`

Ejemplo	Tipo de **`for_each`**	Primera iteración	Segunda iteración

	`{`
	
	`type =`
	` "poodle",`
	
	`name = "Sparky"`
	
	`},`
	
	`{`
	
	`type = "chihuahu`
	`a",`
	
	`name = "Fluffy"`
	
	`}`
	
	`]`
		`type = "poodle",`
		
		`name = "Sparky"`
		
		`}`
		
		`each.value =`
		` null`
			`type = "chihuahu`
			`a",`
			
			`name = "Fluffy"`
			
			`}`
			
			`each.value =`
			` null`

C	`{`
	
	`poodle =`
	` "Sparky",`
	
	`chihuahua =`
	` "Fluffy"`
	
	`}`
		`each.key =`
		` "poodle"`
		
		`each.value =`
		` "Sparky"`
			`each.key =`
			` "chihuahua"`
			
			`each.value =`
			` "Fluffy"`

Ejemplo	Tipo de **`for_each`**	Primera iteración	Segunda iteración

D	`{`
	
	`dogs = {`
	
	`poodle =`
	` "Sparky",`
	
	`chihuahua =`
	` "Fluffy"`
	
	`},`
	
	`cats = {`
	
	`persian =`
	` "Felix",`
	
	`burmese =`
	` "Morris"`
	
	`}`
	
	`}`
		`each.key = "dogs"`
		
		`each.value = {`
		
		`poodle =`
		` "Sparky",`
		
		`chihuahua =`
		` "Fluffy"`
		
		`}`
			`each.key = "cats"`
			
			`each.value = {`
			
			`persian =`
			` "Felix",`
			
			`burmese =`
			` "Morris"`
			
			`}`

Ejemplo	Tipo de **`for_each`**	Primera iteración	Segunda iteración

E	`{`
	
	`dogs = [`
	
	`{`
	
	`type = "poodle",`
	
	`name = "Sparky"`
	
	`},`
	
	`{`
	
	`type = "chihuahua",`
	
	`name = "Fluffy"`
	
	`}`
	
	`],`
	
	`cats = [`
	
	`{`
	
	`type = "persian",`
	
	`name = "Felix"`
	
	`},`
	
	`{`
	
	`type = "burmese",`
		`each.key = "dogs"`
		
		`each.value = [`
		
		`{`
		
		`type = "poodle",`
		
		`name = "Sparky"`
		
		`},`
		
		`{`
		
		`type = "chihuahua",`
		
		`name = "Fluffy"`
		
		`}`
		
		`]`
			`each.key = "cats"`
			
			`each.value = [`
			
			`{`
			
			`type = "persian",`
			
			`name = "Felix"`
			
			`},`
			
			`{`
			
			`type = "burmese",`
			
			`name = "Morris"`
			
			`}`
			
			`]`

Ejemplo	Tipo de **`for_each`**	Primera iteración	Segunda iteración

	`name = "Morris"`
	
	`}`
	
	`]`
	
	`}`		

Entonces, si `var.animals` fuera igual a la fila E, entonces podrías crear un recurso por animal usando el siguiente código.

```
resource "example_resource" "example_resource_name" {
  for_each = var.animals
  type     = each.key
  breeds   = each.value[*].type
  names    = each.value[*].name
}
```

Como alternativa, puedes crear dos recursos por animal mediante el siguiente código.

```
resource "example_resource" "example_resource_name" {
  for_each = var.animals.dogs
  type     = "dogs"
  breeds   = each.value.type
  names    = each.value.name
}

resource "example_resource" "example_resource_name" {
  for_each = var.animals.cats
  type     = "cats"
  breeds   = each.value.type
  names    = each.value.name
}
```

Preguntas frecuentes

¿Cuándo debo usar Terraform en lugar de? CloudFormation

En general, si sus cargas de trabajo se basan principalmente en AWS, AWS CloudFormation proporciona un nivel de soporte nativo que Terraform no puede igualar. Sin embargo, si tus cargas de trabajo incluyen bastantes procesos de terceros o están repartidas entre varios proveedores de servicios en la nube, Terraform es una herramienta que deberías considerar.

¿Cuándo debo usar el AWS CDK en lugar de? CloudFormation

Cuando usas el AWS Cloud Development Kit (AWS CDK), también estás usando CloudFormation. AWS CDK Le permite utilizar un lenguaje de programación común para generar CloudFormation plantillas. Si tiene experiencia en alguno de los lenguajes de programación AWS CDK compatibles, AWS CDK puede reducir el tiempo necesario para generar CloudFormation plantillas.

¿Existe una herramienta como la AWS CDK que genera las configuraciones de Terraform?

En comparación con el AWS CDK, el CDK para Terraform (CDKTF) utiliza la misma biblioteca de construcciones para aprovisionar los recursos y el mismo motor jsii para admitir varios lenguajes de programación. Puede usarla para generar configuraciones de Terraform de la misma manera que genera plantillas. AWS CDK CloudFormation

¿Cómo puedo obtener más información sobre Terraform?

Para obtener más información sobre los conceptos avanzados de Terraform, consulte la documentación de Terraform. También describe los componentes de los principales proveedores y módulos de código abierto.

Recursos relacionados

AWS documentación

- Documentación de AWS CDK
- Documentación de AWS CloudFormation
- Terraform: Más allá de lo básico con AWS (AWS entrada del blog)

Otros recursos

- Documentación de CDK para Terraform
- Documentación de Terraform

Apéndice: Ejemplos de acceso a los atributos de Terraform

Recurso

```
resource "aws_s3_bucket" "myS3Bucket" {
     bucket = "my-s3-bucket"
}

bucketName = aws_s3_bucket.myS3Bucket.bucket
```

Origen de datos

```
data "aws_s3_bucket" "myS3Bucket" {
     bucket = "my-s3-bucket"
}

bucketName = data.aws_s3_bucket.myS3Bucket.bucket
```

Módulo

```
module "eks" {
    source = "terraform-aws-modules/eks/aws"
    version = "20.2.1"
}

vpc_id = module.eks.vpc_id
```

Variable

```
variable "my_variable" = {
   default = "dog"
}

animalType = var.my_variable
```

Local

```
locals {
  type = "dog"
}

animalType = local.type
```

Historial de documentos

En la siguiente tabla, se describen cambios significativos de esta guía. Si quiere recibir notificaciones de futuras actualizaciones, puede suscribirse a las notificaciones RSS.

Cambio	Descripción	Fecha
Publicación inicial	—	29 de marzo de 2024

AWS Glosario de orientación prescriptiva

Los siguientes son términos de uso común en las estrategias, guías y patrones proporcionados por la Guía AWS prescriptiva. Para sugerir entradas, utilice el enlace Enviar comentarios al final del glosario.

Números

Las 7 R

Siete estrategias de migración comunes para trasladar aplicaciones a la nube. Estas estrategias se basan en las 5 R que Gartner identificó en 2011 y consisten en lo siguiente:

- Refactor/re-architect — Mueva una aplicación y modifique su arquitectura aprovechando al máximo las funciones nativas de la nube para mejorar la agilidad, el rendimiento y la escalabilidad. Por lo general, esto implica trasladar el sistema operativo y la base de datos. Ejemplo: migre su base de datos Oracle local a la PostgreSQL-Compatible edición Amazon Aurora.
- Redefinir la plataforma (transportar y redefinir): traslade una aplicación a la nube e introduzca algún nivel de optimización para aprovechar las capacidades de la nube. Ejemplo: Migrar la base de datos Oracle en las instalaciones a Amazon Relational Database Service (Amazon RDS) para Oracle en la nube de Nube de AWS.
- Recomprar (readquirir): cambie a un producto diferente, lo cual se suele llevar a cabo al pasar de una licencia tradicional a un modelo SaaS. Ejemplo: migre su sistema de gestión de relaciones con los clientes (CRM) a Salesforce.com.
- Volver a alojar (migrar mediante lift-and-shift): traslade una aplicación a la nube sin hacer cambios para aprovechar las funcionalidades de la nube. Ejemplo: Migrar la base de datos de Oracle en las instalaciones a Oracle en una instancia de EC2 en la Nube de AWS.
- Reubicar: (migrar el hipervisor mediante lift and shift): traslade la infraestructura a la nube sin comprar equipo nuevo, reescribir aplicaciones o modificar las operaciones actuales. Los servidores se migran de una plataforma en las instalaciones a un servicio en la nube para la misma plataforma. Ejemplo: migrar una Microsoft Hyper-V aplicación a AWS.
- Retener (revisitar): conserve las aplicaciones en el entorno de origen. Estas pueden incluir las aplicaciones que requieren una refactorización importante, que desee posponer para más adelante, y las aplicaciones heredadas que desee retener, ya que no hay ninguna justificación empresarial para migrarlas.

- Retirar: retire o elimine las aplicaciones que ya no sean necesarias en un entorno de origen.

A

A2A () Agent-to-Agent

Un protocolo completo para la colaboración entre agentes que facilita la delegación de tareas y la transferencia de estados.

ABAC

Consulte control de acceso basado en atributos.

servicios abstractos

Consulte servicios administrados.

ACID

Consulte atomicidad, consistencia, aislamiento, durabilidad.

migración activa-activa

Método de migración de bases de datos en el que las bases de datos de origen y destino se mantienen sincronizadas (mediante una herramienta de replicación bidireccional o mediante operaciones de escritura doble) y ambas bases de datos gestionan las transacciones de las aplicaciones conectadas durante la migración. Este método permite la migración en lotes pequeños y controlados, en lugar de requerir una transición única. Es más flexible, pero requiere más trabajo que una migración activa-pasiva.

migración activa-pasiva

Método de migración de bases de datos en el que las bases de datos de origen y destino se mantienen sincronizadas, pero solo la de origen gestiona las transacciones de las aplicaciones conectadas, mientras los datos se replican en la de destino. La base de datos de destino no acepta ninguna transacción durante la migración.

Agente

Un sistema de IA que puede razonar, planificar y tomar medidas de forma autónoma utilizando herramientas para alcanzar los objetivos.

Agent Ops

Prácticas operativas para crear, probar, implementar y ejecutar agentes de IA en producción a escala.

función de agregación

Función SQL que actúa en un grupo de filas y calcula un único valor de devolución para el grupo. Entre los ejemplos de funciones de agregación se incluyen `SUM` y `MAX`.

IA

Consulte inteligencia artificial.

