Commencer à utiliser Terraform : conseils et experts AWS CDK AWS CloudFormation

AWS Directives prescriptives

AWS Directives prescriptives: Commencer à utiliser Terraform : conseils et experts AWS CDK AWS CloudFormation

Table of Contents

Commencer à utiliser Terraform : conseils pour les experts d'AWS CDK et d'AWS CloudFormation

Steven Guggenheimer, Amazon Web Services ()AWS

Mars 2024 (historique du document)

Si votre expérience en matière de provisionnement de ressources cloud se situe exclusivement dans le domaine de AWS, vous avez peut-être une expérience limitée avec les outils d'infrastructure en tant que code (IaC) au-delà de la AWS Cloud Development Kit (AWS CDK)limite. AWS CloudFormation En fait, des outils similaires, tels que Hashicorp Terraform, peuvent vous être totalement inconnus. Cependant, plus vous vous engagez dans votre transition vers le cloud, plus il devient inévitable que vous rencontriez Terraform. Il sera certainement à votre avantage de vous familiariser avec ses concepts fondamentaux.

Bien que Terraform AWS CDK, les et CloudFormation atteignent des objectifs similaires et partagent de nombreux concepts fondamentaux, il existe de nombreuses différences. Vous n'êtes peut-être pas préparé à ces différences si vous contactez Terraform pour la première fois. Après tout, comme AWS CDK les CloudFormation piles sont toutes basées sur l'intérieur Comptes AWS, elles entretiennent ainsi une relation directe avec la plupart des ressources qu'elles gèrent. Terraform n'est basé sur aucun environnement de fournisseur de cloud. Cela lui donne la flexibilité nécessaire pour prendre en charge différents fournisseurs, mais il doit maintenir ses ressources à partir d'un site distant.

Ce guide aide à démystifier les concepts de base de Terraform pour vous aider à relever tous les défis IaC qui se présentent à vous. Il se concentre sur la manière dont Terraform utilise des concepts, tels que les fournisseurs, les modules et les fichiers d'état, pour provisionner des ressources. Il met également en contraste les concepts de Terraform avec la manière dont les AWS CDK et CloudFormation exécutent des opérations similaires.

> **Note**
>
> AWS CDK Cela aide les développeurs à déployer des CloudFormation piles en utilisant des langages de codage programmatiques. Après l'exécution`cdk synth`, votre code est converti en CloudFormation modèles. À partir de ce moment, le processus est identique entre le AWS CDK et CloudFormation. Par souci de concision, ce guide fait généralement référence

au processus AWS IaC en CloudFormation termes, mais les comparaisons sont tout aussi pertinentes pour le AWS CDK.

CloudFormation et terminologie Terraform

Lorsque l'on compare Terraform avec AWS CDK et CloudFormation, il peut être difficile de concilier les concepts fondamentaux d'IaC en raison de la terminologie incohérente utilisée pour les décrire. Voici ces termes et la façon dont ce guide les désignera :

- Pile — Une pile est un IaC déployé dans un pipeline CI/CD et traçable en tant qu'unité unique.Bien que ce terme soit courant dans CloudFormation, Terraform ne l'utilise pas vraiment. Une pile Terraform est un module racine déployé avec tous ses modules enfants. Toutefois, afin d'éviter toute confusion avec le terme module, ce guide utilise le terme stack pour décrire un déploiement unique pour les deux outils.
- État - L'état correspond à toutes les ressources actuellement suivies et à leurs configurations actuelles au sein d'une pile de déploiement IaC. Comme décrit dans la Comprendre les états et les backends de Terraform section, Terraform utilise le terme état plus que. CloudFormation En effet, le maintien de l'état est plus visible dans Terraform, mais le suivi et la mise à jour de l'état sont tout aussi importants pour. CloudFormation
- Fichier IaC — Un fichier IaC est un fichier unique qui contient le langage d'infrastructure en tant que code (IaC). CloudFormation fait référence à un seul CloudFormation fichier en tant que modèle. Cependant, les modèles et les fichiers de modèles dans Terraform sont complètement différents. L'équivalent d'un CloudFormation modèle dans Terraform s'appelle un fichier de configuration. Pour éviter toute confusion dans ce guide, le terme fichier ou fichier iAC est utilisé pour désigner à la fois les CloudFormation modèles et les fichiers de configuration Terraform.

Le tableau suivant compare la terminologie utilisée pour Terraform CloudFormation et Terraform. Le but de ce tableau est de montrer les similitudes. Il ne s'agit pas de one-to-one comparaisons. Chaque concept diffère au moins légèrement entre Terraform CloudFormation et Terraform. Les concepts sont expliqués en détail dans les sections pertinentes de ce guide.

CloudFormation terme	Terme Terraform	Section de ce guide
Interfaces CDK (telles que iBucket)	Source de données	Comprendre les sources de données Terraform

CloudFormation terme	Terme Terraform	Section de ce guide
Modifier le set	Plan	Comprendre les modules Terraform
Fonctions de condition	Expressions conditionnelles	Comprendre les fonctions, expressions et méta-arguments de Terraform
Attribut `DependsOn`	`depends_on` méta-argument	Comprendre les fonctions, expressions et méta-arguments de Terraform
Fonctions intrinsèques	Fonctions	Comprendre les fonctions, expressions et méta-arguments de Terraform
Modules	Modules	Comprendre les modules Terraform
Outputs	Valeurs de sortie	Comprendre les variables, les valeurs locales et les sorties Terraform
Paramètres	Variables	Comprendre les variables, les valeurs locales et les sorties Terraform
Registre	Fournisseurs	Comprendre les fournisseurs de Terraform
Modèle	Fichier de configuration	Tous

Comprendre les ressources de Terraform

La principale raison de l'existence des deux AWS CloudFormation et de Terraform est la création et la maintenance de ressources cloud. Mais qu'est-ce qu'une ressource cloud exactement ? Et les CloudFormation ressources et les ressources Terraform sont-elles la même chose ? La réponse est... oui et non. Pour illustrer cela, ce guide fournit un exemple d'utilisation CloudFormation puis de Terraform pour créer un bucket Amazon Simple Storage Service (Amazon S3).

L'exemple de CloudFormation code suivant crée un exemple de compartiment Amazon S3.

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

L'exemple de code Terraform suivant crée un compartiment Amazon S3 identique.

```
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"
  }
}
```

Pour Terraform, un fournisseur définit la ressource, puis les développeurs déclarent et configurent ces ressources. Les fournisseurs sont un concept que ce guide aborde dans la section suivante. L'exemple Terraform crée des ressources complètement distinctes pour plusieurs paramètres du compartiment S3. La création de ressources distinctes pour les paramètres n'est pas nécessairement typique de la façon dont le AWS fournisseur Terraform traite les AWS ressources. Cependant, cet exemple montre une distinction importante. Bien qu'une CloudFormation ressource soit strictement définie par la spécification de la CloudFormation ressource, Terraform n'a aucune exigence de ce type. Dans Terraform, le concept de ressource est un peu plus nébuleux.

Bien que les outils puissent différer en ce qui concerne les garde-fous exacts qui définissent ce qu'est une ressource unique, une ressource cloud est généralement une entité spécifique qui existe dans le cloud et qui peut être créée, mise à jour ou supprimée. Ainsi, quel que soit le nombre de ressources impliquées, les deux exemples précédents créent exactement la même chose avec exactement les mêmes paramètres dans un Compte AWS.

Comprendre les fournisseurs de Terraform

Dans Terraform, un fournisseur est un plugin qui interagit avec des fournisseurs de cloud, des outils tiers et d'autres API. Pour utiliser Terraform avec AWS, vous utilisez le AWS fournisseur, qui interagit avec les ressources. AWS

Si vous n'avez jamais utilisé le AWS CloudFormation registre pour intégrer des extensions tierces dans vos piles de déploiement, les fournisseurs Terraform auront peut-être besoin d'un certain temps pour s'y habituer. Comme CloudFormation il est natif de AWS, le fournisseur de AWS ressources existe déjà par défaut. Terraform, en revanche, n'a pas de fournisseur par défaut unique, donc rien ne peut être supposé quant à l'origine d'une ressource donnée. Cela signifie que la première chose à déclarer dans un fichier de configuration Terraform est exactement où vont les ressources et comment elles vont y arriver.

Cette distinction ajoute une couche de complexité supplémentaire à Terraform qui n'existe pas avec. CloudFormation Cependant, cette complexité apporte une flexibilité accrue. Vous pouvez déclarer plusieurs fournisseurs au sein d'un même module Terraform, puis les ressources sous-jacentes créées peuvent interagir les unes avec les autres dans le cadre de la même couche de déploiement.

Cela peut être utile de nombreuses manières. Les fournisseurs ne doivent pas nécessairement appartenir à des fournisseurs de cloud distincts. Les fournisseurs peuvent représenter n'importe quelle source de ressources cloud. Prenons l'exemple d'Amazon Elastic Kubernetes Service (Amazon EKS). Lorsque vous provisionnez un cluster Amazon EKS, vous souhaiterez peut-être utiliser des diagrammes Helm pour gérer les extensions tierces et utiliser Kubernetes lui-même pour gérer les ressources des pods. Comme AWSHelm et Kubernetes ont tous leurs propres fournisseurs Terraform, vous pouvez provisionner et intégrer ces ressources en même temps, puis leur transmettre des valeurs.

Dans l'exemple de code suivant pour Terraform, le AWS fournisseur crée un cluster Amazon EKS, puis les informations de configuration Kubernetes qui en résultent sont transmises aux fournisseurs Helm et Kubernetes.

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

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

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

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

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

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

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

```
    }
  }
}

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

Il existe un compromis entre les fournisseurs en ce qui concerne les deux outils IaC. Terraform s'appuie entièrement sur des packages de fournisseurs externes, qui constituent le moteur de ses déploiements. CloudFormation soutient en interne tous les principaux AWS processus. Avec CloudFormation, vous ne devez vous soucier des fournisseurs tiers que si vous souhaitez intégrer une extension tierce. Chaque approche comporte des avantages et des inconvénients. Ce guide ne porte pas sur celui qui vous convient le mieux, mais il est important de garder à l'esprit la différence lors de l'évaluation des deux outils.

Utilisation d'alias Terraform

Dans Terraform, vous pouvez transmettre des configurations personnalisées à chaque fournisseur. Et si vous souhaitez utiliser plusieurs configurations de fournisseurs dans le même module ? Dans ce cas, vous devrez utiliser un alias. Les alias vous aident à sélectionner le fournisseur à utiliser au niveau de la ressource ou du module. Lorsque vous avez plusieurs instances du même fournisseur, vous utilisez un alias pour définir les instances autres que celles par défaut. Par exemple, votre instance de fournisseur par défaut peut être spécifique Région AWS, mais vous utilisez des alias pour définir des régions alternatives.

L'exemple Terraform suivant montre comment utiliser un alias pour approvisionner des buckets dans différents compartiments. Régions AWS La région par défaut du fournisseur est`us-west-2`, mais vous pouvez utiliser l'alias est pour provisionner des ressources`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"
}
```

Lorsque vous utilisez un `alias` avec le `provider` méta-argument, comme indiqué dans l'exemple précédent, vous pouvez spécifier une configuration de fournisseur différente pour des ressources spécifiques. Le provisionnement des ressources en plusieurs Régions AWS dans une seule pile n'est que le début. Les fournisseurs d'alias sont incroyablement pratiques à bien des égards.

Par exemple, il est très courant de provisionner plusieurs clusters Kubernetes à la fois. Les alias peuvent vous aider à configurer des fournisseurs Helm et Kubernetes supplémentaires afin que vous puissiez utiliser ces outils tiers différemment pour les différentes ressources Amazon EKS. L'exemple de code Terraform suivant illustre comment utiliser des alias pour effectuer cette tâche.

```
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"
  }
}
```

Comprendre les modules Terraform

Dans le domaine de l'infrastructure en tant que code (IaC), un module est un bloc de code autonome qui est isolé et empaqueté pour être réutilisé. Le concept de modules est un aspect incontournable du développement de Terraform. Pour plus d'informations, consultez la section Modules dans la documentation Terraform. AWS CloudFormation prend également en charge les modules. Pour plus d'informations, consultez la section Présentation AWS CloudFormation des modules dans le blog sur les opérations et les migrations dans le AWS cloud.

La principale différence entre les modules dans Terraform CloudFormation est que les CloudFormation modules sont importés à l'aide d'un type de ressource spécial ()`AWS::CloudFormation::ModuleVersion`. Dans Terraform, chaque configuration possède au moins un module, appelé module racine. Les ressources Terraform présentes dans le fichier .tf principal ou dans les fichiers d'un fichier de configuration Terraform sont considérées comme se trouvant dans le module racine. Le module racine peut ensuite appeler d'autres modules à inclure dans la pile. L'exemple suivant montre un module racine provisionnant un cluster Amazon Elastic Kubernetes Service (Amazon EKS) à l'aide du module open source eks.

```
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)
  }
```

```
}
```

Vous avez peut-être remarqué que le fichier de configuration ci-dessus n'inclut pas le AWS fournisseur. En effet, les modules sont autonomes et peuvent inclure leurs propres fournisseurs. Les fournisseurs Terraform étant mondiaux, les fournisseurs d'un module enfant peuvent être utilisés dans le module racine. Ce n'est cependant pas vrai pour toutes les valeurs des modules. Les autres valeurs internes d'un module sont limitées par défaut à ce module uniquement et doivent être déclarées en tant que sorties pour être accessibles dans le module racine. Vous pouvez tirer parti des modules open source pour simplifier la création de ressources au sein de votre stack. Par exemple, le module eks ne se contente pas de provisionner un cluster EKS : il fournit un environnement Kubernetes entièrement fonctionnel. Son utilisation peut vous éviter d'écrire des dizaines de lignes de code supplémentaires, à condition que la configuration du module eks réponde à vos besoins.

Modules d'appel

Deux des principales commandes de la CLI Terraform que vous exécutez lors du déploiement de Terraform sont terraform init et terraform apply. L'une des étapes par défaut de la `terraform init` commande consiste à localiser tous les modules enfants et à les importer en tant que dépendances dans le `.terraform/modules` répertoire. Au cours du développement, chaque fois que vous ajoutez un nouveau module externe, vous devez le réinitialiser avant d'utiliser la `apply` commande. Lorsque vous entendez une référence à un module Terraform, cela fait référence aux packages de ce répertoire. À proprement parler, le module que vous déclarez dans votre code est le module appelant. En pratique, le mot-clé module appelle le module lui-même, qui est stocké sous forme de dépendance.

