Erste Schritte mit Terraform: Anleitungen für und Experten AWS CDK AWS CloudFormation

AWS Präskriptive Leitlinien

AWS Präskriptive Leitlinien: Erste Schritte mit Terraform: Anleitungen für und Experten AWS CDK AWS CloudFormation

Table of Contents

Einführung 1
CloudFormation und Terraform-Terminologie 2
Ressourcen 4
Anbieter 6
Terraform-Aliase verwenden 8
Module 12
Module aufrufen 13
Das Root-Modul 14
Staaten und Backends 16
Datenquellen 19
Variablen, lokale Werte und Ausgaben 22
Variablen 22
Lokale Werte 25
Ausgabewerte 25
Funktionen, Ausdrücke und Metaargumente 27
Funktionen 27
Ausdrücke 27
Meta-Argumente 28
Häufig gestellte Fragen 35
Wann sollte ich Terraform anstelle von verwenden? CloudFormation 35
Wann sollte ich das anstelle von verwenden? AWS CDK CloudFormation 35
Gibt es ein Tool wie das AWS CDK , das Terraform-Konfigurationen generiert? 35
Wie erfahre ich mehr über Terraform? 35
Zugehörige Ressourcen 36
AWS Dokumentation 36
Sonstige Ressourcen 36
Anhang: Beispiele für den Zugriff auf Terraform-Attribute 37
Ressource 37
Datenquelle 37
Modul 37
Variable 37
Local 38
Dokumentverlauf 39
Glossar 40

..... 40
A 41
B 44
C 46
D 50
E 54
F 56
G 58
H 59
I 60
L 63
M 64
O 68
P 71
Q 74
R 74
S 77
T 81
U 83
V 83
W 84
Z 85
..... lxxxvi

Erste Schritte mit Terraform: Anleitung für AWS CDK- und AWS-Experten CloudFormation

Steven Guggenheimer, Amazon Web Services ()AWS

März 2024 (Geschichte der Dokumente)

Wenn Ihre Erfahrung mit der Bereitstellung von Cloud-Ressourcen ausschließlich im Bereich von liegt AWS, verfügen Sie möglicherweise nur über begrenzte Erfahrung mit Infrastructure-as-Code-Tools (IaC) jenseits der AWS Cloud Development Kit (AWS CDK)Grenzen. AWS CloudFormation Tatsächlich sind Ihnen ähnliche Tools wie Hashicorp Terraform möglicherweise völlig unbekannt. Je tiefer Sie jedoch in Ihre Cloud-Reise eintauchen, desto unausweichlicher wird es, dass Sie auf Terraform stoßen. Es wird definitiv zu Ihrem Vorteil sein, mit den Kernkonzepten vertraut zu sein.

Terraform, The AWS CDK und, verfolgen zwar ähnliche CloudFormation Ziele und teilen viele Kernkonzepte, es gibt jedoch einige Unterschiede. Sie sind möglicherweise nicht auf diese Unterschiede vorbereitet, wenn Sie sich Terraform zum ersten Mal nähern. Schließlich basieren alle CloudFormation Stapel darin AWS-Konten, sodass sie auf diese Weise eine direkte Beziehung zu den meisten Ressourcen haben, die sie verwalten. AWS CDK Terraform basiert nicht auf der Umgebung eines einzelnen Cloud-Anbieters. Dies gibt dem Unternehmen die Flexibilität, verschiedene Anbieter zu unterstützen, muss jedoch Ressourcen von einem entfernten Standort aus verwalten.

Dieser Leitfaden hilft Ihnen dabei, die Kernkonzepte hinter Terraform zu entmystifizieren, damit Sie alle IaC-Herausforderungen bewältigen können, die Ihnen in den Weg kommen. Es konzentriert sich darauf, wie Terraform Konzepte wie Anbieter, Module und Statusdateien zur Bereitstellung von Ressourcen verwendet. Es stellt auch die Terraform-Konzepte der Art und Weise gegenüber, wie sie ähnliche Operationen ausführen. AWS CDK CloudFormation

> **Note**
>
> Das AWS CDK hilft Entwicklern bei der Bereitstellung von CloudFormation Stacks mithilfe programmatischer Programmiersprachen. Nach der Ausführung `cdk synth` wird Ihr Code in CloudFormation Vorlagen konvertiert. Ab diesem Zeitpunkt ist der Prozess zwischen AWS CDK und identisch CloudFormation. Der Kürze halber bezieht sich dieser Leitfaden in der

Regel auf den AWS IaC-Prozess, CloudFormation aber die Vergleiche eignen sich genauso gut für den. AWS CDK

CloudFormation und Terraform-Terminologie

Beim Vergleich von Terraform mit AWS CDK und kann es aufgrund der CloudFormation inkonsistenten Terminologie, die zu ihrer Beschreibung verwendet wird, schwierig sein, die IaC-Kernkonzepte in Einklang zu bringen. Im Folgenden sind diese Begriffe aufgeführt und wie sich dieser Leitfaden auf sie bezieht:

- Stack — Ein Stack ist IaC, der in einer CI/CD-Pipeline bereitgestellt wird und als einzelne Einheit nachverfolgbar ist.Obwohl dieser Begriff in Terraform üblich ist CloudFormation, verwendet Terraform diesen Begriff nicht wirklich. Ein Terraform-Stack ist ein bereitgestelltes Root-Modul mit all seinen untergeordneten Modulen. Um jedoch Verwechslungen mit dem Begriff Modul zu vermeiden, verwendet dieser Leitfaden den Begriff Stack, um eine einzelne Bereitstellung für beide Tools zu beschreiben.
- Status — Der Status umfasst alle derzeit verfolgten Ressourcen und ihre aktuellen Konfigurationen innerhalb eines IaC-Bereitstellungsstapels. Wie im Terraform-Status und Backends verstehen Abschnitt beschrieben, verwendet Terraform den Begriff State häufiger als. CloudFormation Dies liegt daran, dass die Beibehaltung des Status in Terraform sichtbarer ist, für die Verfolgung und Aktualisierung des Status jedoch genauso wichtig ist. CloudFormation
- IaC-Datei — Eine IaC-Datei ist eine einzelne Datei, die die Sprache Infrastructure as Code (IaC) enthält. CloudFormation bezieht sich auf eine einzelne CloudFormation Datei als Vorlage. Vorlagen und Vorlagendateien in Terraform sind jedoch etwas völlig anderes. Das Äquivalent zu einer CloudFormation Vorlage in Terraform wird als Konfigurationsdatei bezeichnet. Um Verwirrung in diesem Handbuch zu vermeiden, wird der Begriff Datei oder IaC-Datei sowohl für CloudFormation Vorlagen als auch für Terraform-Konfigurationsdateien verwendet.

In der folgenden Tabelle werden die für CloudFormation Terraform und Terraform verwendeten Begriffe verglichen. Diese Tabelle soll Ähnlichkeiten aufzeigen. Dies sind keine one-to-one Vergleiche. Jedes Konzept unterscheidet sich zumindest geringfügig zwischen Terraform CloudFormation und Terraform. Die Konzepte werden in den entsprechenden Abschnitten dieses Handbuchs ausführlich erläutert.

CloudFormation Begriff	Terraform-Begriff	Abschnitt dieses Handbuchs
CDK-Schnittstellen (wie iBucket)	Datenquelle	Terraform-Datenquellen verstehen
Set ändern	Plan	Terraform-Module verstehen
Bedingungsfunktionen	Bedingte Ausdrücke	Terraform-Funktionen, Ausdrücke und Metaargumente verstehen
`DependsOn` Attribut	`depends_on` Metaargument	Terraform-Funktionen, Ausdrücke und Metaargumente verstehen
Intrinsische Funktionen	Funktionen	Terraform-Funktionen, Ausdrücke und Metaargumente verstehen
Module	Module	Terraform-Module verstehen
Outputs	Ausgabewerte	Grundlegendes zu Terraform-Variablen, lokalen Werten und Ausgaben
Parameter	Variablen	Grundlegendes zu Terraform-Variablen, lokalen Werten und Ausgaben
Registrierung	Anbieter	Terraform-Anbieter verstehen
Vorlage	Konfigurationsdatei	Alle

Terraform-Ressourcen verstehen

Der Hauptgrund für die Existenz sowohl AWS CloudFormation von Terraform als auch von Terraform ist die Erstellung und Wartung von Cloud-Ressourcen. Aber was genau ist eine Cloud-Ressource? Und sind CloudFormation Ressourcen und Terraform-Ressourcen dasselbe? Die Antwort lautet... ja und nein. Um dies zu veranschaulichen, finden Sie in diesem Handbuch ein Beispiel für die Verwendung CloudFormation und anschließend Terraform zur Erstellung eines Amazon Simple Storage Service (Amazon S3) -Buckets.

Das folgende CloudFormation Codebeispiel erstellt einen Amazon S3 S3-Beispiel-Bucket.

```
{
    "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"
            }
        }
    }
}
```

Das folgende Terraform-Codebeispiel erstellt einen identischen Amazon S3 S3-Bucket.

```
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"
  }
}
```

Für Terraform definiert ein Anbieter die Ressource, und dann deklarieren und konfigurieren Entwickler diese Ressourcen. Anbieter sind ein Konzept, das in diesem Handbuch im nächsten Abschnitt behandelt wird. Das Terraform-Beispiel erstellt völlig separate Ressourcen für mehrere Einstellungen des S3-Buckets. Das Erstellen separater Ressourcen für Einstellungen ist nicht unbedingt typisch dafür, wie der AWS Terraform-Anbieter mit Ressourcen umgeht. AWS Dieses Beispiel zeigt jedoch einen wichtigen Unterschied. Während eine CloudFormation Ressource durch die CloudFormationRessourcenspezifikation streng definiert ist, hat Terraform keine solche Anforderung. In Terraform ist das Konzept einer Ressource etwas nebulöser.

Obwohl sich die Tools in Bezug auf die genauen Leitplanken, die definieren, was eine einzelne Ressource ist, unterscheiden können, ist eine Cloud-Ressource im Allgemeinen jede bestimmte Entität, die in der Cloud existiert und erstellt, aktualisiert oder gelöscht werden kann. Unabhängig davon, um wie viele Ressourcen es sich handelt, erstellen die beiden vorherigen Beispiele beide exakt dasselbe Objekt mit den exakt gleichen Einstellungen innerhalb eines. AWS-Konto

Terraform-Anbieter verstehen

In Terraform ist ein Anbieter ein Plugin, das mit Cloud-Anbietern, Tools von Drittanbietern und anderen APIs interagiert. Um Terraform mit zu verwenden, verwenden Sie den AWS Anbieter AWS, der mit Ressourcen interagiert. AWS

Wenn Sie die AWS CloudFormation Registrierung noch nie verwendet haben, um Erweiterungen von Drittanbietern in Ihre Deployment-Stacks zu integrieren, ist es möglicherweise gewöhnungsbedürftig, sich an Terraform-Anbieter zu gewöhnen. Da CloudFormation es sich um systemeigene Ressourcen handelt AWS, ist der Anbieter von AWS Ressourcen standardmäßig bereits vorhanden. Terraform hat dagegen keinen einzigen Standardanbieter, sodass nichts über die Herkunft einer bestimmten Ressource angenommen werden kann. Das bedeutet, dass das Erste, was in einer Terraform-Konfigurationsdatei deklariert werden muss, genau ist, wohin die Ressourcen gehen und wie sie dorthin gelangen werden.

Diese Unterscheidung verleiht Terraform eine zusätzliche Komplexitätsebene, die es nicht gibt. CloudFormation Diese Komplexität bietet jedoch eine erhöhte Flexibilität. Sie können mehrere Anbieter innerhalb eines einzigen Terraform-Moduls deklarieren, und dann können die erstellten zugrunde liegenden Ressourcen als Teil derselben Bereitstellungsebene miteinander interagieren.

Dies kann auf vielfältige Weise nützlich sein. Anbieter müssen nicht unbedingt für separate Cloud-Anbieter zuständig sein. Anbieter können jede Quelle für Cloud-Ressourcen repräsentieren. Nehmen wir zum Beispiel Amazon Elastic Kubernetes Service (Amazon EKS). Wenn Sie einen Amazon EKS-Cluster bereitstellen, möchten Sie möglicherweise Helm-Diagramme verwenden, um Erweiterungen von Drittanbietern zu verwalten, und Kubernetes selbst zur Verwaltung von Pod-Ressourcen verwenden. Da AWSHelm und Kubernetes alle ihre eigenen Terraform-Anbieter haben, können Sie diese Ressourcen alle gleichzeitig bereitstellen und integrieren und dann Werte zwischen ihnen weitergeben.

Im folgenden Codebeispiel für Terraform erstellt der AWS Anbieter einen Amazon EKS-Cluster, und dann werden die resultierenden Kubernetes-Konfigurationsinformationen sowohl an den Helm- als auch an den Kubernetes-Anbieter weitergegeben.

```
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"
  }
}
```

Wenn es um die beiden IaC-Tools geht, gibt es einen Kompromiss in Bezug auf Anbieter. Terraform stützt sich ausschließlich auf externe Anbieterpakete, die den Motor für die Bereitstellung bilden. CloudFormation unterstützt intern alle wichtigen Prozesse. AWS Mit CloudFormation müssen Sie sich nur dann Gedanken über Drittanbieter machen, wenn Sie eine Erweiterung eines Drittanbieters integrieren möchten. Jeder Ansatz hat Vor- und Nachteile. Welches für Sie das Richtige ist, würde den Rahmen dieses Leitfadens sprengen, aber es ist wichtig, bei der Bewertung beider Tools den Unterschied zu berücksichtigen.

Terraform-Aliase verwenden

In Terraform können Sie benutzerdefinierte Konfigurationen an jeden Anbieter übergeben. Was ist also, wenn Sie mehrere Anbieterkonfigurationen innerhalb desselben Moduls verwenden möchten? In diesem Fall müssten Sie einen Alias verwenden. Aliase helfen Ihnen bei der Auswahl des Anbieters, der auf Ressourcen- oder Modulebene verwendet werden soll. Wenn Sie mehr als eine Instanz desselben Anbieters haben, verwenden Sie einen Alias, um die nicht standardmäßigen Instanzen zu definieren. Beispielsweise kann es sich bei Ihrer Standardanbieterinstanz um eine bestimmte Instanz handeln AWS-Region, aber Sie verwenden Aliase, um alternative Regionen zu definieren.

Das folgende Terraform-Beispiel zeigt, wie Sie einen Alias verwenden, um Buckets in verschiedenen Bereichen bereitzustellen. AWS-Regionen Die Standardregion für den Anbieter ist`us-west-2`, aber Sie können den Ost-Alias verwenden, um Ressourcen bereitzustellen. `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"
}
```

Wenn Sie `alias` zusammen mit dem `provider` Metaargument ein verwenden, können Sie, wie im vorherigen Beispiel gezeigt, eine andere Anbieterkonfiguration für bestimmte Ressourcen angeben. Die Bereitstellung mehrerer Ressourcen AWS-Regionen in einem einzigen Stapel ist erst der Anfang. Das Aliasing von Anbietern ist in vielerlei Hinsicht unglaublich praktisch.

Beispielsweise ist es sehr üblich, mehrere Kubernetes-Cluster gleichzeitig bereitzustellen. Aliase können Ihnen helfen, zusätzliche Helm- und Kubernetes-Anbieter zu konfigurieren, sodass Sie diese Drittanbieter-Tools für verschiedene Amazon EKS-Ressourcen unterschiedlich verwenden können. Das folgende Terraform-Codebeispiel veranschaulicht, wie Aliase zur Ausführung dieser Aufgabe verwendet werden.

```
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"
  }
}
```

Terraform-Module verstehen

Im Bereich Infrastructure as Code (IaC) ist ein Modul ein eigenständiger Codeblock, der isoliert und zur Wiederverwendung zusammengepackt wird. Das Konzept der Module ist ein unausweichlicher Aspekt der Terraform-Entwicklung. Weitere Informationen finden Sie unter Module in der Terraform-Dokumentation. AWS CloudFormation unterstützt auch Module. Weitere Informationen finden Sie unter Einführung in AWS CloudFormation Module im AWS Cloud Operations and Migrations Blog.

Der Hauptunterschied zwischen Modulen in Terraform besteht CloudFormation darin, dass CloudFormation Module mithilfe eines speziellen Ressourcentyps () importiert werden. `AWS::CloudFormation::ModuleVersion` In Terraform hat jede Konfiguration mindestens ein Modul, das als Root-Modul bezeichnet wird. Terraform-Ressourcen, die sich in der main.tf-Datei oder in Dateien in einer Terraform-Konfigurationsdatei befinden, werden als im Root-Modul befindlich betrachtet. Das Root-Modul kann dann andere Module aufrufen, um sie in den Stack aufzunehmen. Das folgende Beispiel zeigt ein Root-Modul, das mithilfe des Open-Source-EKS-Moduls einen Amazon Elastic Kubernetes Service (Amazon EKS) -Cluster bereitstellt.