AIOps

Consulte operaciones de inteligencia artificial

anonimización

El proceso de eliminar permanentemente la información personal de un conjunto de datos. La anonimización puede ayudar a proteger la privacidad personal. Los datos anonimizados ya no se consideran datos personales.

antipatrones

Una solución que se utiliza con frecuencia para un problema recurrente en el que la solución es contraproducente, ineficaz o menos eficaz que una alternativa.

control de aplicaciones

Enfoque de seguridad que permite usar de manera exclusiva aplicaciones aprobadas para ayudar a proteger un sistema contra el malware.

cartera de aplicaciones

Recopilación de información detallada sobre cada aplicación que utiliza una organización, incluido el costo de creación y mantenimiento de la aplicación y su valor empresarial. Esta información es clave para el proceso de detección y análisis de la cartera y ayuda a identificar y priorizar las aplicaciones que se van a migrar, modernizar y optimizar.

inteligencia artificial (IA)

El campo de la informática que se dedica al uso de tecnologías informáticas para realizar funciones cognitivas que suelen estar asociadas a los seres humanos, como el aprendizaje, la resolución de problemas y el reconocimiento de patrones. Para más información, consulte ¿Qué es la inteligencia artificial?

operaciones de inteligencia artificial (AIOps)

El proceso de utilizar técnicas de machine learning para resolver problemas operativos, reducir los incidentes operativos y la intervención humana, y mejorar la calidad del servicio. Para obtener más información sobre cómo se utiliza AIOps en la estrategia de migración de AWS , consulte la Guía de integración de operaciones.

cifrado asimétrico

Algoritmo de cifrado que utiliza un par de claves, una clave pública para el cifrado y una clave privada para el descifrado. Puede compartir la clave pública porque no se utiliza para el descifrado, pero el acceso a la clave privada debe estar sumamente restringido.

atomicidad, consistencia, aislamiento, durabilidad (ACID)

Conjunto de propiedades de software que garantizan la validez de los datos y la fiabilidad operativa de una base de datos, incluso en caso de errores, cortes de energía u otros problemas.

control de acceso basado en atributos (ABAC)

La práctica de crear permisos detallados basados en los atributos del usuario, como el departamento, el puesto de trabajo y el nombre del equipo. Para obtener más información, consulte ABAC AWS en la documentación AWS Identity and Access Management (IAM).

origen de datos fidedigno

Ubicación en la que se almacena la versión principal de los datos, que se considera la fuente de información más fiable. Puede copiar los datos del origen de datos autorizado a otras ubicaciones con el fin de procesarlos o modificarlos, por ejemplo, anonimizarlos, redactarlos o seudonimizarlos.

Zona de disponibilidad

Una ubicación distinta dentro de una Región de AWS que está aislada de los fallos en otras zonas de disponibilidad y que proporciona una conectividad de red económica y de baja latencia a otras zonas de disponibilidad de la misma región.

AWS Marco de adopción de la nube (AWS CAF)

Un marco de directrices y mejores prácticas AWS para ayudar a las organizaciones a desarrollar un plan eficiente y eficaz para migrar con éxito a la nube. AWS CAF organiza la orientación en seis áreas de enfoque denominadas perspectivas: negocios, personas, gobierno, plataforma, seguridad y operaciones. Las perspectivas empresariales, humanas y de gobernanza se centran en las habilidades y los procesos empresariales; las perspectivas de plataforma, seguridad y

operaciones se centran en las habilidades y los procesos técnicos. Por ejemplo, la perspectiva humana se dirige a las partes interesadas que se ocupan de los Recursos Humanos (RR. HH.), las funciones del personal y la administración de las personas. Desde esta perspectiva, AWS CAF proporciona orientación para el desarrollo, la formación y la comunicación de las personas a fin de preparar a la organización para una adopción exitosa de la nube. Para obtener más información, consulte la Página web de AWS CAF y el Documento técnico de AWS CAF.

AWS Marco de calificación de la carga de trabajo (AWS WQF)

Herramienta que evalúa las cargas de trabajo de migración de bases de datos, recomienda estrategias de migración y proporciona estimaciones de trabajo. AWS WQF se incluye con AWS Schema Conversion Tool ().AWS SCT Analiza los esquemas de bases de datos y los objetos de código, el código de las aplicaciones, las dependencias y las características de rendimiento y proporciona informes de evaluación.

B

bot malicioso

Bot destinado a causar interrupciones o daños a personas u organizaciones.

BCP

Consulte planificación de la continuidad del negocio.

gráfico de comportamiento

Una vista unificada e interactiva del comportamiento de los recursos y de las interacciones a lo largo del tiempo. Puede utilizar un gráfico de comportamiento con Amazon Detective para examinar los intentos de inicio de sesión fallidos, las llamadas sospechosas a la API y acciones similares. Para obtener más información, consulte Datos en un gráfico de comportamiento en la documentación de Detective.

sistema big-endian

Un sistema que almacena primero el byte más significativo. Consulte también endianidad.

clasificación binaria

Un proceso que predice un resultado binario (una de las dos clases posibles). Por ejemplo, es posible que su modelo de ML necesite predecir problemas como “¿Este correo electrónico es spam o no es spam?” o “¿Este producto es un libro o un automóvil?”.

filtro de floración

Estructura de datos probabilística y eficiente en términos de memoria que se utiliza para comprobar si un elemento es miembro de un conjunto.

blue/green despliegue

Estrategia de implementación en la que se crean dos entornos separados, pero idénticos. La versión actual de la aplicación se ejecuta en un entorno (azul) y la nueva versión de la aplicación se ejecuta en el otro entorno (verde). Esta estrategia lo ayuda a hacer reversiones rápidas con un impacto mínimo.

bot

Aplicación de software que ejecuta tareas automatizadas a través de Internet y simula la actividad o interacción humana. Algunos bots son útiles o beneficiosos, como los rastreadores web que indexan la información de Internet. Otros bots, conocidos como bots maliciosos, tienen como objetivo causar interrupciones o daños a personas u organizaciones.

botnet

Redes de bots infectadas por malware y que están bajo el control de una sola parte, conocida como pastor de bots u operador de bots. Las botnets son el mecanismo más conocido para escalar los bots y su impacto.

branch

Área contenida de un repositorio de código. La primera rama que se crea en un repositorio es la rama principal. Puede crear una rama nueva a partir de una rama existente y, a continuación, desarrollar características o corregir errores en la rama nueva. Una rama que se genera para crear una característica se denomina comúnmente rama de característica. Cuando la característica se encuentra lista para su lanzamiento, se vuelve a combinar la rama de característica con la rama principal. Para obtener más información, consulte Acerca de las sucursales (GitHub documentación).

acceso de emergencia

En circunstancias excepcionales y mediante un proceso aprobado, es una forma rápida de que un usuario pueda acceder a un Cuenta de AWS sitio al que normalmente no tiene permisos de acceso. Para obtener más información, consulte el indicador de implementación de procedimientos rompe-cristales en la AWS Well-Architected guía.

estrategia de implementación sobre infraestructura existente

La infraestructura existente en su entorno. Al adoptar una estrategia de implementación sobre infraestructura existente para una arquitectura de sistemas, se diseña la arquitectura en función de las limitaciones de los sistemas y la infraestructura actuales. Si está ampliando la infraestructura existente, puede combinar las estrategias de implementación sobre infraestructuras existentes y de implementación desde cero.

caché de búfer

El área de memoria donde se almacenan los datos a los que se accede con más frecuencia.

capacidad empresarial

Lo que hace una empresa para generar valor (por ejemplo, ventas, servicio al cliente o marketing). Las arquitecturas de microservicios y las decisiones de desarrollo pueden estar impulsadas por las capacidades empresariales. Para obtener más información, consulte la sección Organizado en torno a las capacidades empresariales del documento técnico Ejecutar microservicios en contenedores en AWS.

planificación de la continuidad del negocio (BCP)

Plan que aborda el posible impacto de un evento disruptivo, como una migración a gran escala en las operaciones y permite a la empresa reanudar las operaciones rápidamente.

C

CAF

Consulte AWS Cloud Adoption Framework.

implementación canario

Lanzamiento lento e incremental de una versión para los usuarios finales. Cuando tenga mayor confianza en la nueva versión, la implementa y reemplaza la versión actual en su totalidad.

CCoE

Consulte Centro de excelencia en la nube.

CDC

Consulte captura de datos de cambios.

captura de datos de cambio (CDC)

Proceso de seguimiento de los cambios en un origen de datos, como una tabla de base de datos, y registro de los metadatos relacionados con el cambio. Puede utilizar los CDC para diversos fines, como auditar o replicar los cambios en un sistema de destino para mantener la sincronización.

ingeniería del caos

Introducción intencionada de fallos o eventos disruptivos para poner a prueba la resiliencia de un sistema. Puedes usar AWS Fault Injection Service (AWS FIS) para realizar experimentos que estresen tus AWS cargas de trabajo y evalúen su respuesta.

CI/CD

Consulte integración continua y entrega continua.

clasificación

Un proceso de categorización que permite generar predicciones. Los modelos de ML para problemas de clasificación predicen un valor discreto. Los valores discretos siempre son distintos entre sí. Por ejemplo, es posible que un modelo necesite evaluar si hay o no un automóvil en una imagen.

Desarrollador ciudadano

Un usuario empresarial que crea aplicaciones de IA utilizando plataformas sin code/low código sin conocimientos técnicos especializados.

cifrado del cliente

Cifrado de datos localmente, antes de que el objetivo los Servicio de AWS reciba.

Centro de excelencia en la nube (CCoE)

Equipo multidisciplinario que impulsa los esfuerzos de adopción de la nube en toda la organización, incluido el desarrollo de las prácticas recomendadas en la nube, la movilización de recursos, el establecimiento de plazos de migración y la dirección de la organización durante las transformaciones a gran escala. Para obtener más información, consulte las publicaciones de CCoE en el blog de estrategia Nube de AWS empresarial.

computación en la nube

La tecnología en la nube que se utiliza normalmente para la administración de dispositivos de IoT y el almacenamiento de datos de forma remota. La computación en la nube suele estar relacionada con la tecnología de computación de periferia.

modelo operativo en la nube

En una organización de TI, el modelo operativo que se utiliza para crear, madurar y optimizar uno o más entornos de nube. Para obtener más información, consulte Creación de su modelo operativo de nube.

etapas de adopción de la nube

Las siguientes son las cuatro fases por las que suelen pasar las empresas cuando migran a la Nube de AWS:

- Proyecto: ejecución de algunos proyectos relacionados con la nube con fines de prueba de concepto y aprendizaje
- Fundamento: realización de inversiones fundamentales para escalar la adopción de la nube (p. ej., crear una zona de aterrizaje, definir un CCoE, establecer un modelo de operaciones)
- Migración: migración de aplicaciones individuales
- Re-invention — Optimizar los productos y servicios e innovar en la nube

Stephen Orban definió estas etapas en la entrada del blog The Journey Toward Cloud-First & the Stages of Adoption del blog Nube de AWS Enterprise Strategy. Para obtener información sobre su relación con la estrategia de AWS migración, consulte la guía de preparación para la migración.

