Introdução ao Terraform: orientação AWS CDK e orientação para especialistas AWS CloudFormation

AWS Orientação prescritiva

AWS Orientação prescritiva: Introdução ao Terraform: orientação AWS CDK e orientação para especialistas AWS CloudFormation

Table of Contents

Comece a usar o Terraform: orientação para especialistas em CDK e AWS CloudFormation

Steven Guggenheimer, da Amazon Web Services ()AWS

Março de 2024 (histórico do documento)

Se sua experiência com o provisionamento de recursos de nuvem estiver exclusivamente dentro do domínio de AWS, você pode ter experiência limitada com ferramentas de infraestrutura como código (IaC) além do normal. AWS Cloud Development Kit (AWS CDK)AWS CloudFormation Na verdade, ferramentas semelhantes, como o Hashicorp Terraform, podem ser completamente desconhecidas para você. No entanto, quanto mais você se aprofunda em sua jornada na nuvem, mais inevitável se torna que você encontre o Terraform. Definitivamente, será vantajoso para você se familiarizar com seus conceitos fundamentais.

Embora o Terraform AWS CDK, o e CloudFormation alcancem objetivos semelhantes e compartilhem muitos conceitos básicos, existem algumas diferenças. Talvez você não esteja preparado para essas diferenças se estiver abordando o Terraform pela primeira vez. Afinal, AWS CDK as CloudFormation pilhas são todas baseadas internamente Contas da AWS, então, dessa forma, elas têm uma relação direta com a maioria dos recursos que mantêm. O Terraform não está baseado no ambiente de nenhum único provedor de nuvem. Isso lhe dá a flexibilidade de oferecer suporte a vários provedores diferentes, mas deve manter recursos do que equivale a um local remoto.

Este guia ajuda a desmistificar os principais conceitos por trás do Terraform para ajudá-lo a lidar com qualquer desafio de IaC que surja em seu caminho. Ele se concentra em como o Terraform usa conceitos, como provedores, módulos e arquivos de estado, para provisionar recursos. Também compara os conceitos do Terraform com a forma como o AWS CDK e CloudFormation executam operações semelhantes.

> **Note**
>
> AWS CDK Isso ajuda os desenvolvedores a implantar CloudFormation pilhas usando linguagens de codificação programática. Depois de executado`cdk synth`, seu código é convertido em CloudFormation modelos. Desse ponto em diante, o processo é idêntico entre AWS CDK CloudFormation e. Por uma questão de brevidade, este guia geralmente se refere

ao processo de AWS IaC em CloudFormation termos, mas as comparações são igualmente adequadas para o. AWS CDK

CloudFormation e terminologia do Terraform

Ao comparar o Terraform com o AWS CDK and CloudFormation, reconciliar os conceitos básicos do IaC pode ser difícil devido à terminologia inconsistente usada para descrevê-los. A seguir estão esses termos e como este guia se referirá a eles:

- Pilha — Uma pilha é IaC que é implantada em um pipeline de CI/CD e rastreável como uma única unidade.Embora esse termo seja comum em CloudFormation, o Terraform realmente não usa esse termo. Uma pilha do Terraform é um módulo raiz implantado com todos os seus módulos secundários. No entanto, para evitar confusão com o termo módulo, este guia usa o termo pilha para descrever uma única implantação para ambas as ferramentas.
- Estado - O estado é composto por todos os recursos atualmente rastreados e suas configurações atuais em uma pilha de implantação de IaC. Conforme descrito na Entendendo os estados e back-ends do Terraform seção, o Terraform usa o termo estado mais do que CloudFormation. Isso ocorre porque manter o estado é mais visível no Terraform, mas rastrear e atualizar o estado é igualmente importante para CloudFormation.
- Arquivo IaC — Um arquivo IaC é um arquivo único que contém a linguagem de infraestrutura como código (IaC). CloudFormation se refere a um único CloudFormation arquivo como modelo. No entanto, modelos e arquivos de modelo no Terraform são algo completamente diferente. O equivalente a um CloudFormation modelo no Terraform é chamado de arquivo de configuração. Para minimizar a confusão neste guia, o termo arquivo ou arquivo IaC é usado para se referir aos CloudFormation modelos e aos arquivos de configuração do Terraform.

A tabela a seguir compara a terminologia usada para CloudFormation e o Terraform. A intenção dessa tabela é mostrar semelhanças. Essas não são one-to-one comparações. Cada conceito difere pelo menos um pouco entre o Terraform CloudFormation e o Terraform. Os conceitos são explicados detalhadamente nas seções relevantes deste guia.

CloudFormation termo	Termo Terraform	Seção deste guia
Interfaces CDK (como iBucket)	Fonte de dados	Entendendo as fontes de dados do Terraform

CloudFormation termo	Termo Terraform	Seção deste guia
Conjunto de alterações	Planejamento	Entendendo os módulos do Terraform
Funções de condição	Expressões condicionais	Entendendo as funções, expressões e meta-argumentos do Terraform
Atributo `DependsOn`	`depends_on` meta-argumento	Entendendo as funções, expressões e meta-argumentos do Terraform
Funções intrínsecas	Funções	Entendendo as funções, expressões e meta-argumentos do Terraform
Modules	Modules	Entendendo os módulos do Terraform
Outputs	Valores de saída	Compreendendo as variáveis, valores locais e saídas do Terraform
Parâmetros	Variáveis	Compreendendo as variáveis, valores locais e saídas do Terraform
Registro	Provedores	Entendendo os fornecedores do Terraform
Modelo	Arquivo de configuração	Todos

Compreendendo os recursos do Terraform

O principal motivo da existência de ambos AWS CloudFormation e do Terraform é a criação e manutenção de recursos em nuvem. Mas o que exatamente é um recurso de nuvem? E CloudFormation recursos e recursos do Terraform são a mesma coisa? A resposta é... sim e não. Para ilustrar isso, este guia fornece um exemplo do uso CloudFormation e, em seguida, do Terraform para criar um bucket do Amazon Simple Storage Service (Amazon S3).

O exemplo de CloudFormation código a seguir cria uma amostra de bucket do 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"
            }
        }
    }
}
```

O exemplo de código do Terraform a seguir cria um bucket Amazon S3 idêntico.