```
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)
  }
```

```
}
```

Möglicherweise haben Sie bemerkt, dass die obige Konfigurationsdatei den Provider nicht enthält. AWS Das liegt daran, dass Module eigenständig sind und ihre eigenen Anbieter enthalten können. Da Terraform-Anbieter global sind, können Anbieter aus einem untergeordneten Modul im Root-Modul verwendet werden. Dies gilt jedoch nicht für alle Modulwerte. Andere interne Werte innerhalb eines Moduls sind standardmäßig nur auf dieses Modul beschränkt und müssen als Ausgaben deklariert werden, damit sie im Root-Modul zugänglich sind. Sie können Open-Source-Module nutzen, um die Erstellung von Ressourcen innerhalb Ihres Stacks zu vereinfachen. Das eks-Modul leistet beispielsweise mehr als die Bereitstellung eines EKS-Clusters — es stellt eine voll funktionsfähige Kubernetes-Umgebung bereit. Wenn Sie es verwenden, müssen Sie nicht Dutzende zusätzlicher Codezeilen schreiben, vorausgesetzt, die Konfiguration des EKS-Moduls entspricht Ihren Anforderungen.

Module aufrufen

Zwei der wichtigsten Terraform-CLI-Befehle, die Sie während der Terraform-Bereitstellung ausführen, sind terraform init und terraform apply. Einer der Standardschritte, die der `terraform init` Befehl ausführt, besteht darin, alle untergeordneten Module zu suchen und sie als Abhängigkeiten in das Verzeichnis zu importieren. `.terraform/modules` Wenn Sie während der Entwicklung ein neues Modul aus externen Quellen hinzufügen, müssen Sie es erneut initialisieren, bevor Sie den `apply` Befehl verwenden können. Wenn Sie einen Verweis auf ein Terraform-Modul hören, bezieht sich das auf die Pakete in diesem Verzeichnis. Genau genommen ist das Modul, das Sie in Ihrem Code deklarieren, das aufrufende Modul. In der Praxis ruft das Schlüsselwort module also das eigentliche Modul auf, das als Abhängigkeit gespeichert ist.

Auf diese Weise dient das aufrufende Modul als prägnantere Darstellung des vollständigen Moduls, das bei der Bereitstellung ersetzt werden muss. Sie können sich diese Idee zunutze machen, indem Sie Ihre eigenen Module innerhalb Ihrer Stacks erstellen, um logische Trennungen von Ressourcen zu erzwingen, indem Sie beliebige Kriterien verwenden. Denken Sie daran, dass das Endziel dabei darin bestehen sollte, die Komplexität Ihres Stacks zu reduzieren. Da die gemeinsame Nutzung von Daten zwischen Modulen erfordert, dass Sie diese Daten innerhalb des Moduls ausgeben, kann es manchmal zu kompliziert werden, wenn Sie sich zu stark auf Module verlassen.

Das Root-Modul

Da jede Terraform-Konfiguration mindestens ein Modul hat, kann es hilfreich sein, die Moduleigenschaften des Moduls zu untersuchen, mit dem Sie sich am häufigsten befassen werden: des Root-Moduls. Wann immer Sie an einem Terraform-Projekt arbeiten, besteht das Root-Modul aus allen `.tf` (oder`.tf.json`) Dateien in Ihrem Verzeichnis auf oberster Ebene. Wenn Sie `terraform apply` in diesem Verzeichnis der obersten Ebene ausführen, versucht Terraform, jede Datei auszuführen, die es dort findet. `.tf` Alle Dateien in Unterverzeichnissen werden ignoriert, es sei denn, sie werden in einer dieser Konfigurationsdateien der obersten Ebene aufgerufen.

Dies bietet eine gewisse Flexibilität bei der Strukturierung Ihres Codes. Dies ist auch der Grund, warum es genauer ist, Ihre Terraform-Bereitstellung als Modul als als Datei zu bezeichnen, da mehrere Dateien an einem einzigen Prozess beteiligt sein können. Es gibt eine Standardmodulstruktur, die Terraform als Best Practices empfiehlt. Wenn Sie jedoch eine `.tf` Datei in Ihrem Verzeichnis auf oberster Ebene ablegen würden, würde sie zusammen mit den übrigen Dateien ausgeführt. Tatsächlich werden alle `.tf` Dateien der obersten Ebene in einem Modul bereitgestellt, wenn Sie es ausführen. `terraform apply` Welche Datei wird also zuerst von Terraform ausgeführt? Die Antwort auf diese Frage ist sehr wichtig.

Es gibt eine Reihe von Schritten, die Terraform nach der Initialisierung und vor der Stack-Bereitstellung ausführt. Zuerst werden die vorhandenen Konfigurationen analysiert und dann wird ein Abhängigkeitsdiagramm erstellt. Das Abhängigkeitsdiagramm bestimmt, welche Ressourcen benötigt werden und in welcher Reihenfolge sie adressiert werden sollten. Ressourcen, die Eigenschaften enthalten, auf die in anderen Ressourcen verwiesen wird, würden beispielsweise vor ihren abhängigen Ressourcen behandelt. In ähnlicher Weise würden Ressourcen, die mithilfe des `depends_on` Parameters explizit Abhängigkeit deklarieren, erst nach den Ressourcen behandelt, die sie angeben. Wenn möglich, kann Terraform Parallelität implementieren und gleichzeitig unabhängige Ressourcen verarbeiten. Sie können das Abhängigkeitsdiagramm vor der Bereitstellung mit dem Befehl terraform graph sehen.

Nachdem das Abhängigkeitsdiagramm erstellt wurde, bestimmt Terraform, was während der Bereitstellung getan werden muss. Es vergleicht das Abhängigkeitsdiagramm mit der neuesten Statusdatei. Das Ergebnis dieses Prozesses wird als Plan bezeichnet und ist einem CloudFormation Änderungssatz sehr ähnlich. Sie können den aktuellen Plan mit dem Befehl Terraform Plan anzeigen.

Als bewährte Methode wird empfohlen, so nah wie möglich an der Standardmodulstruktur zu bleiben. In Fällen, in denen Ihre Konfigurationsdateien zu lang werden, um sie effizient zu verwalten, und logische Trennungen die Verwaltung vereinfachen könnten, können Sie Ihren Code auf mehrere

Dateien verteilen. Denken Sie daran, wie das Abhängigkeitsdiagramm und der Planungsprozess funktionieren, damit Ihre Stacks so effizient wie möglich ausgeführt werden.

Terraform-Status und Backends verstehen

Eines der wichtigsten Konzepte in Infrastructure as Code (IaC) ist das Konzept des Zustands. IaC-Dienste behalten den Status bei, sodass Sie eine Ressource in einer IaC-Datei deklarieren können, ohne dass sie bei jeder Bereitstellung neu erstellt werden muss. IaC-Dateien dokumentieren den Status aller Ressourcen am Ende einer Bereitstellung, sodass dieser Status dann mit dem Zielstatus verglichen werden kann, der in der nächsten Bereitstellung deklariert wurde. Wenn der aktuelle Status also einen Amazon Simple Storage Service (Amazon S3) -Bucket mit dem Namen enthält `my-s3-bucket` und die eingehenden Änderungen ebenfalls denselben Bucket enthalten, wendet der neue Prozess alle gefundenen Änderungen auf den vorhandenen Bucket an, anstatt zu versuchen, einen völlig neuen Bucket zu erstellen.

Die folgende Tabelle enthält Beispiele für den allgemeinen IaC-Statusprozess.

Aktueller Status	Zielstatus	Aktion
Es wurde kein S3-Bucket benannt `my-s3-bucket`	S3-Bucket benannt `my-s3-bucket`	Erstellen Sie einen S3-Bucket mit dem Namen `my-s3-bucket`
`my-s3-bucket` ohne konfigurierte Bucket-Versionierung	`my-s3-bucket` ohne konfigurierte Bucket-Versionierung	Keine Aktion
`my-s3-bucket` ohne konfigurierte Bucket-Versionierung	`my-s3-bucket` mit konfigurierter Bucket-Versionierung	Konfigurieren Sie somy-s3-bucket , dass die Bucket-Versionierung aktiviert ist
`my-s3-bucket` mit konfigurierter Bucket-Versionierung	Kein S3-Bucket benannt `my-s3-bucket`	Versuchen Sie zu löschen `my-s3-bucket`

Um zu verstehen, auf welche Art AWS CloudFormation und Weise Terraform den Status verfolgt, ist es wichtig, sich an den ersten grundlegenden Unterschied zwischen den beiden Tools zu erinnern: Es CloudFormation wird innerhalb von gehostet und Terraform ist im AWS Cloud Wesentlichen remote. Diese Tatsache ermöglicht es, den Status intern CloudFormation aufrechtzuerhalten. Sie können zur

CloudFormation Konsole gehen und den Ereignisverlauf eines bestimmten Stacks einsehen, aber der CloudFormation Dienst selbst setzt die Statusregeln für Sie durch.

Die drei Modi, die CloudFormation für eine bestimmte Ressource verwendet werden`Create`, sind`Update`, und`Delete`. Der aktuelle Modus wird anhand der Ereignisse bei der letzten Bereitstellung bestimmt und kann nicht anderweitig beeinflusst werden. Möglicherweise können Sie CloudFormation Ressourcen manuell aktualisieren, um zu beeinflussen, welcher Modus festgelegt wird, aber Sie können keinen Befehl an diese Ressource übergeben, der besagt, CloudFormation dass Sie `Create` im Modus arbeiten müssen.

Da Terraform nicht in der gehostet wird AWS Cloud, muss der Prozess der Aufrechterhaltung des Status besser konfigurierbar sein. Aus diesem Grund wird der Terraform-Status in einer automatisch generierten Statusdatei beibehalten. Ein Terraform-Entwickler muss sich viel direkter mit dem Status befassen, als er es tun würde. CloudFormation Es ist wichtig, sich daran zu erinnern, dass die Statusverfolgung für beide Tools gleich wichtig ist.

Standardmäßig wird die Terraform-Statusdatei lokal auf der obersten Ebene des Hauptverzeichnisses gespeichert, in dem Ihr Terraform-Stack ausgeführt wird. Wenn Sie den `terraform apply` Befehl in Ihrer lokalen Entwicklungsumgebung ausführen, können Sie sehen, wie Terraform die Datei terraform.tfstate generiert, mit der der Status in Echtzeit aufrechterhalten wird. In guten wie in schlechten Zeiten haben Sie dadurch viel mehr Kontrolle über den Status in Terraform als in. CloudFormation Sie sollten die Statusdatei zwar niemals direkt aktualisieren, es gibt jedoch mehrere Terraform-CLI-Befehle, die Sie ausführen können, um den Status zwischen Bereitstellungen zu aktualisieren. Mit dem Terraform-Import können Sie beispielsweise Ressourcen, die außerhalb von Terraform erstellt wurden, zu Ihrem Bereitstellungsstapel hinzufügen. Umgekehrt können Sie eine Ressource aus dem Status entfernen, indem Sie terraform state rm ausführen.

Die Tatsache, dass Terraform seinen Status irgendwo speichern muss, führt zu einem anderen Konzept, das nicht gilt: das Backend. CloudFormation Ein Terraform-Backend ist der Ort, an dem ein Terraform-Stack seine Statusdatei nach der Bereitstellung speichert. Hier erwartet es auch, die Statusdatei zu finden, wenn eine neue Bereitstellung beginnt. Wenn Sie Ihren Stack wie oben beschrieben lokal ausführen, können Sie eine Kopie des Terraform-Status im lokalen Verzeichnis der obersten Ebene speichern. Dies wird als lokales Backend bezeichnet.

Bei der Entwicklung für eine CI/CD-Umgebung (Continuous Integration and Continuous Deployment) ist die lokale Statusdatei im Allgemeinen in der .gitignore-Datei enthalten, damit sie nicht der Versionskontrolle unterliegt. Dann ist in der Pipeline keine lokale Statusdatei vorhanden. Um ordnungsgemäß zu funktionieren, muss diese Pipeline-Phase irgendwo die richtige Statusdatei

finden.Aus diesem Grund enthalten Terraform-Konfigurationsdateien häufig einen Backend-Block. Der Backend-Block zeigt dem Terraform-Stack an, dass er irgendwo neben seinem eigenen Verzeichnis auf oberster Ebene suchen muss, um die Statusdatei zu finden.

Ein Terraform-Backend kann sich fast überall befinden: in einem Amazon S3 S3-Bucket, einem API-Endpunkt oder sogar in einem Remote-Terraform-Workspace. Das Folgende ist ein Beispiel für ein Terraform-Backend, das in einem Amazon S3 S3-Bucket gespeichert ist.

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

Um zu vermeiden, dass vertrauliche Informationen in Terraform-Konfigurationsdateien gespeichert werden, unterstützen Backends auch Teilkonfigurationen. Im vorherigen Beispiel sind die für den Zugriff auf den Bucket erforderlichen Anmeldeinformationen nicht in der Konfiguration vorhanden. Anmeldeinformationen können aus Umgebungsvariablen oder auf andere Weise abgerufen werden, z. AWS Secrets Manager B. Weitere Informationen finden Sie unter Schützen vertraulicher Daten mithilfe von AWS Secrets Manager und HashiCorp Terraform.

Ein übliches Backend-Szenario ist ein lokales Backend, das in Ihrer lokalen Umgebung zu Testzwecken verwendet wird. Die Datei terraform.tfstate ist in der .gitignore-Datei enthalten, sodass sie nicht in das Remote-Repository übertragen wird. Dann würde jede Umgebung innerhalb der CI/CD-Pipeline ihr eigenes Backend verwalten. In diesem Szenario haben möglicherweise mehrere Entwickler Zugriff auf diesen Remote-Status, sodass Sie die Integrität der Statusdatei schützen sollten. Wenn mehrere Bereitstellungen gleichzeitig ausgeführt werden und der Status aktualisiert wird, kann die Statusdatei beschädigt werden. Aus diesem Grund wird die Statusdatei in Situationen mit nicht lokalen Backends normalerweise während der Bereitstellung gesperrt.

Terraform-Datenquellen verstehen

Es ist sehr üblich, dass Deployment-Stacks auf Daten aus zuvor vorhandenen Ressourcen zurückgreifen. Die meisten IaC-Tools bieten die Möglichkeit, Ressourcen zu importieren, die durch einen anderen Prozess erstellt wurden. Diese importierten Ressourcen sind normalerweise schreibgeschützt (obwohl IAM-Rollen eine bemerkenswerte Ausnahme bilden) und werden für den Zugriff auf Daten verwendet, die von Ressourcen innerhalb des Stacks benötigt werden. AWS CloudFormation ermöglicht den Import von Ressourcen, aber diese Idee lässt sich besser erklären, indem man sich die AWS Cloud Development Kit (AWS CDK) ansieht.

Das AWS CDK hilft Entwicklern, bestehende Programmiersprachen zu verwenden, um CloudFormation Vorlagen zu generieren. Das Endergebnis eines AWS CDK Vorgangs ist eine importierte Ressource in CloudFormation. Die mit der verwendete Syntax erleichtert jedoch den AWS CDK Vergleich mit Terraform. Hier ist ein Beispiel für das Importieren einer Ressource mithilfe von. AWS CDK

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

Eine importierte Ressource wird normalerweise erstellt, indem eine statische Methode für dieselbe Klasse aufgerufen wird, mit der Sie eine neue Ressource derselben Art erstellen. Beim Aufrufen `new Bucket(...` würde eine neue Ressource erstellt, und beim Aufrufen wird eine bestehende Ressource `Bucket.fromBucketAttributes(...` importiert. Sie übergeben eine Teilmenge der Eigenschaften des Buckets an die Funktion, damit sie den richtigen Bucket finden AWS CDK kann. Ein weiterer Unterschied besteht jedoch darin, dass beim Erstellen eines neuen Buckets eine vollständige Instanz der `Bucket` Klasse mit allen darin verfügbaren Eigenschaften und Methoden zurückgegeben wird. Beim Import der Ressource wird ein zurückgegeben. `IBucket` Dabei handelt es sich um einen Typ, der nur die Eigenschaften enthält, die `Bucket` vorhanden sein müssen. Sie können zwar eine Ressource aus einem externen Stack importieren, aber die Möglichkeiten, was Sie damit machen können, sind begrenzt.

In Terraform wird ein ähnliches Ziel durch die Verwendung von Datenquellen erreicht. Zu den meisten definierten Terraform-Ressourcen steht zusätzlich eine zugehörige Datenquelle zur Verfügung. Das

Folgende ist ein Beispiel für eine Terraform S3-Bucket-Ressource, gefolgt von der entsprechenden Datenquelle.

```
# 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"
}
```

Der einzige Unterschied zwischen diesen beiden Elementen ist das Namenspräfix. Wie in der Dokumentation zu einer Datenquelle gezeigt, sind weniger Parameter verfügbar, die Sie an eine Datenquelle übergeben können als an eine Ressource. Das liegt daran, dass die Ressource diese Parameter verwendet, um alle Eigenschaften eines neuen S3-Buckets zu deklarieren, während die Datenquelle gerade genügend Informationen benötigt, um die Daten einer vorhandenen Ressource eindeutig zu identifizieren und zu importieren.

Die Ähnlichkeit zwischen der Syntax einer Terraform-Ressource und einer Datenquelle kann praktisch, aber auch problematisch sein. Es ist üblich, dass unerfahrene Terraform-Entwickler in ihrer Konfiguration versehentlich eine Datenquelle anstelle einer Ressource verwenden. Terraform-Datenquellen sind immer schreibgeschützt. Sie können sie anstelle der entsprechenden Ressource für Leseaktionen verwenden (z. B. die Bereitstellung eines ID-Namens für eine andere Ressource). Sie können sie jedoch nicht für Schreibaktionen verwenden, die einen Aspekt der zugrunde liegenden Ressource grundlegend verändern. Aus diesem Grund können Sie sich eine Terraform-Datenquelle als eine geklonte Version der zugrunde liegenden Ressource vorstellen.

Ähnlich wie im vorherigen AWS CDK iBucket-Beispiel sind Datenquellen für schreibgeschützte Szenarien nützlich. Wenn Sie Daten aus einer vorhandenen Ressource abrufen müssen, diese Ressource jedoch nicht in Ihrem Stack verwalten müssen, verwenden Sie eine Datenquelle. Ein gutes Beispiel hierfür ist, wenn Sie eine Amazon EC2 EC2-Instance erstellen, die die Standard-VPC des Kontos verwendet. Da diese VPC bereits existiert, müssen Sie nur ihre Daten abrufen. Das folgende Codebeispiel zeigt, wie Daten verwendet werden, um die Ziel-VPC zu identifizieren.