CMDB

Consulte base de datos de administración de configuración.

repositorio de código

Una ubicación donde el código fuente y otros activos, como documentación, muestras y scripts, se almacenan y actualizan mediante procesos de control de versiones. Algunos repositorios en la nube comunes son GitHub o Bitbucket Cloud. Cada versión del código se denomina rama. En una estructura de microservicios, cada repositorio se encuentra dedicado a una única funcionalidad. Una sola CI/CD canalización puede utilizar varios repositorios.

caché en frío

Una caché de búfer que está vacía no está bien poblada o contiene datos obsoletos o irrelevantes. Esto afecta al rendimiento, ya que la instancia de la base de datos debe leer desde la memoria principal o el disco, lo que es más lento que leer desde la memoria caché del búfer.

datos fríos

Datos a los que se accede con poca frecuencia y que suelen ser históricos. Al consultar este tipo de datos, normalmente se aceptan consultas lentas. Trasladar estos datos a niveles o clases de almacenamiento de menor rendimiento y menos costosos puede reducir los costos.

visión artificial (CV)

Campo de la IA que utiliza el machine learning para analizar y extraer información de formatos visuales, como imágenes y videos digitales. Por ejemplo, Amazon SageMaker AI proporciona algoritmos de procesamiento de imágenes para CV.

deriva de configuración

En el caso de una carga de trabajo, un cambio en la configuración con respecto al estado esperado. Podría provocar que la carga de trabajo deje de cumplir las normas y, por lo general, es gradual e involuntaria.

base de datos de administración de configuración (CMDB)

Repositorio que almacena y administra información sobre una base de datos y su entorno de TI, incluidos los componentes de hardware y software y sus configuraciones. Por lo general, los datos de una CMDB se utilizan en la etapa de detección y análisis de la cartera de productos durante la migración.

paquete de conformidad

Un conjunto de AWS Config reglas y medidas correctivas que puede reunir para personalizar sus controles de conformidad y seguridad. Puede implementar un paquete de conformidad como una entidad única en una región Cuenta de AWS y, o en una organización, mediante una plantilla YAML. Para obtener más información, consulta los paquetes de conformidad en la documentación. AWS Config

integración y entrega continuas () CI/CD

El proceso de automatización de las etapas de origen, creación, prueba, puesta en escena y producción del proceso de publicación del software. CI/CD se describe comúnmente como una canalización. CI/CD puede ayudarlo a automatizar los procesos, mejorar la productividad, mejorar

la calidad del código y entregar más rápido. Para obtener más información, consulte Beneficios de la entrega continua. CD también puede significar implementación continua. Para obtener más información, consulte Entrega continua frente a implementación continua.

CV

Consulte visión artificial.

D

datos en reposo

Datos que están estacionarios en la red, como los datos que se encuentran almacenados.

clasificación de datos

Un proceso para identificar y clasificar los datos de su red en función de su importancia y sensibilidad. Es un componente fundamental de cualquier estrategia de administración de riesgos de ciberseguridad porque lo ayuda a determinar los controles de protección y retención adecuados para los datos. La clasificación de los datos es un componente del pilar de seguridad del AWS Well-Architected Framework. Para obtener más información, consulte Clasificación de datos.

deriva de datos

Una variación significativa entre los datos de producción y los datos que se utilizaron para entrenar un modelo de machine learning, o un cambio significativo en los datos de entrada a lo largo del tiempo. La deriva de datos puede reducir la calidad, la precisión y la imparcialidad generales de las predicciones de los modelos de machine learning.

datos en tránsito

Datos que se mueven de forma activa por la red, por ejemplo, entre los recursos de la red.

malla de datos

Marco de arquitectura que proporciona una propiedad de datos distribuida y descentralizada con una administración y una gobernanza centralizadas.

minimización de datos

El principio de recopilar y procesar solo los datos estrictamente necesarios. Practicar la minimización de los datos Nube de AWS puede reducir los riesgos de privacidad, los costos y la huella de carbono de la analítica.

perímetro de datos

Un conjunto de barreras preventivas en su AWS entorno que ayudan a garantizar que solo las identidades confiables accedan a los recursos confiables desde las redes esperadas. Para obtener más información, consulte Crear un perímetro de datos sobre. AWS

preprocesamiento de datos

Transformar los datos sin procesar en un formato que su modelo de ML pueda analizar fácilmente. El preprocesamiento de datos puede implicar eliminar determinadas columnas o filas y corregir los valores faltantes, incoherentes o duplicados.

procedencia de los datos

El proceso de rastrear el origen y el historial de los datos a lo largo de su ciclo de vida, por ejemplo, la forma en que se generaron, transmitieron y almacenaron los datos.

titular de los datos

Persona cuyos datos se recopilan y procesan.

almacenamiento de datos

Sistema de administración de datos que respalda la inteligencia empresarial, como los análisis. Los almacenes de datos suelen contener grandes cantidades de datos históricos y, por lo general, se utilizan para las consultas y los análisis.

lenguaje de definición de datos (DDL)

Instrucciones o comandos para crear o modificar la estructura de tablas y objetos de una base de datos.

lenguaje de manipulación de datos (DML)

Instrucciones o comandos para modificar (insertar, actualizar y eliminar) la información de una base de datos.

DDL

Consulte lenguaje de definición de bases de datos.

conjunto profundo

Combinar varios modelos de aprendizaje profundo para la predicción. Puede utilizar conjuntos profundos para obtener una predicción más precisa o para estimar la incertidumbre de las predicciones.

aprendizaje profundo

Un subcampo del ML que utiliza múltiples capas de redes neuronales artificiales para identificar el mapeo entre los datos de entrada y las variables objetivo de interés.

defensa en profundidad

Un enfoque de seguridad de la información en el que se distribuyen cuidadosamente una serie de mecanismos y controles de seguridad en una red informática para proteger la confidencialidad, la integridad y la disponibilidad de la red y de los datos que contiene. Al adoptar esta estrategia AWS, se añaden varios controles en diferentes capas de la AWS Organizations estructura para ayudar a proteger los recursos. Por ejemplo, un enfoque de defensa en profundidad podría combinar la autenticación multifactor, la segmentación de la red y el cifrado.

administrador delegado

En AWS Organizations, un servicio compatible puede registrar una cuenta de AWS miembro para administrar las cuentas de la organización y gestionar los permisos de ese servicio. Esta cuenta se denomina administrador delegado para ese servicio. Para obtener más información y una lista de servicios compatibles, consulte Servicios que funcionan con AWS Organizations en la documentación de AWS Organizations .

Implementación

El proceso de hacer que una aplicación, características nuevas o correcciones de código se encuentren disponibles en el entorno de destino. La implementación abarca implementar cambios en una base de código y, a continuación, crear y ejecutar esa base en los entornos de la aplicación.

entorno de desarrollo

Consulte entorno.

control de detección

Un control de seguridad que se ha diseñado para detectar, registrar y alertar después de que se produzca un evento. Estos controles son una segunda línea de defensa, ya que lo advierten sobre los eventos de seguridad que han eludido los controles preventivos establecidos. Para obtener más información, consulte Controles de detección en Implementación de controles de seguridad en AWS.

asignación de flujos de valor para el desarrollo (DVSM)

Proceso que se utiliza para identificar y priorizar las restricciones que afectan negativamente a la velocidad y la calidad en el ciclo de vida del desarrollo de software. DVSM amplía el proceso de asignación del flujo de valor diseñado originalmente para las prácticas de fabricación ajustada. Se centra en los pasos y los equipos necesarios para crear y transferir valor a través del proceso de desarrollo de software.

gemelo digital

Representación virtual de un sistema del mundo real, como un edificio, una fábrica, un equipo industrial o una línea de producción. Los gemelos digitales son compatibles con el mantenimiento predictivo, la supervisión remota y la optimización de la producción.

tabla de dimensiones

En un esquema en estrella, tabla más pequeña que contiene los atributos de datos sobre los datos cuantitativos en una tabla de hechos. Los atributos de la tabla de dimensiones suelen ser campos de texto o números discretos que se comportan como texto. Estos atributos se suelen utilizar para restringir consultas, filtrarlas y etiquetar los conjuntos de resultados.

desastre

Un evento que impide que una carga de trabajo o un sistema cumplan sus objetivos empresariales en su ubicación principal de implementación. Estos eventos pueden ser desastres naturales, fallos técnicos o el resultado de acciones humanas, como una configuración incorrecta involuntaria o un ataque de malware.

recuperación de desastres (DR)

Estrategia y proceso que utiliza para minimizar el tiempo de inactividad y la pérdida de datos a causa de un desastre. Para obtener más información, consulte Recuperación de cargas de trabajo ante desastres en AWS: Recuperación en la nube en el AWS Well-Architected marco.

DML

Consulte lenguaje de manipulación de bases de datos.

diseño basado en el dominio

Un enfoque para desarrollar un sistema de software complejo mediante la conexión de sus componentes a dominios en evolución, o a los objetivos empresariales principales, a los que sirve cada componente. Eric Evans introdujo este concepto en su libro Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Para

obtener información sobre cómo utilizar el diseño basado en dominios con el patrón de higos estranguladores, consulte Modernización gradual de los servicios web antiguos de ASP.NET Microsoft (ASMX) mediante contenedores y Amazon API Gateway.

DR

Consulte recuperación ante desastres.

Detección de desviaciones

Seguimiento de las desviaciones con respecto a una configuración con línea de base. Por ejemplo, puedes usarlo AWS CloudFormation para detectar desviaciones en los recursos del sistema o puedes usarlo AWS Control Tower para detectar cambios en tu landing zone que puedan afectar al cumplimiento de los requisitos de gobierno.

DVSM

Consulte asignación de flujos de valor para el desarrollo.

E

EDA

Consulte análisis de datos de tipo exploratorio.