De cette façon, le module d'appel sert de représentant plus succinct du module complet à remplacer lors du déploiement. Vous pouvez tirer parti de cette idée en créant vos propres modules au sein de vos piles pour appliquer des séparations logiques des ressources en utilisant les critères de votre choix. N'oubliez pas que l'objectif final de cette opération doit être de réduire la complexité de votre stack. Étant donné que le partage de données entre les modules vous oblige à générer ces données depuis le module, le fait de trop compter sur les modules peut parfois compliquer les choses.

Le module racine

Comme chaque configuration Terraform comporte au moins un module, il peut être utile d'examiner les propriétés du module avec lequel vous aurez le plus à traiter : le module racine. Chaque fois

que vous travaillez sur un projet Terraform, le module racine comprend tous les `.tf` (ou `.tf.json`) fichiers de votre répertoire de premier niveau. Lorsque vous exécutez `terraform apply` dans ce répertoire de premier niveau, Terraform tente d'exécuter tous les `.tf` fichiers qu'il y trouve. Tous les fichiers des sous-répertoires sont ignorés sauf s'ils sont appelés dans l'un de ces fichiers de configuration de niveau supérieur.

Cela offre une certaine flexibilité dans la façon dont vous structurez votre code. C'est également la raison pour laquelle il est plus précis de désigner votre déploiement Terraform comme un module que comme un fichier, car plusieurs fichiers peuvent être impliqués dans un seul processus. Terraform recommande une structure de module standard pour les meilleures pratiques. Toutefois, si vous deviez placer un `.tf` fichier dans votre répertoire de premier niveau, il s'exécuterait avec le reste des fichiers. En fait, tous les `.tf` fichiers de premier niveau d'un module sont déployés lorsque vous l'exécutez`terraform apply`. Alors, quel fichier Terraform exécute-t-il en premier ? La réponse à cette question est très importante.

Terraform effectue une série d'étapes après l'initialisation et avant le déploiement de la pile. Les configurations existantes sont d'abord analysées, puis un graphe de dépendance est créé. Le graphe de dépendance détermine quelles ressources sont demandées et dans quel ordre elles doivent être traitées. Les ressources contenant des propriétés référencées dans d'autres ressources, par exemple, seraient traitées avant leurs ressources dépendantes. De même, les ressources qui déclarent explicitement une dépendance à l'aide du `depends_on` paramètre seraient gérées après les ressources qu'elles spécifient. Dans la mesure du possible, Terraform peut implémenter le parallélisme et gérer simultanément des ressources non dépendantes. Vous pouvez voir le graphe de dépendance avant le déploiement à l'aide de la commande terraform graph.

Une fois le graphe de dépendance créé, Terraform détermine ce qui doit être fait pendant le déploiement. Il compare le graphe de dépendance avec le fichier d'état le plus récent. Le résultat de ce processus s'appelle un plan, qui ressemble beaucoup à un ensemble de CloudFormation modifications. Vous pouvez voir le plan actuel à l'aide de la commande terraform plan.

En tant que bonne pratique, il est recommandé de rester aussi proche que possible de la structure du module standard. Si vos fichiers de configuration deviennent trop longs pour être gérés efficacement et que les séparations logiques peuvent simplifier la gestion, vous pouvez répartir votre code sur plusieurs fichiers. Gardez à l'esprit le fonctionnement du graphique des dépendances et du processus de planification pour que vos piles fonctionnent le plus efficacement possible.

Comprendre les états et les backends de Terraform

L'un des concepts les plus importants de l'infrastructure en tant que code (IAc) est le concept d'état. Les services IaC conservent l'état, ce qui vous permet de déclarer une ressource dans un fichier IaC sans la recréer à chaque déploiement. Les fichiers IaC documentent l'état de toutes les ressources à la fin d'un déploiement afin de pouvoir ensuite comparer cet état à l'état cible, tel que déclaré lors du déploiement suivant. Ainsi, si l'état actuel contient un compartiment Amazon Simple Storage Service (Amazon S3) `my-s3-bucket` nommé et que les modifications entrantes contiennent également ce même compartiment, le nouveau processus appliquera toutes les modifications trouvées au compartiment existant plutôt que d'essayer de créer un tout nouveau compartiment.

Le tableau suivant fournit des exemples du processus général d'état de l'IaC.

État actuel	État cible	Action
Aucun compartiment S3 nommé `my-s3-bucket`	Compartiment S3 nommé `my-s3-bucket`	Créez un compartiment S3 nommé `my-s3-bucket`
`my-s3-bucket` sans configuration de version des compartiments	`my-s3-bucket` sans configuration de version des compartiments	Aucune action
`my-s3-bucket` sans configuration de version des compartiments	`my-s3-bucket` avec configuration du versionnement des compartiments	Configurer `my-s3-bucket` pour avoir une gestion des versions des compartiments
`my-s3-bucket` avec configuration du versionnement des compartiments	Aucun compartiment S3 nommé `my-s3-bucket`	Tentative de suppression `my-s3-bucket`

Pour comprendre les différentes manières dont AWS CloudFormation Terraform suit l'état, il est important de se rappeler la première différence fondamentale entre les deux outils : il CloudFormation est hébergé dans le AWS Cloud, et Terraform est essentiellement distant. Ce fait permet de CloudFormation maintenir l'état en interne. Vous pouvez accéder à la CloudFormation console et consulter l'historique des événements d'une pile donnée, mais le CloudFormation service lui-même applique les règles de l'État à votre place.

Les trois modes qui CloudFormation fonctionnent pour une ressource donnée sont `CreateUpdate`, et`Delete`. Le mode actuel est déterminé en fonction de ce qui s'est passé lors du dernier déploiement, et il ne peut pas être influencé autrement. Vous pouvez peut-être mettre à jour les CloudFormation ressources manuellement afin d'influencer le mode déterminé, mais vous ne pouvez pas passer une commande CloudFormation disant « Pour cette ressource, opérez en `Create` mode ».

Terraform n'étant pas hébergé dans le AWS Cloud, le processus de maintien de l'état doit être plus configurable. Pour cette raison, l'état Terraform est conservé dans un fichier d'état généré automatiquement. Un développeur Terraform doit traiter avec l'État beaucoup plus directement qu'il ne le ferait avec. CloudFormation Il est important de se rappeler que l'état du suivi est tout aussi important pour les deux outils.

Par défaut, le fichier d'état Terraform est stocké localement au niveau supérieur du répertoire principal qui exécute votre stack Terraform. Si vous exécutez la `terraform apply` commande depuis votre environnement de développement local, vous pouvez voir Terraform générer le fichier terraform.tfstate qu'il utilise pour maintenir l'état en temps réel. Pour le meilleur ou pour le pire, cela vous donne beaucoup plus de contrôle sur l'état dans Terraform que dans celui-ci. CloudFormation Bien que vous ne deviez jamais mettre à jour directement le fichier d'état, vous pouvez exécuter plusieurs commandes Terraform CLI pour mettre à jour l'état entre les déploiements. Par exemple, l'importation de Terraform vous permet d'ajouter des ressources créées en dehors de Terraform dans votre pile de déploiement. Inversement, vous pouvez supprimer une ressource de l'état en exécutant terraform state rm.

Le fait que Terraform ait besoin de stocker son état quelque part conduit à un autre concept qui ne s'applique pas à CloudFormation : le backend. Un backend Terraform est l'endroit où une pile Terraform stocke son fichier d'état après le déploiement. C'est également là qu'il s'attend à trouver le fichier d'état lorsqu'un nouveau déploiement commence. Lorsque vous exécutez votre stack localement, comme décrit ci-dessus, vous pouvez conserver une copie de l'état de Terraform dans le répertoire local de niveau supérieur. C'est ce que l'on appelle un backend local.

Lors du développement pour un environnement d'intégration et de déploiement continus (CI/CD), le fichier d'état local est généralement inclus dans le fichier .gitignore afin de le soustraire au contrôle de version. Il n'y a alors aucun fichier d'état local présent dans le pipeline. Pour fonctionner correctement, cette étape du pipeline doit trouver le bon fichier d'état quelque part.C'est pourquoi les fichiers de configuration Terraform contiennent souvent un bloc principal. Le bloc principal indique à la pile Terraform qu'elle doit chercher ailleurs que dans son propre répertoire de premier niveau pour trouver le fichier d'état.

Un backend Terraform peut être situé presque n'importe où : un compartiment Amazon S3, un point de terminaison d'API ou même un espace de travail Terraform distant. Voici un exemple de backend Terraform stocké dans un compartiment Amazon S3.

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

Afin d'éviter de stocker des informations sensibles dans les fichiers de configuration Terraform, les backends prennent également en charge les configurations partielles. Dans l'exemple précédent, les informations d'identification nécessaires pour accéder au bucket ne sont pas présentes dans la configuration. Les informations d'identification peuvent être obtenues à partir de variables d'environnement ou par d'autres moyens, tels que AWS Secrets Manager. Pour plus d'informations, consultez Sécurisation des données sensibles à l'aide AWS Secrets Manager de HashiCorp Terraform.

Un scénario de backend courant est un backend local utilisé dans votre environnement local à des fins de test. Le fichier terraform.tfstate est inclus dans le fichier .gitignore afin qu'il ne soit pas transféré vers le référentiel distant. Ensuite, chaque environnement du pipeline CI/CD conserverait son propre backend. Dans ce scénario, plusieurs développeurs peuvent avoir accès à cet état distant. Vous devez donc protéger l'intégrité du fichier d'état. Si plusieurs déploiements sont en cours d'exécution et mettent à jour l'état en même temps, le fichier d'état peut être endommagé. Pour cette raison, dans les situations où les backends ne sont pas locaux, le fichier d'état est généralement verrouillé pendant le déploiement.

Comprendre les sources de données Terraform

Il est très courant que les piles de déploiement s'appuient sur des données provenant de ressources existantes. La plupart des outils IaC ont un moyen d'importer des ressources créées par un autre processus. Ces ressources importées sont généralement en lecture seule (bien que les rôles IAM constituent une exception notable) et sont utilisées pour accéder aux données nécessaires aux ressources de la pile. AWS CloudFormation permet d'importer des ressources, mais cette idée peut être mieux expliquée en examinant le AWS Cloud Development Kit (AWS CDK).

AWS CDK Cela aide les développeurs à utiliser les langages de programmation existants pour générer des CloudFormation modèles. Le résultat final d'une AWS CDK opération est une ressource importée dans CloudFormation. Cependant, la syntaxe utilisée avec le AWS CDK facilite la comparaison avec Terraform. Voici un exemple d'importation d'une ressource à l'aide du AWS CDK.

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

Une ressource importée est généralement créée en appelant une méthode statique sur la même classe que celle que vous utilisez pour créer une nouvelle ressource du même type. L'appel `new Bucket(...` créerait une nouvelle ressource, et l'appel `Bucket.fromBucketAttributes(...` en importe une existante. Vous transmettez un sous-ensemble des propriétés du compartiment à la fonction afin qu'elle AWS CDK puisse trouver le bon compartiment. Une autre différence, cependant, est que la création d'un nouveau bucket renvoie une instance complète de la `Bucket` classe, avec toutes les propriétés et méthodes disponibles. L'importation de la ressource renvoie un`IBucket`, qui est un type qui contient uniquement les `Bucket` propriétés requises. Bien que vous puissiez importer une ressource à partir d'une pile externe, les options d'utilisation de cette ressource sont limitées.

Dans Terraform, un objectif similaire est atteint en utilisant des sources de données. La plupart des ressources Terraform définies sont associées à une source de données disponible. Voici un exemple de ressource de compartiment Terraform S3 suivie de sa source de données correspondante.

```
# S3 Bucket resource:
resource "aws_s3_bucket" "My_S3_Bucket" {
```

```
  bucket = "My_S3_Bucket"
}

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

La seule différence entre ces deux éléments est le préfixe du nom. Comme indiqué dans la documentation relative à une source de données, il existe moins de paramètres que vous pouvez transmettre à une source de données qu'à une ressource. Cela est dû au fait que la ressource utilise ces paramètres pour déclarer toutes les propriétés d'un nouveau compartiment S3, tandis que la source de données a juste besoin de suffisamment d'informations pour identifier et importer de manière unique les données d'une ressource existante.

La similitude entre la syntaxe d'une ressource Terraform et d'une source de données peut être pratique, mais elle peut également être problématique. Il est courant que les développeurs Terraform novices utilisent accidentellement une source de données plutôt qu'une ressource dans leur configuration. Les sources de données Terraform sont toujours en lecture seule. Vous pouvez les utiliser à la place de la ressource correspondante pour les actions de lecture (comme la fourniture d'un nom d'identification à une autre ressource). Cependant, vous ne pouvez pas les utiliser pour des actions d'écriture, qui modifient fondamentalement certains aspects de la ressource sous-jacente. Pour cette raison, vous pouvez considérer une source de données Terraform comme une version clonée de la ressource sous-jacente.

Comme dans le précédent exemple d'AWS CDK iBucket, les sources de données sont utiles pour les scénarios en lecture seule. Si vous avez besoin d'obtenir des données à partir d'une ressource existante mais que vous n'avez pas besoin de conserver cette ressource dans votre pile, utilisez une source de données. C'est le cas par exemple lorsque vous créez une instance Amazon EC2 qui utilise le VPC par défaut du compte. Comme ce VPC existe déjà, il vous suffit d'extraire ses données. L'exemple de code suivant montre comment utiliser les données pour identifier le VPC cible.

```
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
```

```
}
```

Comprendre les variables, les valeurs locales et les sorties Terraform

Les variables améliorent la flexibilité du code en autorisant des espaces réservés dans les blocs de code. Les variables peuvent représenter des valeurs différentes chaque fois que le code est réutilisé. Terraform distingue ses types de variables par leur portée modulaire. Les variables d'entrée sont des valeurs externes qui peuvent être injectées dans un module, les valeurs de sortie sont des valeurs internes qui peuvent être partagées en externe, et les valeurs locales restent toujours dans leur champ d'application d'origine.

Variables

AWS CloudFormation utilise des paramètres pour représenter des valeurs personnalisées qui peuvent être définies et réinitialisées d'un déploiement de stack à l'autre. De même, Terraform utilise des variables d'entrée, ou variables. Les variables peuvent être déclarées n'importe où dans un fichier de configuration Terraform et sont généralement déclarées avec le type de données requis ou la valeur par défaut. Les trois expressions suivantes sont des déclarations de variables Terraform valides.