```
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
}
```

Grundlegendes zu Terraform-Variablen, lokalen Werten und Ausgaben

Variablen erhöhen die Codeflexibilität, indem sie Platzhalter innerhalb von Codeblöcken zulassen. Variablen können unterschiedliche Werte darstellen, wenn der Code wiederverwendet wird. Terraform unterscheidet seine Variablentypen durch ihren modularen Umfang. Eingabevariablen sind externe Werte, die in ein Modul eingefügt werden können, Ausgabewerte sind interne Werte, die extern gemeinsam genutzt werden können, und lokale Werte bleiben immer innerhalb ihres ursprünglichen Gültigkeitsbereichs.

Variablen

AWS CloudFormation verwendet Parameter, um benutzerdefinierte Werte darzustellen, die von einer Stack-Bereitstellung zur nächsten festgelegt und zurückgesetzt werden können. In ähnlicher Weise verwendet Terraform Eingabevariablen oder Variablen. Variablen können an einer beliebigen Stelle in einer Terraform-Konfigurationsdatei deklariert werden und werden normalerweise mit dem erforderlichen Datentyp oder Standardwert deklariert. Alle drei der folgenden Ausdrücke sind gültige Terraform-Variablendeklarationen.

```
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"
    }
```

```
  ]
}
```

Um innerhalb der Konfiguration auf Sparkys Breed zuzugreifen, würden Sie die Variable verwenden. `var.dogs[0].breed` Wenn eine Variable keinen Standard hat und nicht als nullwertfähig eingestuft ist, muss der Wert der Variablen für jede Bereitstellung festgelegt werden. Andernfalls ist es optional, einen neuen Wert für die Variable festzulegen. In einem Root-Modul können Sie aktuelle Variablenwerte in der Befehlszeile, als Umgebungsvariablen oder in der Datei terraform.tfvars festlegen. Das folgende Beispiel zeigt, wie Variablenwerte in die Datei terraform.tfvars eingegeben werden, die im obersten Verzeichnis des Moduls gespeichert ist.

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

random_number = 7

thing_i_made_up = "Kabibble"
```

Der Wert für `dogs` in diesem Beispiel die Datei terraform.tfvars würde den Standardwert in der Variablendeklaration überschreiben. Wenn Sie Variablen innerhalb eines untergeordneten Moduls deklarieren, können Sie die Variablenwerte direkt im Moduldeklarationsblock festlegen, wie im folgenden Beispiel gezeigt.

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

Zu den anderen Argumenten, die Sie bei der Deklaration einer Variablen verwenden können, gehören:

- `sensitive`— Wenn Sie dies auf einstellen, `true` wird verhindert, dass der Variablenwert in den Terraform-Prozessausgaben angezeigt wird.
- `nullable`— Wenn Sie dies auf setzen, `true` kann die Variable keinen Wert haben. Dies ist praktisch für Variablen, bei denen kein Standard gesetzt ist.
- `description`— Fügt den Metadaten für den Stack eine Beschreibung der Variablen hinzu.
- `validation`— Legen Sie Validierungsregeln für die Variable fest.

Einer der praktischsten Aspekte von Terraform-Variablen ist die Möglichkeit, ein oder mehrere Validierungsobjekte innerhalb der Variablendeklaration hinzuzufügen. Sie können Validierungsobjekte verwenden, um eine Bedingung hinzuzufügen, die die Variable erfüllen muss, sonst schlägt die Bereitstellung fehl. Sie können auch eine benutzerdefinierte Fehlermeldung einrichten, die angezeigt wird, wenn die Bedingung verletzt wird.

Sie richten beispielsweise eine Terraform-Konfigurationsdatei ein, die Mitglieder Ihres Teams ausführen werden. Vor der Bereitstellung der Stacks muss ein Teammitglied eine terraform.tfvars-Datei erstellen, um einen wichtigen Konfigurationswert festzulegen. Um sie daran zu erinnern, könnten Sie etwas wie das Folgende tun.

```
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-'"
  }
}
```

Wie in diesem Beispiel gezeigt, können Sie mehrere Bedingungen innerhalb einer einzigen Variablen festlegen. Terraform zeigt nur Fehlermeldungen für fehlgeschlagene Bedingungen an. Auf diese Weise können Sie alle Arten von Regeln für Variablenwerte durchsetzen. Wenn ein Variablenwert einen Pipeline-Ausfall verursacht, wüssten Sie genau, warum.

Lokale Werte

Wenn es in einem Modul Werte gibt, für die Sie einen Alias verwenden möchten, verwenden Sie das `locals` Schlüsselwort, anstatt eine Standardvariable zu deklarieren, die niemals aktualisiert wird. Wie der Name schon sagt, enthält ein `locals` Block Begriffe, die intern auf dieses spezifische Modul beschränkt sind. Wenn Sie einen Zeichenkettenwert transformieren möchten, z. B. indem Sie einem Variablenwert ein Präfix zur Verwendung in einem Ressourcennamen hinzufügen, ist die Verwendung eines lokalen Werts möglicherweise eine gute Lösung. Ein einzelner `locals` Block kann alle lokalen Werte für Ihr Modul deklarieren, wie im folgenden Beispiel gezeigt.

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

Denken Sie daran, dass das `locals` Schlüsselwort beim Zugriff auf den Wert singulär wird, z. B. `local.LocalConfigId`

Ausgabewerte

Wenn Terraform-Eingabevariablen wie CloudFormation Parameter sind, könnte man sagen, dass Terraform-Ausgabewerte wie Ausgaben sind. CloudFormation Beide werden verwendet, um Werte innerhalb eines Deployment-Stacks verfügbar zu machen. Da das Terraform-Modul jedoch stärker in der Struktur des Tools verankert ist, werden Terraform-Ausgabewerte auch verwendet, um Werte innerhalb eines Moduls für ein übergeordnetes Modul oder andere untergeordnete Module verfügbar zu machen, selbst wenn sich diese Module alle innerhalb desselben Bereitstellungsstapels befinden. Wenn Sie zwei benutzerdefinierte Module erstellen und das erste Modul auf den ID-Wert des zweiten Moduls zugreifen muss, müssen Sie dem zweiten Modul den folgenden `output` Block hinzufügen.

```
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
}
```

Da Terraform-Ausgabewerte innerhalb desselben Stacks verwendet werden können, können Sie das `sensitive` Attribut auch in einem `output` Block verwenden, um zu verhindern, dass der Wert in der Stackausgabe angezeigt wird. Darüber hinaus kann ein `output` Block `precondition` Blöcke auf die gleiche Weise verwenden wie Variablen `validation` Blöcke verwenden: um sicherzustellen, dass Variablen bestimmten Regeln folgen. Auf diese Weise können Sie sicherstellen, dass alle Werte innerhalb eines Moduls wie erwartet vorhanden sind, bevor Sie mit der Bereitstellung fortfahren.

```
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."
  }
}
```

Terraform-Funktionen, Ausdrücke und Metaargumente verstehen

Ein Kritikpunkt an IaC-Tools, die deklarative Konfigurationsdateien anstelle gängiger Programmiersprachen verwenden, ist, dass sie die Implementierung benutzerdefinierter Programmlogik erschweren. In Terraform-Konfigurationen wird dieses Problem durch die Verwendung von Funktionen, Ausdrücken und Metaargumenten behoben.

Funktionen

Einer der großen Vorteile der Verwendung von Code zur Bereitstellung Ihrer Infrastruktur ist die Möglichkeit, gängige Workflows zu speichern und immer wieder zu verwenden, wobei oft jedes Mal andere Argumente übergeben werden. Terraform-Funktionen ähneln AWS CloudFormation systeminternen Funktionen, obwohl ihre Syntax eher der Art und Weise ähnelt, wie Funktionen in Programmiersprachen aufgerufen werden. Möglicherweise sind Ihnen in den Beispielen in diesem Handbuch bereits einige Terraform-Funktionen wie substr, concat, length und base64decode aufgefallen. Wie CloudFormation bei intrinsischen Funktionen verfügt Terraform über eine Reihe integrierter Funktionen, die in Ihren Konfigurationen verwendet werden können. Wenn ein bestimmtes Ressourcenattribut beispielsweise ein sehr großes JSON-Objekt benötigt, dessen direktes Einfügen in die Datei ineffizient wäre, könnten Sie das Objekt in eine JSON-Datei einfügen und Terraform-Funktionen verwenden, um darauf zuzugreifen. Im folgenden Beispiel gibt die `file` Funktion den Inhalt der Datei in Zeichenfolgenform zurück und konvertiert ihn dann in einen Objekttyp. `jsondecode`

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

Ausdrücke

Terraform ermöglicht auch bedingte Ausdrücke, die CloudFormation `condition` Funktionen ähneln, außer dass sie die traditionellere ternäre Operatorsyntax verwenden. Im folgenden Beispiel geben die beiden Ausdrücke genau dasselbe Ergebnis zurück. Das zweite Beispiel nennt Terraform einen Splat-Ausdruck. Das Sternchen veranlasst Terraform, die Liste in einer Schleife zu durchlaufen und eine neue Liste zu erstellen, indem nur die Eigenschaft jedes Elements verwendet wird. `id`

```
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
}
```

Meta-Argumente

Im vorherigen Codebeispiel *`list_value_2`* werden sie als Argumente bezeichnet. *`list_value`* Möglicherweise sind Ihnen einige dieser Metaargumente bereits bekannt. Terraform hat auch einige Metaargumente, die sich wie Argumente verhalten, aber mit einigen zusätzlichen Funktionen:

- Das meta-Argument depends_on ist dem Attribut sehr ähnlich. CloudFormation DependsOn
- Das Provider-Metaargument ermöglicht es Ihnen, mehrere Anbieterkonfigurationen gleichzeitig zu verwenden.
- Mit dem Lifecycle-Metaargument können Sie Ressourceneinstellungen anpassen, ähnlich den Richtlinien zum Entfernen und Löschen von. CloudFormation

Andere Metaargumente ermöglichen das direkte Hinzufügen von Funktions- und Ausdrucksfunktionen zu einer Ressource. Das Metaargument count ist beispielsweise ein nützlicher Mechanismus, um mehrere ähnliche Ressourcen gleichzeitig zu erstellen. Das folgende Beispiel zeigt, wie Sie zwei Amazon Elastic Container Service (Amazon EKS) -Cluster erstellen, ohne das `count` Metaargument zu verwenden.

```
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]
  }
}
```

Das folgende Beispiel zeigt, wie das `count` Metaargument verwendet wird, um zwei Amazon EKS-Cluster zu erstellen.

```
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]
  }
}
```

Um jeder Einheit einen Namen zu geben, können Sie auf den Listenindex innerhalb des Ressourcenblocks unter zugreifen. `count.index` Was aber, wenn Sie mehrere ähnliche Ressourcen erstellen möchten, die etwas komplexer sind? Hier kommt das Meta-Argument for_each ins Spiel. Das `for_each` Meta-Argument ist sehr ähnlich`count`, außer dass Sie anstelle einer Zahl eine Liste oder ein Objekt übergeben. Terraform erstellt für jedes Mitglied der Liste oder des Objekts eine neue Ressource. Es ist ähnlich wie wenn Sie festlegen`count = length(list)`, außer dass Sie auf den Inhalt der Liste und nicht auf den Schleifenindex zugreifen können.

Dies funktioniert sowohl für eine Liste von Elementen als auch für ein einzelnes Objekt. Im folgenden Beispiel würden zwei Ressourcen mit `id-0` und `id-1` als IDs erstellt.

```
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
}
```

Im folgenden Beispiel würden ebenfalls zwei Ressourcen erstellt, eine für Sparky, den Pudel, und eine für Fluffy, den Chihuahua.

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

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

So wie Sie mithilfe von count.index auf den Loop-Index in count zugreifen können, können Sie mithilfe des Each-Objekts auf den Schlüssel und den Wert jedes Elements in einer for_each-Schleife zugreifen. Da for_each sowohl über Listen als auch über Objekte iteriert, kann es etwas verwirrend sein, den Überblick über die einzelnen Schlüssel und Werte zu behalten. Die folgende Tabelle zeigt die verschiedenen Möglichkeiten, wie Sie das for_each-Metaargument verwenden können und wie Sie bei jeder Iteration auf die Werte verweisen können.

Beispiel	**for_each**-Typ	Erste Iteration	Zweite Iteration

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

Beispiel	**for_each**-Typ	Erste Iteration	Zweite Iteration
B	`[ { type = "poodle", name = "Sparky" }, { type = "chihuahua", name = "Fluffy" } ]`	`each.key = { type = “poodle”, name = “Sparky” } each.value = null`	`each.key = { type = “chihuahua”, name = “Fluffy” } each.value = null`
C	`{ poodle = “Sparky”, chihuahua = “Fluffy” }`	`each.key = “poodle” each.value = “Sparky”`	`each.key = “chihuahua” each.value = “Fluffy”`

Beispiel	**for_each**-Typ	Erste Iteration	Zweite Iteration

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”`
			
			`}`

Beispiel	**for_each**-Typ	Erste Iteration	Zweite Iteration

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”
			
			}
			
			]

Beispiel	**for_each**-Typ	Erste Iteration	Zweite Iteration

	`name = “Morris”`
	
	`}`
	
	`]`
	
	`}`		

Wenn `var.animals` also Zeile E entspricht, könnten Sie mit dem folgenden Code eine Ressource pro Tier erstellen.

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

Alternativ könnten Sie zwei Ressourcen pro Tier erstellen, indem Sie den folgenden Code verwenden.

```
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
}
```

Häufig gestellte Fragen

Wann sollte ich Terraform anstelle von verwenden? CloudFormation

Wenn Ihre Workloads hauptsächlich darauf basieren, AWS CloudFormation bietet es im Allgemeinen ein Maß an nativer Unterstützung AWS, mit dem Terraform nicht mithalten kann. Wenn Ihre Workloads jedoch eine ganze Reihe von Prozessen von Drittanbietern beinhalten oder diese auf mehrere Cloud-Anbieter verteilt sind, ist Terraform ein Tool, das Sie vielleicht in Betracht ziehen sollten.

Wann sollte ich das anstelle von verwenden? AWS CDK CloudFormation

Wenn Sie das verwenden AWS Cloud Development Kit (AWS CDK), verwenden Sie auch CloudFormation. Das AWS CDK ermöglicht es Ihnen, eine gängige Programmiersprache zum Generieren von CloudFormation Vorlagen zu verwenden. Wenn Sie Erfahrung mit einer der von ihnen AWS CDK unterstützten Programmiersprachen haben, AWS CDK können Sie damit den Zeitaufwand für die Generierung von CloudFormation Vorlagen reduzieren.

Gibt es ein Tool wie das AWS CDK , das Terraform-Konfigurationen generiert?

Im Vergleich zu verwendet das AWS CDKCDK for Terraform (CDKTF) dieselbe Konstruktbibliothek zur Bereitstellung von Ressourcen und dieselbe JSII-Engine zur Unterstützung mehrerer Programmiersprachen. Sie können es verwenden, um Terraform-Konfigurationen auf die gleiche Weise zu generieren wie die generierten Vorlagen. AWS CDK CloudFormation

Wie erfahre ich mehr über Terraform?

Weitere Informationen zu fortgeschrittenen Terraform-Konzepten finden Sie in der Terraform-Dokumentation. Es beschreibt auch die Komponenten aller wichtigen Anbieter und Open-Source-Module.

Zugehörige Ressourcen

AWS Dokumentation

- AWS CDK -Dokumentation
- AWS CloudFormation -Dokumentation
- Terraform: Jenseits der Grundlagen mit AWS (AWS Blogbeitrag)

Sonstige Ressourcen

- CDK für Terraform-Dokumentation
- Terraform-Dokumentation

Anhang: Beispiele für den Zugriff auf Terraform-Attribute

Ressource

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

bucketName = aws_s3_bucket.myS3Bucket.bucket
```

Datenquelle

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

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

Modul