EDI

Consulte intercambio electrónico de datos.

computación en la periferia

La tecnología que aumenta la potencia de cálculo de los dispositivos inteligentes en la periferia de una red de IoT. En comparación con la computación en la nube, la computación de periferia puede reducir la latencia de la comunicación y mejorar el tiempo de respuesta.

intercambio electrónico de datos (EDI)

Intercambio automatizado de documentos comerciales entre organizaciones. Para más información, consulte ¿Qué es el intercambio electrónico de datos?

cifrado

Proceso de computación que transforma datos de texto plano, que son legibles por humanos, en texto cifrado.

clave de cifrado

Cadena criptográfica de bits aleatorios que se genera mediante un algoritmo de cifrado. Las claves pueden variar en longitud y cada una se ha diseñado para ser impredecible y única.

endianidad

El orden en el que se almacenan los bytes en la memoria del ordenador. Big-endian los sistemas almacenan primero el byte más significativo. Little-endian los sistemas almacenan primero el byte menos significativo.

punto de conexión

Consulte punto de conexión de servicio.

servicio de punto de conexión

Servicio que puede alojar en una nube privada virtual (VPC) para compartir con otros usuarios. Puede crear un servicio de punto final con AWS PrivateLink entidades principales Cuentas de AWS o AWS Identity and Access Management (de IAM) y conceder permisos a ellas. Estas cuentas o entidades principales pueden conectarse a su servicio de punto de conexión de forma privada mediante la creación de puntos de conexión de VPC de interfaz. Para obtener más información, consulte Creación de un servicio de punto de conexión en la documentación de Amazon Virtual Private Cloud (Amazon VPC).

planificación de recursos empresariales (ERP)

Sistema que automatiza y administra los procesos empresariales clave (como la contabilidad, MES y la administración de proyectos) de una empresa.

cifrado de sobre

El proceso de cifrar una clave de cifrado con otra clave de cifrado. Para obtener más información, consulte el cifrado de sobres en la documentación de AWS Key Management Service (AWS KMS).

entorno

Una instancia de una aplicación en ejecución. Los siguientes son los tipos de entornos más comunes en la computación en la nube:

- entorno de desarrollo: instancia de una aplicación en ejecución que solo se encuentra disponible para el equipo principal responsable del mantenimiento de la aplicación. Los

entornos de desarrollo se utilizan para probar los cambios antes de promocionarlos a los entornos superiores. Este tipo de entorno a veces se denomina entorno de prueba.

- entornos inferiores: todos los entornos de desarrollo de una aplicación, como los que se utilizan para las compilaciones y pruebas iniciales.
- entorno de producción: instancia de una aplicación en ejecución a la que pueden acceder los usuarios finales. En un CI/CD proceso, el entorno de producción es el último entorno de implementación.
- entornos superiores: todos los entornos a los que pueden acceder usuarios que no sean del equipo de desarrollo principal. Esto puede incluir un entorno de producción, entornos de preproducción y entornos para las pruebas de aceptación por parte de los usuarios.

epopeya

En las metodologías ágiles, son categorías funcionales que ayudan a organizar y priorizar el trabajo. Las epopeyas brindan una descripción detallada de los requisitos y las tareas de implementación. Por ejemplo, las epopeyas AWS de seguridad de CAF incluyen la gestión de identidades y accesos, los controles de detección, la seguridad de la infraestructura, la protección de datos y la respuesta a incidentes. Para obtener más información sobre las epopeyas en la estrategia de migración de AWS , consulte la Guía de implementación del programa.

ERP

Consulte planificación de recursos empresariales.

análisis de datos de tipo exploratorio (EDA)

El proceso de analizar un conjunto de datos para comprender sus características principales. Se recopilan o agregan datos y, a continuación, se realizan las investigaciones iniciales para encontrar patrones, detectar anomalías y comprobar las suposiciones. El EDA se realiza mediante el cálculo de estadísticas resumidas y la creación de visualizaciones de datos.

F

tabla de hechos

Tabla central de un esquema en estrella. Almacena datos cuantitativos sobre operaciones empresariales. Por lo general, una tabla de hechos contiene dos tipos de columnas: las que contienen medidas y las que contienen una clave externa para una tabla de dimensiones.

Fail Fast

Filosofía que utiliza pruebas frecuentes e incrementales para reducir el ciclo de vida del desarrollo. Es una parte fundamental de los enfoques ágiles.

límite de aislamiento de errores

En el Nube de AWS, un límite, como una zona de disponibilidad Región de AWS, un plano de control o un plano de datos, que limita el efecto de una falla y ayuda a mejorar la resiliencia de las cargas de trabajo. Para más información, consulte AWS Fault Isolation Boundaries.

rama de característica

Consulte rama.

características

Los datos de entrada que se utilizan para hacer una predicción. Por ejemplo, en un contexto de fabricación, las características pueden ser imágenes que se capturan periódicamente desde la línea de fabricación.

importancia de las características

La importancia que tiene una característica para las predicciones de un modelo. Por lo general, esto se expresa como una puntuación numérica que se puede calcular mediante diversas técnicas, como las explicaciones aditivas de Shapley (SHAP) y los gradientes integrados. Para obtener más información, consulte Interpretabilidad del modelo de aprendizaje automático con. AWS

transformación de funciones

Optimizar los datos para el proceso de ML, lo que incluye enriquecer los datos con fuentes adicionales, escalar los valores o extraer varios conjuntos de información de un solo campo de datos. Esto permite que el modelo de ML se beneficie de los datos. Por ejemplo, si divide la fecha del “27 de mayo de 2021 00:15:37” en “jueves”, “mayo”, “2021” y “15”, puede ayudar al algoritmo de aprendizaje a aprender patrones matizados asociados a los diferentes componentes de los datos.

peticiones con pocos pasos

Proporcionar a un LLM una pequeña cantidad de ejemplos que demuestren la tarea y el resultado deseado antes de pedirle que lleve a cabo una tarea similar. Esta técnica es una aplicación del aprendizaje contextual, en el que los modelos aprenden a partir de ejemplos (tomas) integrados en las instrucciones. Few-shot Las indicaciones pueden ser eficaces para tareas que requieren

un formato, un razonamiento o un conocimiento del dominio específicos. Consulte también peticiones desde cero.

FGAC

Consulte control de acceso detallado.

control de acceso preciso (FGAC)

El uso de varias condiciones que tienen por objetivo permitir o denegar una solicitud de acceso.

migración relámpago

Método de migración de bases de datos que utiliza la replicación continua de datos mediante la captura de datos de cambio para migrar los datos en el menor tiempo posible, en lugar de utilizar un enfoque gradual. El objetivo es reducir al mínimo el tiempo de inactividad.

FM

Consulte modelo fundacional.

Modelo fundacional (FM)

Gran red neuronal de aprendizaje profundo que se entrenó con conjuntos de datos masivos de datos generalizados y no etiquetados. Los FM pueden hacer una amplia variedad de tareas generales, como comprender el lenguaje, generar texto e imágenes y conversar en lenguaje natural. Para más información, consulte ¿Qué son los modelos fundacionales?

Puerta de enlace FM

Un intermediario centralizado que controla y normaliza el acceso a los modelos básicos. También se conoce como puerta de enlace LLM.

G

IA generativa

Subconjunto de modelos de IA que se entrenaron con grandes cantidades de datos y que pueden utilizar una simple petición de texto para crear contenido y artefactos nuevos, como imágenes, videos, texto y audio. Para más información, consulte ¿Qué es la IA generativa?

bloqueo geográfico

Consulte restricciones geográficas.

restricciones geográficas (bloqueo geográfico)

En Amazon CloudFront, una opción para impedir que los usuarios de países específicos accedan a las distribuciones de contenido. Puede utilizar una lista de permitidos o bloqueados para especificar los países aprobados y prohibidos. Para obtener más información, consulta Restringir la distribución geográfica del contenido en la CloudFront documentación.

Flujo de trabajo de Gitflow

Un enfoque en el que los entornos inferiores y superiores utilizan diferentes ramas en un repositorio de código fuente. El flujo de trabajo de Gitflow se considera heredado, mientras que el flujo de trabajo basado en enlaces troncales es el enfoque moderno preferido.

imagen dorada

Instantánea de un sistema o software que se usa como plantilla para implementar nuevas instancias de ese sistema o software. Por ejemplo, en la fabricación, una imagen dorada se puede utilizar para aprovisionar software en varios dispositivos y ayuda a mejorar la velocidad, la escalabilidad y la productividad de las operaciones de fabricación de dispositivos.

estrategia de implementación desde cero

La ausencia de infraestructura existente en un entorno nuevo. Al adoptar una estrategia de implementación desde cero para una arquitectura de sistemas, puede seleccionar todas las tecnologías nuevas sin que estas deban ser compatibles con una infraestructura existente, lo que también se conoce como implementación sobre infraestructura existente. Si está ampliando la infraestructura existente, puede combinar las estrategias de implementación sobre infraestructuras existentes y de implementación desde cero.

barrera de protección

Una regla de alto nivel que ayuda a regular los recursos, las políticas y la conformidad en todas las unidades organizativas (OU). Las barreras de protección preventivas aplican políticas para garantizar la alineación con los estándares de conformidad. Se implementan mediante políticas de control de servicios y límites de permisos de IAM. Las barreras de protección de detección detectan las vulneraciones de las políticas y los problemas de conformidad, y generan alertas para su corrección. Se implementan mediante Amazon AWS Config AWS Security Hub CSPM GuardDuty AWS Trusted Advisor, Amazon Inspector y AWS Lambda cheques personalizados.

barandas (AI)

Mecanismos de seguridad que filtran, validan y restringen las entradas y salidas de los agentes para ayudar a garantizar un comportamiento responsable y seguro de la IA.

H

HA

Consulte alta disponibilidad.

migración heterogénea de bases de datos

Migración de la base de datos de origen a una base de datos de destino que utilice un motor de base de datos diferente (por ejemplo, de Oracle a Amazon Aurora). La migración heterogénea suele ser parte de un esfuerzo de rediseño de la arquitectura y convertir el esquema puede ser una tarea compleja. AWS ofrece AWS SCT, lo cual ayuda con las conversiones de esquemas.

alta disponibilidad (HA)

La capacidad de una carga de trabajo para funcionar de forma continua, sin intervención, en caso de desafíos o desastres. Los sistemas de alta disponibilidad están diseñados para realizar una conmutación por error automática, ofrecer un rendimiento de alta calidad de forma constante y gestionar diferentes cargas y fallos con un impacto mínimo en el rendimiento.

modernización histórica

Un enfoque utilizado para modernizar y actualizar los sistemas de tecnología operativa (TO) a fin de satisfacer mejor las necesidades de la industria manufacturera. Un histórico es un tipo de base de datos que se utiliza para recopilar y almacenar datos de diversas fuentes en una fábrica.

datos de reserva

Parte de los datos históricos etiquetados que se ocultan de un conjunto de datos que se utiliza para entrenar un modelo de machine learning. Puede utilizar los datos de reserva para evaluar el rendimiento del modelo mediante la comparación de las predicciones del modelo con los datos de reserva.

human-in-the-loop (HiTL)

Un patrón de flujo de trabajo en el que la ejecución de los agentes se detiene para su revisión y aprobación por parte de una persona en los puntos de decisión críticos.

migración homogénea de bases de datos

Migración de la base de datos de origen a una base de datos de destino que comparte el mismo motor de base de datos (por ejemplo, Microsoft SQL Server a Amazon RDS para SQL Server).