```
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"
    }
```

```
  ]
}
```

Pour accéder à la race de Sparky dans la configuration, vous devez utiliser la variable`var.dogs[0].breed`. Si une variable n'a pas de valeur par défaut et n'est pas classée comme nulle, la valeur de la variable doit être définie pour chaque déploiement. Dans le cas contraire, il est facultatif de définir une nouvelle valeur pour la variable. Dans un module racine, vous pouvez définir les valeurs des variables actuelles sur la ligne de commande, en tant que variables d'environnement ou dans le fichier terraform.tfvars. L'exemple suivant montre comment saisir des valeurs de variables dans le fichier terraform.tfvars, qui est stocké dans le répertoire de premier niveau du module.

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

random_number = 7

thing_i_made_up = "Kabibble"
```

`dogs`Dans cet exemple, la valeur du fichier terraform.tfvars remplacerait la valeur par défaut dans la déclaration de variable. Si vous déclarez des variables dans un module enfant, vous pouvez définir les valeurs des variables directement dans le bloc de déclaration du module, comme indiqué dans l'exemple suivant.

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

Parmi les autres arguments que vous pouvez utiliser lors de la déclaration d'une variable, citons :

- `sensitive`— Définissez ce paramètre pour `true` empêcher que la valeur de la variable ne soit exposée dans les sorties du processus Terraform.
- `nullable`— La définition de cette valeur `true` permet à la variable de ne pas avoir de valeur. Cela est pratique pour les variables pour lesquelles aucune valeur par défaut n'est définie.
- `description`— Ajoutez une description de la variable aux métadonnées de la pile.
- `validation`— Définissez les règles de validation pour la variable.

L'un des aspects les plus pratiques des variables Terraform est la possibilité d'ajouter un ou plusieurs objets de validation dans la déclaration de variable. Vous pouvez utiliser des objets de validation pour ajouter une condition selon laquelle la variable doit satisfaire, faute de quoi le déploiement échoue. Vous pouvez également définir un message d'erreur personnalisé à afficher chaque fois que la condition n'est pas respectée.

Par exemple, vous configurez un fichier de configuration Terraform que les membres de votre équipe exécuteront. Avant de déployer les stacks, un membre de l'équipe doit créer un fichier terraform.tfvars pour définir une valeur de configuration importante. Pour le leur rappeler, vous pouvez faire quelque chose comme ceci.

```
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-'"
  }
}
```

Comme le montre cet exemple, vous pouvez définir plusieurs conditions dans une seule variable. Terraform affiche uniquement les messages d'erreur en cas d'échec. De cette façon, vous pouvez appliquer toutes sortes de règles sur les valeurs des variables. Si une valeur variable provoque une défaillance du pipeline, vous saurez exactement pourquoi.

Valeurs locales

Si vous souhaitez créer un alias dans un module pour certaines valeurs, utilisez le `locals` mot-clé plutôt que de déclarer une variable par défaut qui ne sera jamais mise à jour. Comme son nom l'indique, un `locals` bloc contient des termes dont le champ d'application est interne à ce module spécifique. Si vous souhaitez transformer une valeur de chaîne, par exemple en ajoutant un préfixe à une valeur de variable à utiliser dans le nom d'une ressource, l'utilisation d'une valeur locale peut être une bonne solution. Un seul `locals` bloc peut déclarer toutes les valeurs locales de votre module, comme illustré dans l'exemple suivant.

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

N'oubliez pas que lorsque vous accédez à la valeur, le `locals` mot-clé devient singulier, par exemple`local.LocalConfigId`.

Valeurs de sortie

Si les variables d'entrée Terraform sont comme des CloudFormation paramètres, on peut dire que les valeurs de sortie Terraform sont comme des sorties. CloudFormation Les deux sont utilisés pour exposer les valeurs issues d'une pile de déploiement. Cependant, étant donné que le module Terraform est davantage ancré dans la structure de l'outil, les valeurs de sortie Terraform sont également utilisées pour exposer les valeurs d'un module à un module parent ou à d'autres modules enfants, même si ces modules font tous partie de la même pile de déploiement. Si vous créez deux modules personnalisés et que le premier module doit accéder à la valeur ID du second module, vous devez ajouter le `output` bloc suivant au second module.

```
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
}
```

Étant donné que les valeurs de sortie Terraform peuvent être utilisées dans la même pile, vous pouvez également utiliser l'`sensitive`attribut dans un `output` bloc pour empêcher l'affichage de la valeur dans la sortie de la pile. De plus, un `output` bloc peut utiliser des `precondition` blocs de la même manière que les variables utilisent des `validation` blocs : pour garantir que les variables suivent un certain ensemble de règles. Cela permet de s'assurer que toutes les valeurs d'un module existent comme prévu avant de procéder au déploiement.

```
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."
  }
}
```

Comprendre les fonctions, expressions et méta-arguments de Terraform

L'une des critiques à l'encontre des outils IaC qui utilisent des fichiers de configuration déclaratifs plutôt que des langages de programmation courants est qu'ils compliquent la mise en œuvre d'une logique de programmation personnalisée. Dans les configurations Terraform, ce problème est résolu à l'aide de fonctions, d'expressions et de méta-arguments.

Fonctions

L'un des grands avantages de l'utilisation du code pour provisionner votre infrastructure est la possibilité de stocker des flux de travail courants et de les réutiliser encore et encore, en utilisant souvent des arguments différents à chaque fois. Les fonctions Terraform sont similaires aux fonctions AWS CloudFormation intrinsèques, bien que leur syntaxe soit plus proche de la façon dont les fonctions sont appelées dans les langages de programmation. Vous avez peut-être déjà remarqué certaines fonctions Terraform, telles que substr, concat, length et base64decode, dans les exemples de ce guide. Comme CloudFormation pour les fonctions intrinsèques, Terraform possède une série de fonctions intégrées qui peuvent être utilisées dans vos configurations. Par exemple, si un attribut de ressource particulier prend un objet JSON très volumineux qu'il serait inefficace de coller directement dans le fichier, vous pouvez placer l'objet dans un fichier .json et utiliser les fonctions Terraform pour y accéder. Dans l'exemple suivant, la `file` fonction renvoie le contenu du fichier sous forme de chaîne, puis le `jsondecode` convertit en un type d'objet.

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

Expressions

Terraform autorise également les expressions conditionnelles, qui sont similaires aux CloudFormation `condition` fonctions, sauf qu'elles utilisent la syntaxe plus traditionnelle des opérateurs ternaires. Dans l'exemple suivant, les deux expressions renvoient exactement le même résultat. Le deuxième exemple est ce que Terraform appelle une expression splat. L'astérisque permet à Terraform de parcourir la liste en boucle et de créer une nouvelle liste en utilisant uniquement la `id` propriété de chaque élément.

```
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
}
```

Méta-arguments

Dans l'exemple de code précédent, `list_value` et `list_value_2` sont appelés arguments. Vous connaissez peut-être déjà certains de ces méta-arguments. Terraform possède également quelques méta-arguments, qui agissent comme des arguments mais avec quelques fonctionnalités supplémentaires :

- Le méta-argument depends_on est très similaire à l'attribut. CloudFormation DependsOn
- Le méta-argument du fournisseur vous permet d'utiliser plusieurs configurations de fournisseur à la fois.
- Le méta-argument du cycle de vie vous permet de personnaliser les paramètres des ressources, de la même manière que les politiques de suppression et de suppression dans. CloudFormation

D'autres méta-arguments permettent d'ajouter des fonctionnalités de fonction et d'expression directement à une ressource. Par exemple, le méta-argument count est un mécanisme utile pour créer plusieurs ressources similaires en même temps. L'exemple suivant montre comment créer deux clusters Amazon Elastic Container Service (Amazon EKS) sans utiliser le `count` méta-argument.

```
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]
  }
}
```

L'exemple suivant montre comment utiliser le `count` méta-argument pour créer deux clusters Amazon EKS.

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

Pour attribuer un nom d'unité à chacune, vous pouvez accéder à l'index de la liste dans le bloc de ressources à l'adresse`count.index`. Mais que faire si vous souhaitez créer plusieurs ressources similaires un peu plus complexes ? C'est là qu'intervient le méta-argument for_each. Le `for_each` méta-argument est très similaire à`count`, sauf que vous transmettez une liste ou un objet au lieu d'un nombre. Terraform crée une nouvelle ressource pour chaque membre de la liste ou de l'objet. C'est la même chose que si vous définissez`count = length(list)`, sauf que vous pouvez accéder au contenu de la liste plutôt qu'à l'index de la boucle.

Cela fonctionne à la fois pour une liste d'éléments ou pour un seul objet. L'exemple suivant créerait deux ressources dotées d'un `id-0` identifiant et `id-1` d'un identifiant.

```
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
}
```

L'exemple suivant créerait également deux ressources, l'une pour Sparky, le caniche, et l'autre pour Fluffy, le chihuahua.

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

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

Tout comme vous pouvez accéder à l'index de boucle dans count en utilisant count.index, vous pouvez accéder à la clé et à la valeur de chaque élément d'une boucle for_each en utilisant l'objet each. Comme for_each itère à la fois sur les listes et sur les objets, le suivi de chaque clé et de chaque valeur peut être un peu confus. Le tableau suivant montre les différentes manières d'utiliser le méta-argument for_each et la manière dont vous pouvez référencer les valeurs à chaque itération.

Exemple	**`for_each`** type	Première itération	Deuxième itération

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

Exemple	**for_each** type	Première itération	Deuxième itération
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”`

Exemple	**for_each** type	Première itération	Deuxième itération

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”`
			
			`}`

Exemple	**for_each** type	Première itération	Deuxième itération
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” }]

Exemple	**for_each** type	Première itération	Deuxième itération
	name = “Morris” }] }		

Donc, si `var.animals` c'est égal à la ligne E, vous pouvez créer une ressource par animal en utilisant le code suivant.

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

Vous pouvez également créer deux ressources par animal en utilisant le code suivant.

```
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
}
```

FAQ

Quand dois-je utiliser Terraform au lieu de ? CloudFormation

En général, si vos charges de travail sont principalement basées sur AWS, AWS CloudFormation fournit un niveau de support natif que Terraform ne peut égaler. Toutefois, si vos charges de travail incluent un certain nombre de processus tiers ou si elles sont réparties entre plusieurs fournisseurs de cloud, Terraform est un outil que vous devriez peut-être envisager.

Quand dois-je utiliser le au AWS CDK lieu de CloudFormation ?

Lorsque vous utilisez le AWS Cloud Development Kit (AWS CDK), vous utilisez également CloudFormation. Vous AWS CDK permet d'utiliser un langage de programmation courant pour générer des CloudFormation modèles. Si vous maîtrisez l'un des langages de programmation pris AWS CDK en charge, AWS CDK vous pouvez réduire le temps nécessaire à la génération de CloudFormation modèles.

Existe-t-il un outil comme celui-ci AWS CDK qui génère des configurations Terraform ?

Par rapport au AWS CDK, le CDK pour Terraform (CDKTF) utilise la même bibliothèque de constructions pour approvisionner les ressources et le même moteur jsii pour prendre en charge plusieurs langages de programmation. Vous pouvez l'utiliser pour générer des configurations Terraform de la même manière que AWS CDK pour générer CloudFormation des modèles.

Comment puis-je en savoir plus sur Terraform ?

Pour plus d'informations sur les concepts avancés de Terraform, consultez la documentation Terraform. Il décrit également les composants de tous les principaux fournisseurs et modules open source.

Ressources connexes

AWS documentation

- Documentation AWS CDK
- Documentation AWS CloudFormation
- Terraform : Au-delà des bases avec AWS (AWS article de blog)

Autres ressources

- Documentation du CDK pour Terraform
- Documentation Terraform

Annexe : Exemples d'accès aux attributs Terraform

Ressource

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

bucketName = aws_s3_bucket.myS3Bucket.bucket
```

Source de données

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

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

Module