```
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
```

Dokumentverlauf

In der folgenden Tabelle werden wichtige Änderungen in diesem Leitfaden beschrieben. Um Benachrichtigungen über zukünftige Aktualisierungen zu erhalten, können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Erste Veröffentlichung	—	29. März 2024

AWS Glossar zu präskriptiven Leitlinien

Im Folgenden finden Sie häufig verwendete Begriffe in Strategien, Leitfäden und Mustern von AWS Prescriptive Guidance. Um Einträge vorzuschlagen, verwenden Sie bitte den Link Feedback geben am Ende des Glossars.

Zahlen

7 Rs

Sieben gängige Migrationsstrategien für die Verlagerung von Anwendungen in die Cloud. Diese Strategien bauen auf den 5 Rs auf, die Gartner 2011 identifiziert hat, und bestehen aus folgenden Elementen:

- Faktorwechsel/Architekturwechsel – Verschieben Sie eine Anwendung und ändern Sie ihre Architektur, indem Sie alle Vorteile cloudnativer Feature nutzen, um Agilität, Leistung und Skalierbarkeit zu verbessern. Dies beinhaltet in der Regel die Portierung des Betriebssystems und der Datenbank. Beispiel: Migrieren Sie Ihre lokale Oracle-Datenbank auf die Amazon Aurora PostgreSQL-kompatible Edition.
- Plattformwechsel (Lift and Reshape) – Verschieben Sie eine Anwendung in die Cloud und führen Sie ein gewisses Maß an Optimierung ein, um die Cloud-Funktionen zu nutzen. Beispiel: Migrieren Sie Ihre lokale Oracle-Datenbank zu Amazon Relational Database Service (Amazon RDS) für Oracle in der. AWS Cloud
- Neukauf (Drop and Shop) – Wechseln Sie zu einem anderen Produkt, indem Sie typischerweise von einer herkömmlichen Lizenz zu einem SaaS-Modell wechseln. Beispiel: Migrieren Sie Ihr CRM-System (Customer Relationship Management) zu Salesforce.com.
- Hostwechsel (Lift and Shift) – Verschieben Sie eine Anwendung in die Cloud, ohne Änderungen vorzunehmen, um die Cloud-Funktionen zu nutzen. Beispiel: Migrieren Sie Ihre lokale Oracle-Datenbank zu Oracle auf einer EC2-Instanz in der. AWS Cloud
- Verschieben (Lift and Shift auf Hypervisor-Ebene) – Verlagern Sie die Infrastruktur in die Cloud, ohne neue Hardware kaufen, Anwendungen umschreiben oder Ihre bestehenden Abläufe ändern zu müssen. Sie migrieren Server von einer lokalen Plattform zu einem Cloud-Dienst für dieselbe Plattform. Beispiel: Migrieren Sie eine Microsoft Hyper-V Anwendung zu AWS.
- Beibehaltung (Wiederaufgreifen) – Bewahren Sie Anwendungen in Ihrer Quellumgebung auf. Dazu können Anwendungen gehören, die einen umfangreichen Faktorwechsel erfordern und

die Sie auf einen späteren Zeitpunkt verschieben möchten, sowie ältere Anwendungen, die Sie beibehalten möchten, da es keine geschäftliche Rechtfertigung für ihre Migration gibt.

- Außerbetriebnahme – Dekommissionierung oder Entfernung von Anwendungen, die in Ihrer Quellumgebung nicht mehr benötigt werden.

A

ABAC

Siehe attributbasierte Zugriffskontrolle.

abstrahierte Dienste

Weitere Informationen finden Sie unter Managed Services.

ACID

Siehe Atomarität, Konsistenz, Isolierung und Haltbarkeit.

Aktiv-Aktiv-Migration

Eine Datenbankmigrationsmethode, bei der die Quell- und Zieldatenbanken synchron gehalten werden (mithilfe eines bidirektionalen Replikationstools oder dualer Schreibvorgänge) und beide Datenbanken Transaktionen von miteinander verbundenen Anwendungen während der Migration verarbeiten. Diese Methode unterstützt die Migration in kleinen, kontrollierten Batches, anstatt einen einmaligen Cutover zu erfordern. Es ist flexibler, erfordert aber mehr Arbeit als eine aktiv-passive Migration.

Aktiv-Passiv-Migration

Eine Datenbankmigrationsmethode, bei der die Quell- und Zieldatenbanken synchron gehalten werden, aber nur die Quelldatenbank Transaktionen von verbindenden Anwendungen verarbeitet, während Daten in die Zieldatenbank repliziert werden. Die Zieldatenbank akzeptiert während der Migration keine Transaktionen.

Aggregatfunktion

Eine SQL-Funktion, die mit einer Gruppe von Zeilen arbeitet und einen einzelnen Rückgabewert für die Gruppe berechnet. Beispiele für Aggregatfunktionen sind `SUM` und`MAX`.

AI

Siehe künstliche Intelligenz.

AIOps

Siehe Operationen mit künstlicher Intelligenz.

Anonymisierung

Der Prozess des dauerhaften Löschens personenbezogener Daten in einem Datensatz. Anonymisierung kann zum Schutz der Privatsphäre beitragen. Anonymisierte Daten gelten nicht mehr als personenbezogene Daten.

Anti-Muster

Eine häufig verwendete Lösung für ein wiederkehrendes Problem, bei dem die Lösung kontraproduktiv, ineffektiv oder weniger wirksam als eine Alternative ist.

Anwendungssteuerung

Ein Sicherheitsansatz, bei dem nur zugelassene Anwendungen verwendet werden können, um ein System vor Schadsoftware zu schützen.

Anwendungsportfolio

Eine Sammlung detaillierter Informationen zu jeder Anwendung, die von einer Organisation verwendet wird, einschließlich der Kosten für die Erstellung und Wartung der Anwendung und ihres Geschäftswerts. Diese Informationen sind entscheidend für den Prozess der Portfoliofindung und -analyse und hilft bei der Identifizierung und Priorisierung der Anwendungen, die migriert, modernisiert und optimiert werden sollen.

künstliche Intelligenz (KI)

Das Gebiet der Datenverarbeitungswissenschaft, das sich der Nutzung von Computertechnologien zur Ausführung kognitiver Funktionen widmet, die typischerweise mit Menschen in Verbindung gebracht werden, wie Lernen, Problemlösen und Erkennen von Mustern. Weitere Informationen finden Sie unter Was ist künstliche Intelligenz?

Operationen mit künstlicher Intelligenz (AIOps)

Der Prozess des Einsatzes von Techniken des Machine Learning zur Lösung betrieblicher Probleme, zur Reduzierung betrieblicher Zwischenfälle und menschlicher Eingriffe sowie zur Steigerung der Servicequalität. Weitere Informationen zur Verwendung von AIOps in der AWS -Migrationsstrategie finden Sie im Leitfaden zur Betriebsintegration.

Asymmetrische Verschlüsselung

Ein Verschlüsselungsalgorithmus, der ein Schlüsselpaar, einen öffentlichen Schlüssel für die Verschlüsselung und einen privaten Schlüssel für die Entschlüsselung verwendet. Sie können den öffentlichen Schlüssel teilen, da er nicht für die Entschlüsselung verwendet wird. Der Zugriff auf den privaten Schlüssel sollte jedoch stark eingeschränkt sein.

Atomizität, Konsistenz, Isolierung, Haltbarkeit (ACID)

Eine Reihe von Softwareeigenschaften, die die Datenvalidität und betriebliche Zuverlässigkeit einer Datenbank auch bei Fehlern, Stromausfällen oder anderen Problemen gewährleisten.

Attributbasierte Zugriffskontrolle (ABAC)

Die Praxis, detaillierte Berechtigungen auf der Grundlage von Benutzerattributen wie Abteilung, Aufgabenrolle und Teamname zu erstellen. Weitere Informationen finden Sie unter ABAC AWS in der AWS Identity and Access Management (IAM-) Dokumentation.

autoritative Datenquelle

Ein Ort, an dem Sie die primäre Version der Daten speichern, die als die zuverlässigste Informationsquelle angesehen wird. Sie können Daten aus der maßgeblichen Datenquelle an andere Speicherorte kopieren, um die Daten zu verarbeiten oder zu ändern, z. B. zu anonymisieren, zu redigieren oder zu pseudonymisieren.

Availability Zone

Ein bestimmter Standort innerhalb einer AWS-Region , der vor Ausfällen in anderen Availability Zones geschützt ist und kostengünstige Netzwerkkonnektivität mit niedriger Latenz zu anderen Availability Zones in derselben Region bietet.

AWS Framework für die Cloud-Einführung (AWS CAF)

Ein Framework mit Richtlinien und bewährten Verfahren, das Unternehmen bei der Entwicklung eines effizienten und effektiven Plans für den erfolgreichen Umstieg auf die Cloud unterstützt. AWS AWS CAF unterteilt die Leitlinien in sechs Schwerpunktbereiche, die als Perspektiven bezeichnet werden: Unternehmen, Mitarbeiter, Unternehmensführung, Plattform, Sicherheit und Betrieb. Die Perspektiven Geschäft, Mitarbeiter und Unternehmensführung konzentrieren sich auf Geschäftskompetenzen und -prozesse, während sich die Perspektiven Plattform, Sicherheit und Betriebsabläufe auf technische Fähigkeiten und Prozesse konzentrieren. Die Personalperspektive zielt beispielsweise auf Stakeholder ab, die sich mit Personalwesen (HR), Personalfunktionen und Personalmanagement befassen. Aus dieser Perspektive bietet AWS CAF Leitlinien für Personalentwicklung, Schulung und Kommunikation, um das Unternehmen auf eine erfolgreiche

Cloud-Einführung vorzubereiten. Weitere Informationen finden Sie auf der AWS -CAF-Webseite und dem AWS -CAF-Whitepaper.

AWS Workload-Qualifizierungsrahmen (AWS WQF)

Ein Tool, das Workloads bei der Datenbankmigration bewertet, Migrationsstrategien empfiehlt und Arbeitsschätzungen bereitstellt. AWS WQF ist in () enthalten. AWS Schema Conversion Tool AWS SCT Es analysiert Datenbankschemas und Codeobjekte, Anwendungscode, Abhängigkeiten und Leistungsmerkmale und stellt Bewertungsberichte bereit.

B

schlechter Bot

Ein Bot, der Einzelpersonen oder Organisationen stören oder ihnen Schaden zufügen soll.

BCP

Siehe Planung der Geschäftskontinuität.

Verhaltensdiagramm

Eine einheitliche, interaktive Ansicht des Ressourcenverhaltens und der Interaktionen im Laufe der Zeit. Sie können ein Verhaltensdiagramm mit Amazon Detective verwenden, um fehlgeschlagene Anmeldeversuche, verdächtige API-Aufrufe und ähnliche Vorgänge zu untersuchen. Weitere Informationen finden Sie unter Daten in einem Verhaltensdiagramm in der Detective-Dokumentation.

Big-Endian-System

Ein System, welches das höchstwertige Byte zuerst speichert. Siehe auch Endianness.

Binäre Klassifikation

Ein Prozess, der ein binäres Ergebnis vorhersagt (eine von zwei möglichen Klassen). Beispielsweise könnte Ihr ML-Modell möglicherweise Probleme wie „Handelt es sich bei dieser E-Mail um Spam oder nicht?“ vorhersagen müssen oder „Ist dieses Produkt ein Buch oder ein Auto?“

Bloom-Filter

Eine probabilistische, speichereffiziente Datenstruktur, mit der getestet wird, ob ein Element Teil einer Menge ist.

Blau/Grün-Bereitstellung

Eine Bereitstellungsstrategie, bei der Sie zwei separate, aber identische Umgebungen erstellen. Sie führen die aktuelle Anwendungsversion in einer Umgebung (blau) und die neue Anwendungsversion in der anderen Umgebung (grün) aus. Mit dieser Strategie können Sie schnell und mit minimalen Auswirkungen ein Rollback durchführen.

Bot

Eine Softwareanwendung, die automatisierte Aufgaben über das Internet ausführt und menschliche Aktivitäten oder Interaktionen simuliert. Manche Bots sind nützlich oder nützlich, wie z. B. Webcrawler, die Informationen im Internet indexieren. Einige andere Bots, die als bösartige Bots bezeichnet werden, sollen Einzelpersonen oder Organisationen stören oder ihnen Schaden zufügen.

Botnetz

Netzwerke von Bots, die mit Malware infiziert sind und unter der Kontrolle einer einzigen Partei stehen, die als Bot-Herder oder Bot-Operator bezeichnet wird. Botnetze sind der bekannteste Mechanismus zur Skalierung von Bots und ihrer Wirkung.

branch

Ein containerisierter Bereich eines Code-Repositorys. Der erste Zweig, der in einem Repository erstellt wurde, ist der Hauptzweig. Sie können einen neuen Zweig aus einem vorhandenen Zweig erstellen und dann Feature entwickeln oder Fehler in dem neuen Zweig beheben. Ein Zweig, den Sie erstellen, um ein Feature zu erstellen, wird allgemein als Feature-Zweig bezeichnet. Wenn das Feature zur Veröffentlichung bereit ist, führen Sie den Feature-Zweig wieder mit dem Hauptzweig zusammen. Weitere Informationen finden Sie unter Über Branches (GitHub Dokumentation).

Zugang durch Glasbruch

Unter außergewöhnlichen Umständen und im Rahmen eines genehmigten Verfahrens ist dies eine schnelle Methode für einen Benutzer, auf einen Bereich zuzugreifen AWS-Konto , für den er normalerweise keine Zugriffsrechte besitzt. Weitere Informationen finden Sie unter dem Indikator Implementation break-glass procedures in den AWS Well-Architected-Leitlinien.

Brownfield-Strategie

Die bestehende Infrastruktur in Ihrer Umgebung. Wenn Sie eine Brownfield-Strategie für eine Systemarchitektur anwenden, richten Sie sich bei der Gestaltung der Architektur nach den

Einschränkungen der aktuellen Systeme und Infrastruktur. Wenn Sie die bestehende Infrastruktur erweitern, könnten Sie Brownfield- und Greenfield-Strategien mischen.

Puffer-Cache

Der Speicherbereich, in dem die am häufigsten abgerufenen Daten gespeichert werden.

Geschäftsfähigkeit

Was ein Unternehmen tut, um Wert zu generieren (z. B. Vertrieb, Kundenservice oder Marketing). Microservices-Architekturen und Entwicklungsentscheidungen können von den Geschäftskapazitäten beeinflusst werden. Weitere Informationen finden Sie im Abschnitt Organisiert nach Geschäftskapazitäten des Whitepapers Ausführen von containerisierten Microservices in AWS.

Planung der Geschäftskontinuität (BCP)

Ein Plan, der die potenziellen Auswirkungen eines störenden Ereignisses, wie z. B. einer groß angelegten Migration, auf den Betrieb berücksichtigt und es einem Unternehmen ermöglicht, den Betrieb schnell wieder aufzunehmen.

C

CAF

Weitere Informationen finden Sie unter Framework für die AWS Cloud-Einführung.

Bereitstellung auf Kanaren

Die langsame und schrittweise Veröffentlichung einer Version für Endbenutzer. Wenn Sie sich sicher sind, stellen Sie die neue Version bereit und ersetzen die aktuelle Version vollständig.

CCoE

Weitere Informationen finden Sie im Cloud Center of Excellence.

CDC

Siehe Erfassung von Änderungsdaten.

Erfassung von Datenänderungen (CDC)

Der Prozess der Nachverfolgung von Änderungen an einer Datenquelle, z. B. einer Datenbanktabelle, und der Aufzeichnung von Metadaten zu der Änderung. Sie können CDC für

verschiedene Zwecke verwenden, z. B. für die Prüfung oder Replikation von Änderungen in einem Zielsystem, um die Synchronisation aufrechtzuerhalten.

Chaos-Technik

Absichtliches Einführen von Ausfällen oder Störungsereignissen, um die Widerstandsfähigkeit eines Systems zu testen. Sie können AWS Fault Injection Service (AWS FIS) verwenden, um Experimente durchzuführen, die Ihre AWS Workloads stress, und deren Reaktion zu bewerten.

CI/CD

Siehe Continuous Integration und Continuous Delivery.

Klassifizierung

Ein Kategorisierungsprozess, der bei der Erstellung von Vorhersagen hilft. ML-Modelle für Klassifikationsprobleme sagen einen diskreten Wert voraus. Diskrete Werte unterscheiden sich immer voneinander. Beispielsweise muss ein Modell möglicherweise auswerten, ob auf einem Bild ein Auto zu sehen ist oder nicht.

clientseitige Verschlüsselung

Lokale Verschlüsselung von Daten, bevor das Ziel sie AWS-Service empfängt.

Cloud-Kompetenzzentrum (CCoE)

Ein multidisziplinäres Team, das die Cloud-Einführung in der gesamten Organisation vorantreibt, einschließlich der Entwicklung bewährter Cloud-Methoden, der Mobilisierung von Ressourcen, der Festlegung von Migrationszeitplänen und der Begleitung der Organisation durch groß angelegte Transformationen. Weitere Informationen finden Sie in den CCoE-Beiträgen im AWS Cloud Enterprise Strategy Blog.

Cloud Computing

Die Cloud-Technologie, die typischerweise für die Ferndatenspeicherung und das IoT-Gerätemanagement verwendet wird. Cloud Computing ist häufig mit Edge-Computing-Technologie verbunden.

Cloud-Betriebsmodell

In einer IT-Organisation das Betriebsmodell, das zum Aufbau, zur Weiterentwicklung und Optimierung einer oder mehrerer Cloud-Umgebungen verwendet wird. Weitere Informationen finden Sie unter Aufbau Ihres Cloud-Betriebsmodells.

Phasen der Einführung der Cloud

Die vier Phasen, die Unternehmen bei der Migration in der Regel durchlaufen AWS Cloud:

- Projekt – Durchführung einiger Cloud-bezogener Projekte zu Machbarkeitsnachweisen und zu Lernzwecken
- Fundament – Grundlegende Investitionen tätigen, um Ihre Cloud-Einführung zu skalieren (z. B. Einrichtung einer Landing Zone, Definition eines CCoE, Einrichtung eines Betriebsmodells)
- Migration – Migrieren einzelner Anwendungen
- Neuentwicklung – Optimierung von Produkten und Services und Innovation in der Cloud

Diese Phasen wurden von Stephen Orban im Blogbeitrag The Journey Toward Cloud-First & the Stages of Adoption im AWS Cloud Enterprise Strategy-Blog definiert. Informationen darüber, wie sie mit der AWS Migrationsstrategie zusammenhängen, finden Sie im Leitfaden zur Vorbereitung der Migration.

CMDB

Siehe Datenbank für das Konfigurationsmanagement.