La migración homogénea suele formar parte de un esfuerzo para volver a alojar o redefinir la plataforma. Puede utilizar las utilidades de bases de datos nativas para migrar el esquema.

datos recientes

Datos a los que se accede con frecuencia, como datos en tiempo real o datos traslacionales recientes. Por lo general, estos datos requieren un nivel o una clase de almacenamiento de alto rendimiento para proporcionar respuestas rápidas a las consultas.

hotfix

Una solución urgente para un problema crítico en un entorno de producción. Debido a su urgencia, una revisión suele realizarse fuera del flujo de trabajo habitual de las DevOps versiones.

periodo de hiperatención

Periodo, inmediatamente después de la transición, durante el cual un equipo de migración administra y monitorea las aplicaciones migradas en la nube para solucionar cualquier problema. Por lo general, este periodo dura de 1 a 4 días. Al final del periodo de hiperatención, el equipo de migración suele transferir la responsabilidad de las aplicaciones al equipo de operaciones en la nube.

I

IaC

Consulte infraestructura como código.

políticas basadas en identidades

Política asociada a uno o más directores de IAM que define sus permisos en el entorno. Nube de AWS

aplicación inactiva

Aplicación que utiliza un promedio de CPU y memoria de entre 5 y 20 por ciento durante un periodo de 90 días. En un proyecto de migración, es habitual retirar estas aplicaciones o mantenerlas en las instalaciones.

IIoT

Consulte Internet de las cosas industrial.

infraestructura inmutable

Modelo que implementa una nueva infraestructura para las cargas de trabajo de producción en lugar de actualizar o modificar la infraestructura existente o aplicarle revisiones. Las infraestructuras inmutables son de manera intrínseca más coherentes, fiables y predecibles que las infraestructuras mutables. Para obtener más información, consulte las mejores prácticas del Framework para implementar con una infraestructura inmutable. AWS Well-Architected

VPC entrante (de entrada)

En una arquitectura de AWS cuentas múltiples, una VPC que acepta, inspecciona y enruta las conexiones de red desde fuera de una aplicación. La Arquitectura de referencia de seguridad de AWS recomienda configurar su cuenta de red con VPC entrantes, salientes y de inspección para proteger la interfaz bidireccional entre su aplicación e Internet en general.

migración gradual

Estrategia de transición en la que se migra la aplicación en partes pequeñas en lugar de realizar una transición única y completa. Por ejemplo, puede trasladar inicialmente solo unos pocos microservicios o usuarios al nuevo sistema. Tras comprobar que todo funciona correctamente, puede trasladar microservicios o usuarios adicionales de forma gradual hasta que pueda retirar su sistema heredado. Esta estrategia reduce los riesgos asociados a las grandes migraciones.

Industria 4.0

Un término que Klaus Schwab introdujo en 2016 para referirse a la modernización de los procesos de fabricación mediante avances en la conectividad, los datos en tiempo real, la automatización, el análisis y. AI/ML

infraestructura

Todos los recursos y activos que se encuentran en el entorno de una aplicación.

infraestructura como código (IaC)

Proceso de aprovisionamiento y administración de la infraestructura de una aplicación mediante un conjunto de archivos de configuración. La IaC se ha diseñado para ayudarlo a centralizar la administración de la infraestructura, estandarizar los recursos y escalar con rapidez a fin de que los entornos nuevos sean repetibles, fiables y consistentes.

Internet de las cosas industrial (IIoT)

El uso de sensores y dispositivos conectados a Internet en los sectores industriales, como el productivo, el eléctrico, el automotriz, el sanitario, el de las ciencias de la vida y el de la

agricultura. Para obtener más información, consulte Creación de una estrategia de transformación digital del Internet de las cosas industrial (IIoT).

VPC de inspección

En una arquitectura de AWS cuentas múltiples, una VPC centralizada que gestiona las inspecciones del tráfico de red entre las VPC (iguales o Regiones de AWS diferentes), Internet y las redes locales. La Arquitectura de referencia de seguridad de AWS recomienda configurar su cuenta de red con VPC entrantes, salientes y de inspección para proteger la interfaz bidireccional entre su aplicación e Internet en general.

Internet de las cosas (IoT)

Red de objetos físicos conectados con sensores o procesadores integrados que se comunican con otros dispositivos y sistemas a través de Internet o de una red de comunicación local. Para obtener más información, consulte ¿Qué es IoT?.

interpretabilidad

Característica de un modelo de machine learning que describe el grado en que un ser humano puede entender cómo las predicciones del modelo dependen de sus entradas. Para obtener más información, consulte Interpretabilidad del modelo de aprendizaje automático con. AWS

IoT

Consulte Internet de las cosas.

biblioteca de información de TI (ITIL)

Conjunto de prácticas recomendadas para ofrecer servicios de TI y alinearlos con los requisitos empresariales. La ITIL proporciona la base para la ITSM.

administración de servicios de TI (ITSM)

Actividades asociadas con el diseño, la implementación, la administración y el soporte de los servicios de TI para una organización. Para obtener información sobre la integración de las operaciones en la nube con las herramientas de ITSM, consulte la Guía de integración de operaciones.

ITIL

Consulte biblioteca de información de TI.

ITSM

Consulte administración de servicios de TI.

L

control de acceso basado en etiquetas (LBAC)

Una implementación del control de acceso obligatorio (MAC) en la que a los usuarios y a los propios datos se les asigna explícitamente un valor de etiqueta de seguridad. La intersección entre la etiqueta de seguridad del usuario y la etiqueta de seguridad de los datos determina qué filas y columnas puede ver el usuario.

zona de aterrizaje

Una landing zone es un AWS entorno multicuenta bien diseñado, escalable y seguro. Este es un punto de partida desde el cual las empresas pueden lanzar e implementar rápidamente cargas de trabajo y aplicaciones con confianza en su entorno de seguridad e infraestructura. Para obtener más información sobre las zonas de aterrizaje, consulte Configuración de un entorno de AWS seguro y escalable con varias cuentas.

modelo de lenguaje de gran tamaño (LLM)

Modelo de IA de aprendizaje profundo que se entrenó previamente con una gran cantidad de datos. Un LLM puede llevar a cabo varias tareas, como responder preguntas, resumir documentos, traducir textos a otros idiomas y completar oraciones. Para más información, consulte ¿Qué es un LLM (modelo de lenguaje de gran tamaño)?

migración grande

Migración de 300 servidores o más.

LBAC

Consulte control de acceso basado en etiquetas.

privilegio mínimo

La práctica recomendada de seguridad que consiste en conceder los permisos mínimos necesarios para realizar una tarea. Para obtener más información, consulte Aplicar permisos de privilegio mínimo en la documentación de IAM.

migrar mediante lift-and-shift

Consulte Las 7 R.

sistema little-endian

Un sistema que almacena primero el byte menos significativo. Consulte también endianidad.

LLM

Consulte modelo de lenguaje de gran tamaño.

entornos inferiores

Consulte entorno.

M

machine learning (ML)

Un tipo de inteligencia artificial que utiliza algoritmos y técnicas para el reconocimiento y el aprendizaje de patrones. El ML analiza y aprende de los datos registrados, como los datos del Internet de las cosas (IoT), para generar un modelo estadístico basado en patrones. Para más información, consulte Machine learning.

rama principal

Consulte rama.

malware

Software diseñado para comprometer la seguridad o la privacidad de la computadora. El malware podría interrumpir los sistemas informáticos, filtrar información confidencial u obtener acceso no autorizado. Algunos ejemplos de malware son los virus, los gusanos, el ransomware, los troyanos, el spyware y los registradores de pulsaciones de teclas.

Servicios administrados

Servicios de AWS en el que AWS opera la capa de infraestructura, el sistema operativo y las plataformas, y se accede a los puntos finales para almacenar y recuperar datos. Amazon Simple Storage Service (Amazon S3) y Amazon DynamoDB son ejemplos de servicios administrados. También se conocen como servicios abstractos.

sistema de ejecución de fabricación (MES)

Sistema de software para seguir, supervisar, documentar y controlar los procesos de producción que convierten las materias primas en productos acabados en la zona de producción.

MAP

Consulte Programa de aceleración de la migración.

MCP

Consulte Model Context Protocol.

Protocolo de contexto para modelos (MCP)

Un protocolo sin estado para la comunicación entre el agente y la herramienta.

Servidor MCP

Un servicio que expone una o más herramientas a través del protocolo Model Context.

mecanismo

Proceso completo mediante el que se crea una herramienta, se impulsa su adopción y, a continuación, se inspeccionan los resultados para hacer ajustes. Un mecanismo es un ciclo que se refuerza y mejora por sí mismo a medida que funciona. Para obtener más información, consulte Creación de mecanismos en el AWS Well-Architected marco.

cuenta de miembro

Todas las Cuentas de AWS demás cuentas, excepto la de administración, que forman parte de una organización AWS Organizations. Una cuenta no puede pertenecer a más de una organización a la vez.

MES

Consulte sistema de ejecución de fabricación.

Message Queuing Telemetry Transport (MQTT)

Un protocolo de comunicación ligero de máquina a máquina (M2M), basado en el publish/subscribepatrón, para dispositivos de IoT con recursos limitados.

microservicio

Un servicio pequeño e independiente que se comunica a través de API bien definidas y que, por lo general, es propiedad de equipos pequeños e independientes. Por ejemplo, un sistema de seguros puede incluir microservicios que se adapten a las capacidades empresariales, como las de ventas o marketing, o a subdominios, como las de compras, reclamaciones o análisis. Los beneficios de los microservicios incluyen la agilidad, la escalabilidad flexible, la facilidad de implementación, el código reutilizable y la resiliencia. Para obtener más información, consulte Integrar microservicios mediante servicios sin servidor. AWS

arquitectura de microservicios

Un enfoque para crear una aplicación con componentes independientes que ejecutan cada proceso de la aplicación como un microservicio. Estos microservicios se comunican a través de una interfaz bien definida mediante API ligeras. Cada microservicio de esta arquitectura se puede actualizar, implementar y escalar para satisfacer la demanda de funciones específicas de una aplicación. Para obtener más información, consulte Implementación de microservicios en. AWS

Programa de aceleración de la migración (MAP)

Un AWS programa que proporciona soporte de consultoría, formación y servicios para ayudar a las organizaciones a crear una base operativa sólida para migrar a la nube y para ayudar a compensar el costo inicial de las migraciones. El MAP incluye una metodología de migración para ejecutar las migraciones antiguas de forma metódica y un conjunto de herramientas para automatizar y acelerar los escenarios de migración más comunes.

migración a escala

Proceso de transferencia de la mayoría de la cartera de aplicaciones a la nube en oleadas, con más aplicaciones desplazadas a un ritmo más rápido en cada oleada. En esta fase, se utilizan las prácticas recomendadas y las lecciones aprendidas en las fases anteriores para implementar una fábrica de migración de equipos, herramientas y procesos con el fin de agilizar la migración de las cargas de trabajo mediante la automatización y la entrega ágil. Esta es la tercera fase de la estrategia de migración de AWS.

fábrica de migración

Cross-functional equipos que agilizan la migración de las cargas de trabajo mediante enfoques ágiles y automatizados. Los equipos de las fábricas de migración suelen estar compuestos por analistas y propietarios de operaciones, ingenieros de migración, desarrolladores y DevOps profesionales que trabajan a pasos agigantados. Entre el 20 y el 50 por ciento de la cartera de aplicaciones empresariales se compone de patrones repetidos que pueden optimizarse mediante un enfoque de fábrica. Para obtener más información, consulte la discusión sobre las fábricas de migración y la Guía de fábricas de migración a la nube en este contenido.

metadatos de migración

Información sobre la aplicación y el servidor que se necesita para completar la migración. Cada patrón de migración requiere un conjunto diferente de metadatos de migración. Algunos ejemplos de metadatos de migración son la subred de destino, el grupo de seguridad y AWS la cuenta.

patrón de migración

Tarea de migración repetible que detalla la estrategia de migración, el destino de la migración y la aplicación o el servicio de migración utilizados. Ejemplo: rehospede la migración a Amazon EC2 AWS con Application Migration Service.