```
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
```

Historique du document

Le tableau suivant décrit les modifications importantes apportées à ce guide. Pour être averti des mises à jour à venir, abonnez-vous à un fil RSS.

Modification	Description	Date
Publication initiale	—	29 mars 2024

AWS Glossaire des directives prescriptives

Les termes suivants sont couramment utilisés dans les stratégies, les guides et les modèles fournis par les directives AWS prescriptives. Pour suggérer des entrées, veuillez utiliser le lien Faire un commentaire à la fin du glossaire.

Nombres

7 R

Sept politiques de migration courantes pour transférer des applications vers le cloud. Ces politiques s'appuient sur les 5 R identifiés par Gartner en 2011 et sont les suivantes :

- Refactorisation/réarchitecture : transférez une application et modifiez son architecture en tirant pleinement parti des fonctionnalités natives cloud pour améliorer l'agilité, les performances et la capacité de mise à l'échelle. Cela implique généralement le transfert du système d'exploitation et de la base de données. Exemple : migrez votre base de données Oracle sur site vers l'édition compatible avec Amazon Aurora PostgreSQL.
- Replateformer (déplacer et remodeler) : transférez une application vers le cloud et introduisez un certain niveau d'optimisation pour tirer parti des fonctionnalités du cloud. Exemple : migrez votre base de données Oracle sur site vers Amazon Relational Database Service (Amazon RDS) pour Oracle dans le. AWS Cloud
- Racheter (rachat) : optez pour un autre produit, généralement en passant d'une licence traditionnelle à un modèle SaaS. Exemple : migrez votre système de gestion de la relation client (CRM) vers Salesforce.com.
- Réhéberger (lift and shift) : transférez une application vers le cloud sans apporter de modifications pour tirer parti des fonctionnalités du cloud. Exemple : migrez votre base de données Oracle sur site vers Oracle sur une instance EC2 dans le. AWS Cloud
- Relocaliser (lift and shift au niveau de l'hyperviseur) : transférez l'infrastructure vers le cloud sans acheter de nouveau matériel, réécrire des applications ou modifier vos opérations existantes. Vous migrez des serveurs d'une plateforme sur site vers un service cloud pour la même plateforme. Exemple : migrer une Microsoft Hyper-V application vers AWS.
- Retenir : conservez les applications dans votre environnement source. Il peut s'agir d'applications nécessitant une refactorisation majeure, que vous souhaitez retarder, et d'applications existantes que vous souhaitez retenir, car rien ne justifie leur migration sur le plan commercial.

- Retirer : mettez hors service ou supprimez les applications dont vous n'avez plus besoin dans votre environnement source.

A

ABAC

Voir contrôle d'accès basé sur les attributs.

services abstraits

Consultez la section Services gérés.

ACIDE

Voir atomicité, consistance, isolation, durabilité.

migration active-active

Méthode de migration de base de données dans laquelle la synchronisation des bases de données source et cible est maintenue (à l'aide d'un outil de réplication bidirectionnelle ou d'opérations d'écriture double), tandis que les deux bases de données gèrent les transactions provenant de la connexion d'applications pendant la migration. Cette méthode prend en charge la migration par petits lots contrôlés au lieu d'exiger un basculement ponctuel. Elle est plus flexible mais demande plus de travail qu'une migration active-passive.

migration active-passive

Méthode de migration de base de données dans laquelle la synchronisation des bases de données source et cible est maintenue, mais seule la base de données source gère les transactions provenant de la connexion d'applications pendant que les données sont répliquées vers la base de données cible. La base de données cible n'accepte aucune transaction pendant la migration.

fonction d'agrégation

Fonction SQL qui agit sur un groupe de lignes et calcule une valeur de retour unique pour le groupe. Des exemples de fonctions d'agrégation incluent `SUM` et`MAX`.

AI

Voir intelligence artificielle.

AIOps

Voir les opérations d'intelligence artificielle.

anonymisation

Processus de suppression définitive d'informations personnelles dans un ensemble de données. L'anonymisation peut contribuer à protéger la vie privée. Les données anonymisées ne sont plus considérées comme des données personnelles.

anti-motif

Solution fréquemment utilisée pour un problème récurrent lorsque la solution est contre-productive, inefficace ou moins efficace qu'une solution alternative.

contrôle des applications

Une approche de sécurité qui permet d'utiliser uniquement des applications approuvées afin de protéger un système contre les logiciels malveillants.

portefeuille d'applications

Ensemble d'informations détaillées sur chaque application utilisée par une organisation, y compris le coût de génération et de maintenance de l'application, ainsi que sa valeur métier. Ces informations sont essentielles pour le processus de découverte et d'analyse du portefeuille et permettent d'identifier et de prioriser les applications à migrer, à moderniser et à optimiser.

intelligence artificielle (IA)

Domaine de l'informatique consacré à l'utilisation des technologies de calcul pour exécuter des fonctions cognitives généralement associées aux humains, telles que l'apprentissage, la résolution de problèmes et la reconnaissance de modèles. Pour plus d'informations, veuillez consulter Qu'est-ce que l'intelligence artificielle ?

opérations d'intelligence artificielle (AIOps)

Processus consistant à utiliser des techniques de machine learning pour résoudre les problèmes opérationnels, réduire les incidents opérationnels et les interventions humaines, mais aussi améliorer la qualité du service. Pour plus d'informations sur la façon dont les AIOps sont utilisées dans la stratégie de migration AWS , veuillez consulter le guide d'intégration des opérations.

chiffrement asymétrique

Algorithme de chiffrement qui utilise une paire de clés, une clé publique pour le chiffrement et une clé privée pour le déchiffrement. Vous pouvez partager la clé publique, car elle n'est pas utilisée pour le déchiffrement, mais l'accès à la clé privée doit être très restreint.

atomicité, cohérence, isolement, durabilité (ACID)

Ensemble de propriétés logicielles garantissant la validité des données et la fiabilité opérationnelle d'une base de données, même en cas d'erreur, de panne de courant ou d'autres problèmes.

contrôle d'accès par attributs (ABAC)

Pratique qui consiste à créer des autorisations détaillées en fonction des attributs de l'utilisateur, tels que le service, le poste et le nom de l'équipe. Pour plus d'informations, consultez ABAC pour AWS dans la documentation AWS Identity and Access Management (IAM).

source de données faisant autorité

Emplacement où vous stockez la version principale des données, considérée comme la source d'information la plus fiable. Vous pouvez copier les données de la source de données officielle vers d'autres emplacements à des fins de traitement ou de modification des données, par exemple en les anonymisant, en les expurgant ou en les pseudonymisant.

Zone de disponibilité

Un emplacement distinct au sein d'un Région AWS réseau isolé des défaillances dans d'autres zones de disponibilité et fournissant une connectivité réseau peu coûteuse et à faible latence aux autres zones de disponibilité de la même région.

AWS Cadre d'adoption du cloud (AWS CAF)

Un cadre de directives et de meilleures pratiques visant AWS à aider les entreprises à élaborer un plan efficace pour réussir leur migration vers le cloud. AWS La CAF organise ses conseils en six domaines prioritaires appelés perspectives : les affaires, les personnes, la gouvernance, les plateformes, la sécurité et les opérations. Les perspectives d'entreprise, de personnes et de gouvernance mettent l'accent sur les compétences et les processus métier, tandis que les perspectives relatives à la plateforme, à la sécurité et aux opérations se concentrent sur les compétences et les processus techniques. Par exemple, la perspective liée aux personnes cible les parties prenantes qui s'occupent des ressources humaines (RH), des fonctions de dotation en personnel et de la gestion des personnes. Dans cette perspective, la AWS CAF fournit des conseils pour le développement du personnel, la formation et les communications afin de préparer

l'organisation à une adoption réussie du cloud. Pour plus d'informations, veuillez consulter le site Web AWS CAF et le livre blanc AWS CAF.

AWS Cadre de qualification de la charge de travail (AWS WQF)

Outil qui évalue les charges de travail liées à la migration des bases de données, recommande des stratégies de migration et fournit des estimations de travail. AWS Le WQF est inclus avec AWS Schema Conversion Tool (AWS SCT). Il analyse les schémas de base de données et les objets de code, le code d'application, les dépendances et les caractéristiques de performance, et fournit des rapports d'évaluation.

B

mauvais bot

Un bot destiné à perturber ou à nuire à des individus ou à des organisations.

BCP

Consultez la section Planification de la continuité des activités.

graphique de comportement

Vue unifiée et interactive des comportements des ressources et des interactions au fil du temps. Vous pouvez utiliser un graphique de comportement avec Amazon Detective pour examiner les tentatives de connexion infructueuses, les appels d'API suspects et les actions similaires. Pour plus d'informations, veuillez consulter Data in a behavior graph dans la documentation Detective.

système de poids fort

Système qui stocke d'abord l'octet le plus significatif. Voir aussi endianité.

classification binaire

Processus qui prédit un résultat binaire (l'une des deux classes possibles). Par exemple, votre modèle de machine learning peut avoir besoin de prévoir des problèmes tels que « Cet e-mail est-il du spam ou non ? » ou « Ce produit est-il un livre ou une voiture ? ».

filtre de Bloom

Structure de données probabiliste et efficace en termes de mémoire qui est utilisée pour tester si un élément fait partie d'un ensemble.

déploiement bleu/vert

Stratégie de déploiement dans laquelle vous créez deux environnements distincts mais identiques. Vous exécutez la version actuelle de l'application dans un environnement (bleu) et la nouvelle version de l'application dans l'autre environnement (vert). Cette stratégie vous permet de revenir rapidement en arrière avec un impact minimal.

bot

Application logicielle qui exécute des tâches automatisées sur Internet et simule l'activité ou l'interaction humaine. Certains robots sont utiles ou bénéfiques, comme les robots d'exploration Web qui indexent des informations sur Internet. D'autres robots, connus sous le nom de mauvais robots, sont destinés à perturber ou à nuire à des individus ou à des organisations.

botnet

Réseaux de robots infectés par des logiciels malveillants et contrôlés par une seule entité, connue sous le nom d'herder ou d'opérateur de bots. Les botnets sont le mécanisme le plus connu pour faire évoluer les bots et leur impact.

branche

Zone contenue d'un référentiel de code. La première branche créée dans un référentiel est la branche principale. Vous pouvez créer une branche à partir d'une branche existante, puis développer des fonctionnalités ou corriger des bogues dans la nouvelle branche. Une branche que vous créez pour générer une fonctionnalité est communément appelée branche de fonctionnalités. Lorsque la fonctionnalité est prête à être publiée, vous fusionnez à nouveau la branche de fonctionnalités dans la branche principale. Pour plus d'informations, consultez À propos des branches (GitHub documentation).

accès par brise-vitre

Dans des circonstances exceptionnelles et par le biais d'un processus approuvé, c'est un moyen rapide pour un utilisateur d'accéder à un accès auquel Compte AWS il n'est généralement pas autorisé. Pour plus d'informations, consultez l'indicateur Implementation break-glass procedures dans le guide Well-Architected AWS .

stratégie existante (brownfield)

L'infrastructure existante de votre environnement. Lorsque vous adoptez une stratégie existante pour une architecture système, vous concevez l'architecture en fonction des contraintes des systèmes et de l'infrastructure actuels. Si vous étendez l'infrastructure existante, vous pouvez combiner des politiques brownfield (existantes) et greenfield (inédites).

cache de tampon

Zone de mémoire dans laquelle sont stockées les données les plus fréquemment consultées.

capacité métier

Ce que fait une entreprise pour générer de la valeur (par exemple, les ventes, le service client ou le marketing). Les architectures de microservices et les décisions de développement peuvent être dictées par les capacités métier. Pour plus d'informations, veuillez consulter la section Organisation en fonction des capacités métier du livre blanc Exécution de microservices conteneurisés sur AWS.

planification de la continuité des activités (BCP)

Plan qui tient compte de l'impact potentiel d'un événement perturbateur, tel qu'une migration à grande échelle, sur les opérations, et qui permet à une entreprise de reprendre ses activités rapidement.

C

CAF

Voir le cadre d'adoption du AWS cloud.

déploiement de Canary

Diffusion lente et progressive d'une version pour les utilisateurs finaux. Lorsque vous êtes sûr, vous déployez la nouvelle version et remplacez la version actuelle dans son intégralité.

CCoE

Voir le Centre d'excellence du cloud.

CDC

Consultez la section Capture des données de modification.

capture des données de modification (CDC)

Processus de suivi des modifications apportées à une source de données, telle qu'une table de base de données, et d'enregistrement des métadonnées relatives à ces modifications. Vous pouvez utiliser la CDC à diverses fins, telles que l'audit ou la réplication des modifications dans un système cible afin de maintenir la synchronisation.

ingénierie du chaos

Introduire intentionnellement des défaillances ou des événements perturbateurs pour tester la résilience d'un système. Vous pouvez utiliser AWS Fault Injection Service (AWS FIS) pour effectuer des expériences qui stressent vos AWS charges de travail et évaluer leur réponse.

CI/CD

Découvrez l'intégration continue et la livraison continue.

classification

Processus de catégorisation qui permet de générer des prédictions. Les modèles de ML pour les problèmes de classification prédisent une valeur discrète. Les valeurs discrètes se distinguent toujours les unes des autres. Par exemple, un modèle peut avoir besoin d'évaluer la présence ou non d'une voiture sur une image.

chiffrement côté client

Chiffrement des données localement, avant que la cible ne les Service AWS reçoive.

Centre d'excellence cloud (CCoE)

Une équipe multidisciplinaire qui dirige les efforts d'adoption du cloud au sein d'une organisation, notamment en développant les bonnes pratiques en matière de cloud, en mobilisant des ressources, en établissant des délais de migration et en guidant l'organisation dans le cadre de transformations à grande échelle. Pour plus d'informations, consultez les articles du CCoE sur le blog de stratégie AWS Cloud d'entreprise.

cloud computing

Technologie cloud généralement utilisée pour le stockage de données à distance et la gestion des appareils IoT. Le cloud computing est généralement associé à la technologie informatique de pointe.

modèle d'exploitation du cloud

Dans une organisation informatique, modèle d'exploitation utilisé pour créer, faire évoluer et optimiser un ou plusieurs environnements cloud. Pour plus d'informations, consultez la section Création de votre modèle d'exploitation cloud.

étapes d'adoption du cloud

Les quatre phases que les entreprises traversent généralement lorsqu'elles migrent vers AWS Cloud :

- Projet : exécution de quelques projets liés au cloud à des fins de preuve de concept et d'apprentissage
- Base : réaliser des investissements fondamentaux pour mettre à l'échelle l'adoption du cloud (par exemple, en créant une zone de destination, en définissant un CCoE ou en établissant un modèle opérationnel)
- Migration : migration d'applications individuelles
- Réinvention : optimisation des produits et services et innovation dans le cloud

Ces étapes ont été définies par Stephen Orban dans le billet de blog The Journey Toward Cloud-First & the Stages of Adoption publié sur le blog AWS Cloud Enterprise Strategy. Pour plus d'informations sur leur lien avec la stratégie de AWS migration, consultez le guide de préparation à la migration.

CMDB

Voir base de données de gestion de configuration.

référentiel de code