```
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"
  }
}
```

Para o Terraform, um provedor define o recurso e, em seguida, os desenvolvedores declaram e configuram esses recursos. Provedores são um conceito que este guia abordará na próxima seção. O exemplo do Terraform cria recursos completamente separados para várias configurações do bucket do S3. Criar recursos separados para configurações não é necessariamente típico de como o Terraform AWS Provider trata AWS os recursos. No entanto, esse exemplo mostra uma distinção importante. Embora um CloudFormation recurso seja estritamente definido pela especificação do CloudFormation recurso, o Terraform não tem esse requisito. No Terraform, o conceito de recurso é um pouco mais nebuloso.

Embora as ferramentas possam diferir em relação às barreiras exatas que definem o que é um único recurso, em geral, um recurso de nuvem é qualquer entidade específica que existe na nuvem e que pode ser criada, atualizada ou excluída. Portanto, independentemente de quantos recursos estejam envolvidos, os dois exemplos anteriores criam exatamente a mesma coisa com exatamente as mesmas configurações em um Conta da AWS.

Entendendo os fornecedores do Terraform

No Terraform, um provedor é um plug-in que interage com provedores de nuvem, ferramentas de terceiros e outras APIs. Para usar o Terraform com AWS, você usa o AWS Provider, que interage com AWS os recursos.

Se você nunca usou o AWS CloudFormation registro para incorporar extensões de terceiros em suas pilhas de implantação, talvez os fornecedores do Terraform levem algum tempo para se acostumar. Por ser CloudFormation nativo do AWS, o provedor de AWS recursos já está lá por padrão. O Terraform, por outro lado, não tem um único provedor padrão, então nada pode ser assumido sobre as origens de um determinado recurso. Isso significa que a primeira coisa que precisa ser declarada em um arquivo de configuração do Terraform é exatamente para onde os recursos estão indo e como eles vão chegar lá.

Essa distinção adiciona uma camada extra de complexidade ao Terraform que não existe com o. CloudFormation No entanto, essa complexidade proporciona maior flexibilidade. Você pode declarar vários provedores em um único módulo do Terraform e, em seguida, os recursos subjacentes criados podem interagir entre si como parte da mesma camada de implantação.

Isso pode ser útil de várias maneiras. Os provedores não precisam necessariamente ser provedores de nuvem separados. Os provedores podem representar qualquer fonte de recursos de nuvem. Por exemplo, pegue o Amazon Elastic Kubernetes Service (Amazon EKS). Ao provisionar um cluster Amazon EKS, talvez você queira usar gráficos do Helm para gerenciar extensões de terceiros e usar o próprio Kubernetes para gerenciar recursos de pod. Como AWS o Helm e o Kubernetes têm seus próprios provedores do Terraform, você pode provisionar e integrar esses recursos ao mesmo tempo e depois passar valores entre eles.

No exemplo de código a seguir para o Terraform, o AWS provedor cria um cluster Amazon EKS e, em seguida, as informações de configuração do Kubernetes resultantes são passadas para os provedores Helm e 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"
  }
}
```

Há uma desvantagem em relação aos fornecedores quando se trata das duas ferramentas de IaC. O Terraform depende totalmente de pacotes de fornecedores localizados externamente, que são o mecanismo que impulsiona suas implantações. CloudFormation suporta internamente todos os principais AWS processos. Com isso CloudFormation, você precisa se preocupar com fornecedores terceirizados somente se quiser incorporar uma extensão de terceiros. Há prós e contras em cada abordagem. Qual é a melhor opção para você está além do escopo deste guia, mas é importante lembrar a diferença ao avaliar as duas ferramentas.

Usando aliases do Terraform

No Terraform, você pode passar configurações personalizadas para cada provedor. E se você quiser usar várias configurações de provedor no mesmo módulo? Nesse caso, você teria que usar um alias. Os aliases ajudam você a selecionar qual provedor usar em um nível por recurso ou por módulo. Quando você tem mais de uma instância do mesmo provedor, você usa um alias para definir as instâncias não padrão. Por exemplo, sua instância de provedor padrão pode ser específica Região da AWS, mas você usa aliases para definir regiões alternativas.

O exemplo do Terraform a seguir mostra como usar um alias para provisionar buckets em diferentes. Regiões da AWS A região padrão para o provedor é`us-west-2`, mas você pode usar o alias east para provisionar recursos em`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"
}
```

Ao usar um `alias` junto com o `provider` meta-argumento, conforme mostrado no exemplo anterior, você pode especificar uma configuração de provedor diferente para recursos específicos. Provisionar recursos em vários Regiões da AWS em uma única pilha é só o começo. Os provedores de aliases são incrivelmente convenientes de várias maneiras.

Por exemplo, é muito comum provisionar vários clusters Kubernetes ao mesmo tempo. Os aliases podem ajudá-lo a configurar provedores adicionais de Helm e Kubernetes para que você possa usar essas ferramentas de terceiros de forma diferente para diferentes recursos do Amazon EKS. O exemplo de código do Terraform a seguir ilustra como usar aliases para realizar essa tarefa.

```
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"
  }
}
```

Entendendo os módulos do Terraform

No reino da infraestrutura como código (IaC), um módulo é um bloco de código independente que é isolado e empacotado para reutilização. O conceito de módulos é um aspecto inevitável do desenvolvimento do Terraform. Para obter mais informações, consulte Módulos na documentação do Terraform. AWS CloudFormation também suporta módulos. Para obter mais informações, consulte Introdução aos AWS CloudFormation módulos no blog de operações e migrações na AWS nuvem.

A principal diferença entre os módulos no Terraform CloudFormation é que os CloudFormation módulos são importados usando um tipo de recurso especial (`AWS::CloudFormation::ModuleVersion`). No Terraform, cada configuração tem pelo menos um módulo, conhecido como módulo raiz. Os recursos do Terraform que estão no arquivo main.tf ou arquivos em um arquivo de configuração do Terraform são considerados como estando no módulo raiz. O módulo raiz pode então chamar outros módulos para inclusão na pilha. O exemplo a seguir mostra um módulo raiz provisionando um cluster do Amazon Elastic Kubernetes Service (Amazon EKS) usando o módulo eks de código aberto.