Migration Portfolio Assessment (MPA)

Herramienta en línea que proporciona información a fin de validar los argumentos comerciales necesarios para migrar a la Nube de AWS. La MPA ofrece una evaluación detallada de la cartera (adecuación del tamaño de los servidores, precios, comparaciones del costo total de propiedad, análisis de los costos de migración), así como una planificación de la migración (análisis y recopilación de datos de aplicaciones, agrupación de aplicaciones, priorización de la migración y planificación de oleadas). La herramienta MPA (requiere iniciar sesión) está disponible de forma gratuita para todos los AWS consultores y consultores de los socios de APN.

Evaluación de la preparación para la migración (MRA)

Proceso que consiste en obtener información sobre el estado de preparación de una organización para la nube, identificar sus puntos fuertes y débiles y elaborar un plan de acción para cerrar las brechas identificadas mediante el AWS CAF. Para obtener más información, consulte la Guía de preparación para la migración. La MRA es la primera fase de la estrategia de migración de AWS.

estrategia de migración

Enfoque utilizado para migrar una carga de trabajo a la Nube de AWS. Para más información, consulte la entrada Las 7 R de este glosario y también Mobilize your organization to accelerate large-scale migrations.

ML

Consulte machine learning.

modernización

Transformar una aplicación obsoleta (antigua o monolítica) y su infraestructura en un sistema ágil, elástico y de alta disponibilidad en la nube para reducir los gastos, aumentar la eficiencia y aprovechar las innovaciones. Para más información, consulte Strategy for modernizing applications in the Nube de AWS.

evaluación de la preparación para la modernización

Evaluación que ayuda a determinar la preparación para la modernización de las aplicaciones de una organización; identifica los beneficios, los riesgos y las dependencias; y determina qué

tan bien la organización puede soportar el estado futuro de esas aplicaciones. El resultado de la evaluación es un esquema de la arquitectura objetivo, una hoja de ruta que detalla las fases de desarrollo y los hitos del proceso de modernización y un plan de acción para abordar las brechas identificadas. Para más información, consulte Evaluating modernization readiness for applications in the Nube de AWS.

aplicaciones monolíticas (monolitos)

Aplicaciones que se ejecutan como un único servicio con procesos estrechamente acoplados. Las aplicaciones monolíticas presentan varios inconvenientes. Si una característica de la aplicación experimenta un aumento en la demanda, se debe escalar toda la arquitectura. Agregar o mejorar las características de una aplicación monolítica también se vuelve más complejo a medida que crece la base de código. Para solucionar problemas con la aplicación, puede utilizar una arquitectura de microservicios. Para obtener más información, consulte Descomposición de monolitos en microservicios.

MPA

Consulte Migration Portfolio Assessment.

MQTT

Consulte Message Queuing Telemetry Transport.

clasificación multiclase

Un proceso que ayuda a generar predicciones para varias clases (predice uno de más de dos resultados). Por ejemplo, un modelo de ML podría preguntar “¿Este producto es un libro, un automóvil o un teléfono?” o “¿Qué categoría de productos es más interesante para este cliente?”.

infraestructura mutable

Modelo que actualiza y modifica la infraestructura actual para las cargas de trabajo de producción. Para mejorar la coherencia, la confiabilidad y la previsibilidad, el AWS Well-Architected Marco recomienda el uso de una infraestructura inmutable como práctica recomendada.

O

OAC

Consulte control de acceso de origen.

OAI

Consulte identidad de acceso de origen.

OCM

Consulte administración del cambio organizacional.

migración fuera de línea

Método de migración en el que la carga de trabajo de origen se elimina durante el proceso de migración. Este método implica un tiempo de inactividad prolongado y, por lo general, se utiliza para cargas de trabajo pequeñas y no críticas.

OI

Consulte integración de operaciones.

OLA

Consulte acuerdo de nivel operativo.

migración en línea

Método de migración en el que la carga de trabajo de origen se copia al sistema de destino sin que se desconecte. Las aplicaciones que están conectadas a la carga de trabajo pueden seguir funcionando durante la migración. Este método implica un tiempo de inactividad nulo o mínimo y, por lo general, se utiliza para cargas de trabajo de producción críticas.

OPC-UA

Consulte Open Process Communications: arquitectura unificada.

Comunicaciones de proceso abierto: arquitectura unificada () OPC-UA

Un protocolo de comunicación de máquina a máquina (M2M) para la automatización industrial. OPC-UA proporciona un estándar de interoperabilidad con esquemas de cifrado, autenticación y autorización de datos.

acuerdo de nivel operativo (OLA)

Acuerdo que aclara lo que los grupos de TI operativos se comprometen a ofrecerse entre sí, para respaldar un acuerdo de nivel de servicio (SLA).

revisión de la preparación operativa (ORR)

Lista de comprobación de preguntas y prácticas recomendadas asociadas que son útiles para comprender, evaluar, prevenir o reducir el alcance de los incidentes y posibles errores. Para

obtener más información, consulte las revisiones de preparación operativa (ORR) en el AWS Well-Architected marco.

tecnología operativa (TO)

Sistemas de hardware y software que funcionan con el entorno físico para controlar las operaciones, los equipos y la infraestructura industriales. En el sector de la fabricación, la integración de los sistemas de TO y tecnología de la información (TI) es un enfoque clave para las transformaciones de la industria 4.0.

integración de operaciones (OI)

Proceso de modernización de las operaciones en la nube, que implica la planificación de la preparación, la automatización y la integración. Para obtener más información, consulte la Guía de integración de las operaciones.

registro de seguimiento organizativo

Un registro creado por y AWS CloudTrail que registra todos los eventos Cuentas de AWS de una organización AWS Organizations. Este registro de seguimiento se crea en cada Cuenta de AWS que forma parte de la organización y realiza un seguimiento de la actividad en cada cuenta. Para obtener más información, consulte Crear un registro para una organización en la CloudTrail documentación.

administración del cambio organizacional (OCM)

Marco para administrar las transformaciones empresariales importantes y disruptivas desde la perspectiva de las personas, la cultura y el liderazgo. La OCM ayuda a las empresas a prepararse para nuevos sistemas y estrategias y a realizar la transición a ellos, al acelerar la adopción de cambios, abordar los problemas de transición e impulsar cambios culturales y organizacionales. En la estrategia de AWS migración, este marco se denomina aceleración de personal, debido a la velocidad de cambio que requieren los proyectos de adopción de la nube. Para obtener más información, consulte la Guía de OCM.

control de acceso de origen (OAC)

En CloudFront, una opción mejorada para restringir el acceso y proteger el contenido del Amazon Simple Storage Service (Amazon S3). El OAC admite todos los buckets de S3 Regiones de AWS, el cifrado del lado del servidor con AWS KMS (SSE-KMS) y `DELETE` las solicitudes `PUT` y dinámicas al bucket de S3.

identidad de acceso de origen (OAI)

En CloudFront, una opción para restringir el acceso y proteger el contenido de Amazon S3. Cuando utiliza OAI, CloudFront crea un principal con el que Amazon S3 puede autenticarse. Los directores autenticados solo pueden acceder al contenido de un bucket de S3 a través de una distribución específica. CloudFront Consulte también el OAC, que proporciona un control de acceso más detallado y mejorado.

ORR

Consulte revisión de la preparación operativa.

OT

Consulte tecnología operativa.

VPC saliente (de salida)

En una arquitectura de AWS cuentas múltiples, una VPC que gestiona las conexiones de red que se inician desde una aplicación. La Arquitectura de referencia de seguridad de AWS recomienda configurar su cuenta de red con VPC entrantes, salientes y de inspección para proteger la interfaz bidireccional entre su aplicación e Internet en general.

P

límite de permisos

Una política de administración de IAM que se adjunta a las entidades principales de IAM para establecer los permisos máximos que puede tener el usuario o el rol. Para obtener más información, consulte Límites de permisos en la documentación de IAM.

información de identificación personal (PII)

Información que, vista directamente o combinada con otros datos relacionados, puede utilizarse para deducir de manera razonable la identidad de una persona. Algunos ejemplos de información de identificación personal son los nombres, las direcciones y la información de contacto.

PII

Consulte información de identificación personal.

manual de estrategias

Conjunto de pasos predefinidos que capturan el trabajo asociado a las migraciones, como la entrega de las funciones de operaciones principales en la nube. Un manual puede adoptar la forma de scripts, manuales de procedimientos automatizados o resúmenes de los procesos o pasos necesarios para operar un entorno modernizado.

PLC

Consulte controlador lógico programable.

PLM

Consulte administración del ciclo de vida del producto.

policy

Objeto que puede definir permisos (consulte política basada en identidad), especificar las condiciones de acceso (consulte política basada en recursos) o definir los permisos máximos para todas las cuentas de una organización de AWS Organizations (consulte política de control de servicio).

persistencia políglota

Elegir de forma independiente la tecnología de almacenamiento de datos de un microservicio en función de los patrones de acceso a los datos y otros requisitos. Si sus microservicios tienen la misma tecnología de almacenamiento de datos, pueden enfrentarse a desafíos de implementación o experimentar un rendimiento deficiente. Los microservicios se implementan más fácilmente y logran un mejor rendimiento y escalabilidad si utilizan el almacén de datos que mejor se adapte a sus necesidades.

evaluación de cartera

Proceso de detección, análisis y priorización de la cartera de aplicaciones para planificar la migración. Para obtener más información, consulte la Evaluación de la preparación para la migración.

predicate

Condición de consulta que devuelve `true` o `false`. En general, se encuentra en una cláusula `WHERE`.

inserción de predicados

Técnica de optimización de consultas en bases de datos que filtra los datos de la consulta antes de transferirlos. Esta técnica reduce la cantidad de datos de la base de datos relacional que se tienen que recuperar y procesar. Además, mejora el rendimiento de las consultas.

control preventivo

Un control de seguridad diseñado para evitar que ocurra un evento. Estos controles son la primera línea de defensa para evitar el acceso no autorizado o los cambios no deseados en la red. Para obtener más información, consulte Controles preventivos en Implementación de controles de seguridad en AWS.

entidad principal

Una entidad AWS que puede realizar acciones y acceder a los recursos. Esta entidad suele ser un usuario raíz para un Cuenta de AWS rol de IAM o un usuario. Para obtener más información, consulte Entidad principal en Términos y conceptos de roles en la documentación de IAM.