Emplacement où le code source et d'autres ressources, comme la documentation, les exemples et les scripts, sont stockés et mis à jour par le biais de processus de contrôle de version. Les référentiels cloud courants incluent GitHub ou AWS CodeCommit. Chaque version du code est appelée branche. Dans une structure de microservice, chaque référentiel est consacré à une seule fonctionnalité. Un seul pipeline CI/CD peut utiliser plusieurs référentiels.

cache passif

Cache tampon vide, mal rempli ou contenant des données obsolètes ou non pertinentes. Cela affecte les performances, car l'instance de base de données doit lire à partir de la mémoire principale ou du disque, ce qui est plus lent que la lecture à partir du cache tampon.

données gelées

Données rarement consultées et généralement historiques. Lorsque vous interrogez ce type de données, les requêtes lentes sont généralement acceptables. Le transfert de ces données vers des niveaux ou classes de stockage moins performants et moins coûteux peut réduire les coûts.

vision par ordinateur (CV)

Domaine de l'IA qui utilise l'apprentissage automatique pour analyser et extraire des informations à partir de formats visuels tels que des images numériques et des vidéos. Par exemple, AWS

Panorama propose des appareils qui ajoutent des CV aux réseaux de caméras locaux, et Amazon SageMaker fournit des algorithmes de traitement d'image pour les CV.

dérive de configuration

Pour une charge de travail, une modification de configuration par rapport à l'état attendu. Cela peut entraîner une non-conformité de la charge de travail, et cela est généralement progressif et involontaire.

base de données de gestion des configurations (CMDB)

Référentiel qui stocke et gère les informations relatives à une base de données et à son environnement informatique, y compris les composants matériels et logiciels ainsi que leurs configurations. Vous utilisez généralement les données d'une CMDB lors de la phase de découverte et d'analyse du portefeuille de la migration.

pack de conformité

Ensemble de AWS Config règles et d'actions correctives que vous pouvez assembler pour personnaliser vos contrôles de conformité et de sécurité. Vous pouvez déployer un pack de conformité en tant qu'entité unique dans une région Compte AWS et, ou au sein d'une organisation, à l'aide d'un modèle YAML. Pour plus d'informations, consultez la section Packs de conformité dans la AWS Config documentation.

intégration continue et livraison continue (CI/CD)

Processus d'automatisation des étapes source, de génération, de test, intermédiaire et de production du processus de publication du logiciel. CI/CD est communément décrit comme un pipeline. CI/CD peut vous aider à automatiser les processus, à améliorer la productivité, à améliorer la qualité du code et à accélérer les livraisons. Pour plus d'informations, veuillez consulter Avantages de la livraison continue. CD peut également signifier déploiement continu. Pour plus d'informations, veuillez consulter Livraison continue et déploiement continu.

CV

Voir vision par ordinateur.

D

données au repos

Données stationnaires dans votre réseau, telles que les données stockées.

classification des données

Processus permettant d'identifier et de catégoriser les données de votre réseau en fonction de leur sévérité et de leur sensibilité. Il s'agit d'un élément essentiel de toute stratégie de gestion des risques de cybersécurité, car il vous aide à déterminer les contrôles de protection et de conservation appropriés pour les données. La classification des données est une composante du pilier de sécurité du AWS Well-Architected Framework. Pour plus d'informations, veuillez consulter Classification des données.

dérive des données

Une variation significative entre les données de production et les données utilisées pour entraîner un modèle ML, ou une modification significative des données d'entrée au fil du temps. La dérive des données peut réduire la qualité, la précision et l'équité globales des prédictions des modèles ML.

données en transit

Données qui circulent activement sur votre réseau, par exemple entre les ressources du réseau.

maillage de données

Un cadre architectural qui fournit une propriété des données distribuée et décentralisée avec une gestion et une gouvernance centralisées.

minimisation des données

Le principe de collecte et de traitement des seules données strictement nécessaires. La pratique de la minimisation des données AWS Cloud peut réduire les risques liés à la confidentialité, les coûts et l'empreinte carbone de vos analyses.

périmètre de données

Ensemble de garde-fous préventifs dans votre AWS environnement qui permettent de garantir que seules les identités fiables accèdent aux ressources fiables des réseaux attendus. Pour plus d'informations, voir Création d'un périmètre de données sur AWS.

prétraitement des données

Pour transformer les données brutes en un format facile à analyser par votre modèle de ML. Le prétraitement des données peut impliquer la suppression de certaines colonnes ou lignes et le traitement des valeurs manquantes, incohérentes ou en double.

provenance des données

Le processus de suivi de l'origine et de l'historique des données tout au long de leur cycle de vie, par exemple la manière dont les données ont été générées, transmises et stockées.

sujet des données

Personne dont les données sont collectées et traitées.

entrepôt des données

Un système de gestion des données qui prend en charge les informations commerciales, telles que les analyses. Les entrepôts de données contiennent généralement de grandes quantités de données historiques et sont généralement utilisés pour les requêtes et les analyses.

langage de définition de base de données (DDL)

Instructions ou commandes permettant de créer ou de modifier la structure des tables et des objets dans une base de données.

langage de manipulation de base de données (DML)

Instructions ou commandes permettant de modifier (insérer, mettre à jour et supprimer) des informations dans une base de données.

DDL

Voir langage de définition de base de données.

ensemble profond

Sert à combiner plusieurs modèles de deep learning à des fins de prédiction. Vous pouvez utiliser des ensembles profonds pour obtenir une prévision plus précise ou pour estimer l'incertitude des prédictions.

deep learning

Un sous-champ de ML qui utilise plusieurs couches de réseaux neuronaux artificiels pour identifier le mappage entre les données d'entrée et les variables cibles d'intérêt.

defense-in-depth

Approche de la sécurité de l'information dans laquelle une série de mécanismes et de contrôles de sécurité sont judicieusement répartis sur l'ensemble d'un réseau informatique afin de protéger la confidentialité, l'intégrité et la disponibilité du réseau et des données qu'il contient. Lorsque vous adoptez cette stratégie AWS, vous ajoutez plusieurs contrôles à différentes couches de

la AWS Organizations structure afin de sécuriser les ressources. Par exemple, une defense-in-depth approche peut combiner l'authentification multifactorielle, la segmentation du réseau et le chiffrement.

administrateur délégué

Dans AWS Organizations, un service compatible peut enregistrer un compte AWS membre pour administrer les comptes de l'organisation et gérer les autorisations pour ce service. Ce compte est appelé administrateur délégué pour ce service. Pour plus d'informations et une liste des services compatibles, veuillez consulter la rubrique Services qui fonctionnent avec AWS Organizations dans la documentation AWS Organizations .

déploiement

Processus de mise à disposition d'une application, de nouvelles fonctionnalités ou de corrections de code dans l'environnement cible. Le déploiement implique la mise en œuvre de modifications dans une base de code, puis la génération et l'exécution de cette base de code dans les environnements de l'application.

environnement de développement

Voir environnement.

contrôle de détection

Contrôle de sécurité conçu pour détecter, journaliser et alerter après la survenue d'un événement. Ces contrôles constituent une deuxième ligne de défense et vous alertent en cas d'événements de sécurité qui ont contourné les contrôles préventifs en place. Pour plus d'informations, veuillez consulter la rubrique Contrôles de détection dans Implementing security controls on AWS.

cartographie de la chaîne de valeur du développement (DVSM)

Processus utilisé pour identifier et hiérarchiser les contraintes qui nuisent à la rapidité et à la qualité du cycle de vie du développement logiciel. DVSM étend le processus de cartographie de la chaîne de valeur initialement conçu pour les pratiques de production allégée. Il met l'accent sur les étapes et les équipes nécessaires pour créer et transférer de la valeur tout au long du processus de développement logiciel.

jumeau numérique

Représentation virtuelle d'un système réel, tel qu'un bâtiment, une usine, un équipement industriel ou une ligne de production. Les jumeaux numériques prennent en charge la maintenance prédictive, la surveillance à distance et l'optimisation de la production.

tableau des dimensions

Dans un schéma en étoile, table plus petite contenant les attributs de données relatifs aux données quantitatives d'une table de faits. Les attributs des tables de dimensions sont généralement des champs de texte ou des nombres discrets qui se comportent comme du texte. Ces attributs sont couramment utilisés pour la contrainte des requêtes, le filtrage et l'étiquetage des ensembles de résultats.

catastrophe

Un événement qui empêche une charge de travail ou un système d'atteindre ses objectifs commerciaux sur son site de déploiement principal. Ces événements peuvent être des catastrophes naturelles, des défaillances techniques ou le résultat d'actions humaines, telles qu'une mauvaise configuration involontaire ou une attaque de logiciel malveillant.

reprise après sinistre (DR)

La stratégie et le processus que vous utilisez pour minimiser les temps d'arrêt et les pertes de données causés par un sinistre. Pour plus d'informations, consultez Disaster Recovery of Workloads on AWS : Recovery in the Cloud in the AWS Well-Architected Framework.

DML

Voir langage de manipulation de base de données.

conception axée sur le domaine

Approche visant à développer un système logiciel complexe en connectant ses composants à des domaines évolutifs, ou objectifs métier essentiels, que sert chaque composant. Ce concept a été introduit par Eric Evans dans son ouvrage Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston : Addison-Wesley Professional, 2003). Pour plus d'informations sur l'utilisation du design piloté par domaine avec le modèle de figuier étrangleur, veuillez consulter Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

DR

Consultez la section Reprise après sinistre.

détection de dérive

Suivi des écarts par rapport à une configuration de référence. Par exemple, vous pouvez l'utiliser AWS CloudFormation pour détecter la dérive des ressources du système ou AWS Control Tower

pour détecter les modifications de votre zone d'atterrissage susceptibles d'affecter le respect des exigences de gouvernance.

DVSM

Voir la cartographie de la chaîne de valeur du développement.

E

EDA

Voir analyse exploratoire des données.

informatique de périphérie

Technologie qui augmente la puissance de calcul des appareils intelligents en périphérie d'un réseau IoT. Comparé au cloud computing, l'informatique de pointe peut réduire la latence des communications et améliorer le temps de réponse.

chiffrement

Processus informatique qui transforme des données en texte clair, lisibles par l'homme, en texte chiffré.

clé de chiffrement

Chaîne cryptographique de bits aléatoires générée par un algorithme cryptographique. La longueur des clés peut varier, et chaque clé est conçue pour être imprévisible et unique.

endianisme

Ordre selon lequel les octets sont stockés dans la mémoire de l'ordinateur. Les systèmes de poids fort stockent d'abord l'octet le plus significatif. Les systèmes de poids faible stockent d'abord l'octet le moins significatif.

point de terminaison

Voir point de terminaison de service.

service de point de terminaison

Service que vous pouvez héberger sur un cloud privé virtuel (VPC) pour le partager avec d'autres utilisateurs. Vous pouvez créer un service de point de terminaison avec AWS PrivateLink et accorder des autorisations à d'autres principaux Comptes AWS ou à AWS Identity and Access Management (IAM) principaux. Ces comptes ou principaux peuvent se connecter à votre

service de point de terminaison de manière privée en créant des points de terminaison d'un VPC d'interface. Pour plus d'informations, veuillez consulter Création d'un service de point de terminaison dans la documentation Amazon Virtual Private Cloud (Amazon VPC).

planification des ressources d'entreprise (ERP)

Système qui automatise et gère les principaux processus métier (tels que la comptabilité, le MES et la gestion de projet) pour une entreprise.

chiffrement d'enveloppe

Processus de chiffrement d'une clé de chiffrement à l'aide d'une autre clé de chiffrement. Pour plus d'informations, consultez la section Chiffrement des enveloppes dans la documentation AWS Key Management Service (AWS KMS).

environment

Instance d'une application en cours d'exécution. Les types d'environnement les plus courants dans le cloud computing sont les suivants :

- Environnement de développement : instance d'une application en cours d'exécution à laquelle seule l'équipe principale chargée de la maintenance de l'application peut accéder. Les environnements de développement sont utilisés pour tester les modifications avant de les promouvoir dans les environnements supérieurs. Ce type d'environnement est parfois appelé environnement de test.
- Environnements inférieurs : tous les environnements de développement d'une application, tels que ceux utilisés pour les générations et les tests initiaux.
- Environnement de production : instance d'une application en cours d'exécution à laquelle les utilisateurs finaux peuvent accéder. Dans un pipeline CI/CD, l'environnement de production est le dernier environnement de déploiement.
- Environnements supérieurs : tous les environnements accessibles aux utilisateurs autres que l'équipe de développement principale. Ils peuvent inclure un environnement de production, des environnements de préproduction et des environnements pour les tests d'acceptation par les utilisateurs.

épopée

Dans les méthodologies agiles, catégories fonctionnelles qui aident à organiser et à prioriser votre travail. Les épopées fournissent une description détaillée des exigences et des tâches d'implémentation. Par exemple, les points forts de la AWS CAF en matière de sécurité incluent la gestion des identités et des accès, les contrôles de détection, la sécurité des infrastructures,

la protection des données et la réponse aux incidents. Pour plus d'informations sur les épopées dans la stratégie de migration AWS , veuillez consulter le guide d'implémentation du programme.

ERP

Voir Planification des ressources d'entreprise.

analyse exploratoire des données (EDA)

Processus d'analyse d'un jeu de données pour comprendre ses principales caractéristiques. Vous collectez ou agrégez des données, puis vous effectuez des enquêtes initiales pour trouver des modèles, détecter des anomalies et vérifier les hypothèses. L'EDA est réalisée en calculant des statistiques récapitulatives et en créant des visualisations de données.

F

tableau des faits

La table centrale dans un schéma en étoile. Il stocke des données quantitatives sur les opérations commerciales. Généralement, une table de faits contient deux types de colonnes : celles qui contiennent des mesures et celles qui contiennent une clé étrangère pour une table de dimensions.

échouer rapidement

Une philosophie qui utilise des tests fréquents et progressifs pour réduire le cycle de vie du développement. C'est un élément essentiel d'une approche agile.

limite d'isolation des défauts

Dans le AWS Cloud, une limite telle qu'une zone de disponibilité Région AWS, un plan de contrôle ou un plan de données qui limite l'effet d'une panne et contribue à améliorer la résilience des charges de travail. Pour plus d'informations, consultez la section Limites d'isolation des AWS pannes.

branche de fonctionnalités

Voir la succursale.

fonctionnalités

Les données d'entrée que vous utilisez pour faire une prédiction. Par exemple, dans un contexte de fabrication, les fonctionnalités peuvent être des images capturées périodiquement à partir de la ligne de fabrication.

importance des fonctionnalités

Le niveau d'importance d'une fonctionnalité pour les prédictions d'un modèle. Il s'exprime généralement sous la forme d'un score numérique qui peut être calculé à l'aide de différentes techniques, telles que la méthode Shapley Additive Explanations (SHAP) et les gradients intégrés. Pour plus d'informations, voir Interprétabilité du modèle d'apprentissage automatique avec :AWS.