Code-Repository

Ein Ort, an dem Quellcode und andere Komponenten wie Dokumentation, Beispiele und Skripts gespeichert und im Rahmen von Versionskontrollprozessen aktualisiert werden. Zu den gängigen Cloud-Repositorys gehören GitHub oder AWS CodeCommit. Jede Version des Codes wird Zweig genannt. In einer Microservice-Struktur ist jedes Repository einer einzelnen Funktionalität gewidmet. Eine einzelne CI/CD-Pipeline kann mehrere Repositorien verwenden.

Kalter Cache

Ein Puffer-Cache, der leer oder nicht gut gefüllt ist oder veraltete oder irrelevante Daten enthält. Dies beeinträchtigt die Leistung, da die Datenbank-Instance aus dem Hauptspeicher oder der Festplatte lesen muss, was langsamer ist als das Lesen aus dem Puffercache.

Kalte Daten

Daten, auf die selten zugegriffen wird und die in der Regel historisch sind. Bei der Abfrage dieser Art von Daten sind langsame Abfragen in der Regel akzeptabel. Durch die Verlagerung dieser Daten auf leistungsschwächere und kostengünstigere Speicherstufen oder -klassen können Kosten gesenkt werden.

Computer Vision (CV)

Ein Bereich der KI, der maschinelles Lernen nutzt, um Informationen aus visuellen Formaten wie digitalen Bildern und Videos zu analysieren und zu extrahieren. AWS Panorama Bietet beispielsweise Geräte an, die CV zu lokalen Kameranetzwerken hinzufügen, und Amazon SageMaker stellt Bildverarbeitungsalgorithmen für CV bereit.

Drift in der Konfiguration

Bei einer Arbeitslast eine Änderung der Konfiguration gegenüber dem erwarteten Zustand. Dies kann dazu führen, dass der Workload nicht mehr richtlinienkonform wird, und zwar in der Regel schrittweise und unbeabsichtigt.

Verwaltung der Datenbankkonfiguration (CMDB)

Ein Repository, das Informationen über eine Datenbank und ihre IT-Umgebung speichert und verwaltet, inklusive Hardware- und Softwarekomponenten und deren Konfigurationen. In der Regel verwenden Sie Daten aus einer CMDB in der Phase der Portfolioerkennung und -analyse der Migration.

Konformitätspaket

Eine Sammlung von AWS Config Regeln und Abhilfemaßnahmen, die Sie zusammenstellen können, um Ihre Konformitäts- und Sicherheitsprüfungen individuell anzupassen. Mithilfe einer YAML-Vorlage können Sie ein Conformance Pack als einzelne Entität in einer AWS-Konto AND-Region oder unternehmensweit bereitstellen. Weitere Informationen finden Sie in der Dokumentation unter Conformance Packs. AWS Config

Kontinuierliche Bereitstellung und kontinuierliche Integration (CI/CD)

Der Prozess der Automatisierung der Quell-, Build-, Test-, Staging- und Produktionsphasen des Softwareveröffentlichungsprozesses. CI/CD wird allgemein als Pipeline beschrieben. CI/CD kann Ihnen helfen, Prozesse zu automatisieren, die Produktivität zu steigern, die Codequalität zu verbessern und schneller zu liefern. Weitere Informationen finden Sie unter Vorteile der kontinuierlichen Auslieferung. CD kann auch für kontinuierliche Bereitstellung stehen. Weitere Informationen finden Sie unterKontinuierliche Auslieferung im Vergleich zu kontinuierlicher Bereitstellung.

CV

Siehe Computer Vision.

D

Daten im Ruhezustand

Daten, die in Ihrem Netzwerk stationär sind, z. B. Daten, die sich im Speicher befinden.

Datenklassifizierung

Ein Prozess zur Identifizierung und Kategorisierung der Daten in Ihrem Netzwerk auf der Grundlage ihrer Kritikalität und Sensitivität. Sie ist eine wichtige Komponente jeder Strategie für das Management von Cybersecurity-Risiken, da sie Ihnen hilft, die geeigneten Schutz- und Aufbewahrungskontrollen für die Daten zu bestimmen. Die Datenklassifizierung ist ein Bestandteil der Sicherheitssäule im AWS Well-Architected Framework. Weitere Informationen finden Sie unter Datenklassifizierung.

Datendrift

Eine signifikante Abweichung zwischen den Produktionsdaten und den Daten, die zum Trainieren eines ML-Modells verwendet wurden, oder eine signifikante Änderung der Eingabedaten im Laufe der Zeit. Datendrift kann die Gesamtqualität, Genauigkeit und Fairness von ML-Modellvorhersagen beeinträchtigen.

Daten während der Übertragung

Daten, die sich aktiv durch Ihr Netzwerk bewegen, z. B. zwischen Netzwerkressourcen.

Datennetz

Ein architektonisches Framework, das verteilte, dezentrale Dateneigentum mit zentraler Verwaltung und Steuerung ermöglicht.

Datenminimierung

Das Prinzip, nur die Daten zu sammeln und zu verarbeiten, die unbedingt erforderlich sind. Durch Datenminimierung im AWS Cloud können Datenschutzrisiken, Kosten und der CO2-Fußabdruck Ihrer Analysen reduziert werden.

Datenperimeter

Eine Reihe präventiver Schutzmaßnahmen in Ihrer AWS Umgebung, die sicherstellen, dass nur vertrauenswürdige Identitäten auf vertrauenswürdige Ressourcen von erwarteten Netzwerken zugreifen. Weitere Informationen finden Sie unter Aufbau eines Datenperimeters auf. AWS

Vorverarbeitung der Daten

Rohdaten in ein Format umzuwandeln, das von Ihrem ML-Modell problemlos verarbeitet werden kann. Die Vorverarbeitung von Daten kann bedeuten, dass bestimmte Spalten oder Zeilen entfernt und fehlende, inkonsistente oder doppelte Werte behoben werden.

Herkunft der Daten

Der Prozess der Nachverfolgung des Ursprungs und der Geschichte von Daten während ihres gesamten Lebenszyklus, z. B. wie die Daten generiert, übertragen und gespeichert wurden.

betroffene Person

Eine Person, deren Daten gesammelt und verarbeitet werden.

Data Warehouse

Ein Datenverwaltungssystem, das Business Intelligence wie Analysen unterstützt. Data Warehouses enthalten in der Regel große Mengen an historischen Daten und werden in der Regel für Abfragen und Analysen verwendet.

Datenbankdefinitionssprache (DDL)

Anweisungen oder Befehle zum Erstellen oder Ändern der Struktur von Tabellen und Objekten in einer Datenbank.

Datenbankmanipulationssprache (DML)

Anweisungen oder Befehle zum Ändern (Einfügen, Aktualisieren und Löschen) von Informationen in einer Datenbank.

DDL

Siehe Datenbankdefinitionssprache.

Deep-Ensemble

Mehrere Deep-Learning-Modelle zur Vorhersage kombinieren. Sie können Deep-Ensembles verwenden, um eine genauere Vorhersage zu erhalten oder um die Unsicherheit von Vorhersagen abzuschätzen.

Deep Learning

Ein ML-Teilbereich, der mehrere Schichten künstlicher neuronaler Netzwerke verwendet, um die Zuordnung zwischen Eingabedaten und Zielvariablen von Interesse zu ermitteln.

defense-in-depth

Ein Ansatz zur Informationssicherheit, bei dem eine Reihe von Sicherheitsmechanismen und -kontrollen sorgfältig in einem Computernetzwerk verteilt werden, um die Vertraulichkeit, Integrität und Verfügbarkeit des Netzwerks und der darin enthaltenen Daten zu schützen. Wenn Sie diese Strategie anwenden AWS, fügen Sie mehrere Steuerelemente auf verschiedenen Ebenen der AWS Organizations Struktur hinzu, um die Ressourcen zu schützen. Ein defense-in-depth Ansatz könnte beispielsweise Multi-Faktor-Authentifizierung, Netzwerksegmentierung und Verschlüsselung kombinieren.

delegierter Administrator

In AWS Organizations kann ein kompatibler Dienst ein AWS Mitgliedskonto registrieren, um die Konten der Organisation und die Berechtigungen für diesen Dienst zu verwalten. Dieses Konto wird als delegierter Administrator für diesen Service bezeichnet. Weitere Informationen und eine Liste kompatibler Services finden Sie unter Services, die mit AWS Organizations funktionieren in der AWS Organizations -Dokumentation.

Bereitstellung

Der Prozess, bei dem eine Anwendung, neue Feature oder Codekorrekturen in der Zielumgebung verfügbar gemacht werden. Die Bereitstellung umfasst das Implementieren von Änderungen an einer Codebasis und das anschließende Erstellen und Ausführen dieser Codebasis in den Anwendungsumgebungen.

Entwicklungsumgebung

Siehe Umgebung.

Detektivische Kontrolle

Eine Sicherheitskontrolle, die darauf ausgelegt ist, ein Ereignis zu erkennen, zu protokollieren und zu warnen, nachdem ein Ereignis eingetreten ist. Diese Kontrollen stellen eine zweite Verteidigungslinie dar und warnen Sie vor Sicherheitsereignissen, bei denen die vorhandenen präventiven Kontrollen umgangen wurden. Weitere Informationen finden Sie unter Detektivische Kontrolle in Implementierung von Sicherheitskontrollen in AWS.

Abbildung des Wertstroms in der Entwicklung (DVSM)

Ein Prozess zur Identifizierung und Priorisierung von Einschränkungen, die sich negativ auf Geschwindigkeit und Qualität im Lebenszyklus der Softwareentwicklung auswirken. DVSM erweitert den Prozess der Wertstromanalyse, der ursprünglich für Lean-Manufacturing-Praktiken

konzipiert wurde. Es konzentriert sich auf die Schritte und Teams, die erforderlich sind, um durch den Softwareentwicklungsprozess Mehrwert zu schaffen und zu steigern.

digitaler Zwilling

Eine virtuelle Darstellung eines realen Systems, z. B. eines Gebäudes, einer Fabrik, einer Industrieanlage oder einer Produktionslinie. Digitale Zwillinge unterstützen vorausschauende Wartung, Fernüberwachung und Produktionsoptimierung.

Maßtabelle

In einem Sternschema eine kleinere Tabelle, die Datenattribute zu quantitativen Daten in einer Faktentabelle enthält. Bei Attributen von Dimensionstabellen handelt es sich in der Regel um Textfelder oder diskrete Zahlen, die sich wie Text verhalten. Diese Attribute werden häufig zum Einschränken von Abfragen, zum Filtern und zur Kennzeichnung von Ergebnismengen verwendet.

Katastrophe

Ein Ereignis, das verhindert, dass ein Workload oder ein System seine Geschäftsziele an seinem primären Einsatzort erfüllt. Diese Ereignisse können Naturkatastrophen, technische Ausfälle oder das Ergebnis menschlichen Handelns sein, z. B. unbeabsichtigte Fehlkonfigurationen oder Malware-Angriffe.

Notfallwiederherstellung (DR)

Die Strategie und der Prozess, die Sie zur Minimierung von Ausfallzeiten und Datenverlusten aufgrund einer Katastrophe anwenden. Weitere Informationen finden Sie unter Disaster Recovery von Workloads unter AWS: Wiederherstellung in der Cloud im AWS Well-Architected Framework.

DML

Siehe Sprache zur Datenbankmanipulation.

Domainorientiertes Design

Ein Ansatz zur Entwicklung eines komplexen Softwaresystems, bei dem seine Komponenten mit sich entwickelnden Domains oder Kerngeschäftszielen verknüpft werden, denen jede Komponente dient. Dieses Konzept wurde von Eric Evans in seinem Buch Domaingesteuertes Design: Bewältigen der Komplexität im Herzen der Software (Boston: Addison-Wesley Professional, 2003) vorgestellt. Informationen darüber, wie Sie domaingesteuertes Design mit dem Strangler-Fig-Muster verwenden können, finden Sie unter Schrittweises Modernisieren älterer Microsoft ASP.NET (ASMX)-Webservices mithilfe von Containern und Amazon API Gateway.

DR

Siehe Disaster Recovery.

Erkennung von Driften

Verfolgung von Abweichungen von einer Basiskonfiguration Sie können es beispielsweise verwenden, AWS CloudFormation um Abweichungen bei den Systemressourcen zu erkennen, oder Sie können AWS Control Tower damit Änderungen in Ihrer landing zone erkennen, die sich auf die Einhaltung von Governance-Anforderungen auswirken könnten.

DVSM

Siehe Abbildung des Wertstroms in der Entwicklung.

E

EDA

Siehe explorative Datenanalyse.

Edge-Computing

Die Technologie, die die Rechenleistung für intelligente Geräte an den Rändern eines IoT-Netzwerks erhöht. Im Vergleich zu Cloud Computing kann Edge Computing die Kommunikationslatenz reduzieren und die Reaktionszeit verbessern.

Verschlüsselung

Ein Rechenprozess, der Klartextdaten, die für Menschen lesbar sind, in Chiffretext umwandelt.

Verschlüsselungsschlüssel

Eine kryptografische Zeichenfolge aus zufälligen Bits, die von einem Verschlüsselungsalgorithmus generiert wird. Schlüssel können unterschiedlich lang sein, und jeder Schlüssel ist so konzipiert, dass er unvorhersehbar und einzigartig ist.

Endianismus

Die Reihenfolge, in der Bytes im Computerspeicher gespeichert werden. Big-Endian-Systeme speichern das höchstwertige Byte zuerst. Little-Endian-Systeme speichern das niedrigwertigste Byte zuerst.

Endpunkt

Siehe Service-Endpunkt.

Endpunkt-Services

Ein Service, den Sie in einer Virtual Private Cloud (VPC) hosten können, um ihn mit anderen Benutzern zu teilen. Sie können einen Endpunktdienst mit anderen AWS-Konten oder AWS Identity and Access Management (IAM AWS PrivateLink -) Prinzipalen erstellen und diesen Berechtigungen gewähren. Diese Konten oder Prinzipale können sich privat mit Ihrem Endpunktservice verbinden, indem sie Schnittstellen-VPC-Endpunkte erstellen. Weitere Informationen finden Sie unter Einen Endpunkt-Service erstellen in der Amazon Virtual Private Cloud (Amazon VPC)-Dokumentation.

Unternehmensressourcenplanung (ERP)

Ein System, das wichtige Geschäftsprozesse (wie Buchhaltung, MES und Projektmanagement) für ein Unternehmen automatisiert und verwaltet.

Envelope-Verschlüsselung

Der Prozess der Verschlüsselung eines Verschlüsselungsschlüssels mit einem anderen Verschlüsselungsschlüssel. Weitere Informationen finden Sie unter Envelope-Verschlüsselung in der AWS Key Management Service (AWS KMS) -Dokumentation.

Umgebung

Eine Instance einer laufenden Anwendung. Die folgenden Arten von Umgebungen sind beim Cloud-Computing üblich:

- Entwicklungsumgebung – Eine Instance einer laufenden Anwendung, die nur dem Kernteam zur Verfügung steht, das für die Wartung der Anwendung verantwortlich ist. Entwicklungsumgebungen werden verwendet, um Änderungen zu testen, bevor sie in höhere Umgebungen übertragen werden. Diese Art von Umgebung wird manchmal als Testumgebung bezeichnet.
- Niedrigere Umgebungen – Alle Entwicklungsumgebungen für eine Anwendung, z. B. solche, die für erste Builds und Tests verwendet wurden.
- Produktionsumgebung – Eine Instance einer laufenden Anwendung, auf die Endbenutzer zugreifen können. In einer CI/CD-Pipeline ist die Produktionsumgebung die letzte Bereitstellungsumgebung.

- Höhere Umgebungen – Alle Umgebungen, auf die auch andere Benutzer als das Kernentwicklungsteam zugreifen können. Dies kann eine Produktionsumgebung, Vorproduktionsumgebungen und Umgebungen für Benutzerakzeptanztests umfassen.

Epics

In der agilen Methodik sind dies funktionale Kategorien, die Ihnen helfen, Ihre Arbeit zu organisieren und zu priorisieren. Epics bieten eine allgemeine Beschreibung der Anforderungen und Implementierungsaufgaben. Zu den Sicherheitsthemen AWS von CAF gehören beispielsweise Identitäts- und Zugriffsmanagement, Detektivkontrollen, Infrastruktursicherheit, Datenschutz und Reaktion auf Vorfälle. Weitere Informationen zu Epics in der AWS - Migrationsstrategie finden Sie im Leitfaden zur Programm-Implementierung.

ERP

Siehe Enterprise Resource Planning.

Explorative Datenanalyse (EDA)

Der Prozess der Analyse eines Datensatzes, um seine Hauptmerkmale zu verstehen. Sie sammeln oder aggregieren Daten und führen dann erste Untersuchungen durch, um Muster zu finden, Anomalien zu erkennen und Annahmen zu überprüfen. EDA wird durchgeführt, indem zusammenfassende Statistiken berechnet und Datenvisualisierungen erstellt werden.

F

Faktentabelle

Die zentrale Tabelle in einem Sternschema. Sie speichert quantitative Daten über den Geschäftsbetrieb. In der Regel enthält eine Faktentabelle zwei Arten von Spalten: Spalten, die Kennzahlen enthalten, und Spalten, die einen Fremdschlüssel für eine Dimensionstabelle enthalten.

schnell scheitern

Eine Philosophie, die häufige und inkrementelle Tests verwendet, um den Entwicklungslebenszyklus zu verkürzen. Dies ist ein wichtiger Bestandteil eines agilen Ansatzes.

Grenze zur Fehlerisolierung

Dabei handelt es sich um eine Grenze AWS Cloud, z. B. eine Availability Zone AWS-Region, eine Steuerungsebene oder eine Datenebene, die die Auswirkungen eines Fehlers begrenzt und die

Widerstandsfähigkeit von Workloads verbessert. Weitere Informationen finden Sie unter Grenzen zur AWS Fehlerisolierung.

Feature-Zweig

Siehe Zweig.

Features

Die Eingabedaten, die Sie verwenden, um eine Vorhersage zu treffen. In einem Fertigungskontext könnten Feature beispielsweise Bilder sein, die regelmäßig von der Fertigungslinie aus aufgenommen werden.

Bedeutung der Feature

Wie wichtig ein Feature für die Vorhersagen eines Modells ist. Dies wird in der Regel als numerischer Wert ausgedrückt, der mit verschiedenen Techniken wie Shapley Additive Explanations (SHAP) und integrierten Gradienten berechnet werden kann. Weitere Informationen finden Sie unter Interpretierbarkeit von Modellen für maschinelles Lernen mit:AWS.