Privacidad desde el diseño

Enfoque de ingeniería de sistemas que tiene en cuenta la privacidad durante todo el proceso de desarrollo.

zonas alojadas privadas

Contenedor que aloja información acerca de cómo desea que responda Amazon Route 53 a las consultas de DNS de un dominio y sus subdominios en una o varias VPC. Para obtener más información, consulte Uso de zonas alojadas privadas en la documentación de Route 53.

control proactivo

Control de seguridad que se diseñó para evitar la implementación de recursos que no cumplan con la normativa. Estos controles analizan los recursos antes de aprovisionarlos. Si el recurso no cumple con los requisitos del control, no se aprovisiona. Para obtener más información, consulte la guía de referencia de controles en la AWS Control Tower documentación y consulte Controles proactivos en Implementación de controles de seguridad en AWS.

administración del ciclo de vida del producto (PLM)

Administración de los datos y los procesos de un producto a lo largo de todo su ciclo de vida, desde el diseño, el desarrollo y el lanzamiento, pasando por el crecimiento y la madurez, hasta la reducción de su uso y su retirada.

entorno de producción

Consulte entorno.

controlador lógico programable (PLC)

En el sector de la fabricación, computadora adaptable y altamente fiable que supervisa las máquinas y automatiza los procesos de fabricación.

encadenamiento de peticiones

Uso de la salida de una petición de LLM como entrada para la siguiente petición a fin de generar mejores respuestas. Esta técnica se utiliza para dividir una tarea compleja en tareas secundarias o para refinar o ampliar de forma iterativa una respuesta preliminar. Ayuda a mejorar la precisión y la relevancia de las respuestas de un modelo y permite obtener resultados más detallados y personalizados.

seudonimización

El proceso de reemplazar los identificadores personales de un conjunto de datos por valores de marcadores de posición. La seudonimización puede ayudar a proteger la privacidad personal. Los datos seudonimizados siguen considerándose datos personales.

publish/subscribe (pub/sub)

Patrón que permite establecer comunicaciones asíncronas entre microservicios para mejorar la escalabilidad y la capacidad de respuesta. Por ejemplo, en un MES basado en microservicios, un microservicio puede publicar mensajes de eventos en un canal al que se pueden suscribir otros microservicios. El sistema puede agregar nuevos microservicios sin cambiar el servicio de publicación.

Q

plan de consulta

Serie de pasos, como instrucciones, que se utilizan para acceder a los datos de un sistema de base de datos relacional SQL.

regresión del plan de consulta

El optimizador de servicios de la base de datos elige un plan menos óptimo que antes de un cambio determinado en el entorno de la base de datos. Los cambios en estadísticas,

restricciones, configuración del entorno, enlaces de parámetros de consultas y actualizaciones del motor de base de datos PostgreSQL pueden provocar una regresión del plan.

R

Matriz RACI

Consulte responsable, fiable, consultada e informada (RACI).

RAG

Consulte generación aumentada por recuperación.

ransomware

Software malicioso que se ha diseñado para bloquear el acceso a un sistema informático o a los datos hasta que se efectúe un pago.

Matriz RASCI

Consulte responsable, fiable, consultada e informada (RACI).

RCAC

Consulte control de acceso por filas y columnas.

réplica de lectura

Una copia de una base de datos que se utiliza con fines de solo lectura. Puede enrutar las consultas a la réplica de lectura para reducir la carga en la base de datos principal.

rediseñar

Consulte Las 7 R.

objetivo de punto de recuperación (RPO)

La cantidad de tiempo máximo aceptable desde el último punto de recuperación de datos. Esto determina qué se considera una pérdida de datos aceptable entre el último punto de recuperación y la interrupción del servicio.

objetivo de tiempo de recuperación (RTO)

La demora máxima aceptable entre la interrupción del servicio y el restablecimiento del servicio.

refactorizar

Consulte Las 7 R.

Region

Conjunto de AWS recursos en un área geográfica. Cada uno Región de AWS está aislado e independiente de los demás para proporcionar tolerancia a las fallas, estabilidad y resiliencia. Para más información, consulte Specify which Regiones de AWS your account can use.

regresión

Una técnica de ML que predice un valor numérico. Por ejemplo, para resolver el problema de "¿A qué precio se venderá esta casa?", un modelo de ML podría utilizar un modelo de regresión lineal para predecir el precio de venta de una vivienda en función de datos conocidos sobre ella (por ejemplo, los metros cuadrados).

volver a alojar

Consulte Las 7 R.

versión

En un proceso de implementación, el acto de promover cambios en un entorno de producción.

reubicar

Consulte Las 7 R.

redefinir la plataforma

Consulte Las 7 R.

recomprar

Consulte Las 7 R.

resiliencia

Capacidad de una aplicación para resistir interrupciones o recuperarse de ellas. Al planificar la resiliencia en la Nube de AWS, la alta disponibilidad y la recuperación ante desastres son consideraciones comunes. Para más información, consulte Resiliencia en la Nube de AWS.

política basada en recursos

Una política asociada a un recurso, como un bucket de Amazon S3, un punto de conexión o una clave de cifrado. Este tipo de política especifica a qué entidades principales se les permite el acceso, las acciones compatibles y cualquier otra condición que deba cumplirse.

matriz responsable, confiable, consultada e informada (RACI)

Una matriz que define las funciones y responsabilidades de todas las partes involucradas en las actividades de migración y las operaciones de la nube. El nombre de la matriz se deriva de los tipos de responsabilidad definidos en la matriz: responsable (R), contable (A), consultado (C) e informado (I). El tipo de soporte (S) es opcional. Si incluye el soporte, la matriz se denomina matriz RASCI y, si la excluye, se denomina matriz RACI.

control receptivo

Un control de seguridad que se ha diseñado para corregir los eventos adversos o las desviaciones con respecto a su base de seguridad. Para obtener más información, consulte Controles receptivos en Implementación de controles de seguridad en AWS.

retain

Consulte Las 7 R.

retirar

Consulte Las 7 R.

Generación aumentada de recuperación (RAG)

Tecnología de IA generativa mediante la que un LLM hace referencia a un origen de datos autorizado que se encuentra fuera de sus orígenes de datos de entrenamiento antes de generar una respuesta. Por ejemplo, un modelo de RAG podría hacer una búsqueda semántica en la base de conocimientos o en los datos personalizados de una organización. Para más información, consulte ¿Qué es RAG (generación aumentada por recuperación)?

rotación

Proceso mediante el que periódicamente se actualiza un secreto para que resulte más difícil que un atacante pueda acceder a las credenciales.

control de acceso por filas y columnas (RCAC)

El uso de expresiones SQL básicas y flexibles que tienen reglas de acceso definidas. El RCAC consta de permisos de fila y máscaras de columnas.

RPO

Consulte objetivo de punto de recuperación.

RTO

Consulte objetivo de tiempo de recuperación.

manual de procedimientos

Conjunto de procedimientos manuales o automatizados necesarios para realizar una tarea específica. Por lo general, se diseñan para agilizar las operaciones o los procedimientos repetitivos con altas tasas de error.

S

SAML 2.0

Un estándar abierto que utilizan muchos proveedores de identidad (IdPs). Esta función permite el inicio de sesión único (SSO) federado, de modo que los usuarios pueden iniciar sesión Consola de administración de AWS o llamar a las operaciones de la AWS API sin tener que crear un usuario en IAM para todos los miembros de la organización. Para obtener más información sobre la federación basada en SAML 2.0, consulte Acerca de la federación basada en SAML 2.0 en la documentación de IAM.

SCADA

Consulte control de supervisión y adquisición de datos.

SCP

Consulte política de control de servicio.

secreta

En AWS Secrets Manager, información confidencial o restringida, como una contraseña o credenciales de usuario, que se almacena de forma cifrada. Se compone del valor del secreto y de sus metadatos. El valor del secreto puede ser binario, una sola cadena o varias cadenas. Para más información, consulte What's in a Secrets Manager secret? en la documentación de Secrets Manager.

seguridad desde el diseño

Enfoque de ingeniería de sistemas que tiene en cuenta la seguridad durante todo el proceso de desarrollo.

control de seguridad

Barrera de protección técnica o administrativa que impide, detecta o reduce la capacidad de un agente de amenazas para aprovechar una vulnerabilidad de seguridad. Existen cuatro tipos de controles de seguridad principales: preventivos, de detección, de respuesta y proactivos.

refuerzo de la seguridad

Proceso de reducir la superficie expuesta a ataques para hacerla más resistente a los ataques. Esto puede incluir acciones, como la eliminación de los recursos que ya no se necesitan, la implementación de prácticas recomendadas de seguridad consistente en conceder privilegios mínimos o la desactivación de características innecesarias en los archivos de configuración.

sistema de información sobre seguridad y administración de eventos (SIEM)

Herramientas y servicios que combinan sistemas de administración de información sobre seguridad (SIM) y de administración de eventos de seguridad (SEM). Un sistema de SIEM recopila, monitorea y analiza los datos de servidores, redes, dispositivos y otras fuentes para detectar amenazas y brechas de seguridad y generar alertas.

automatización de la respuesta de seguridad

Acción predefinida y programada que está diseñada para responder automáticamente a un evento de seguridad o corregirlo. Estas automatizaciones sirven como controles de seguridad preventivos o adaptables que le ayudan a implementar las mejores prácticas AWS de seguridad. La modificación de un grupo de seguridad de VPC, la aplicación de revisiones a una instancia de Amazon EC2 o la rotación de credenciales son algunos ejemplos de acciones de respuesta automatizadas.

cifrado del servidor

Cifrado de los datos en su destino, por parte de Servicio de AWS quien los recibe.

política de control de servicio (SCP)

Una política que proporciona un control centralizado de los permisos de todas las cuentas de una organización en AWS Organizations. Las SCP definen barreras de protección o establecen límites a las acciones que un administrador puede delegar en los usuarios o roles. Puede utilizar las SCP como listas de permitidos o rechazados, para especificar qué servicios o acciones se encuentra permitidos o prohibidos. Para obtener más información, consulte las políticas de control del servicio en la AWS Organizations documentación.

punto de enlace de servicio

La URL del punto de entrada de un Servicio de AWS. Para conectarse mediante programación a un servicio de destino, puede utilizar un punto de conexión. Para obtener más información, consulte Puntos de conexión de Servicio de AWS en Referencia general de AWS.

acuerdo de nivel de servicio (SLA)

Acuerdo que aclara lo que un equipo de TI se compromete a ofrecer a los clientes, como el tiempo de actividad y el rendimiento del servicio.

indicador de nivel de servicio (SLI)

Medición de un aspecto del rendimiento de un servicio, como la tasa de errores, la disponibilidad o el rendimiento.

objetivo de nivel de servicio (SLO)

Métrica objetivo que representa el estado de un servicio medido mediante un indicador de nivel de servicio.

modelo de responsabilidad compartida

Un modelo que describe la responsabilidad con AWS la que compartes la seguridad y el cumplimiento de la nube. AWS es responsable de la seguridad de la nube, mientras que usted es responsable de la seguridad en la nube. Para obtener más información, consulte el Modelo de responsabilidad compartida.