transformation de fonctionnalité

Optimiser les données pour le processus de ML, notamment en enrichissant les données avec des sources supplémentaires, en mettant à l'échelle les valeurs ou en extrayant plusieurs ensembles d'informations à partir d'un seul champ de données. Cela permet au modèle de ML de tirer parti des données. Par exemple, si vous décomposez la date « 2021-05-27 00:15:37 » en « 2021 », « mai », « jeudi » et « 15 », vous pouvez aider l'algorithme d'apprentissage à apprendre des modèles nuancés associés à différents composants de données.

FGAC

Découvrez le contrôle d'accès détaillé.

contrôle d'accès détaillé (FGAC)

Utilisation de plusieurs conditions pour autoriser ou refuser une demande d'accès.

migration instantanée (flash-cut)

Méthode de migration de base de données qui utilise la réplication continue des données via la capture des données de modification afin de migrer les données dans les plus brefs délais, au lieu d'utiliser une approche progressive. L'objectif est de réduire au maximum les temps d'arrêt.

G

blocage géographique

Voir les restrictions géographiques.

restrictions géographiques (blocage géographique)

Sur Amazon CloudFront, option permettant d'empêcher les utilisateurs de certains pays d'accéder aux distributions de contenu. Vous pouvez utiliser une liste d'autorisation ou une liste de blocage pour spécifier les pays approuvés et interdits. Pour plus d'informations, consultez la section Restreindre la distribution géographique de votre contenu dans la CloudFront documentation.

Flux de travail Gitflow

Approche dans laquelle les environnements inférieurs et supérieurs utilisent différentes branches dans un référentiel de code source. Le flux de travail Gitflow est considéré comme existant, et le flux de travail basé sur les troncs est l'approche moderne préférée.

stratégie inédite

L'absence d'infrastructures existantes dans un nouvel environnement. Lorsque vous adoptez une stratégie inédite pour une architecture système, vous pouvez sélectionner toutes les nouvelles technologies sans restriction de compatibilité avec l'infrastructure existante, également appelée brownfield. Si vous étendez l'infrastructure existante, vous pouvez combiner des politiques brownfield (existantes) et greenfield (inédites).

barrière de protection

Règle de haut niveau qui permet de régir les ressources, les politiques et la conformité au sein des unités d'organisation (UO). Les barrières de protection préventives appliquent des politiques pour garantir l'alignement sur les normes de conformité. Elles sont mises en œuvre à l'aide de politiques de contrôle des services et de limites des autorisations IAM. Les barrières de protection de détection détectent les violations des politiques et les problèmes de conformité, et génèrent des alertes pour y remédier. Ils sont implémentés à l'aide d'Amazon AWS Config AWS Security Hub GuardDuty AWS Trusted Advisor, d'Amazon Inspector et de AWS Lambda contrôles personnalisés.

H

HA

Découvrez la haute disponibilité.

migration de base de données hétérogène

Migration de votre base de données source vers une base de données cible qui utilise un moteur de base de données différent (par exemple, Oracle vers Amazon Aurora). La migration hétérogène fait généralement partie d'un effort de réarchitecture, et la conversion du schéma peut s'avérer une tâche complexe. AWS propose AWS SCT qui facilite les conversions de schémas.

haute disponibilité (HA)

Capacité d'une charge de travail à fonctionner en continu, sans intervention, en cas de difficultés ou de catastrophes. Les systèmes HA sont conçus pour basculer automatiquement, fournir

constamment des performances de haute qualité et gérer différentes charges et défaillances avec un impact minimal sur les performances.

modernisation de l'historien

Approche utilisée pour moderniser et mettre à niveau les systèmes de technologie opérationnelle (OT) afin de mieux répondre aux besoins de l'industrie manufacturière. Un historien est un type de base de données utilisé pour collecter et stocker des données provenant de diverses sources dans une usine.

migration de base de données homogène

Migration de votre base de données source vers une base de données cible qui partage le même moteur de base de données (par exemple, Microsoft SQL Server vers Amazon RDS for SQL Server). La migration homogène s'inscrit généralement dans le cadre d'un effort de réhébergement ou de replateforme. Vous pouvez utiliser les utilitaires de base de données natifs pour migrer le schéma.

données chaudes

Données fréquemment consultées, telles que les données en temps réel ou les données translationnelles récentes. Ces données nécessitent généralement un niveau ou une classe de stockage à hautes performances pour fournir des réponses rapides aux requêtes.

correctif

Solution d'urgence à un problème critique dans un environnement de production. En raison de son urgence, un correctif est généralement créé en dehors du flux de travail de DevOps publication habituel.

période de soins intensifs

Immédiatement après le basculement, période pendant laquelle une équipe de migration gère et surveille les applications migrées dans le cloud afin de résoudre les problèmes éventuels. En règle générale, cette période dure de 1 à 4 jours. À la fin de la période de soins intensifs, l'équipe de migration transfère généralement la responsabilité des applications à l'équipe des opérations cloud.

I

IaC

Considérez l'infrastructure comme un code.

politique basée sur l'identité

Politique attachée à un ou plusieurs principaux IAM qui définit leurs autorisations au sein de l'AWS Cloud environnement.

application inactive

Application dont l'utilisation moyenne du processeur et de la mémoire se situe entre 5 et 20 % sur une période de 90 jours. Dans un projet de migration, il est courant de retirer ces applications ou de les retenir sur site.

IIoT

Voir Internet industriel des objets.

infrastructure immuable

Modèle qui déploie une nouvelle infrastructure pour les charges de travail de production au lieu de mettre à jour, d'appliquer des correctifs ou de modifier l'infrastructure existante. Les infrastructures immuables sont intrinsèquement plus cohérentes, fiables et prévisibles que les infrastructures mutables. Pour plus d'informations, consultez les meilleures pratiques de déploiement à l'aide d'une infrastructure immuable dans le AWS Well-Architected Framework.

VPC entrant (d'entrée)

Dans une architecture AWS multi-comptes, un VPC qui accepte, inspecte et achemine les connexions réseau depuis l'extérieur d'une application. L'architecture de référence de sécuritéAWS recommande de configurer votre compte réseau avec des VPC entrants, sortants et d'inspection afin de protéger l'interface bidirectionnelle entre votre application et Internet en général.

migration incrémentielle

Stratégie de basculement dans le cadre de laquelle vous migrez votre application par petites parties au lieu d'effectuer un basculement complet unique. Par exemple, il se peut que vous ne transfériez que quelques microservices ou utilisateurs vers le nouveau système dans un

premier temps. Après avoir vérifié que tout fonctionne correctement, vous pouvez transférer progressivement des microservices ou des utilisateurs supplémentaires jusqu'à ce que vous puissiez mettre hors service votre système hérité. Cette stratégie réduit les risques associés aux migrations de grande ampleur.

Industry 4.0

Terme introduit par Klaus Schwab en 2016 pour désigner la modernisation des processus de fabrication grâce aux avancées en matière de connectivité, de données en temps réel, d'automatisation, d'analyse et d'IA/ML.

infrastructure

Ensemble des ressources et des actifs contenus dans l'environnement d'une application.

infrastructure en tant que code (IaC)

Processus de mise en service et de gestion de l'infrastructure d'une application via un ensemble de fichiers de configuration. IaC est conçue pour vous aider à centraliser la gestion de l'infrastructure, à normaliser les ressources et à mettre à l'échelle rapidement afin que les nouveaux environnements soient reproductibles, fiables et cohérents.

internet industriel des objets (IIoT)

L'utilisation de capteurs et d'appareils connectés à Internet dans les secteurs industriels tels que la fabrication, l'énergie, l'automobile, les soins de santé, les sciences de la vie et l'agriculture. Pour plus d'informations, veuillez consulter Building an industrial Internet of Things (IIoT) digital transformation strategy.

VPC d'inspection

Dans une architecture AWS multi-comptes, un VPC centralisé qui gère les inspections du trafic réseau entre les VPC (identiques ou Régions AWS différents), Internet et les réseaux sur site. L'architecture de référence de sécuritéAWS recommande de configurer votre compte réseau avec des VPC entrants, sortants et d'inspection afin de protéger l'interface bidirectionnelle entre votre application et Internet en général.

Internet des objets (IoT)

Réseau d'objets physiques connectés dotés de capteurs ou de processeurs intégrés qui communiquent avec d'autres appareils et systèmes via Internet ou via un réseau de communication local. Pour plus d'informations, veuillez consulter la section Qu'est-ce que l'IoT ?.

interprétabilité

Caractéristique d'un modèle de machine learning qui décrit dans quelle mesure un être humain peut comprendre comment les prédictions du modèle dépendent de ses entrées. Pour plus d'informations, veuillez consulter Machine learning model interpretability with AWS.

IoT

Voir Internet des objets.

Bibliothèque d'informations informatiques (ITIL)

Ensemble de bonnes pratiques pour proposer des services informatiques et les aligner sur les exigences métier. L'ITIL constitue la base de l'ITSM.

gestion des services informatiques (ITSM)

Activités associées à la conception, à la mise en œuvre, à la gestion et à la prise en charge de services informatiques d'une organisation. Pour plus d'informations sur l'intégration des opérations cloud aux outils ITSM, veuillez consulter le guide d'intégration des opérations.

ITIL

Consultez la bibliothèque d'informations informatiques.

ITSM

Consultez la section Gestion des services informatiques.

L

contrôle d'accès basé sur des étiquettes (LBAC)

Une implémentation du contrôle d'accès obligatoire (MAC) dans laquelle une valeur d'étiquette de sécurité est explicitement attribuée aux utilisateurs et aux données elles-mêmes. L'intersection entre l'étiquette de sécurité utilisateur et l'étiquette de sécurité des données détermine les lignes et les colonnes visibles par l'utilisateur.

zone de destination

Une zone d'atterrissage est un AWS environnement multi-comptes bien conçu, évolutif et sécurisé. Il s'agit d'un point de départ à partir duquel vos entreprises peuvent rapidement lancer et déployer des charges de travail et des applications en toute confiance dans leur environnement

de sécurité et d'infrastructure. Pour plus d'informations sur les zones de destination, veuillez consulter Setting up a secure and scalable multi-account AWS environment.

migration de grande envergure

Migration de 300 serveurs ou plus.

LBAC

Voir contrôle d'accès basé sur des étiquettes.

principe de moindre privilège

Bonne pratique de sécurité qui consiste à accorder les autorisations minimales nécessaires à l'exécution d'une tâche. Pour plus d'informations, veuillez consulter la rubrique Accorder les autorisations de moindre privilège dans la documentation IAM.

lift and shift

Voir 7 Rs.

système de poids faible

Système qui stocke d'abord l'octet le moins significatif. Voir aussi endianité.

environnements inférieurs

Voir environnement.

M

machine learning (ML)

Type d'intelligence artificielle qui utilise des algorithmes et des techniques pour la reconnaissance et l'apprentissage de modèles. Le ML analyse et apprend à partir de données enregistrées, telles que les données de l'Internet des objets (IoT), pour générer un modèle statistique basé sur des modèles. Pour plus d'informations, veuillez consulter Machine Learning.

branche principale

Voir la succursale.

malware

Logiciel conçu pour compromettre la sécurité ou la confidentialité de l'ordinateur. Les logiciels malveillants peuvent perturber les systèmes informatiques, divulguer des informations sensibles

ou obtenir un accès non autorisé. Parmi les malwares, on peut citer les virus, les vers, les rançongiciels, les chevaux de Troie, les logiciels espions et les enregistreurs de frappe.

services gérés

Services AWS qui AWS gère la couche d'infrastructure, le système d'exploitation et les plateformes, et vous accédez aux points de terminaison pour stocker et récupérer des données. Amazon Simple Storage Service (Amazon S3) et Amazon DynamoDB sont des exemples de services gérés. Ils sont également connus sous le nom de services abstraits.

système d'exécution de la fabrication (MES)

Un système logiciel pour le suivi, la surveillance, la documentation et le contrôle des processus de production qui convertissent les matières premières en produits finis dans l'atelier.

MAP

Voir Migration Acceleration Program.

mécanisme

Processus complet au cours duquel vous créez un outil, favorisez son adoption, puis inspectez les résultats afin de procéder aux ajustements nécessaires. Un mécanisme est un cycle qui se renforce et s'améliore lorsqu'il fonctionne. Pour plus d'informations, voir Création de mécanismes dans le cadre AWS Well-Architected.

compte membre

Tous, à l' Comptes AWS exception du compte de gestion, qui font partie d'une organisation dans AWS Organizations. Un compte ne peut être membre que d'une seule organisation à la fois.

MAILLES

Voir le système d'exécution de la fabrication.

Transport télémétrique en file d'attente de messages (MQTT)

Protocole de communication léger machine-to-machine (M2M), basé sur le modèle de publication/d'abonnement, pour les appareils IoT aux ressources limitées.

microservice

Petit service indépendant qui communique via des API bien définies et qui est généralement détenu par de petites équipes autonomes. Par exemple, un système d'assurance peut inclure des microservices qui mappent à des capacités métier, telles que les ventes ou le marketing, ou

à des sous-domaines, tels que les achats, les réclamations ou l'analytique. Les avantages des microservices incluent l'agilité, la flexibilité de la mise à l'échelle, la facilité de déploiement, la réutilisation du code et la résilience. Pour plus d'informations, consultez la section Intégration de microservices à l'aide de services AWS sans serveur.

architecture de microservices

Approche de création d'une application avec des composants indépendants qui exécutent chaque processus d'application en tant que microservice. Ces microservices communiquent via une interface bien définie à l'aide d'API légères. Chaque microservice de cette architecture peut être mis à jour, déployé et mis à l'échelle pour répondre à la demande de fonctions spécifiques d'une application. Pour plus d'informations, consultez la section Implémentation de microservices sur AWS.

Programme d'accélération des migrations (MAP)

Un AWS programme qui fournit un support de conseil, des formations et des services pour aider les entreprises à établir une base opérationnelle solide pour passer au cloud, et pour aider à compenser le coût initial des migrations. MAP inclut une méthodologie de migration pour exécuter les migrations héritées de manière méthodique, ainsi qu'un ensemble d'outils pour automatiser et accélérer les scénarios de migration courants.

migration à grande échelle

Processus consistant à transférer la majeure partie du portefeuille d'applications vers le cloud par vagues, un plus grand nombre d'applications étant déplacées plus rapidement à chaque vague. Cette phase utilise les bonnes pratiques et les enseignements tirés des phases précédentes pour implémenter une usine de migration d'équipes, d'outils et de processus en vue de rationaliser la migration des charges de travail grâce à l'automatisation et à la livraison agile. Il s'agit de la troisième phase de la stratégie de migration AWS.

usine de migration

Équipes interfonctionnelles qui rationalisent la migration des charges de travail grâce à des approches automatisées et agiles. Les équipes de Migration Factory comprennent généralement les opérations, les analystes commerciaux et les propriétaires, les ingénieurs de migration, les développeurs et les DevOps professionnels travaillant dans le cadre de sprints. Entre 20 et 50 % du portefeuille d'applications d'entreprise est constitué de modèles répétés qui peuvent être optimisés par une approche d'usine. Pour plus d'informations, veuillez consulter la rubrique discussion of migration factories et le guide Cloud Migration Factory dans cet ensemble de contenus.