```
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)
  }
```

```
}
```

Você deve ter notado que o arquivo de configuração acima não inclui o AWS provedor. Isso porque os módulos são independentes e podem incluir seus próprios fornecedores. Como os provedores do Terraform são globais, os provedores de um módulo filho podem ser usados no módulo raiz. No entanto, isso não é verdade para todos os valores do módulo. Outros valores internos em um módulo têm como escopo definido, por padrão, somente esse módulo e precisam ser declarados como saídas para serem acessíveis no módulo raiz. Você pode aproveitar os módulos de código aberto para simplificar a criação de recursos em sua pilha. Por exemplo, o módulo eks faz mais do que provisionar um cluster EKS — ele provisiona um ambiente Kubernetes totalmente funcional. Usá-lo pode evitar que você escreva dezenas de linhas extras de código, desde que a configuração do módulo eks atenda às suas necessidades.

Módulos de chamada

Dois dos principais comandos da CLI do Terraform que você executa durante a implantação do Terraform são terraform init e terraform apply. Uma das etapas padrão que o `terraform init` comando executa é localizar todos os módulos secundários e importá-los como dependências para o `.terraform/modules` diretório. Durante o desenvolvimento, sempre que você adiciona um novo módulo de origem externa, é necessário reinicializar antes de usar o comando. `apply` Quando você ouve uma referência a um módulo do Terraform, ela está se referindo aos pacotes nesse diretório. Estritamente falando, o módulo que você declara em seu código é o módulo de chamada, então, na prática, a palavra-chave module chama o módulo real, que é armazenado como uma dependência.

Dessa forma, o módulo de chamada serve como um representante mais sucinto do módulo completo a ser substituído quando a implantação ocorrer. Você pode aproveitar essa ideia criando seus próprios módulos em suas pilhas para impor separações lógicas de recursos usando os critérios que desejar. Lembre-se de que o objetivo final de fazer isso deve ser reduzir a complexidade da pilha. Como o compartilhamento de dados entre módulos exige que você produza esses dados de dentro do módulo, às vezes confiar demais nos módulos pode complicar demais as coisas.

O módulo raiz

Como cada configuração do Terraform tem pelo menos um módulo, pode ser útil examinar as propriedades do módulo com o qual você mais lidará: o módulo raiz. Sempre que você estiver trabalhando em um projeto do Terraform, o módulo raiz consiste em todos os `.tf` (ou`.tf.json`) arquivos em seu diretório de nível superior. Quando você executa `terraform apply` nesse

diretório de nível superior, o Terraform tenta executar todos os `.tf` arquivos que encontra lá. Todos os arquivos nos subdiretórios são ignorados, a menos que sejam chamados em um desses arquivos de configuração de nível superior.

Isso fornece alguma flexibilidade na forma como você estrutura seu código. Também é por isso que é mais preciso se referir à implantação do Terraform como um módulo do que como um arquivo, pois vários arquivos podem estar envolvidos em um único processo. Há uma estrutura de módulo padrão que o Terraform recomenda para as melhores práticas. No entanto, se você colocasse qualquer `.tf` arquivo em seu diretório de nível superior, ele será executado junto com o resto dos arquivos. Na verdade, todos os `.tf` arquivos de nível superior em um módulo são implantados quando você executa. `terraform apply` Então, qual arquivo o Terraform executa primeiro? A resposta a essa pergunta é muito importante.

Há uma série de etapas que o Terraform executa após a inicialização e antes da implantação da pilha. Primeiro, as configurações existentes são analisadas e, em seguida, um gráfico de dependências é criado. O gráfico de dependências determina quais recursos são necessários e em que ordem eles devem ser tratados. Recursos que contêm propriedades referenciadas em outros recursos, por exemplo, seriam tratados antes dos recursos dependentes. Da mesma forma, os recursos que declaram explicitamente a dependência usando o `depends_on` parâmetro seriam tratados após os recursos que eles especificam. Quando possível, o Terraform pode implementar paralelismo e lidar com recursos não dependentes simultaneamente. Você pode ver o gráfico de dependências antes da implantação usando o comando terraform graph.

Depois que o gráfico de dependências é criado, o Terraform determina o que precisa ser feito durante a implantação. Ele compara o gráfico de dependências com o arquivo de estado mais recente. O resultado desse processo é chamado de plano e é muito parecido com um conjunto de CloudFormation mudanças. Você pode ver o plano atual usando o comando terraform plan.

Como prática recomendada, é recomendável ficar o mais próximo possível da estrutura padrão do módulo. Nos casos em que seus arquivos de configuração estão ficando muito longos para serem gerenciados com eficiência e as separações lógicas podem simplificar o gerenciamento, você pode distribuir seu código em vários arquivos. Lembre-se de como o gráfico de dependências e o processo de planejamento funcionam para que suas pilhas funcionem da forma mais eficiente possível.

Entendendo os estados e back-ends do Terraform

Um dos conceitos mais importantes em infraestrutura como código (IaC) é o conceito de estado. Os serviços de IaC mantêm o estado, o que permite declarar um recurso em um arquivo IaC sem que ele seja recriado toda vez que você implanta. Os arquivos IaC documentam o estado de todos os recursos no final de uma implantação para que ele possa comparar esse estado com o estado de destino, conforme declarado na próxima implantação. Portanto, se o estado atual contiver um bucket do Amazon Simple Storage Service (Amazon S3) `my-s3-bucket` chamado e as alterações recebidas também contiverem o mesmo bucket, o novo processo aplicará todas as alterações encontradas ao bucket existente em vez de tentar criar um bucket totalmente novo.

A tabela a seguir fornece exemplos do processo geral do estado do IaC.

Estado atual	Estado alvo	Ação
Nenhum bucket S3 chamado `my-s3-bucket`	Bucket S3 chamado `my-s3-bucket`	Crie um bucket S3 chamado `my-s3-bucket`
`my-s3-bucket` sem o controle de versão do bucket configurado	`my-s3-bucket` sem o controle de versão do bucket configurado	Nenhuma ação
`my-s3-bucket` sem o controle de versão do bucket configurado	`my-s3-bucket` com o controle de versão do bucket configurado	Configure `my-s3-bucket` para ter controle de versão do bucket
`my-s3-bucket` com o controle de versão do bucket configurado	Nenhum bucket S3 chamado `my-s3-bucket`	Tentativa de excluir `my-s3-bucket`

Para entender as diferentes maneiras pelas quais AWS CloudFormation o Terraform rastreia o estado, é importante lembrar a primeira diferença básica entre as duas ferramentas: CloudFormation está hospedada dentro do e o Nuvem AWS Terraform é essencialmente remoto. Esse fato permite CloudFormation manter o estado internamente. Você pode acessar o CloudFormation console e ver o histórico de eventos de uma determinada pilha, mas o CloudFormation serviço em si impõe as regras de estado para você.

Os três modos que CloudFormation operam em um determinado recurso são `CreateUpdate`, `Delete` e. O modo atual é determinado com base no que aconteceu na última implantação e não pode ser influenciado de outra forma. Talvez você possa atualizar CloudFormation os recursos manualmente para influenciar qual modo é determinado, mas não pode passar um comando CloudFormation que diga “Para este recurso, opere `Create` no modo”.

Como o Terraform não está hospedado no Nuvem AWS, o processo de manutenção do estado deve ser mais configurável. Por esse motivo, o estado do Terraform é mantido em um arquivo de estado gerado automaticamente. Um desenvolvedor do Terraform precisa lidar com o estado de forma muito mais direta do que faria com CloudFormation ele. É importante lembrar que o estado de rastreamento é igualmente importante para ambas as ferramentas.

Por padrão, o arquivo de estado do Terraform é armazenado localmente no nível superior do diretório principal que executa sua pilha do Terraform. Se você executar o `terraform apply` comando em seu ambiente de desenvolvimento local, poderá ver o Terraform gerar o arquivo terraform.tfstate que ele usa para manter o estado em tempo real. Para o bem ou para o mal, isso lhe dá muito mais controle sobre o estado no Terraform do que você tem no CloudFormation. Embora você nunca deva atualizar o arquivo de estado diretamente, há vários comandos da CLI do Terraform que você pode executar para atualizar o estado entre as implantações. Por exemplo, a importação do terraform permite que você adicione recursos criados fora do Terraform à sua pilha de implantação. Por outro lado, você pode remover um recurso do estado executando terraform state rm.

O fato de o Terraform precisar armazenar seu estado em algum lugar leva a outro conceito que não se aplica a CloudFormation: o back-end. Um back-end do Terraform é o local em que uma pilha do Terraform armazena seu arquivo de estado após a implantação. Também é aqui que ele espera encontrar o arquivo de estado quando uma nova implantação começar. Ao executar sua pilha localmente, conforme descrito acima, você pode manter uma cópia do estado do Terraform no diretório local de nível superior. Isso é conhecido como back-end local.

Ao desenvolver para um ambiente de integração contínua e implantação contínua (CI/CD), o arquivo de estado local geralmente é incluído no arquivo.gitignore para mantê-lo fora do controle de versão. Então, não há nenhum arquivo de estado local presente no pipeline. Para funcionar corretamente, esse estágio do pipeline precisa encontrar o arquivo de estado correto em algum lugar.É por isso que os arquivos de configuração do Terraform geralmente contêm um bloco de back-end. O bloco de back-end indica à pilha do Terraform que ela precisa procurar em algum lugar além de seu próprio diretório de nível superior para encontrar o arquivo de estado.

Um back-end do Terraform pode estar localizado em praticamente qualquer lugar: um bucket Amazon S3, um endpoint de API ou até mesmo um espaço de trabalho remoto do Terraform. Veja a seguir um exemplo de um back-end do Terraform armazenado em um bucket do Amazon S3.

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

Para evitar o armazenamento de informações confidenciais nos arquivos de configuração do Terraform, os back-ends também oferecem suporte a configurações parciais. No exemplo anterior, as credenciais necessárias para acessar o bucket não estão presentes na configuração. As credenciais podem ser obtidas a partir de variáveis de ambiente ou usando outros meios, como AWS Secrets Manager. Para obter mais informações, consulte Protegendo dados confidenciais usando o AWS Secrets Manager HashiCorp Terraform.

Um cenário de back-end comum é um back-end local usado em seu ambiente local para fins de teste. O arquivo terraform.tfstate está incluído no arquivo.gitignore para que não seja enviado ao repositório remoto. Então, cada ambiente dentro do pipeline de CI/CD manteria seu próprio back-end. Nesse cenário, vários desenvolvedores podem ter acesso a esse estado remoto, então você gostaria de proteger a integridade do arquivo de estado. Se várias implantações estiverem em execução e atualizando o estado ao mesmo tempo, o arquivo de estado poderá ficar corrompido. Por esse motivo, em situações com back-ends não locais, o arquivo de estado normalmente é bloqueado durante a implantação.

Entendendo as fontes de dados do Terraform

É muito comum que as pilhas de implantação dependam de dados de recursos existentes anteriormente. A maioria das ferramentas de IaC tem uma forma de importar recursos que foram criados por algum outro processo. Esses recursos importados geralmente são somente para leitura (embora as funções do IAM sejam uma exceção notável) e são usados para acessar os dados necessários aos recursos dentro da pilha. AWS CloudFormation permite a importação de recursos, mas essa ideia pode ser melhor explicada examinando o. AWS Cloud Development Kit (AWS CDK)

AWS CDK Isso ajuda os desenvolvedores a usar as linguagens de programação existentes para gerar CloudFormation modelos. O resultado final de uma AWS CDK operação é um recurso importado em CloudFormation. No entanto, a sintaxe usada com o AWS CDK facilita a comparação com o Terraform. Aqui está um exemplo de importação de um recurso usando o. AWS CDK

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

Um recurso importado geralmente é criado chamando um método estático na mesma classe que você usa para criar um novo recurso do mesmo tipo. A chamada `new Bucket(...` criaria um novo recurso e a chamada `Bucket.fromBucketAttributes(...` importaria um existente. Você passa um subconjunto das propriedades do intervalo para a função para que ela AWS CDK possa encontrar o intervalo certo. Outra diferença, no entanto, é que a criação de um novo bucket retorna uma instância completa da `Bucket` classe, com todas as propriedades e métodos disponíveis nela. A importação do recurso retorna an`IBucket`, que é um tipo que contém somente as propriedades que `Bucket` devem ter. Embora você possa importar um recurso de uma pilha externa, as opções do que você pode fazer com ele são limitadas.

No Terraform, uma meta semelhante é alcançada usando fontes de dados. A maioria dos recursos definidos do Terraform tem uma fonte de dados complementar disponível junto com ela. Veja a seguir um exemplo de um recurso de bucket do Terraform S3 seguido por sua fonte de dados correspondente.