Shadow AI

Aplicaciones de IA no autorizadas creadas o utilizadas fuera de los canales regulados dentro de una organización.

SIEM

Consulte sistema de administración de eventos e información de seguridad.

único punto de error (SPOF)

Error en un único componente crítico de una aplicación que puede interrumpir el sistema.

SLA

Consulte acuerdo de nivel de servicio.

SLI

Consulte indicador de nivel de servicio.

SLO

Consulte objetivo de nivel de servicio.

modelo de dividir y sembrar

Un patrón para escalar y acelerar los proyectos de modernización. A medida que se definen las nuevas funciones y los lanzamientos de los productos, el equipo principal se divide para crear nuevos equipos de productos. Esto ayuda a ampliar las capacidades y los servicios de su organización, mejora la productividad de los desarrolladores y apoya la innovación rápida. Para más información, consulte Phased approach to modernizing applications in the Nube de AWS.

SPOF

Consulte único punto de error.

esquema en estrella

Estructura organizativa de una base de datos que utiliza una tabla de hechos de gran tamaño para almacenar datos transaccionales o medidos y una o varias tablas dimensionales más pequeñas para almacenar los atributos de los datos. Esta estructura está diseñada para utilizarse en un almacén de datos o con fines de inteligencia empresarial.

patrón de higo estrangulador

Un enfoque para modernizar los sistemas monolíticos mediante la reescritura y el reemplazo gradual de las funciones del sistema hasta que se pueda desmantelar el sistema heredado. Este patrón utiliza la analogía de una higuera que crece hasta convertirse en un árbol estable y, finalmente, se apodera y reemplaza a su host. El patrón fue presentado por Martin Fowler como una forma de gestionar el riesgo al reescribir sistemas monolíticos. Para ver un ejemplo de cómo aplicar este patrón, consulte Modernización gradual de los servicios web antiguos de Microsoft ASP.NET (ASMX) mediante contenedores y Amazon API Gateway.

subred

Un intervalo de direcciones IP en la VPC. Una subred debe residir en una sola zona de disponibilidad.

control de supervisión y adquisición de datos (SCADA)

En el sector de la fabricación, sistema que utiliza hardware y software para supervisar los activos físicos y las operaciones de producción.

cifrado simétrico

Un algoritmo de cifrado que utiliza la misma clave para cifrar y descifrar los datos.

pruebas sintéticas

Prueba de un sistema de manera que simule las interacciones de los usuarios para detectar posibles problemas o supervisar el rendimiento. Puede usar Amazon CloudWatch Synthetics para crear estas pruebas.

petición del sistema

Técnica para proporcionar contexto, instrucciones o pautas a un LLM para dirigir su comportamiento. Las peticiones del sistema ayudan a establecer el contexto y las reglas para las interacciones con los usuarios.

T

etiquetas

Key-value pares que actúan como metadatos para organizar sus AWS recursos. Las etiquetas pueden ayudar a administrar, identificar, organizar, buscar y filtrar recursos de . Para obtener más información, consulte Etiquetado de los recursos de AWS.

variable de destino

El valor que intenta predecir en el ML supervisado. Esto también se conoce como variable de resultado. Por ejemplo, en un entorno de fabricación, la variable objetivo podría ser un defecto del producto.

lista de tareas

Herramienta que se utiliza para hacer un seguimiento del progreso mediante un manual de procedimientos. La lista de tareas contiene una descripción general del manual de procedimientos y una lista de las tareas generales que deben completarse. Para cada tarea general, se incluye la cantidad estimada de tiempo necesario, el propietario y el progreso.

entorno de prueba

Consulte entorno.

entrenamiento

Proporcionar datos de los que pueda aprender su modelo de ML. Los datos de entrenamiento deben contener la respuesta correcta. El algoritmo de aprendizaje encuentra patrones en los

datos de entrenamiento que asignan los atributos de los datos de entrada al destino (la respuesta que desea predecir). Genera un modelo de ML que captura estos patrones. Luego, el modelo de ML se puede utilizar para obtener predicciones sobre datos nuevos para los que no se conoce el destino.

herramienta

Una función o API que un agente puede invocar para realizar operaciones en sistemas externos.

puerta de enlace de tránsito

Centro de tránsito de red que puede utilizar para interconectar las VPC y las redes en las instalaciones. Para obtener más información, consulte Qué es una pasarela de tránsito en la AWS Transit Gateway documentación.

flujo de trabajo basado en enlaces troncales

Un enfoque en el que los desarrolladores crean y prueban características de forma local en una rama de característica y, a continuación, combinan esos cambios en la rama principal. Luego, la rama principal se adapta a los entornos de desarrollo, preproducción y producción, de forma secuencial.

acceso de confianza

Otorgar permisos a un servicio que especifique para realizar tareas en su organización AWS Organizations y en sus cuentas en su nombre. El servicio de confianza crea un rol vinculado al servicio en cada cuenta, cuando ese rol es necesario, para realizar las tareas de administración por usted. Para obtener más información, consulte AWS Organizations Utilización con otros AWS servicios en la AWS Organizations documentación.

ajuste

Cambiar aspectos de su proceso de formación a fin de mejorar la precisión del modelo de ML. Por ejemplo, puede entrenar el modelo de ML al generar un conjunto de etiquetas, incorporar etiquetas y, luego, repetir estos pasos varias veces con diferentes ajustes para optimizar el modelo.

equipo de dos pizzas

Un DevOps equipo pequeño al que puedes alimentar con dos pizzas. Un equipo formado por dos integrantes garantiza la mejor oportunidad posible de colaboración en el desarrollo de software.

U

incertidumbre

Un concepto que hace referencia a información imprecisa, incompleta o desconocida que puede socavar la fiabilidad de los modelos predictivos de ML. Hay dos tipos de incertidumbre: la incertidumbre epistémica se debe a datos limitados e incompletos, mientras que la incertidumbre aleatoria se debe al ruido y la aleatoriedad inherentes a los datos.

tareas indiferenciadas

También conocido como tareas arduas, es el trabajo que es necesario para crear y operar una aplicación, pero que no proporciona un valor directo al usuario final ni proporciona una ventaja competitiva. Algunos ejemplos de tareas indiferenciadas son la adquisición, el mantenimiento y la planificación de la capacidad.

entornos superiores

Consulte entorno.

V

succión

Una operación de mantenimiento de bases de datos que implica limpiar después de las actualizaciones incrementales para recuperar espacio de almacenamiento y mejorar el rendimiento.

control de versión

Procesos y herramientas que realizan un seguimiento de los cambios, como los cambios en el código fuente de un repositorio.

Emparejamiento de VPC

Conexión entre dos VPC que permite enrutar el tráfico mediante direcciones IP privadas. Para obtener más información, consulte ¿Qué es una interconexión de VPC? en la documentación de Amazon VPC.

vulnerabilidad

Defecto de software o hardware que pone en peligro la seguridad del sistema.

W

caché caliente

Un búfer caché que contiene datos actuales y relevantes a los que se accede con frecuencia. La instancia de base de datos puede leer desde la caché del búfer, lo que es más rápido que leer desde la memoria principal o el disco.

datos templados

Datos a los que el acceso es infrecuente. Al consultar este tipo de datos, normalmente se aceptan consultas moderadamente lentas.

función de ventana

Función SQL que hace un cálculo en un grupo de filas que se relacionan de alguna manera con el registro actual. Las funciones de ventana son útiles para las tareas de procesamiento, como calcular una media móvil o acceder al valor de las filas en función de la posición relativa de la fila actual.

carga de trabajo

Conjunto de recursos y código que ofrece valor comercial, como una aplicación orientada al cliente o un proceso de backend.

flujo de trabajo

Grupos funcionales de un proyecto de migración que son responsables de un conjunto específico de tareas. Cada flujo de trabajo es independiente, pero respalda a los demás flujos de trabajo del proyecto. Por ejemplo, el flujo de trabajo de la cartera es responsable de priorizar las aplicaciones, planificar las oleadas y recopilar los metadatos de migración. El flujo de trabajo de la cartera entrega estos recursos al flujo de trabajo de migración, que luego migra los servidores y las aplicaciones.

WORM

Consulte escritura única y lectura múltiple.

WQF

Consulte AWS Workload Qualification Framework.

escritura única y lectura múltiple (WORM)

Modelo de almacenamiento que escribe los datos una sola vez y evita que se eliminen o modifiquen. Los usuarios autorizados pueden leer los datos tantas veces como sea necesario, pero no los pueden cambiar. Esta infraestructura de almacenamiento de datos se considera inmutable.

Z

ataque de día cero

Ataque, normalmente de malware, que se aprovecha de una vulnerabilidad de día cero.

vulnerabilidad de día cero

Un defecto o una vulnerabilidad sin mitigación en un sistema de producción. Los agentes de amenazas pueden usar este tipo de vulnerabilidad para atacar el sistema. Los desarrolladores suelen darse cuenta de la vulnerabilidad a raíz del ataque.

peticiones desde cero

Proporcionar a un LLM instrucciones para llevar a cabo una tarea, pero sin ejemplos (pasos) que puedan ayudar a guiarlo. El LLM debe usar los conocimientos del entrenamiento previo para llevar a cabo la tarea. La eficacia de la petición desde cero depende de la complejidad de la tarea y de la calidad de la petición. Consulte también peticiones con pocos pasos.

aplicación zombi

Aplicación que utiliza un promedio de CPU y memoria menor al 5 por ciento. En un proyecto de migración, es habitual retirar estas aplicaciones.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.