Featuretransformation

Daten für den ML-Prozess optimieren, einschließlich der Anreicherung von Daten mit zusätzlichen Quellen, der Skalierung von Werten oder der Extraktion mehrerer Informationssätze aus einem einzigen Datenfeld. Das ermöglicht dem ML-Modell, von den Daten profitieren. Wenn Sie beispielsweise das Datum „27.05.2021 00:15:37“ in „2021“, „Mai“, „Donnerstag“ und „15“ aufschlüsseln, können Sie dem Lernalgorithmus helfen, nuancierte Muster zu erlernen, die mit verschiedenen Datenkomponenten verknüpft sind.

FGAC

Weitere Informationen finden Sie unter detaillierter Zugriffskontrolle.

Feinkörnige Zugriffskontrolle (FGAC)

Die Verwendung mehrerer Bedingungen, um eine Zugriffsanfrage zuzulassen oder abzulehnen.

Flash-Cut-Migration

Eine Datenbankmigrationsmethode, bei der eine kontinuierliche Datenreplikation durch Erfassung von Änderungsdaten verwendet wird, um Daten in kürzester Zeit zu migrieren, anstatt einen schrittweisen Ansatz zu verwenden. Ziel ist es, Ausfallzeiten auf ein Minimum zu beschränken.

G

Geoblocking

Siehe geografische Einschränkungen.

Geografische Einschränkungen (Geoblocking)

Bei Amazon eine Option CloudFront, um zu verhindern, dass Benutzer in bestimmten Ländern auf Inhaltsverteilungen zugreifen. Sie können eine Zulassungsliste oder eine Sperrliste verwenden, um zugelassene und gesperrte Länder anzugeben. Weitere Informationen finden Sie in der Dokumentation unter Beschränkung der geografischen Verteilung Ihrer Inhalte. CloudFront

Gitflow-Workflow

Ein Ansatz, bei dem niedrigere und höhere Umgebungen unterschiedliche Zweige in einem Quellcode-Repository verwenden. Der Gitflow-Workflow gilt als veraltet, und der Trunk-basierte Workflow ist der moderne, bevorzugte Ansatz.

Greenfield-Strategie

Das Fehlen vorhandener Infrastruktur in einer neuen Umgebung. Bei der Einführung einer Neuausrichtung einer Systemarchitektur können Sie alle neuen Technologien ohne Einschränkung der Kompatibilität mit der vorhandenen Infrastruktur auswählen, auch bekannt als Brownfield. Wenn Sie die bestehende Infrastruktur erweitern, könnten Sie Brownfield- und Greenfield-Strategien mischen.

Integritätsschutz

Eine allgemeine Regel, die dabei hilft, Ressourcen, Richtlinien und die Einhaltung von Vorschriften in allen Organisationseinheiten (OUs) zu regeln. Präventiver Integritätsschutz setzt Richtlinien durch, um die Einhaltung von Standards zu gewährleisten. Sie werden mithilfe von Service-Kontrollrichtlinien und IAM-Berechtigungsgrenzen implementiert. Detektivischer Integritätsschutz erkennt Richtlinienverstöße und Compliance-Probleme und generiert Warnmeldungen zur Abhilfe. Sie werden mithilfe von AWS Config, AWS Security Hub, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector und benutzerdefinierten AWS Lambda Prüfungen implementiert.

H

HEKTAR

Siehe Hochverfügbarkeit.

Heterogene Datenbankmigration

Migrieren Sie Ihre Quelldatenbank in eine Zieldatenbank, die eine andere Datenbank-Engine verwendet (z. B. Oracle zu Amazon Aurora). Eine heterogene Migration ist in der Regel Teil einer Neuarchitektur, und die Konvertierung des Schemas kann eine komplexe Aufgabe sein. AWS bietet AWS SCT, welches bei Schemakonvertierungen hilft.

hohe Verfügbarkeit (HA)

Die Fähigkeit eines Workloads, im Falle von Herausforderungen oder Katastrophen kontinuierlich und ohne Eingreifen zu arbeiten. HA-Systeme sind so konzipiert, dass sie automatisch ein Failover durchführen, gleichbleibend hohe Leistung bieten und unterschiedliche Lasten und Ausfälle mit minimalen Leistungseinbußen bewältigen.

historische Modernisierung

Ein Ansatz zur Modernisierung und Aufrüstung von Betriebstechnologiesystemen (OT), um den Bedürfnissen der Fertigungsindustrie besser gerecht zu werden. Ein Historian ist eine Art von Datenbank, die verwendet wird, um Daten aus verschiedenen Quellen in einer Fabrik zu sammeln und zu speichern.

Homogene Datenbankmigration

Migrieren Sie Ihre Quelldatenbank zu einer Zieldatenbank, die dieselbe Datenbank-Engine verwendet (z. B. Microsoft SQL Server zu Amazon RDS für SQL Server). Eine homogene Migration ist in der Regel Teil eines Hostwechsels oder eines Plattformwechsels. Sie können native Datenbankserviceprogramme verwenden, um das Schema zu migrieren.

heiße Daten

Daten, auf die häufig zugegriffen wird, z. B. Echtzeitdaten oder aktuelle Translationsdaten. Für diese Daten ist in der Regel eine leistungsstarke Speicherebene oder -klasse erforderlich, um schnelle Abfrageantworten zu ermöglichen.

Hotfix

Eine dringende Lösung für ein kritisches Problem in einer Produktionsumgebung. Aufgrund seiner Dringlichkeit wird ein Hotfix normalerweise außerhalb des typischen DevOps Release-Workflows erstellt.

Hypercare-Phase

Unmittelbar nach dem Cutover, der Zeitraum, in dem ein Migrationsteam die migrierten Anwendungen in der Cloud verwaltet und überwacht, um etwaige Probleme zu beheben. In der Regel dauert dieser Zeitraum 1–4 Tage. Am Ende der Hypercare-Phase überträgt das Migrationsteam in der Regel die Verantwortung für die Anwendungen an das Cloud-Betriebsteam.

I

IaC

Sehen Sie sich Infrastruktur als Code an.

Identitätsbasierte Richtlinie

Eine Richtlinie, die einem oder mehreren IAM-Prinzipalen zugeordnet ist und deren Berechtigungen innerhalb der AWS Cloud Umgebung definiert.

Leerlaufanwendung

Eine Anwendung mit einer durchschnittlichen CPU- und Arbeitsspeicherauslastung zwischen 5 und 20 Prozent über einen Zeitraum von 90 Tagen. In einem Migrationsprojekt ist es üblich, diese Anwendungen außer Betrieb zu nehmen oder sie On-Premises beizubehalten.

IIoT

Siehe Industrielles Internet der Dinge.

unveränderliche Infrastruktur

Ein Modell, das eine neue Infrastruktur für Produktionsworkloads bereitstellt, anstatt die bestehende Infrastruktur zu aktualisieren, zu patchen oder zu modifizieren. Unveränderliche Infrastrukturen sind von Natur aus konsistenter, zuverlässiger und vorhersehbarer als veränderliche Infrastrukturen. Weitere Informationen finden Sie in der Best Practice Deploy using immutable infrastructure im AWS Well-Architected Framework.

Eingehende (ingress) VPC

In einer Architektur AWS mit mehreren Konten ist dies eine VPC, die Netzwerkverbindungen von außerhalb einer Anwendung akzeptiert, überprüft und weiterleitet. Die AWS -Referenzarchitektur für die Sicherheit empfiehlt, Ihr Netzwerkkonto mit eingehenden und ausgehenden VPCs und Inspektions-VPCs einzurichten, um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet zu schützen.

Inkrementelle Migration

Eine Cutover-Strategie, bei der Sie Ihre Anwendung in kleinen Teilen migrieren, anstatt eine einziges vollständiges Cutover durchzuführen. Beispielsweise könnten Sie zunächst nur einige Microservices oder Benutzer auf das neue System umstellen. Nachdem Sie sich vergewissert haben, dass alles ordnungsgemäß funktioniert, können Sie weitere Microservices oder Benutzer schrittweise verschieben, bis Sie Ihr Legacy-System außer Betrieb nehmen können. Diese Strategie reduziert die mit großen Migrationen verbundenen Risiken.

Industrie 4.0

Ein Begriff, der 2016 von Klaus Schwab eingeführt wurde und sich auf die Modernisierung von Fertigungsprozessen durch Fortschritte in den Bereichen Konnektivität, Echtzeitdaten, Automatisierung, Analytik und KI/ML bezieht.

Infrastruktur

Alle Ressourcen und Komponenten, die in der Umgebung einer Anwendung enthalten sind.

Infrastructure as Code (IaC)

Der Prozess der Bereitstellung und Verwaltung der Infrastruktur einer Anwendung mithilfe einer Reihe von Konfigurationsdateien. IaC soll Ihnen helfen, das Infrastrukturmanagement zu zentralisieren, Ressourcen zu standardisieren und schnell zu skalieren, sodass neue Umgebungen wiederholbar, zuverlässig und konsistent sind.

Industrielles Internet der Dinge (IIoT)

Einsatz von mit dem Internet verbundenen Sensoren und Geräten in Industriesektoren wie Fertigung, Energie, Automobilindustrie, Gesundheitswesen, Biowissenschaften und Landwirtschaft. Mehr Informationen finden Sie unter Aufbau einer digitalen Transformationsstrategie für das industrielle Internet der Dinge (IIoT).

Inspektions-VPC

In einer Architektur AWS mit mehreren Konten eine zentralisierte VPC, die Inspektionen des Netzwerkverkehrs zwischen VPCs (in derselben oder unterschiedlichen AWS-Regionen), dem Internet und lokalen Netzwerken verwaltet. Die AWS -Referenzarchitektur für die Sicherheit empfiehlt, Ihr Netzwerkkonto mit eingehenden und ausgehenden VPCs und Inspektions-VPCs einzurichten, um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet zu schützen.

Internet of Things (IoT)

Das Netzwerk verbundener physischer Objekte mit eingebetteten Sensoren oder Prozessoren, das über das Internet oder über ein lokales Kommunikationsnetzwerk mit anderen Geräten und Systemen kommuniziert. Weitere Informationen finden Sie unter Was ist IoT?

Interpretierbarkeit

Ein Merkmal eines Modells für Machine Learning, das beschreibt, inwieweit ein Mensch verstehen kann, wie die Vorhersagen des Modells von seinen Eingaben abhängen. Weitere Informationen finden Sie unter Interpretierbarkeit von Modellen für Machine Learning mit AWS.

IoT

Siehe Internet der Dinge.

IT information library (ITIL, IT-Informationsbibliothek)

Eine Reihe von bewährten Methoden für die Bereitstellung von IT-Services und die Abstimmung dieser Services auf die Geschäftsanforderungen. ITIL bietet die Grundlage für ITSM.

T service management (ITSM, IT-Servicemanagement)

Aktivitäten im Zusammenhang mit der Gestaltung, Implementierung, Verwaltung und Unterstützung von IT-Services für eine Organisation. Informationen zur Integration von Cloud-Vorgängen mit ITSM-Tools finden Sie im Leitfaden zur Betriebsintegration.

BIS

Weitere Informationen finden Sie in der IT-Informationsbibliothek.

ITSM

Siehe IT-Servicemanagement.

L

Labelbasierte Zugangskontrolle (LBAC)

Eine Implementierung der Mandatory Access Control (MAC), bei der den Benutzern und den Daten selbst jeweils explizit ein Sicherheitslabelwert zugewiesen wird. Die Schnittmenge zwischen der Benutzersicherheitsbeschriftung und der Datensicherheitsbeschriftung bestimmt, welche Zeilen und Spalten für den Benutzer sichtbar sind.

Landing Zone

Eine landing zone ist eine gut strukturierte AWS Umgebung mit mehreren Konten, die skalierbar und sicher ist. Dies ist ein Ausgangspunkt, von dem aus Ihre Organisationen Workloads und Anwendungen schnell und mit Vertrauen in ihre Sicherheits- und Infrastrukturumgebung starten und bereitstellen können. Weitere Informationen zu Landing Zones finden Sie unter Einrichtung einer sicheren und skalierbaren AWS -Umgebung mit mehreren Konten..

Große Migration

Eine Migration von 300 oder mehr Servern.

SCHWARZ

Weitere Informationen finden Sie unter Label-basierte Zugriffskontrolle.

Geringste Berechtigung

Die bewährte Sicherheitsmethode, bei der nur die für die Durchführung einer Aufgabe erforderlichen Mindestberechtigungen erteilt werden. Weitere Informationen finden Sie unter Geringste Berechtigungen anwenden in der IAM-Dokumentation.

Lift and Shift

Siehe 7 Rs.

Little-Endian-System

Ein System, welches das niedrigwertigste Byte zuerst speichert. Siehe auch Endianness.

Niedrigere Umgebungen

Siehe Umwelt.

M

Machine Learning (ML)

Eine Art künstlicher Intelligenz, die Algorithmen und Techniken zur Mustererkennung und zum Lernen verwendet. ML analysiert aufgezeichnete Daten, wie z. B. Daten aus dem Internet der Dinge (IoT), und lernt daraus, um ein statistisches Modell auf der Grundlage von Mustern zu erstellen. Weitere Informationen finden Sie unter Machine Learning.

Hauptzweig

Siehe Filiale.

Malware

Software, die entwickelt wurde, um die Computersicherheit oder den Datenschutz zu gefährden. Malware kann Computersysteme stören, vertrauliche Informationen durchsickern lassen oder sich unbefugten Zugriff verschaffen. Beispiele für Malware sind Viren, Würmer, Ransomware, Trojaner, Spyware und Keylogger.

verwaltete Dienste

AWS-Services für die die Infrastrukturebene, das Betriebssystem und die Plattformen AWS betrieben werden, und Sie greifen auf die Endgeräte zu, um Daten zu speichern und abzurufen. Amazon Simple Storage Service (Amazon S3) und Amazon DynamoDB sind Beispiele für Managed Services. Diese werden auch als abstrakte Dienste bezeichnet.

Manufacturing Execution System (MES)

Ein Softwaresystem zur Nachverfolgung, Überwachung, Dokumentation und Steuerung von Produktionsprozessen, bei denen Rohstoffe in der Fertigung zu fertigen Produkten umgewandelt werden.

MAP

Siehe Migration Acceleration Program.

Mechanismus

Ein vollständiger Prozess, bei dem Sie ein Tool erstellen, die Akzeptanz des Tools vorantreiben und anschließend die Ergebnisse überprüfen, um Anpassungen vorzunehmen. Ein Mechanismus ist ein Zyklus, der sich im Laufe seiner Tätigkeit selbst verstärkt und verbessert. Weitere Informationen finden Sie unter Aufbau von Mechanismen im AWS Well-Architected Framework.

Mitgliedskonto

Alle AWS-Konten außer dem Verwaltungskonto, die Teil einer Organisation in sind. AWS Organizations Ein Konto kann jeweils nur einer Organisation angehören.

DURCHEINANDER

Siehe Manufacturing Execution System.

Message Queuing-Telemetrietransport (MQTT)

Ein leichtes machine-to-machine (M2M) -Kommunikationsprotokoll, das auf dem Publish/Subscribe-Muster für IoT-Geräte mit beschränkten Ressourcen basiert.

Microservice

Ein kleiner, unabhängiger Service, der über klar definierte APIs kommuniziert und in der Regel kleinen, eigenständigen Teams gehört. Ein Versicherungssystem kann beispielsweise Microservices beinhalten, die Geschäftsfunktionen wie Vertrieb oder Marketing oder Subdomains wie Einkauf, Schadenersatz oder Analytik zugeordnet sind. Zu den Vorteilen von Microservices gehören Agilität, flexible Skalierung, einfache Bereitstellung, wiederverwendbarer Code und Ausfallsicherheit. Weitere Informationen finden Sie unter Integration von Microservices mithilfe serverloser Dienste. AWS

Microservices-Architekturen

Ein Ansatz zur Erstellung einer Anwendung mit unabhängigen Komponenten, die jeden Anwendungsprozess als Microservice ausführen. Diese Microservices kommunizieren über eine klar definierte Schnittstelle mithilfe einfacher APIs. Jeder Microservice in dieser Architektur kann aktualisiert, bereitgestellt und skaliert werden, um den Bedarf an bestimmten Funktionen einer Anwendung zu decken. Weitere Informationen finden Sie unter Implementierung von Microservices auf. AWS

Migration Acceleration Program (MAP)

Ein AWS Programm, das Beratung, Unterstützung, Schulungen und Services bietet, um Unternehmen dabei zu unterstützen, eine solide betriebliche Grundlage für die Umstellung auf die Cloud zu schaffen und die anfänglichen Kosten von Migrationen auszugleichen. MAP umfasst eine Migrationsmethode für die methodische Durchführung von Legacy-Migrationen sowie eine Reihe von Tools zur Automatisierung und Beschleunigung gängiger Migrationsszenarien.

Migration in großem Maßstab

Der Prozess, bei dem der Großteil des Anwendungsportfolios in Wellen in die Cloud verlagert wird, wobei in jeder Welle mehr Anwendungen schneller migriert werden. In dieser Phase werden die bewährten Verfahren und Erkenntnisse aus den früheren Phasen zur Implementierung einer Migrationsfabrik von Teams, Tools und Prozessen zur Optimierung der Migration von Workloads durch Automatisierung und agile Bereitstellung verwendet. Dies ist die dritte Phase der AWS - Migrationsstrategie.

Migrationsfabrik

Funktionsübergreifende Teams, die die Migration von Workloads durch automatisierte, agile Ansätze optimieren. Zu den Teams in der Migrationsabteilung gehören in der Regel Betriebsabläufe, Geschäftsanalysten und Eigentümer, Migrationsingenieure, Entwickler und DevOps Experten, die in Sprints arbeiten. Zwischen 20 und 50 Prozent eines Unternehmensanwendungsportfolios bestehen aus sich wiederholenden Mustern, die durch einen Fabrik-Ansatz optimiert werden können. Weitere Informationen finden Sie in Diskussion über Migrationsfabriken und den Leitfaden zur Cloud-Migration-Fabrik in diesem Inhaltssatz.

Migrationsmetadaten

Die Informationen über die Anwendung und den Server, die für den Abschluss der Migration benötigt werden. Für jedes Migrationsmuster ist ein anderer Satz von Migrationsmetadaten erforderlich. Beispiele für Migrationsmetadaten sind das Zielsubnetz, die Sicherheitsgruppe und AWS das Konto.