```
# 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"
}
```

A única diferença entre esses dois itens é o prefixo do nome. Conforme mostrado na documentação de uma fonte de dados, há menos parâmetros disponíveis que você pode passar para uma fonte de dados do que para um recurso. Isso ocorre porque o recurso usa esses parâmetros para declarar todas as propriedades de um novo bucket do S3, enquanto a fonte de dados precisa apenas de informações suficientes para identificar e importar de forma exclusiva os dados de um recurso existente.

A semelhança entre a sintaxe de um recurso do Terraform e uma fonte de dados pode ser conveniente, mas também pode ser problemática. É comum que desenvolvedores novatos do Terraform usem acidentalmente uma fonte de dados em vez de um recurso em sua configuração. As fontes de dados do Terraform são sempre somente para leitura. Você pode usá-los no lugar do recurso correspondente para ações de leitura (como fornecer um nome de ID para outro recurso). No entanto, você não pode usá-los para escrever ações, que fundamentalmente alteram alguns aspectos do recurso subjacente. Por esse motivo, você pode pensar em uma fonte de dados do Terraform como uma versão clonada do recurso subjacente.

Semelhante ao exemplo anterior do AWS CDK iBucket, as fontes de dados são úteis para cenários somente de leitura. Se você precisar obter dados de um recurso existente, mas não precisar manter esse recurso em sua pilha, use uma fonte de dados. Um bom exemplo disso é quando você está criando uma instância do Amazon EC2 que usa a VPC padrão da conta. Como essa VPC já existe, tudo o que você precisa fazer é extrair seus dados. O exemplo de código a seguir mostra como usar dados para identificar a VPC de destino.

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

resource "aws_instance" "instance1" {
  ami           = "ami-123456"
  instance_type = "t2.micro"
  subnet_id     = data.aws_vpc.default.main_route_table_id
```

```
}
```

Compreendendo as variáveis, valores locais e saídas do Terraform

As variáveis aprimoram a flexibilidade do código ao permitir espaços reservados em blocos de código. As variáveis podem representar valores diferentes sempre que o código é reutilizado. O Terraform distingue entre seus tipos de variáveis por seu escopo modular. As variáveis de entrada são valores externos que podem ser injetados em um módulo, os valores de saída são valores internos que podem ser compartilhados externamente e os valores locais sempre permanecem dentro do escopo original.

Variáveis

AWS CloudFormation usa parâmetros para representar valores personalizados que podem ser definidos e redefinidos de uma implantação de pilha para a próxima. Da mesma forma, o Terraform usa variáveis de entrada ou variáveis. As variáveis podem ser declaradas em qualquer lugar em um arquivo de configuração do Terraform e geralmente são declaradas com o tipo de dados necessário ou o valor padrão. Todas as três expressões a seguir são declarações de variáveis válidas do Terraform.

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

variable "random_number" {
  default = 5
}

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

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

```
  ]
}
```

Para acessar a raça do Sparky dentro da configuração, você usaria a variável`var.dogs[0].breed`. Se uma variável não tiver padrão e não for classificada como anulável, o valor da variável deverá ser definido para cada implantação. Caso contrário, é opcional definir um novo valor para a variável. Em um módulo raiz, você pode definir os valores das variáveis atuais na linha de comando, como variáveis de ambiente ou no arquivo terraform.tfvars. O exemplo a seguir mostra como inserir valores variáveis no arquivo terraform.tfvars, que é armazenado no diretório de nível superior do módulo.

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

random_number = 7

thing_i_made_up = "Kabibble"
```

O valor do arquivo terraform.tfvars neste exemplo substituiria o valor padrão na declaração da variável. `dogs` Se você estiver declarando variáveis em um módulo filho, poderá definir os valores das variáveis diretamente no bloco de declaração do módulo, conforme mostrado no exemplo a seguir.

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

Alguns dos outros argumentos que você pode usar ao declarar uma variável incluem:

- `sensitive`— Definir isso para `true` evitar que o valor da variável seja exposto nas saídas do processo do Terraform.
- `nullable`— Definir isso para `true` permitir que a variável não tenha valor. Isso é conveniente para variáveis em que um padrão não está definido.
- `description`— Adicione uma descrição da variável aos metadados da pilha.
- `validation`— Defina regras de validação para a variável.

Um dos aspectos mais convenientes das variáveis do Terraform é a capacidade de adicionar um ou mais objetos de validação na declaração da variável. Você pode usar objetos de validação para adicionar uma condição que a variável deve passar, caso contrário, a implantação falhará. Você também pode definir uma mensagem de erro personalizada para ser exibida sempre que a condição for violada.

Por exemplo, você está configurando um arquivo de configuração do Terraform que os membros da sua equipe executarão. Antes de implantar as pilhas, um membro da equipe precisa criar um arquivo terraform.tfvars para definir um valor de configuração importante. Para lembrá-los, você pode fazer algo como o seguinte.