métadonnées de migration

Informations relatives à l'application et au serveur nécessaires pour finaliser la migration. Chaque modèle de migration nécessite un ensemble de métadonnées de migration différent. Les exemples de métadonnées de migration incluent le sous-réseau cible, le groupe de sécurité et le AWS compte.

modèle de migration

Tâche de migration reproductible qui détaille la stratégie de migration, la destination de la migration et l'application ou le service de migration utilisé. Exemple : réorganisez la migration vers Amazon EC2 AWS avec le service de migration d'applications.

Évaluation du portefeuille de migration (MPA)

Outil en ligne qui fournit des informations pour valider l'analyse de rentabilisation en faveur de la migration vers le. AWS Cloud La MPA propose une évaluation détaillée du portefeuille (dimensionnement approprié des serveurs, tarification, comparaison du coût total de possession, analyse des coûts de migration), ainsi que la planification de la migration (analyse et collecte des données d'applications, regroupement des applications, priorisation des migrations et planification des vagues). L'outil MPA (connexion requise) est disponible gratuitement pour tous les AWS consultants et consultants APN Partner.

Évaluation de la préparation à la migration (MRA)

Processus qui consiste à obtenir des informations sur l'état de préparation d'une organisation au cloud, à identifier les forces et les faiblesses et à élaborer un plan d'action pour combler les lacunes identifiées, à l'aide du AWS CAF. Pour plus d'informations, veuillez consulter le guide de préparation à la migration. La MRA est la première phase de la stratégie de migration AWS.

stratégie de migration

L'approche utilisée pour migrer une charge de travail vers le AWS Cloud. Pour plus d'informations, reportez-vous aux 7 R de ce glossaire et à Mobiliser votre organisation pour accélérer les migrations à grande échelle.

ML

Voir apprentissage automatique.

modernisation

Transformation d'une application obsolète (héritée ou monolithique) et de son infrastructure en un système agile, élastique et hautement disponible dans le cloud afin de réduire les coûts, de

gagner en efficacité et de tirer parti des innovations. Pour plus d'informations, consultez la section Stratégie de modernisation des applications dans le AWS Cloud.

évaluation de la préparation à la modernisation

Évaluation qui permet de déterminer si les applications d'une organisation sont prêtes à être modernisées, d'identifier les avantages, les risques et les dépendances, et qui détermine dans quelle mesure l'organisation peut prendre en charge l'état futur de ces applications. Le résultat de l'évaluation est un plan de l'architecture cible, une feuille de route détaillant les phases de développement et les étapes du processus de modernisation, ainsi qu'un plan d'action pour combler les lacunes identifiées. Pour plus d'informations, consultez la section Évaluation de l'état de préparation à la modernisation des applications dans le AWS Cloud.

applications monolithiques (monolithes)

Applications qui s'exécutent en tant que service unique avec des processus étroitement couplés. Les applications monolithiques ont plusieurs inconvénients. Si une fonctionnalité de l'application connaît un pic de demande, l'architecture entière doit être mise à l'échelle. L'ajout ou l'amélioration des fonctionnalités d'une application monolithique devient également plus complexe lorsque la base de code s'élargit. Pour résoudre ces problèmes, vous pouvez utiliser une architecture de microservices. Pour plus d'informations, veuillez consulter Decomposing monoliths into microservices.

MPA

Voir Évaluation du portefeuille de migration.

MQTT

Voir Message Queuing Telemetry Transport.

classification multi-classes

Processus qui permet de générer des prédictions pour plusieurs classes (prédiction d'un résultat parmi plus de deux). Par exemple, un modèle de ML peut demander « Ce produit est-il un livre, une voiture ou un téléphone ? » ou « Quelle catégorie de produits intéresse le plus ce client ? ».

infrastructure mutable

Modèle qui met à jour et modifie l'infrastructure existante pour les charges de travail de production. Pour améliorer la cohérence, la fiabilité et la prévisibilité, le AWS Well-Architected Framework recommande l'utilisation d'une infrastructure immuable comme meilleure pratique.

O

OAC

Voir Contrôle d'accès à l'origine.

OAI

Voir l'identité d'accès à l'origine.

OCM

Voir gestion du changement organisationnel.

migration hors ligne

Méthode de migration dans laquelle la charge de travail source est supprimée au cours du processus de migration. Cette méthode implique un temps d'arrêt prolongé et est généralement utilisée pour de petites charges de travail non critiques.

OI

Consultez la section Intégration des opérations.

OLA

Voir l'accord au niveau opérationnel.

migration en ligne

Méthode de migration dans laquelle la charge de travail source est copiée sur le système cible sans être mise hors ligne. Les applications connectées à la charge de travail peuvent continuer à fonctionner pendant la migration. Cette méthode implique un temps d'arrêt nul ou minimal et est généralement utilisée pour les charges de travail de production critiques.

OPC-UA

Voir Open Process Communications - Architecture unifiée.

Communications par processus ouvert - Architecture unifiée (OPC-UA)

Un protocole de communication machine-to-machine (M2M) pour l'automatisation industrielle. L'OPC-UA fournit une norme d'interopérabilité avec des schémas de cryptage, d'authentification et d'autorisation des données.

accord au niveau opérationnel (OLA)

Accord qui précise ce que les groupes informatiques fonctionnels s'engagent à fournir les uns aux autres, afin de prendre en charge un contrat de niveau de service (SLA).

examen de l'état de préparation opérationnelle (ORR)

Une liste de questions et de bonnes pratiques associées qui vous aident à comprendre, évaluer, prévenir ou réduire l'ampleur des incidents et des défaillances possibles. Pour plus d'informations, voir Operational Readiness Reviews (ORR) dans le AWS Well-Architected Framework.

technologie opérationnelle (OT)

Systèmes matériels et logiciels qui fonctionnent avec l'environnement physique pour contrôler les opérations, les équipements et les infrastructures industriels. Dans le secteur manufacturier, l'intégration des systèmes OT et des technologies de l'information (IT) est au cœur des transformations de l'industrie 4.0.

intégration des opérations (OI)

Processus de modernisation des opérations dans le cloud, qui implique la planification de la préparation, l'automatisation et l'intégration. Pour en savoir plus, veuillez consulter le guide d'intégration des opérations.

journal de suivi d'organisation

Un parcours créé par AWS CloudTrail qui enregistre tous les événements pour tous les membres Comptes AWS d'une organisation dans AWS Organizations. Ce journal de suivi est créé dans chaque Compte AWS qui fait partie de l'organisation et suit l'activité de chaque compte. Pour plus d'informations, consultez la section Création d'un suivi pour une organisation dans la CloudTrail documentation.

gestion du changement organisationnel (OCM)

Cadre pour gérer les transformations métier majeures et perturbatrices du point de vue des personnes, de la culture et du leadership. L'OCM aide les organisations à se préparer et à effectuer la transition vers de nouveaux systèmes et de nouvelles politiques en accélérant l'adoption des changements, en abordant les problèmes de transition et en favorisant des changements culturels et organisationnels. Dans la stratégie de AWS migration, ce cadre est appelé accélération du personnel, en raison de la rapidité du changement requise dans les projets d'adoption du cloud. Pour plus d'informations, veuillez consulter le guide OCM.

contrôle d'accès d'origine (OAC)

Dans CloudFront, une option améliorée pour restreindre l'accès afin de sécuriser votre contenu Amazon Simple Storage Service (Amazon S3). L'OAC prend en charge tous les compartiments S3 dans leur ensemble Régions AWS, le chiffrement côté serveur avec AWS KMS (SSE-KMS) et les requêtes dynamiques PUT adressées au compartiment S3. DELETE

identité d'accès d'origine (OAI)

Dans CloudFront, une option permettant de restreindre l'accès afin de sécuriser votre contenu Amazon S3. Lorsque vous utilisez OAI, il CloudFront crée un principal auprès duquel Amazon S3 peut s'authentifier. Les principaux authentifiés ne peuvent accéder au contenu d'un compartiment S3 que par le biais d'une distribution spécifique CloudFront . Voir également OAC, qui fournit un contrôle d'accès plus précis et amélioré.

OU

Voir l'examen de l'état de préparation opérationnelle.

DE

Voir technologie opérationnelle.

VPC sortant (de sortie)

Dans une architecture AWS multi-comptes, un VPC qui gère les connexions réseau initiées depuis une application. L'architecture de référence de sécuritéAWS recommande de configurer votre compte réseau avec des VPC entrants, sortants et d'inspection afin de protéger l'interface bidirectionnelle entre votre application et Internet en général.

P

limite des autorisations

Politique de gestion IAM attachée aux principaux IAM pour définir les autorisations maximales que peut avoir l'utilisateur ou le rôle. Pour plus d'informations, veuillez consulter la rubrique Limites des autorisations dans la documentation IAM.

informations personnelles identifiables (PII)

Informations qui, lorsqu'elles sont consultées directement ou associées à d'autres données connexes, peuvent être utilisées pour déduire raisonnablement l'identité d'une personne. Les

exemples d'informations personnelles incluent les noms, les adresses et les informations de contact.

PII

Voir les informations personnelles identifiables.

manuel stratégique

Ensemble d'étapes prédéfinies qui capturent le travail associé aux migrations, comme la fourniture de fonctions d'opérations de base dans le cloud. Un manuel stratégique peut revêtir la forme de scripts, de runbooks automatisés ou d'un résumé des processus ou des étapes nécessaires au fonctionnement de votre environnement modernisé.

PLC

Voir contrôleur logique programmable.

PLM

Consultez la section Gestion du cycle de vie des produits.

politique

Objet capable de définir les autorisations (voir la politique basée sur l'identité), de spécifier les conditions d'accès (voir la politique basée sur les ressources) ou de définir les autorisations maximales pour tous les comptes d'une organisation dans AWS Organizations (voir la politique de contrôle des services).

persistance polyglotte

Choix indépendant de la technologie de stockage de données d'un microservice en fonction des modèles d'accès aux données et d'autres exigences. Si vos microservices utilisent la même technologie de stockage de données, ils peuvent rencontrer des difficultés d'implémentation ou présenter des performances médiocres. Les microservices sont plus faciles à mettre en œuvre, atteignent de meilleures performances, ainsi qu'une meilleure capacité de mise à l'échelle s'ils utilisent l'entrepôt de données le mieux adapté à leurs besoins. Pour plus d'informations, veuillez consulter Enabling data persistence in microservices.

évaluation du portefeuille

Processus de découverte, d'analyse et de priorisation du portefeuille d'applications afin de planifier la migration. Pour plus d'informations, veuillez consulter Evaluating migration readiness.

predicate

Une condition de requête qui renvoie `true` ou`false`, généralement située dans une `WHERE` clause.

prédicat pushdown

Technique d'optimisation des requêtes de base de données qui filtre les données de la requête avant le transfert. Cela réduit la quantité de données qui doivent être extraites et traitées à partir de la base de données relationnelle et améliore les performances des requêtes.

contrôle préventif

Contrôle de sécurité conçu pour empêcher qu'un événement ne se produise. Ces contrôles constituent une première ligne de défense pour empêcher tout accès non autorisé ou toute modification indésirable de votre réseau. Pour plus d'informations, veuillez consulter Preventative controls dans Implementing security controls on AWS.

principal

Entité AWS capable d'effectuer des actions et d'accéder aux ressources. Cette entité est généralement un utilisateur root pour un Compte AWS rôle IAM ou un utilisateur. Pour plus d'informations, veuillez consulter la rubrique Principal dans Termes et concepts relatifs aux rôles, dans la documentation IAM.

Confidentialité dès la conception

Une approche de l'ingénierie des systèmes qui prend en compte la confidentialité tout au long du processus d'ingénierie.

zones hébergées privées

Conteneur qui contient des informations concernant la façon dont vous souhaitez qu'Amazon Route 53 réponde aux requêtes DNS pour un domaine et ses sous-domaines dans un ou plusieurs VPC. Pour plus d'informations, veuillez consulter Working with private hosted zones dans la documentation Route 53.

contrôle proactif

Contrôle de sécurité conçu pour empêcher le déploiement de ressources non conformes. Ces contrôles analysent les ressources avant qu'elles ne soient provisionnées. Si la ressource n'est pas conforme au contrôle, elle n'est pas provisionnée. Pour plus d'informations, consultez le guide de référence sur les contrôles dans la AWS Control Tower documentation et consultez la section Contrôles proactifs dans Implémentation des contrôles de sécurité sur AWS.

gestion du cycle de vie des produits (PLM)

Gestion des données et des processus d'un produit tout au long de son cycle de vie, depuis la conception, le développement et le lancement, en passant par la croissance et la maturité, jusqu'au déclin et au retrait.

environnement de production

Voir environnement.

contrôleur logique programmable (PLC)

Dans le secteur manufacturier, un ordinateur hautement fiable et adaptable qui surveille les machines et automatise les processus de fabrication.

pseudonymisation

Processus de remplacement des identifiants personnels dans un ensemble de données par des valeurs fictives. La pseudonymisation peut contribuer à protéger la vie privée. Les données pseudonymisées sont toujours considérées comme des données personnelles.

publier/souscrire (pub/sub)

Modèle qui permet des communications asynchrones entre les microservices afin d'améliorer l'évolutivité et la réactivité. Par exemple, dans un MES basé sur des microservices, un microservice peut publier des messages d'événements sur un canal auquel d'autres microservices peuvent s'abonner. Le système peut ajouter de nouveaux microservices sans modifier le service de publication.

Q

plan de requête

Série d'étapes, telles que des instructions, utilisées pour accéder aux données d'un système de base de données relationnelle SQL.

régression du plan de requêtes

Le cas où un optimiseur de service de base de données choisit un plan moins optimal qu'avant une modification donnée de l'environnement de base de données. Cela peut être dû à des changements en termes de statistiques, de contraintes, de paramètres d'environnement, de liaisons de paramètres de requêtes et de mises à jour du moteur de base de données.

R

Matrice RACI

Voir responsable, responsable, consulté, informé (RACI).

rançongiciel

Logiciel malveillant conçu pour bloquer l'accès à un système informatique ou à des données jusqu'à ce qu'un paiement soit effectué.

Matrice RASCI

Voir responsable, responsable, consulté, informé (RACI).

RCAC

Voir contrôle d'accès aux lignes et aux colonnes.

réplica en lecture

Copie d'une base de données utilisée en lecture seule. Vous pouvez acheminer les requêtes vers le réplica de lecture pour réduire la charge sur votre base de données principale.

réarchitecte

Voir 7 Rs.

objectif de point de récupération (RPO)

Durée maximale acceptable depuis le dernier point de récupération des données. Cela permet de déterminer ce qui est considéré comme une perte de données acceptable entre le dernier point de restauration et l'interruption du service.

objectif de temps de récupération (RTO)

Le délai maximum acceptable entre l'interruption du service et le rétablissement du service.

refactoriser

Voir 7 Rs.

Région

Un ensemble de AWS ressources dans une zone géographique. Chacun Région AWS est isolé et indépendant des autres pour garantir tolérance aux pannes, stabilité et résilience. Pour plus d'informations, voir Spécifier ce que Régions AWS votre compte peut utiliser.

régression