Migrationsmuster

Eine wiederholbare Migrationsaufgabe, in der die Migrationsstrategie, das Migrationsziel und die verwendete Migrationsanwendung oder der verwendete Migrationsservice detailliert beschrieben werden. Beispiel: Rehost-Migration zu Amazon EC2 mit AWS Application Migration Service.

Migration Portfolio Assessment (MPA)

Ein Online-Tool, das Informationen zur Validierung des Geschäftsszenarios für die Migration auf das bereitstellt. AWS Cloud MPA bietet eine detaillierte Portfoliobewertung (richtige Servergröße, Preisgestaltung, Gesamtbetriebskostenanalyse, Migrationskostenanalyse) sowie Migrationsplanung (Anwendungsdatenanalyse und Datenerfassung, Anwendungsgruppierung, Migrationspriorisierung und Wellenplanung). Das MPA-Tool (Anmeldung erforderlich) steht allen AWS Beratern und APN-Partnerberatern kostenlos zur Verfügung.

Migration Readiness Assessment (MRA)

Der Prozess, bei dem mithilfe des AWS CAF Erkenntnisse über den Cloud-Bereitschaftsstatus eines Unternehmens gewonnen, Stärken und Schwächen identifiziert und ein Aktionsplan zur Schließung festgestellter Lücken erstellt wird. Weitere Informationen finden Sie im Benutzerhandbuch für Migration Readiness. MRA ist die erste Phase der AWS - Migrationsstrategie.

Migrationsstrategie

Der Ansatz, der verwendet wurde, um einen Workload auf den AWS Cloud zu migrieren. Weitere Informationen finden Sie im Eintrag 7 Rs in diesem Glossar und unter Mobilisieren Sie Ihr Unternehmen, um umfangreiche Migrationen zu beschleunigen.

ML

Siehe maschinelles Lernen.

Modernisierung

Umwandlung einer veralteten (veralteten oder monolithischen) Anwendung und ihrer Infrastruktur in ein agiles, elastisches und hochverfügbares System in der Cloud, um Kosten zu senken, die Effizienz zu steigern und Innovationen zu nutzen. Weitere Informationen finden Sie unter Strategie zur Modernisierung von Anwendungen in der AWS Cloud.

Bewertung der Modernisierungsfähigkeit

Eine Bewertung, anhand derer festgestellt werden kann, ob die Anwendungen einer Organisation für die Modernisierung bereit sind, Vorteile, Risiken und Abhängigkeiten identifiziert und ermittelt wird, wie gut die Organisation den zukünftigen Status dieser Anwendungen unterstützen kann. Das Ergebnis der Bewertung ist eine Vorlage der Zielarchitektur, eine Roadmap, in der die Entwicklungsphasen und Meilensteine des Modernisierungsprozesses detailliert beschrieben werden, sowie ein Aktionsplan zur Behebung festgestellter Lücken. Weitere Informationen finden Sie unter Evaluierung der Modernisierungsbereitschaft von Anwendungen in der AWS Cloud.

Monolithische Anwendungen (Monolithen)

Anwendungen, die als ein einziger Service mit eng gekoppelten Prozessen ausgeführt werden. Monolithische Anwendungen haben verschiedene Nachteile. Wenn ein Anwendungs-Feature stark nachgefragt wird, muss die gesamte Architektur skaliert werden. Das Hinzufügen oder Verbessern der Feature einer monolithischen Anwendung wird ebenfalls komplexer, wenn die Codebasis wächst. Um diese Probleme zu beheben, können Sie eine Microservices-Architektur verwenden. Weitere Informationen finden Sie unter Zerlegen von Monolithen in Microservices.

MPA

Siehe Bewertung des Migrationsportfolios.

MQTT

Siehe Message Queuing-Telemetrietransport.

Mehrklassen-Klassifizierung

Ein Prozess, der dabei hilft, Vorhersagen für mehrere Klassen zu generieren (wobei eines von mehr als zwei Ergebnissen vorhergesagt wird). Ein ML-Modell könnte beispielsweise fragen: „Ist dieses Produkt ein Buch, ein Auto oder ein Telefon?“ oder „Welche Kategorie von Produkten ist für diesen Kunden am interessantesten?“

veränderbare Infrastruktur

Ein Modell, das die bestehende Infrastruktur für Produktionsworkloads aktualisiert und modifiziert. Für eine verbesserte Konsistenz, Zuverlässigkeit und Vorhersagbarkeit empfiehlt das AWS Well-Architected Framework die Verwendung einer unveränderlichen Infrastruktur als bewährte Methode.

O

OAC

Weitere Informationen finden Sie unter Origin Access Control.

OAI

Siehe Zugriffsidentität von Origin.

COM

Siehe organisatorisches Change-Management.

Offline-Migration

Eine Migrationsmethode, bei der der Quell-Workload während des Migrationsprozesses heruntergefahren wird. Diese Methode ist mit längeren Ausfallzeiten verbunden und wird in der Regel für kleine, unkritische Workloads verwendet.

OI

Siehe Betriebsintegration.

OLA

Siehe Vereinbarung auf operativer Ebene.

Online-Migration

Eine Migrationsmethode, bei der der Quell-Workload auf das Zielsystem kopiert wird, ohne offline genommen zu werden. Anwendungen, die mit dem Workload verbunden sind, können während der Migration weiterhin funktionieren. Diese Methode beinhaltet keine bis minimale Ausfallzeit und wird in der Regel für kritische Produktionsworkloads verwendet.

OPC-UA

Siehe Open Process Communications — Unified Architecture.

Offene Prozesskommunikation — Einheitliche Architektur (OPC-UA)

Ein machine-to-machine (M2M) -Kommunikationsprotokoll für die industrielle Automatisierung. OPC-UA bietet einen Interoperabilitätsstandard mit Datenverschlüsselungs-, Authentifizierungs- und Autorisierungsschemata.

Vereinbarung auf Betriebsebene (OLA)

Eine Vereinbarung, in der klargestellt wird, welche funktionalen IT-Gruppen sich gegenseitig versprechen zu liefern, um ein Service Level Agreement (SLA) zu unterstützen.

Überprüfung der Betriebsbereitschaft (ORR)

Eine Checkliste mit Fragen und zugehörigen bewährten Methoden, die Ihnen helfen, Vorfälle und mögliche Ausfälle zu verstehen, zu bewerten, zu verhindern oder deren Umfang zu reduzieren. Weitere Informationen finden Sie unter Operational Readiness Reviews (ORR) im AWS Well-Architected Framework.

Betriebstechnologie (OT)

Hardware- und Softwaresysteme, die mit der physischen Umgebung zusammenarbeiten, um industrielle Abläufe, Ausrüstung und Infrastruktur zu steuern. In der Fertigung ist die Integration von OT- und Informationstechnologie (IT) -Systemen ein zentraler Schwerpunkt der Industrie 4.0-Transformationen.

Betriebsintegration (OI)

Der Prozess der Modernisierung von Abläufen in der Cloud, der Bereitschaftsplanung, Automatisierung und Integration umfasst. Weitere Informationen finden Sie im Leitfaden zur Betriebsintegration.

Organisationspfad

Ein Pfad, der von erstellt wird und in AWS CloudTrail dem alle Ereignisse für alle AWS-Konten in einer Organisation protokolliert werden. AWS Organizations Diese Spur wird in jedem AWS-Konto , der Teil der Organisation ist, erstellt und verfolgt die Aktivität in jedem Konto. Weitere Informationen finden Sie in der CloudTrail Dokumentation unter Erstellen eines Pfads für eine Organisation.

Organisatorisches Veränderungsmanagement (OCM)

Ein Framework für das Management wichtiger, disruptiver Geschäftstransformationen aus Sicht der Mitarbeiter, der Kultur und der Führung. OCM hilft Organisationen dabei, sich auf neue Systeme und Strategien vorzubereiten und auf diese umzustellen, indem es die Akzeptanz von Veränderungen beschleunigt, Übergangsprobleme angeht und kulturelle und organisatorische Veränderungen vorantreibt. In der AWS Migrationsstrategie wird dieses Framework aufgrund der Geschwindigkeit des Wandels, der bei Projekten zur Cloud-Einführung erforderlich ist, als Mitarbeiterbeschleunigung bezeichnet. Weitere Informationen finden Sie im OCM-Handbuch.

Ursprungszugriffskontrolle (OAC)

In CloudFront, eine erweiterte Option zur Zugriffsbeschränkung, um Ihre Amazon Simple Storage Service (Amazon S3) -Inhalte zu sichern. OAC unterstützt alle S3-Buckets insgesamt AWS-Regionen, serverseitige Verschlüsselung mit AWS KMS (SSE-KMS) sowie dynamische `PUT` und `DELETE` Anfragen an den S3-Bucket.

Ursprungszugriffsidentität (OAI)

In CloudFront, eine Option zur Zugriffsbeschränkung, um Ihre Amazon S3 S3-Inhalte zu sichern. Wenn Sie OAI verwenden, CloudFront erstellt es einen Principal, mit dem sich Amazon S3 authentifizieren kann. Authentifizierte Principals können nur über eine bestimmte Distribution auf Inhalte in einem S3-Bucket zugreifen. CloudFront Siehe auch OAC, das eine detailliertere und verbesserte Zugriffskontrolle bietet.

ODER

Siehe Überprüfung der Betriebsbereitschaft.

NICHT

Siehe Betriebstechnologie.

Ausgehende (egress) VPC

In einer Architektur AWS mit mehreren Konten eine VPC, die Netzwerkverbindungen verarbeitet, die von einer Anwendung aus initiiert werden. Die AWS -Referenzarchitektur für die Sicherheit empfiehlt, Ihr Netzwerkkonto mit eingehenden und ausgehenden VPCs und Inspektions-VPCs einzurichten, um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet zu schützen.

P

Berechtigungsgrenze

Eine IAM-Verwaltungsrichtlinie, die den IAM-Prinzipalen zugeordnet ist, um die maximalen Berechtigungen festzulegen, die der Benutzer oder die Rolle haben kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitys in der IAM-Dokumentation.

persönlich identifizierbare Informationen (PII)

Informationen, die, wenn sie direkt betrachtet oder mit anderen verwandten Daten kombiniert werden, verwendet werden können, um vernünftige Rückschlüsse auf die Identität einer Person zu ziehen. Beispiele für personenbezogene Daten sind Namen, Adressen und Kontaktinformationen.

Personenbezogene Daten

Siehe persönlich identifizierbare Informationen.

Playbook

Eine Reihe vordefinierter Schritte, die die mit Migrationen verbundenen Aufgaben erfassen, z. B. die Bereitstellung zentraler Betriebsfunktionen in der Cloud. Ein Playbook kann die Form von Skripten, automatisierten Runbooks oder einer Zusammenfassung der Prozesse oder Schritte annehmen, die für den Betrieb Ihrer modernisierten Umgebung erforderlich sind.

PLC

Siehe programmierbare Logiksteuerung.

PLM

Siehe Produktlebenszyklusmanagement.

policy

Ein Objekt, das Berechtigungen definieren (siehe identitätsbasierte Richtlinie), Zugriffsbedingungen spezifizieren (siehe ressourcenbasierte Richtlinie) oder die maximalen Berechtigungen für alle Konten in einer Organisation definieren kann AWS Organizations (siehe Dienststeuerungsrichtlinie).

Polyglotte Beharrlichkeit

Unabhängige Auswahl der Datenspeichertechnologie eines Microservices auf der Grundlage von Datenzugriffsmustern und anderen Anforderungen. Wenn Ihre Microservices über dieselbe Datenspeichertechnologie verfügen, kann dies zu Implementierungsproblemen oder zu Leistungseinbußen führen. Microservices lassen sich leichter implementieren und erzielen eine bessere Leistung und Skalierbarkeit, wenn sie den Datenspeicher verwenden, der ihren Anforderungen am besten entspricht. Weitere Informationen finden Sie unter Datenpersistenz in Microservices aktivieren.

Portfoliobewertung

Ein Prozess, bei dem das Anwendungsportfolio ermittelt, analysiert und priorisiert wird, um die Migration zu planen. Weitere Informationen finden Sie in Bewerten der Migrationsbereitschaft.

predicate

Eine Abfragebedingung, die `true` oder zurückgibt`false`, was üblicherweise in einer Klausel vorkommt. `WHERE`

Prädikat Pushdown

Eine Technik zur Optimierung von Datenbankabfragen, bei der die Daten in der Abfrage vor der Übertragung gefiltert werden. Dadurch wird die Datenmenge reduziert, die aus der relationalen Datenbank abgerufen und verarbeitet werden muss, und die Abfrageleistung wird verbessert.

Präventive Kontrolle

Eine Sicherheitskontrolle, die verhindern soll, dass ein Ereignis eintritt. Diese Kontrollen stellen eine erste Verteidigungslinie dar, um unbefugten Zugriff oder unerwünschte Änderungen an Ihrem Netzwerk zu verhindern. Weitere Informationen finden Sie unter Präventive Kontrolle in Implementierung von Sicherheitskontrollen in AWS.

Prinzipal

Eine Entität AWS , die Aktionen ausführen und auf Ressourcen zugreifen kann. Bei dieser Entität handelt es sich in der Regel um einen Root-Benutzer für eine AWS-Konto, eine IAM-Rolle oder

einen Benutzer. Weitere Informationen finden Sie unter Prinzipal in Rollenbegriffe und -konzepte in der IAM-Dokumentation.

Datenschutz durch Design

Ein Ansatz in der Systemtechnik, der den Datenschutz während des gesamten Engineering-Prozesses berücksichtigt.

Privat gehostete Zonen

Ein Container, der Informationen darüber enthält, wie Amazon Route 53 auf DNS-Abfragen für eine Domain und ihre Subdomains innerhalb einer oder mehrerer VPCs reagieren soll. Weitere Informationen finden Sie unter Arbeiten mit privat gehosteten Zonen in der Route-53-Dokumentation.

proaktive Steuerung

Eine Sicherheitskontrolle, die den Einsatz nicht richtlinienkonformer Ressourcen verhindern soll. Mit diesen Steuerelementen werden Ressourcen gescannt, bevor sie bereitgestellt werden. Wenn die Ressource nicht mit der Steuerung konform ist, wird sie nicht bereitgestellt. Weitere Informationen finden Sie im Referenzhandbuch zu Kontrollen in der AWS Control Tower Dokumentation und unter Proaktive Kontrollen unter Implementierung von Sicherheitskontrollen am AWS.

Produktlebenszyklusmanagement (PLM)

Das Management von Daten und Prozessen für ein Produkt während seines gesamten Lebenszyklus, vom Design, der Entwicklung und Markteinführung über Wachstum und Reife bis hin zur Markteinführung und Markteinführung.

Produktionsumgebung

Siehe Umgebung.

Speicherprogrammierbare Steuerung (SPS)

In der Fertigung ein äußerst zuverlässiger, anpassungsfähiger Computer, der Maschinen überwacht und Fertigungsprozesse automatisiert.

Pseudonymisierung

Der Prozess, bei dem persönliche Identifikatoren in einem Datensatz durch Platzhalterwerte ersetzt werden. Pseudonymisierung kann zum Schutz der Privatsphäre beitragen. Pseudonymisierte Daten gelten weiterhin als personenbezogene Daten.

veröffentlichen/abonnieren (pub/sub)

Ein Muster, das asynchrone Kommunikation zwischen Microservices ermöglicht, um die Skalierbarkeit und Reaktionsfähigkeit zu verbessern. In einem auf Microservices basierenden MES kann ein Microservice beispielsweise Ereignismeldungen in einem Kanal veröffentlichen, den andere Microservices abonnieren können. Das System kann neue Microservices hinzufügen, ohne den Veröffentlichungsservice zu ändern.

Q

Abfrageplan

Eine Reihe von Schritten, wie Anweisungen, die für den Zugriff auf die Daten in einem relationalen SQL-Datenbanksystem verwendet werden.

Abfrageplanregression

Wenn ein Datenbankserviceoptimierer einen weniger optimalen Plan wählt als vor einer bestimmten Änderung der Datenbankumgebung. Dies kann durch Änderungen an Statistiken, Beschränkungen, Umgebungseinstellungen, Abfrageparameter-Bindungen und Aktualisierungen der Datenbank-Engine verursacht werden.

R

RACI-Matrix

Siehe verantwortlich, rechenschaftspflichtig, konsultiert, informiert (RACI).

Ransomware

Eine bösartige Software, die entwickelt wurde, um den Zugriff auf ein Computersystem oder Daten zu blockieren, bis eine Zahlung erfolgt ist.

RASCI-Matrix

Siehe verantwortlich, rechenschaftspflichtig, konsultiert, informiert (RACI).

RCAC

Siehe Zugriffskontrolle für Zeilen und Spalten.

Read Replica

Eine Kopie einer Datenbank, die nur für Lesezwecke verwendet wird. Sie können Abfragen an das Lesereplikat weiterleiten, um die Belastung auf Ihrer Primärdatenbank zu reduzieren.

neu strukturieren

Siehe 7 Rs.

Recovery Point Objective (RPO)

Die maximal zulässige Zeitspanne seit dem letzten Datenwiederherstellungspunkt. Dies bestimmt, was als akzeptabler Datenverlust zwischen dem letzten Wiederherstellungspunkt und der Betriebsunterbrechung angesehen wird.

Wiederherstellungszeitziel (RTO)

Die maximal zulässige Verzögerung zwischen der Betriebsunterbrechung und der Wiederherstellung des Dienstes.

Refaktorierung

Siehe 7 Rs.

Region

Eine Sammlung von AWS Ressourcen in einem geografischen Gebiet. Jeder AWS-Region ist isoliert und unabhängig von den anderen, um Fehlertoleranz, Stabilität und Belastbarkeit zu gewährleisten. Weitere Informationen finden Sie unter Geben Sie an, was AWS-Regionen Ihr Konto verwenden kann.

Regression

Eine ML-Technik, die einen numerischen Wert vorhersagt. Zum Beispiel, um das Problem „Zu welchem Preis wird dieses Haus verkauft werden?“ zu lösen Ein ML-Modell könnte ein lineares Regressionsmodell verwenden, um den Verkaufspreis eines Hauses auf der Grundlage bekannter Fakten über das Haus (z. B. die Quadratmeterzahl) vorherzusagen.

rehosten

Siehe 7 Rs.

Veröffentlichung

In einem Bereitstellungsprozess der Akt der Förderung von Änderungen an einer Produktionsumgebung.