```
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-'"
  }
}
```

Conforme mostrado neste exemplo, você pode definir várias condições dentro de uma única variável. O Terraform mostra apenas mensagens de erro para condições com falha. Dessa forma, você pode aplicar todos os tipos de regras em valores variáveis. Se o valor de uma variável causar uma falha na tubulação, você saberia exatamente o motivo.

Valores locais

Se houver algum valor em um módulo que você queira criar um alias, use a `locals` palavra-chave em vez de declarar uma variável padrão que nunca será atualizada. Como o nome sugere, um `locals` bloco contém termos que têm como escopo interno esse módulo específico. Se você quiser transformar um valor de string, por exemplo, adicionando um prefixo a um valor de variável para uso em um nome de recurso, usar um valor local pode ser uma boa solução. Um único `locals` bloco pode declarar todos os valores locais do seu módulo, conforme mostrado no exemplo a seguir.

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

Lembre-se de que, quando você está acessando o valor, a `locals` palavra-chave se torna singular, como`local.LocalConfigId`.

Valores de saída

Se as variáveis de entrada do Terraform forem como CloudFormation parâmetros, você poderia dizer que os valores de saída do Terraform são como CloudFormation saídas. Ambos são usados para expor valores de dentro de uma pilha de implantação. No entanto, como o módulo Terraform está mais enraizado na estrutura da ferramenta, os valores de saída do Terraform também são usados para expor valores dentro de um módulo a um módulo pai ou a outros módulos secundários, mesmo que esses módulos estejam todos na mesma pilha de implantação. Se você estiver criando dois módulos personalizados e o primeiro módulo precisar acessar o valor de ID do segundo módulo, você precisará adicionar o `output` bloco a seguir ao segundo módulo.

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

resource "example_resource" "example_resource_name" {
  module_id = module.first_module.module_id
```

```
}
```

Como os valores de saída do Terraform podem ser usados na mesma pilha, você também pode usar o `sensitive` atributo em um `output` bloco para impedir que o valor seja exibido na saída da pilha. Além disso, um `output` bloco pode usar `precondition` blocos da mesma forma que as variáveis usam `validation` blocos: para garantir que as variáveis sigam um determinado conjunto de regras. Isso ajuda a garantir que todos os valores em um módulo existam conforme o esperado antes de prosseguir com a implantação.

```
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."
  }
}
```

Entendendo as funções, expressões e meta-argumentos do Terraform

Uma crítica às ferramentas de IaC que usam arquivos de configuração declarativos em vez de linguagens de programação comuns é que elas dificultam a implementação da lógica programática personalizada. Nas configurações do Terraform, esse problema é resolvido usando funções, expressões e meta-argumentos.

Funções

Uma das grandes vantagens de usar código para provisionar sua infraestrutura é a capacidade de armazenar fluxos de trabalho comuns e reutilizá-los repetidamente, geralmente transmitindo argumentos diferentes a cada vez. As funções do Terraform são semelhantes às funções AWS CloudFormation intrínsecas, embora sua sintaxe seja mais semelhante à forma como as funções são chamadas em linguagens programáticas. Talvez você já tenha notado algumas funções do Terraform, como substr, concat, length e base64decode, nos exemplos deste guia. Assim como CloudFormation as funções intrínsecas, o Terraform tem uma série de funções integradas que estão disponíveis para uso em suas configurações. Por exemplo, se um determinado atributo de recurso usa um objeto JSON muito grande que seria ineficiente para colar diretamente no arquivo, você poderia colocar o objeto em um arquivo.json e usar as funções do Terraform para acessá-lo. No exemplo a seguir, a `file` função retorna o conteúdo do arquivo na forma de string e, em seguida, a `jsondecode` função o converte em um tipo de objeto.

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

Expressões

O Terraform também permite expressões condicionais, que são semelhantes às CloudFormation `condition` funções, exceto pelo fato de usarem a sintaxe mais tradicional do operador ternário. No exemplo a seguir, as duas expressões retornam exatamente o mesmo resultado. O segundo exemplo é o que o Terraform chama de expressão splat. O asterisco faz com que o Terraform percorra a lista e crie uma nova lista usando apenas a `id` propriedade de cada item.

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

Meta-argumentos

No exemplo de código anterior, `list_value` e `list_value_2` são chamados de argumentos. Talvez você já esteja familiarizado com alguns desses meta-argumentos. O Terraform também tem alguns meta-argumentos, que funcionam como argumentos, mas com algumas funcionalidades extras:

- O meta-argumento depends_on é muito semelhante ao atributo. CloudFormation DependsOn
- O meta-argumento do provedor permite que você use várias configurações de provedor ao mesmo tempo.
- O meta-argumento do ciclo de vida permite que você personalize as configurações de recursos, de forma semelhante às políticas de remoção e exclusão em. CloudFormation

Outros meta-argumentos permitem que a funcionalidade de função e expressão seja adicionada diretamente a um recurso. Por exemplo, o meta-argumento count é um mecanismo útil para criar vários recursos semelhantes ao mesmo tempo. O exemplo a seguir demonstra como criar dois clusters do Amazon Elastic Container Service (Amazon EKS) sem usar o `count` meta-argumento.

```
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]
  }
}
```

O exemplo a seguir demonstra como usar o `count` meta-argumento para criar dois clusters do Amazon EKS.

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

Para dar um nome de unidade a cada uma, você pode acessar o índice da lista dentro do bloco de recursos em`count.index`. Mas e se você quiser criar vários recursos semelhantes que sejam um pouco mais complexos? É aí que entra o meta-argumento for_each. O `for_each` meta-argumento é muito semelhante a`count`, exceto que você passa uma lista ou um objeto em vez de um número. O Terraform cria um novo recurso para cada membro da lista ou objeto. É semelhante a se você definir`count = length(list)`, exceto que você pode acessar o conteúdo da lista em vez do índice do loop.

Isso funciona tanto para uma lista de itens quanto para um único objeto. O exemplo a seguir criaria dois recursos que têm `id-0` e `id-1` como seus IDs.

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

```
    { id = "id-1" },
  ]
}

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

O exemplo a seguir também criaria dois recursos, um para Sparky, o poodle, e outro para Fluffy, o chihuahua.

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

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

Assim como você pode acessar o índice do loop em count usando count.index, você pode acessar a chave e o valor de cada item em um loop for_each usando o objeto each. Como for_each itera tanto em listas quanto em objetos, o controle de cada chave e valor pode ser um pouco confuso. A tabela a seguir mostra as diferentes maneiras de usar o meta-argumento for_each e como você pode referenciar os valores em cada iteração.

Exemplo	Tipo de **`for_each`**	Primeira iteração	Segunda iteração

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

Exemplo	Tipo de **`for_each`**	Primeira iteração	Segunda iteração
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”`

Exemplo	Tipo de **`for_each`**	Primeira iteração	Segunda iteração

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”`
			
			`}`

Exemplo	Tipo de **`for_each`**	Primeira iteração	Segunda iteração

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”`
			
			`}`
			
			`]`

Exemplo	Tipo de **`for_each`**	Primeira iteração	Segunda iteração

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

Portanto, se `var.animals` fosse igual à linha E, você poderia criar um recurso por animal usando o código a seguir.

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

Como alternativa, você pode criar dois recursos por animal usando o código a seguir.