Technique de ML qui prédit une valeur numérique. Par exemple, pour résoudre le problème « Quel sera le prix de vente de cette maison ? », un modèle de ML pourrait utiliser un modèle de régression linéaire pour prédire le prix de vente d'une maison sur la base de faits connus à son sujet (par exemple, la superficie en mètres carrés).

réhéberger

Voir 7 Rs.

version

Dans un processus de déploiement, action visant à promouvoir les modifications apportées à un environnement de production.

déplacer

Voir 7 Rs.

replateforme

Voir 7 Rs.

rachat

Voir 7 Rs.

résilience

La capacité d'une application à résister aux perturbations ou à s'en remettre. La haute disponibilité et la reprise après sinistre sont des considérations courantes lors de la planification de la résilience dans le AWS Cloud. Pour plus d'informations, consultez AWS Cloud Résilience.

politique basée sur les ressources

Politique attachée à une ressource, comme un compartiment Amazon S3, un point de terminaison ou une clé de chiffrement. Ce type de politique précise les principaux auxquels l'accès est autorisé, les actions prises en charge et toutes les autres conditions qui doivent être remplies.

matrice responsable, redevable, consulté et informé (RACI)

Une matrice qui définit les rôles et les responsabilités de toutes les parties impliquées dans les activités de migration et les opérations cloud. Le nom de la matrice est dérivé des types de responsabilité définis dans la matrice : responsable (R), responsable (A), consulté (C) et informé (I). Le type de support (S) est facultatif. Si vous incluez le support, la matrice est appelée matrice RASCI, et si vous l'excluez, elle est appelée matrice RACI.

contrôle réactif

Contrôle de sécurité conçu pour permettre de remédier aux événements indésirables ou aux écarts par rapport à votre référence de sécurité. Pour plus d'informations, veuillez consulter la rubrique Responsive controls dans Implementing security controls on AWS.

retain

Voir 7 Rs.

se retirer

Voir 7 Rs.

rotation

Processus de mise à jour périodique d'un secret pour empêcher un attaquant d'accéder aux informations d'identification.

contrôle d'accès aux lignes et aux colonnes (RCAC)

Utilisation d'expressions SQL simples et flexibles dotées de règles d'accès définies. Le RCAC comprend des autorisations de ligne et des masques de colonnes.

RPO

Voir l'objectif du point de récupération.

RTO

Voir l'objectif en matière de temps de rétablissement.

runbook

Ensemble de procédures manuelles ou automatisées nécessaires à l'exécution d'une tâche spécifique. Elles visent généralement à rationaliser les opérations ou les procédures répétitives présentant des taux d'erreur élevés.

S

SAML 2.0

Un standard ouvert utilisé par de nombreux fournisseurs d'identité (IdPs). Cette fonctionnalité permet l'authentification unique fédérée (SSO), afin que les utilisateurs puissent se connecter AWS Management Console ou appeler les opérations d' AWS API sans que vous ayez à créer

un utilisateur dans IAM pour tous les membres de votre organisation. Pour plus d'informations sur la fédération SAML 2.0, veuillez consulter À propos de la fédération SAML 2.0 dans la documentation IAM.

SCADA

Voir Contrôle de supervision et acquisition de données.

SCP

Voir la politique de contrôle des services.

secret

Dans AWS Secrets Manager des informations confidentielles ou restreintes, telles qu'un mot de passe ou des informations d'identification utilisateur, que vous stockez sous forme cryptée. Il comprend la valeur secrète et ses métadonnées. La valeur secrète peut être binaire, une chaîne unique ou plusieurs chaînes. Pour plus d'informations, voir Que contient le secret d'un Secrets Manager ? dans la documentation de Secrets Manager.

contrôle de sécurité

Barrière de protection technique ou administrative qui empêche, détecte ou réduit la capacité d'un assaillant d'exploiter une vulnérabilité de sécurité. Il existe quatre principaux types de contrôles de sécurité : préventifs, détectifs, réactifs et proactifs.

renforcement de la sécurité

Processus qui consiste à réduire la surface d'attaque pour la rendre plus résistante aux attaques. Cela peut inclure des actions telles que la suppression de ressources qui ne sont plus requises, la mise en œuvre des bonnes pratiques de sécurité consistant à accorder le moindre privilège ou la désactivation de fonctionnalités inutiles dans les fichiers de configuration.

système de gestion des informations et des événements de sécurité (SIEM)

Outils et services qui associent les systèmes de gestion des informations de sécurité (SIM) et de gestion des événements de sécurité (SEM). Un système SIEM collecte, surveille et analyse les données provenant de serveurs, de réseaux, d'appareils et d'autres sources afin de détecter les menaces et les failles de sécurité, mais aussi de générer des alertes.

automatisation des réponses de sécurité

Action prédéfinie et programmée conçue pour répondre automatiquement à un événement de sécurité ou y remédier. Ces automatisations servent de contrôles de sécurité détectifs ou réactifs

qui vous aident à mettre en œuvre les meilleures pratiques AWS de sécurité. Parmi les actions de réponse automatique, citons la modification d'un groupe de sécurité VPC, l'application de correctifs à une instance Amazon EC2 ou la rotation des informations d'identification.

chiffrement côté serveur

Chiffrement des données à destination, par celui Service AWS qui les reçoit.

Politique de contrôle des services (SCP)

Politique qui propose un contrôle centralisé des autorisations pour tous les comptes d'une organisation dans AWS Organizations. Les SCP définissent des barrières de protection ou des limites aux actions qu'un administrateur peut déléguer à des utilisateurs ou à des rôles. Vous pouvez utiliser les SCP comme listes d'autorisation ou de refus, pour indiquer les services ou les actions autorisés ou interdits. Pour plus d'informations, consultez la section Politiques de contrôle des services dans la AWS Organizations documentation.

point de terminaison du service

URL du point d'entrée pour un Service AWS. Pour vous connecter par programmation au service cible, vous pouvez utiliser un point de terminaison. Pour plus d'informations, veuillez consulter la rubrique Service AWS endpoints dans Références générales AWS.

contrat de niveau de service (SLA)

Accord qui précise ce qu'une équipe informatique promet de fournir à ses clients, comme le temps de disponibilité et les performances des services.

indicateur de niveau de service (SLI)

Mesure d'un aspect des performances d'un service, tel que son taux d'erreur, sa disponibilité ou son débit.

objectif de niveau de service (SLO)

Mesure cible qui représente l'état d'un service, tel que mesuré par un indicateur de niveau de service.

modèle de responsabilité partagée

Un modèle décrivant la responsabilité que vous partagez en matière AWS de sécurité et de conformité dans le cloud. AWS est responsable de la sécurité du cloud, alors que vous êtes responsable de la sécurité dans le cloud. Pour de plus amples informations, veuillez consulter Modèle de responsabilité partagée.

SIEM

Consultez les informations de sécurité et le système de gestion des événements.

point de défaillance unique (SPOF)

Défaillance d'un seul composant critique d'une application susceptible de perturber le système.

SLA

Voir le contrat de niveau de service.

SLI

Voir l'indicateur de niveau de service.

SLO

Voir l'objectif de niveau de service.

split-and-seed modèle

Modèle permettant de mettre à l'échelle et d'accélérer les projets de modernisation. Au fur et à mesure que les nouvelles fonctionnalités et les nouvelles versions de produits sont définies, l'équipe principale se divise pour créer des équipes de produit. Cela permet de mettre à l'échelle les capacités et les services de votre organisation, d'améliorer la productivité des développeurs et de favoriser une innovation rapide. Pour plus d'informations, consultez la section Approche progressive de la modernisation des applications dans le. AWS Cloud

SPOF

Voir point de défaillance unique.

schéma en étoile

Structure organisationnelle de base de données qui utilise une grande table de faits pour stocker les données transactionnelles ou mesurées et utilise une ou plusieurs tables dimensionnelles plus petites pour stocker les attributs des données. Cette structure est conçue pour être utilisée dans un entrepôt de données ou à des fins de business intelligence.

modèle de figuier étrangleur

Approche de modernisation des systèmes monolithiques en réécrivant et en remplaçant progressivement les fonctionnalités du système jusqu'à ce que le système hérité puisse être mis hors service. Ce modèle utilise l'analogie d'un figuier de vigne qui se développe dans un arbre existant et qui finit par supplanter son hôte. Le schéma a été présenté par Martin Fowler comme

un moyen de gérer les risques lors de la réécriture de systèmes monolithiques. Pour obtenir un exemple d'application de ce modèle, veuillez consulter Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

sous-réseau

Plage d'adresses IP dans votre VPC. Un sous-réseau doit se trouver dans une seule zone de disponibilité.

contrôle de supervision et acquisition de données (SCADA)

Dans le secteur manufacturier, un système qui utilise du matériel et des logiciels pour surveiller les actifs physiques et les opérations de production.

chiffrement symétrique

Algorithme de chiffrement qui utilise la même clé pour chiffrer et déchiffrer les données.

tests synthétiques

Tester un système de manière à simuler les interactions des utilisateurs afin de détecter les problèmes potentiels ou de surveiller les performances. Vous pouvez utiliser Amazon CloudWatch Synthetics pour créer ces tests.

T

balises

Des paires clé-valeur qui agissent comme des métadonnées pour organiser vos AWS ressources. Les balises peuvent vous aider à gérer, identifier, organiser, rechercher et filtrer des ressources. Pour plus d'informations, veuillez consulter la rubrique Balisage de vos AWS ressources.

variable cible

La valeur que vous essayez de prédire dans le cadre du ML supervisé. Elle est également qualifiée de variable de résultat. Par exemple, dans un environnement de fabrication, la variable cible peut être un défaut du produit.

liste de tâches

Outil utilisé pour suivre les progrès dans un runbook. Liste de tâches qui contient une vue d'ensemble du runbook et une liste des tâches générales à effectuer. Pour chaque tâche générale, elle inclut le temps estimé nécessaire, le propriétaire et l'avancement.

environnement de test

Voir environnement.

entraînement

Pour fournir des données à partir desquelles votre modèle de ML peut apprendre. Les données d'entraînement doivent contenir la bonne réponse. L'algorithme d'apprentissage identifie des modèles dans les données d'entraînement, qui mettent en correspondance les attributs des données d'entrée avec la cible (la réponse que vous souhaitez prédire). Il fournit un modèle de ML qui capture ces modèles. Vous pouvez alors utiliser le modèle de ML pour obtenir des prédictions sur de nouvelles données pour lesquelles vous ne connaissez pas la cible.

passerelle de transit

Hub de transit de réseau que vous pouvez utiliser pour relier vos VPC et vos réseaux sur site. Pour plus d'informations, voir Qu'est-ce qu'une passerelle de transit dans la AWS Transit Gateway documentation.

flux de travail basé sur jonction

Approche selon laquelle les développeurs génèrent et testent des fonctionnalités localement dans une branche de fonctionnalités, puis fusionnent ces modifications dans la branche principale. La branche principale est ensuite intégrée aux environnements de développement, de préproduction et de production, de manière séquentielle.

accès sécurisé

Accorder des autorisations à un service que vous spécifiez pour effectuer des tâches au sein de votre organisation AWS Organizations et dans ses comptes en votre nom. Le service de confiance crée un rôle lié au service dans chaque compte, lorsque ce rôle est nécessaire, pour effectuer des tâches de gestion à votre place. Pour plus d'informations, consultez la section Utilisation AWS Organizations avec d'autres AWS services dans la AWS Organizations documentation.

réglage

Pour modifier certains aspects de votre processus d'entraînement afin d'améliorer la précision du modèle de ML. Par exemple, vous pouvez entraîner le modèle de ML en générant un ensemble d'étiquetage, en ajoutant des étiquettes, puis en répétant ces étapes plusieurs fois avec différents paramètres pour optimiser le modèle.

équipe de deux pizzas

Une petite DevOps équipe que vous pouvez nourrir avec deux pizzas. Une équipe de deux pizzas garantit les meilleures opportunités de collaboration possible dans le développement de logiciels.

U

incertitude

Un concept qui fait référence à des informations imprécises, incomplètes ou inconnues susceptibles de compromettre la fiabilité des modèles de ML prédictifs. Il existe deux types d'incertitude : l'incertitude épistémique est causée par des données limitées et incomplètes, alors que l'incertitude aléatoire est causée par le bruit et le caractère aléatoire inhérents aux données. Pour plus d'informations, veuillez consulter le guide Quantifying uncertainty in deep learning systems.

tâches indifférenciées

Également connu sous le nom de « levage de charges lourdes », ce travail est nécessaire pour créer et exploiter une application, mais qui n'apporte pas de valeur directe à l'utilisateur final ni d'avantage concurrentiel. Les exemples de tâches indifférenciées incluent l'approvisionnement, la maintenance et la planification des capacités.

environnements supérieurs

Voir environnement.

V

mise à vide

Opération de maintenance de base de données qui implique un nettoyage après des mises à jour incrémentielles afin de récupérer de l'espace de stockage et d'améliorer les performances.

contrôle de version

Processus et outils permettant de suivre les modifications, telles que les modifications apportées au code source dans un référentiel.

Appairage de VPC

Connexion entre deux VPC qui vous permet d'acheminer le trafic à l'aide d'adresses IP privées. Pour plus d'informations, veuillez consulter la rubrique Qu'est-ce que l'appairage de VPC ? dans la documentation Amazon VPC.

vulnérabilités

Défaut logiciel ou matériel qui compromet la sécurité du système.

W

cache actif

Cache tampon qui contient les données actuelles et pertinentes fréquemment consultées. L'instance de base de données peut lire à partir du cache tampon, ce qui est plus rapide que la lecture à partir de la mémoire principale ou du disque.

données chaudes

Données rarement consultées. Lorsque vous interrogez ce type de données, des requêtes modérément lentes sont généralement acceptables.

fonction de fenêtre

Fonction SQL qui effectue un calcul sur un groupe de lignes liées d'une manière ou d'une autre à l'enregistrement en cours. Les fonctions de fenêtre sont utiles pour traiter des tâches, telles que le calcul d'une moyenne mobile ou l'accès à la valeur des lignes en fonction de la position relative de la ligne en cours.

charge de travail

Ensemble de ressources et de code qui fournit une valeur métier, par exemple une application destinée au client ou un processus de backend.

flux de travail

Groupes fonctionnels d'un projet de migration chargés d'un ensemble de tâches spécifique. Chaque flux de travail est indépendant, mais prend en charge les autres flux de travail du projet. Par exemple, le flux de travail du portefeuille est chargé de prioriser les applications, de planifier les vagues et de collecter les métadonnées de migration. Le flux de travail du portefeuille fournit ces actifs au flux de travail de migration, qui migre ensuite les serveurs et les applications.

VER

Voir écrire une fois, lire plusieurs.

WQF

Consultez le cadre de qualification des charges de travail AWS.

écrire une fois, lire plusieurs (WORM)

Modèle de stockage qui écrit les données une seule fois et empêche leur suppression ou leur modification. Les utilisateurs autorisés peuvent lire les données autant de fois que nécessaire, mais ils ne peuvent pas les modifier. Cette infrastructure de stockage de données est considérée comme immuable.

Z

exploit Zero-Day

Une attaque, généralement un logiciel malveillant, qui tire parti d'une vulnérabilité de type « jour zéro ».

vulnérabilité de type « jour zéro »

Une faille ou une vulnérabilité non atténuée dans un système de production. Les acteurs malveillants peuvent utiliser ce type de vulnérabilité pour attaquer le système. Les développeurs prennent souvent conscience de la vulnérabilité à la suite de l'attaque.

application zombie

Application dont l'utilisation moyenne du processeur et de la mémoire est inférieure à 5 %. Dans un projet de migration, il est courant de retirer ces applications.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.