umziehen

Siehe 7 Rs.

neue Plattform

Siehe 7 Rs.

Rückkauf

Siehe 7 Rs.

Ausfallsicherheit

Die Fähigkeit einer Anwendung, Störungen zu widerstehen oder sich von ihnen zu erholen. Hochverfügbarkeit und Notfallwiederherstellung sind häufig Überlegungen bei der Planung der Ausfallsicherheit in der. AWS Cloud Weitere Informationen finden Sie unter AWS Cloud Resilienz.

Ressourcenbasierte Richtlinie

Eine mit einer Ressource verknüpfte Richtlinie, z. B. ein Amazon-S3-Bucket, ein Endpunkt oder ein Verschlüsselungsschlüssel. Diese Art von Richtlinie legt fest, welchen Prinzipalen der Zugriff gewährt wird, welche Aktionen unterstützt werden und welche anderen Bedingungen erfüllt sein müssen.

RACI-Matrix (verantwortlich, rechenschaftspflichtig, konsultiert, informiert)

Eine Matrix, die die Rollen und Verantwortlichkeiten aller an Migrationsaktivitäten und Cloud-Operationen beteiligten Parteien definiert. Der Matrixname leitet sich von den in der Matrix definierten Zuständigkeitstypen ab: verantwortlich (R), rechenschaftspflichtig (A), konsultiert (C) und informiert (I). Der Unterstützungstyp (S) ist optional. Wenn Sie Unterstützung einbeziehen, wird die Matrix als RASCI-Matrix bezeichnet, und wenn Sie sie ausschließen, wird sie als RACI-Matrix bezeichnet.

Reaktive Kontrolle

Eine Sicherheitskontrolle, die darauf ausgelegt ist, die Behebung unerwünschter Ereignisse oder Abweichungen von Ihren Sicherheitsstandards voranzutreiben. Weitere Informationen finden Sie unter Reaktive Kontrolle in Implementieren von Sicherheitskontrollen in AWS.

Beibehaltung

Siehe 7 Rs.

zurückziehen

Siehe 7 Rs.

Drehung

Der Vorgang, bei dem ein Geheimnis regelmäßig aktualisiert wird, um es einem Angreifer zu erschweren, auf die Anmeldeinformationen zuzugreifen.

Zugriffskontrolle für Zeilen und Spalten (RCAC)

Die Verwendung einfacher, flexibler SQL-Ausdrücke mit definierten Zugriffsregeln. RCAC besteht aus Zeilenberechtigungen und Spaltenmasken.

RPO

Siehe Recovery Point Objective.

RTO

Siehe Ziel der Wiederherstellungszeit.

Runbook

Eine Reihe manueller oder automatisierter Verfahren, die zur Ausführung einer bestimmten Aufgabe erforderlich sind. Diese sind in der Regel darauf ausgelegt, sich wiederholende Operationen oder Verfahren mit hohen Fehlerquoten zu rationalisieren.

S

SAML 2.0

Ein offener Standard, den viele Identitätsanbieter (IdPs) verwenden. Diese Funktion ermöglicht föderiertes Single Sign-On (SSO), sodass sich Benutzer bei den API-Vorgängen anmelden AWS Management Console oder die AWS API-Operationen aufrufen können, ohne dass Sie einen Benutzer in IAM für alle in Ihrer Organisation erstellen müssen. Weitere Informationen zum SAML-2.0.-basierten Verbund finden Sie unter Über den SAML-2.0-basierten Verbund in der IAM-Dokumentation.

SCADA

Siehe Aufsichtskontrolle und Datenerfassung.

SCP

Siehe Richtlinie zur Dienstkontrolle.

Secret

Interne AWS Secrets Manager, vertrauliche oder eingeschränkte Informationen, wie z. B. ein Passwort oder Benutzeranmeldeinformationen, die Sie in verschlüsselter Form speichern. Es besteht aus dem geheimen Wert und seinen Metadaten. Der geheime Wert kann binär, eine einzelne Zeichenfolge oder mehrere Zeichenketten sein. Weitere Informationen finden Sie unter Was ist in einem Secrets Manager Manager-Geheimnis? in der Secrets Manager Manager-Dokumentation.

Sicherheitskontrolle

Ein technischer oder administrativer Integritätsschutz, der die Fähigkeit eines Bedrohungsakteurs, eine Schwachstelle auszunutzen, verhindert, erkennt oder einschränkt. Es gibt vier Haupttypen von Sicherheitskontrollen: präventiv, detektiv, reaktionsschnell und proaktiv.

Härtung der Sicherheit

Der Prozess, bei dem die Angriffsfläche reduziert wird, um sie widerstandsfähiger gegen Angriffe zu machen. Dies kann Aktionen wie das Entfernen von Ressourcen, die nicht mehr benötigt werden, die Implementierung der bewährten Sicherheitsmethode der Gewährung geringster Berechtigungen oder die Deaktivierung unnötiger Feature in Konfigurationsdateien umfassen.

System zur Verwaltung von Sicherheitsinformationen und Ereignissen (security information and event management – SIEM)

Tools und Services, die Systeme für das Sicherheitsinformationsmanagement (SIM) und das Management von Sicherheitsereignissen (SEM) kombinieren. Ein SIEM-System sammelt, überwacht und analysiert Daten von Servern, Netzwerken, Geräten und anderen Quellen, um Bedrohungen und Sicherheitsverletzungen zu erkennen und Warnmeldungen zu generieren.

Automatisierung von Sicherheitsreaktionen

Eine vordefinierte und programmierte Aktion, die darauf ausgelegt ist, automatisch auf ein Sicherheitsereignis zu reagieren oder es zu beheben. Diese Automatisierungen dienen als detektive oder reaktionsschnelle Sicherheitskontrollen, die Sie bei der Implementierung bewährter AWS Sicherheitsmethoden unterstützen. Beispiele für automatisierte Antwortaktionen sind das Ändern einer VPC-Sicherheitsgruppe, das Patchen einer Amazon EC2 EC2-Instance oder das Rotieren von Anmeldeinformationen.

Serverseitige Verschlüsselung

Verschlüsselung von Daten am Zielort durch denjenigen AWS-Service , der sie empfängt.

Service-Kontrollrichtlinie (SCP)

Eine Richtlinie, die eine zentrale Kontrolle über die Berechtigungen für alle Konten in einer Organisation in AWS Organizations ermöglicht. SCPs definieren Integritätsschutz oder legen Grenzwerte für Aktionen fest, die ein Administrator an Benutzer oder Rollen delegieren kann. Sie können SCPs als Zulassungs- oder Ablehnungslisten verwenden, um festzulegen, welche Services oder Aktionen zulässig oder verboten sind. Weitere Informationen finden Sie in der AWS Organizations Dokumentation unter Richtlinien zur Dienststeuerung.

Service-Endpunkt

Die URL des Einstiegspunkts für einen AWS-Service. Sie können den Endpunkt verwenden, um programmgesteuert eine Verbindung zum Zielservice herzustellen. Weitere Informationen finden Sie unter AWS-Service -Endpunkte in der Allgemeine AWS-Referenz.

Service Level Agreement (SLA)

Eine Vereinbarung, in der klargestellt wird, was ein IT-Team seinen Kunden zu bieten verspricht, z. B. in Bezug auf Verfügbarkeit und Leistung der Services.

Service-Level-Indikator (SLI)

Eine Messung eines Leistungsaspekts eines Dienstes, z. B. seiner Fehlerrate, Verfügbarkeit oder Durchsatz.

Service-Level-Ziel (SLO)

Eine Zielkennzahl, die den Zustand eines Dienstes darstellt, gemessen anhand eines Service-Level-Indikators.

Modell der geteilten Verantwortung

Ein Modell, das die Verantwortung beschreibt, mit der Sie gemeinsam AWS für Cloud-Sicherheit und Compliance verantwortlich sind. AWS ist für die Sicherheit der Cloud verantwortlich, wohingegen Sie für die Sicherheit in der Cloud verantwortlich sind. Weitere Informationen finden Sie unter Modell der geteilten Verantwortung.

SIEM

Siehe Sicherheitsinformations- und Event-Management-System.

Single Point of Failure (SPOF)

Ein Fehler in einer einzelnen, kritischen Komponente einer Anwendung, der das System stören kann.

SLA

Siehe Service Level Agreement.

SLI

Siehe Service-Level-Indikator.

ALSO

Siehe Service-Level-Ziel.

split-and-seed Modell

Ein Muster für die Skalierung und Beschleunigung von Modernisierungsprojekten. Sobald neue Features und Produktversionen definiert werden, teilt sich das Kernteam auf, um neue Produktteams zu bilden. Dies trägt zur Skalierung der Fähigkeiten und Services Ihrer Organisation bei, verbessert die Produktivität der Entwickler und unterstützt schnelle Innovationen. Weitere Informationen finden Sie unter Schrittweiser Ansatz zur Modernisierung von Anwendungen in der AWS Cloud.

SPOTTEN

Siehe Single Point of Failure.

Sternschema

Eine Datenbank-Organisationsstruktur, die eine große Faktentabelle zum Speichern von Transaktions- oder Messdaten und eine oder mehrere kleinere dimensionale Tabellen zum Speichern von Datenattributen verwendet. Diese Struktur ist für die Verwendung in einem Data Warehouse oder für Business Intelligence-Zwecke konzipiert.

Strangler-Fig-Muster

Ein Ansatz zur Modernisierung monolithischer Systeme, bei dem die Systemfunktionen schrittweise umgeschrieben und ersetzt werden, bis das Legacy-System außer Betrieb genommen werden kann. Dieses Muster verwendet die Analogie einer Feigenrebe, die zu einem etablierten Baum heranwächst und schließlich ihren Wirt überwindet und ersetzt. Das Muster wurde eingeführt von Martin Fowler als Möglichkeit, Risiken beim Umschreiben

monolithischer Systeme zu managen. Ein Beispiel für die Anwendung dieses Musters finden Sie unter Schrittweises Modernisieren älterer Microsoft ASP.NET (ASMX)-Webservices mithilfe von Containern und Amazon API Gateway.

Subnetz

Ein Bereich von IP-Adressen in Ihrer VPC. Ein Subnetz muss sich in einer einzigen Availability Zone befinden.

Aufsichtskontrolle und Datenerfassung (SCADA)

In der Fertigung ein System, das Hardware und Software zur Überwachung von Sachanlagen und Produktionsabläufen verwendet.

Symmetrische Verschlüsselung

Ein Verschlüsselungsalgorithmus, der denselben Schlüssel zum Verschlüsseln und Entschlüsseln der Daten verwendet.

synthetisches Testen

Testen eines Systems auf eine Weise, die Benutzerinteraktionen simuliert, um potenzielle Probleme zu erkennen oder die Leistung zu überwachen. Sie können Amazon CloudWatch Synthetics verwenden, um diese Tests zu erstellen.

T

tags

Schlüssel-Wert-Paare, die als Metadaten für die Organisation Ihrer Ressourcen dienen. AWS Mit Tags können Sie Ressourcen verwalten, identifizieren, organisieren, suchen und filtern. Weitere Informationen finden Sie unter Markieren Ihrer AWS -Ressourcen.

Zielvariable

Der Wert, den Sie in überwachtem ML vorhersagen möchten. Dies wird auch als Ergebnisvariable bezeichnet. In einer Fertigungsumgebung könnte die Zielvariable beispielsweise ein Produktfehler sein.

Aufgabenliste

Ein Tool, das verwendet wird, um den Fortschritt anhand eines Runbooks zu verfolgen. Eine Aufgabenliste enthält eine Übersicht über das Runbook und eine Liste mit allgemeinen Aufgaben,

die erledigt werden müssen. Für jede allgemeine Aufgabe werden der geschätzte Zeitaufwand, der Eigentümer und der Fortschritt angegeben.

Testumgebungen

Siehe Umgebung.

Training

Daten für Ihr ML-Modell bereitstellen, aus denen es lernen kann. Die Trainingsdaten müssen die richtige Antwort enthalten. Der Lernalgorithmus findet Muster in den Trainingsdaten, die die Attribute der Input-Daten dem Ziel (die Antwort, die Sie voraussagen möchten) zuordnen. Es gibt ein ML-Modell aus, das diese Muster erfasst. Sie können dann das ML-Modell verwenden, um Voraussagen für neue Daten zu erhalten, bei denen Sie das Ziel nicht kennen.

Transit-Gateway

Ein Transit-Gateway ist ein Netzwerk-Transit-Hub, mit dem Sie Ihre VPCs und On-Premises-Netzwerke miteinander verbinden können. Weitere Informationen finden Sie in der AWS Transit Gateway Dokumentation unter Was ist ein Transit-Gateway.

Stammbasierter Workflow

Ein Ansatz, bei dem Entwickler Feature lokal in einem Feature-Zweig erstellen und testen und diese Änderungen dann im Hauptzweig zusammenführen. Der Hauptzweig wird dann sequentiell für die Entwicklungs-, Vorproduktions- und Produktionsumgebungen erstellt.

Vertrauenswürdiger Zugriff

Gewährung von Berechtigungen für einen Dienst, den Sie angeben, um Aufgaben in Ihrer Organisation AWS Organizations und in deren Konten in Ihrem Namen auszuführen. Der vertrauenswürdige Service erstellt in jedem Konto eine mit dem Service verknüpfte Rolle, wenn diese Rolle benötigt wird, um Verwaltungsaufgaben für Sie auszuführen. Weitere Informationen finden Sie in der AWS Organizations Dokumentation unter Verwendung AWS Organizations mit anderen AWS Diensten.

Optimieren

Aspekte Ihres Trainingsprozesses ändern, um die Genauigkeit des ML-Modells zu verbessern. Sie können das ML-Modell z. B. trainieren, indem Sie einen Beschriftungssatz generieren, Beschriftungen hinzufügen und diese Schritte dann mehrmals unter verschiedenen Einstellungen wiederholen, um das Modell zu optimieren.

Zwei-Pizzen-Team

Ein kleines DevOps Team, das Sie mit zwei Pizzen ernähren können. Eine Teamgröße von zwei Pizzen gewährleistet die bestmögliche Gelegenheit zur Zusammenarbeit bei der Softwareentwicklung.

U

Unsicherheit

Ein Konzept, das sich auf ungenaue, unvollständige oder unbekannte Informationen bezieht, die die Zuverlässigkeit von prädiktiven ML-Modellen untergraben können. Es gibt zwei Arten von Unsicherheit: Epistemische Unsicherheit wird durch begrenzte, unvollständige Daten verursacht, wohingegen aleatorische Unsicherheit durch Rauschen und Randomisierung verursacht wird, die in den Daten liegt. Weitere Informationen finden Sie im Leitfaden Quantifizieren der Unsicherheit in Deep-Learning-Systemen.

undifferenzierte Aufgaben

Diese Arbeit wird auch als Schwerstarbeit bezeichnet. Dabei handelt es sich um Arbeiten, die zwar für die Erstellung und den Betrieb einer Anwendung erforderlich sind, aber dem Endbenutzer keinen direkten Mehrwert bieten oder keinen Wettbewerbsvorteil bieten. Beispiele für undifferenzierte Aufgaben sind Beschaffung, Wartung und Kapazitätsplanung.

höhere Umgebungen

Siehe Umgebung.

V

Vacuuming

Ein Vorgang zur Datenbankwartung, bei dem die Datenbank nach inkrementellen Aktualisierungen bereinigt wird, um Speicherplatz zurückzugewinnen und die Leistung zu verbessern.

Versionskontrolle

Prozesse und Tools zur Nachverfolgung von Änderungen, z. B. Änderungen am Quellcode in einem Repository.

VPC-Peering

Eine Verbindung zwischen zwei VPCs, mit der Sie den Datenverkehr mithilfe von privaten IP-Adressen weiterleiten können. Weitere Informationen finden Sie unter Was ist VPC-Peering? in der Amazon-VPC-Dokumentation.

Schwachstelle

Ein Software- oder Hardwarefehler, der die Sicherheit des Systems gefährdet.

W

Warmer Cache

Ein Puffer-Cache, der aktuelle, relevante Daten enthält, auf die häufig zugegriffen wird. Die Datenbank-Instance kann aus dem Puffer-Cache lesen, was schneller ist als das Lesen aus dem Hauptspeicher oder von der Festplatte.

warme Daten

Daten, auf die selten zugegriffen wird. Bei der Abfrage dieser Art von Daten sind mäßig langsame Abfragen in der Regel akzeptabel.

Fensterfunktion

Eine SQL-Funktion, die eine Berechnung für eine Gruppe von Zeilen durchführt, die sich in irgendeiner Weise auf den aktuellen Datensatz beziehen. Fensterfunktionen sind nützlich für die Verarbeitung von Aufgaben wie die Berechnung eines gleitenden Durchschnitts oder für den Zugriff auf den Wert von Zeilen auf der Grundlage der relativen Position der aktuellen Zeile.

Workload

Ein Workload ist eine Sammlung von Ressourcen und Code, die einen Unternehmenswert bietet, wie z. B. eine kundenorientierte Anwendung oder ein Backend-Prozess.

Workstream

Funktionsgruppen in einem Migrationsprojekt, die für eine bestimmte Reihe von Aufgaben verantwortlich sind. Jeder Workstream ist unabhängig, unterstützt aber die anderen Workstreams im Projekt. Der Portfolio-Workstream ist beispielsweise für die Priorisierung von Anwendungen, die Wellenplanung und die Erfassung von Migrationsmetadaten verantwortlich. Der Portfolio-Workstream liefert diese Komponenten an den Migrations-Workstream, der dann die Server und Anwendungen migriert.

WURM

Sehen Sie einmal schreiben, viele lesen.

WQF

Weitere Informationen finden Sie unter AWS Workload Qualification Framework.

einmal schreiben, viele lesen (WORM)

Ein Speichermodell, das Daten ein einziges Mal schreibt und verhindert, dass die Daten gelöscht oder geändert werden. Autorisierte Benutzer können die Daten so oft wie nötig lesen, aber sie können sie nicht ändern. Diese Datenspeicherinfrastruktur gilt als unveränderlich.

Z

Zero-Day-Exploit

Ein Angriff, in der Regel Malware, der eine Zero-Day-Sicherheitslücke ausnutzt.

Zero-Day-Sicherheitslücke

Ein unfehlbarer Fehler oder eine Sicherheitslücke in einem Produktionssystem. Bedrohungsakteure können diese Art von Sicherheitslücke nutzen, um das System anzugreifen. Entwickler werden aufgrund des Angriffs häufig auf die Sicherheitsanfälligkeit aufmerksam.

Zombie-Anwendung

Eine Anwendung, deren durchschnittliche CPU- und Arbeitsspeichernutzung unter 5 Prozent liegt. In einem Migrationsprojekt ist es üblich, diese Anwendungen außer Betrieb zu nehmen.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.