```
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
}
```

Perguntas frequentes

Quando devo usar o Terraform em vez de CloudFormation?

Em geral, se suas cargas de trabalho se baseiam principalmente AWS, AWS CloudFormation fornece um nível de suporte nativo que o Terraform não pode igualar. No entanto, se suas cargas de trabalho incluírem vários processos de terceiros ou estiverem espalhadas entre vários provedores de nuvem, o Terraform é uma ferramenta que você talvez queira considerar.

Quando devo usar o AWS CDK em vez de CloudFormation?

Quando você usa o AWS Cloud Development Kit (AWS CDK), você também está usando CloudFormation. AWS CDK Isso permite que você use uma linguagem de programação comum para gerar CloudFormation modelos. Se você tiver experiência em qualquer uma das linguagens de programação AWS CDK compatíveis, AWS CDK poderá reduzir o tempo necessário para gerar CloudFormation modelos.

Existe uma ferramenta como a AWS CDK que gera configurações do Terraform?

Em comparação com o AWS CDK, o CDK for Terraform (CDKTF) usa a mesma biblioteca de construção para provisionar recursos e o mesmo mecanismo jsii para oferecer suporte a várias linguagens de programação. Você pode usá-lo para gerar configurações do Terraform da mesma forma que AWS CDK gera CloudFormation modelos.

Como faço para saber mais sobre o Terraform?

Para obter mais informações sobre conceitos avançados do Terraform, consulte a documentação do Terraform. Ele também descreve os componentes de todos os principais fornecedores e módulos de código aberto.

Recursos relacionados

AWS documentação

- Documentação da AWS CDK
- Documentação da AWS CloudFormation
- Terraform: Além do básico com AWS (AWS postagem do blog)

Outros recursos

- Documentação do CDK para Terraform
- Documentação do Terraform

Apêndice: exemplos de acesso ao atributo Terraform

Recurso

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

bucketName = aws_s3_bucket.myS3Bucket.bucket
```

Fonte de dados

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

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

Módulo

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

vpc_id = module.eks.vpc_id
```

Variável

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

animalType = var.my_variable
```

Local

```
locals {
  type = "dog"
}

animalType = local.type
```

Histórico do documento

A tabela a seguir descreve alterações significativas feitas neste guia. Se desejar receber notificações sobre futuras atualizações, inscreva-se em um feed RSS.

Alteração	Descrição	Data
Publicação inicial	—	29 de março de 2024

AWS Glossário de orientação prescritiva

A seguir estão os termos comumente usados em estratégias, guias e padrões fornecidos pela Orientação AWS Prescritiva. Para sugerir entradas, use o link Fornecer feedback no final do glossário.

Números

7 Rs

Sete estratégias comuns de migração para mover aplicações para a nuvem. Essas estratégias baseiam-se nos 5 Rs identificados pela Gartner em 2011 e consistem em:

- Refatorar/rearquitetar: mova uma aplicação e modifique sua arquitetura aproveitando ao máximo os recursos nativos de nuvem para melhorar a agilidade, a performance e a escalabilidade. Isso normalmente envolve a portabilidade do sistema operacional e do banco de dados. Exemplo: migre seu banco de dados Oracle local para a edição compatível com o Amazon Aurora PostgreSQL.
- Redefinir a plataforma (mover e redefinir [mover e redefinir (lift-and-reshape]): mova uma aplicação para a nuvem e introduza algum nível de otimização a fim de aproveitar os recursos da nuvem. Exemplo: Migre seu banco de dados Oracle local para o Amazon Relational Database Service (Amazon RDS) for Oracle no. Nuvem AWS
- Recomprar (drop and shop): mude para um produto diferente, normalmente migrando de uma licença tradicional para um modelo SaaS. Exemplo: migre seu sistema de gerenciamento de relacionamento com o cliente (CRM) para a Salesforce.com.
- Redefinir a hospedagem (mover sem alterações [lift-and-shift])mover uma aplicação para a nuvem sem fazer nenhuma alteração a fim de aproveitar os recursos da nuvem. Exemplo: Migre seu banco de dados Oracle local para o Oracle em uma instância do EC2 no. Nuvem AWS
- Realocar (mover o hipervisor sem alterações [hypervisor-level lift-and-shift]): mover a infraestrutura para a nuvem sem comprar novo hardware, reescrever aplicações ou modificar suas operações existentes. Você migra servidores de uma plataforma local para um serviço em nuvem para a mesma plataforma. Exemplo: migrar um Microsoft Hyper-V aplicativo para o. AWS
- Reter (revisitar): mantenha as aplicações em seu ambiente de origem. Isso pode incluir aplicações que exigem grande refatoração, e você deseja adiar esse trabalho para um

momento posterior, e aplicações antigas que você deseja manter porque não há justificativa comercial para migrá-las.

- Retirar: desative ou remova aplicações que não são mais necessárias em seu ambiente de origem.

A

ABAC

Consulte controle de acesso baseado em atributos.

serviços abstratos

Veja os serviços gerenciados.

ACID

Veja atomicidade, consistência, isolamento, durabilidade.

migração ativa-ativa

Um método de migração de banco de dados no qual os bancos de dados de origem e de destino são mantidos em sincronia (por meio de uma ferramenta de replicação bidirecional ou operações de gravação dupla), e ambos os bancos de dados lidam com transações de aplicações conectadas durante a migração. Esse método oferece suporte à migração em lotes pequenos e controlados, em vez de exigir uma substituição única. É mais flexível, mas exige mais trabalho do que a migração ativa-passiva.

migração ativa-passiva

Um método de migração de banco de dados no qual os bancos de dados de origem e de destino são mantidos em sincronia, mas somente o banco de dados de origem manipula as transações das aplicações conectadas enquanto os dados são replicados no banco de dados de destino. O banco de dados de destino não aceita nenhuma transação durante a migração.

função agregada

Uma função SQL que opera em um grupo de linhas e calcula um único valor de retorno para o grupo. Exemplos de funções agregadas incluem `SUM` e. `MAX`

AI

Veja inteligência artificial.

AIOps

Veja as operações de inteligência artificial.

anonimização

O processo de excluir permanentemente informações pessoais em um conjunto de dados. A anonimização pode ajudar a proteger a privacidade pessoal. Dados anônimos não são mais considerados dados pessoais.

antipadrões

Uma solução frequentemente usada para um problema recorrente em que a solução é contraproducente, ineficaz ou menos eficaz do que uma alternativa.

controle de aplicativos

Uma abordagem de segurança que permite o uso somente de aplicativos aprovados para ajudar a proteger um sistema contra malware.

portfólio de aplicações

Uma coleção de informações detalhadas sobre cada aplicação usada por uma organização, incluindo o custo para criar e manter a aplicação e seu valor comercial. Essas informações são fundamentais para o processo de descoberta e análise de portfólio e ajudam a identificar e priorizar as aplicações a serem migradas, modernizadas e otimizadas.

inteligência artificial (IA)

O campo da ciência da computação que se dedica ao uso de tecnologias de computação para desempenhar funções cognitivas normalmente associadas aos humanos, como aprender, resolver problemas e reconhecer padrões. Para obter mais informações, consulte O que é inteligência artificial?

operações de inteligência artificial (AIOps)

O processo de usar técnicas de machine learning para resolver problemas operacionais, reduzir incidentes operacionais e intervenção humana e aumentar a qualidade do serviço. Para obter mais informações sobre como as AIOps são usadas na estratégia de migração para a AWS , consulte o guia de integração de operações.

criptografia assimétrica

Um algoritmo de criptografia que usa um par de chaves, uma chave pública para criptografia e uma chave privada para descriptografia. É possível compartilhar a chave pública porque ela não é usada na descriptografia, mas o acesso à chave privada deve ser altamente restrito.

atomicidade, consistência, isolamento, durabilidade (ACID)

Um conjunto de propriedades de software que garantem a validade dos dados e a confiabilidade operacional de um banco de dados, mesmo no caso de erros, falhas de energia ou outros problemas.

controle de acesso por atributo (ABAC)

A prática de criar permissões minuciosas com base nos atributos do usuário, como departamento, cargo e nome da equipe. Para obter mais informações, consulte ABAC AWS na documentação AWS Identity and Access Management (IAM).

fonte de dados autorizada

Um local onde você armazena a versão principal dos dados, que é considerada a fonte de informações mais confiável. Você pode copiar dados da fonte de dados autorizada para outros locais com o objetivo de processar ou modificar os dados, como anonimizá-los, redigi-los ou pseudonimizá-los.

Availability Zone (zona de disponibilidade)

Um local distinto dentro de um Região da AWS que está isolado de falhas em outras zonas de disponibilidade e fornece conectividade de rede barata e de baixa latência a outras zonas de disponibilidade na mesma região.

AWS Estrutura de adoção da nuvem (AWS CAF)

Uma estrutura de diretrizes e melhores práticas AWS para ajudar as organizações a desenvolver um plano eficiente e eficaz para migrar com sucesso para a nuvem. AWS O CAF organiza a orientação em seis áreas de foco chamadas perspectivas: negócios, pessoas, governança, plataforma, segurança e operações. As perspectivas de negócios, pessoas e governança têm como foco habilidades e processos de negócios; as perspectivas de plataforma, segurança e operações concentram-se em habilidades e processos técnicos. Por exemplo, a perspectiva das pessoas tem como alvo as partes interessadas que lidam com recursos humanos (RH), funções de pessoal e gerenciamento de pessoal. Nessa perspectiva, o AWS CAF fornece orientação para desenvolvimento, treinamento e comunicação de pessoas para ajudar a preparar a organização

para a adoção bem-sucedida da nuvem. Para obter mais informações, consulte o site da AWS CAF e o whitepaper da AWS CAF.

AWS Estrutura de qualificação da carga de trabalho (AWS WQF)

Uma ferramenta que avalia as cargas de trabalho de migração do banco de dados, recomenda estratégias de migração e fornece estimativas de trabalho. AWS O WQF está incluído com AWS Schema Conversion Tool ()AWS SCT. Ela analisa esquemas de banco de dados e objetos de código, código de aplicações, dependências e características de performance, além de fornecer relatórios de avaliação.

B

bot ruim

Um bot destinado a perturbar ou causar danos a indivíduos ou organizações.

BCP

Veja o planejamento de continuidade de negócios.

gráfico de comportamento

Uma visualização unificada e interativa do comportamento e das interações de recursos ao longo do tempo. É possível usar um gráfico de comportamento com o Amazon Detective para examinar tentativas de login malsucedidas, chamadas de API suspeitas e ações similares. Para obter mais informações, consulte Dados em um gráfico de comportamento na documentação do Detective.

sistema big-endian

Um sistema que armazena o byte mais significativo antes. Veja também endianness.

classificação binária

Um processo que prevê um resultado binário (uma de duas classes possíveis). Por exemplo, seu modelo de ML pode precisar prever problemas como “Este e-mail é ou não é spam?” ou “Este produto é um livro ou um carro?”

filtro de bloom

Uma estrutura de dados probabilística e eficiente em termos de memória que é usada para testar se um elemento é membro de um conjunto.

blue/green deployment (implantação azul/verde)

Uma estratégia de implantação em que você cria dois ambientes separados, mas idênticos. Você executa a versão atual do aplicativo em um ambiente (azul) e a nova versão do aplicativo no outro ambiente (verde). Essa estratégia ajuda você a reverter rapidamente com o mínimo de impacto.

bot

Um aplicativo de software que executa tarefas automatizadas pela Internet e simula a atividade ou interação humana. Alguns bots são úteis ou benéficos, como rastreadores da Web que indexam informações na Internet. Alguns outros bots, conhecidos como bots ruins, têm como objetivo perturbar ou causar danos a indivíduos ou organizações.

botnet

Redes de bots infectadas por malware e sob o controle de uma única parte, conhecidas como pastor de bots ou operador de bots. As redes de bots são o mecanismo mais conhecido para escalar bots e seu impacto.

ramo

Uma área contida de um repositório de código. A primeira ramificação criada em um repositório é a ramificação principal. Você pode criar uma nova ramificação a partir de uma ramificação existente e, em seguida, desenvolver recursos ou corrigir bugs na nova ramificação. Uma ramificação que você cria para gerar um recurso é comumente chamada de ramificação de recurso. Quando o recurso estiver pronto para lançamento, você mesclará a ramificação do recurso de volta com a ramificação principal. Para obter mais informações, consulte Sobre filiais (GitHub documentação).

acesso em vidro quebrado

Em circunstâncias excepcionais e por meio de um processo aprovado, um meio rápido para um usuário obter acesso a um Conta da AWS que ele normalmente não tem permissão para acessar. Para obter mais informações, consulte o indicador Implementar procedimentos de quebra de vidro na orientação do Well-Architected AWS .

estratégia brownfield

A infraestrutura existente em seu ambiente. Ao adotar uma estratégia brownfield para uma arquitetura de sistema, você desenvolve a arquitetura de acordo com as restrições dos sistemas e da infraestrutura atuais. Se estiver expandindo a infraestrutura existente, poderá combinar as estratégias brownfield e greenfield.

cache do buffer

A área da memória em que os dados acessados com mais frequência são armazenados.

capacidade de negócios

O que uma empresa faz para gerar valor (por exemplo, vendas, atendimento ao cliente ou marketing). As arquiteturas de microsserviços e as decisões de desenvolvimento podem ser orientadas por recursos de negócios. Para obter mais informações, consulte a seção Organizados de acordo com as capacidades de negócios do whitepaper Executar microsserviços conteinerizados na AWS.

planejamento de continuidade de negócios (BCP)

Um plano que aborda o impacto potencial de um evento disruptivo, como uma migração em grande escala, nas operações e permite que uma empresa retome as operações rapidamente.

C

CAF

Consulte Estrutura de adoção da AWS nuvem.

implantação canária

O lançamento lento e incremental de uma versão para usuários finais. Quando estiver confiante, você implanta a nova versão e substituirá a versão atual em sua totalidade.

CCoE

Veja o Centro de Excelência em Nuvem.

CDC

Veja a captura de dados de alterações.

captura de dados de alterações (CDC)

O processo de rastrear alterações em uma fonte de dados, como uma tabela de banco de dados, e registrar metadados sobre a alteração. É possível usar o CDC para várias finalidades, como auditar ou replicar alterações em um sistema de destino para manter a sincronização.

engenharia do caos

Introduzir intencionalmente falhas ou eventos disruptivos para testar a resiliência de um sistema. Você pode usar AWS Fault Injection Service (AWS FIS) para realizar experimentos que estressam suas AWS cargas de trabalho e avaliar sua resposta.

CI/CD

Veja a integração e a entrega contínuas.

classificação

Um processo de categorização que ajuda a gerar previsões. Os modelos de ML para problemas de classificação predizem um valor discreto. Os valores discretos são sempre diferentes uns dos outros. Por exemplo, um modelo pode precisar avaliar se há ou não um carro em uma imagem.

criptografia no lado do cliente

Criptografia de dados localmente, antes que o alvo os AWS service (Serviço da AWS) receba.

Centro de Excelência da Nuvem (CCoE)

Uma equipe multidisciplinar que impulsiona os esforços de adoção da nuvem em toda a organização, incluindo o desenvolvimento de práticas recomendas de nuvem, a mobilização de recursos, o estabelecimento de cronogramas de migração e a liderança da organização em transformações em grande escala. Para obter mais informações, consulte as postagens do CCoE no blog de estratégia Nuvem AWS corporativa.

computação em nuvem

A tecnologia de nuvem normalmente usada para armazenamento de dados remoto e gerenciamento de dispositivos de IoT. A computação em nuvem geralmente está conectada à tecnologia de computação de ponta.

modelo operacional em nuvem

Em uma organização de TI, o modelo operacional usado para criar, amadurecer e otimizar um ou mais ambientes de nuvem. Para obter mais informações, consulte Criar seu modelo operacional de nuvem.

estágios de adoção da nuvem

As quatro fases pelas quais as organizações normalmente passam quando migram para o Nuvem AWS:

- Projeto: executar alguns projetos relacionados à nuvem para fins de prova de conceito e aprendizado
- Fundação: realizar investimentos fundamentais para escalar sua adoção da nuvem (por exemplo, criar uma zona de pouso, definir um CCoE, estabelecer um modelo de operações)
- Migração: migrar aplicações individuais
- Reinvenção: otimizar produtos e serviços e inovar na nuvem

Esses estágios foram definidos por Stephen Orban na postagem do blog The Journey Toward Cloud-First & the Stages of Adoption no blog de estratégia Nuvem AWS empresarial. Para obter informações sobre como eles se relacionam com a estratégia de AWS migração, consulte o guia de preparação para migração.

CMDB

Consulte o banco de dados de gerenciamento de configuração.

repositório de código

Um local onde o código-fonte e outros ativos, como documentação, amostras e scripts, são armazenados e atualizados por meio de processos de controle de versão. Os repositórios de nuvem comuns incluem GitHub ou AWS CodeCommit. Cada versão do código é chamada de ramificação. Em uma estrutura de microsserviços, cada repositório é dedicado a uma única peça de funcionalidade. Um único pipeline de CI/CD pode usar vários repositórios.

cache frio

Um cache de buffer que está vazio, não está bem preenchido ou contém dados obsoletos ou irrelevantes. Isso afeta a performance porque a instância do banco de dados deve ler da memória principal ou do disco, um processo que é mais lento do que a leitura do cache do buffer.

dados frios

Dados que raramente são acessados e geralmente são históricos. Ao consultar esse tipo de dados, consultas lentas geralmente são aceitáveis. Mover esses dados para níveis ou classes de armazenamento de baixo desempenho e menos caros pode reduzir os custos.

visão computacional (CV)

Um campo da IA que usa aprendizado de máquina para analisar e extrair informações de formatos visuais, como imagens e vídeos digitais. Por exemplo, AWS Panorama oferece dispositivos que adicionam CV às redes de câmeras locais, e a Amazon SageMaker fornece algoritmos de processamento de imagem para CV.

desvio de configuração

Para uma carga de trabalho, uma alteração de configuração em relação ao estado esperado. Isso pode fazer com que a carga de trabalho se torne incompatível e, normalmente, é gradual e não intencional.

banco de dados de gerenciamento de configuração (CMDB)

Um repositório que armazena e gerencia informações sobre um banco de dados e seu ambiente de TI, incluindo componentes de hardware e software e suas configurações. Normalmente, os dados de um CMDB são usados no estágio de descoberta e análise do portfólio da migração.

pacote de conformidade

Um conjunto de AWS Config regras e ações de remediação que você pode montar para personalizar suas verificações de conformidade e segurança. Você pode implantar um pacote de conformidade como uma entidade única em uma Conta da AWS região ou em uma organização usando um modelo YAML. Para obter mais informações, consulte Pacotes de conformidade na documentação. AWS Config

integração contínua e entrega contínua (CI/CD)

O processo de automatizar os estágios de origem, criação, teste, preparação e produção do processo de lançamento do software. O CI/CD é comumente descrito como um pipeline. O CI/CD pode ajudar você a automatizar processos, melhorar a produtividade, melhorar a qualidade do código e entregar com mais rapidez. Para obter mais informações, consulte Benefícios da entrega contínua. CD também pode significar implantação contínua. Para obter mais informações, consulte Entrega contínua versus implantação contínua.

CV

Veja visão computacional.

D

dados em repouso

Dados estacionários em sua rede, por exemplo, dados que estão em um armazenamento.

classificação de dados

Um processo para identificar e categorizar os dados em sua rede com base em criticalidade e confidencialidade. É um componente crítico de qualquer estratégia de gerenciamento de riscos de

segurança cibernética, pois ajuda a determinar os controles adequados de proteção e retenção para os dados. A classificação de dados é um componente do pilar de segurança no AWS Well-Architected Framework. Para obter mais informações, consulte Classificação de dados.

desvio de dados

Uma variação significativa entre os dados de produção e os dados usados para treinar um modelo de ML ou uma alteração significativa nos dados de entrada ao longo do tempo. O desvio de dados pode reduzir a qualidade geral, a precisão e a imparcialidade das previsões do modelo de ML.

dados em trânsito

Dados que estão se movendo ativamente pela sua rede, como entre os recursos da rede.

malha de dados

Uma estrutura arquitetônica que fornece propriedade de dados distribuída e descentralizada com gerenciamento e governança centralizados.

minimização de dados

O princípio de coletar e processar apenas os dados estritamente necessários. Praticar a minimização de dados no Nuvem AWS pode reduzir os riscos de privacidade, os custos e a pegada de carbono de sua análise.

perímetro de dados

Um conjunto de proteções preventivas em seu AWS ambiente que ajudam a garantir que somente identidades confiáveis acessem recursos confiáveis das redes esperadas. Para obter mais informações, consulte Construindo um perímetro de dados em. AWS

pré-processamento de dados

A transformação de dados brutos em um formato que seja facilmente analisado por seu modelo de ML. O pré-processamento de dados pode significar a remoção de determinadas colunas ou linhas e o tratamento de valores ausentes, inconsistentes ou duplicados.

proveniência dos dados

O processo de rastrear a origem e o histórico dos dados ao longo de seu ciclo de vida, por exemplo, como os dados foram gerados, transmitidos e armazenados.

titular dos dados

Um indivíduo cujos dados estão sendo coletados e processados.

data warehouse

Um sistema de gerenciamento de dados que oferece suporte à inteligência comercial, como análises. Os data warehouses geralmente contêm grandes quantidades de dados históricos e geralmente são usados para consultas e análises.

linguagem de definição de dados (DDL)

Instruções ou comandos para criar ou modificar a estrutura de tabelas e objetos em um banco de dados.

linguagem de manipulação de dados (DML)

Instruções ou comandos para modificar (inserir, atualizar e excluir) informações em um banco de dados.

DDL

Consulte a linguagem de definição de banco de dados.

deep ensemble

A combinação de vários modelos de aprendizado profundo para gerar previsões. Os deep ensembles podem ser usados para produzir uma previsão mais precisa ou para estimar a incerteza nas previsões.

Aprendizado profundo

Um subcampo do ML que usa várias camadas de redes neurais artificiais para identificar o mapeamento entre os dados de entrada e as variáveis-alvo de interesse.

defense-in-depth

Uma abordagem de segurança da informação na qual uma série de mecanismos e controles de segurança são cuidadosamente distribuídos por toda a rede de computadores para proteger a confidencialidade, a integridade e a disponibilidade da rede e dos dados nela contidos. Ao adotar essa estratégia AWS, você adiciona vários controles em diferentes camadas da AWS Organizations estrutura para ajudar a proteger os recursos. Por exemplo, uma defense-in-depth abordagem pode combinar autenticação multifatorial, segmentação de rede e criptografia.

administrador delegado

Em AWS Organizations, um serviço compatível pode registrar uma conta de AWS membro para administrar as contas da organização e gerenciar as permissões desse serviço. Essa conta

é chamada de administrador delegado para esse serviço Para obter mais informações e uma lista de serviços compatíveis, consulte Serviços que funcionam com o AWS Organizations na documentação do AWS Organizations .

implantação

O processo de criar uma aplicação, novos recursos ou correções de código disponíveis no ambiente de destino. A implantação envolve a implementação de mudanças em uma base de código e, em seguida, a criação e execução dessa base de código nos ambientes da aplicação

ambiente de desenvolvimento

Veja o ambiente.

controle detectivo

Um controle de segurança projetado para detectar, registrar e alertar após a ocorrência de um evento. Esses controles são uma segunda linha de defesa, alertando você sobre eventos de segurança que contornaram os controles preventivos em vigor. Para obter mais informações, consulte Controles detectivos em Como implementar controles de segurança na AWS.

mapeamento do fluxo de valor de desenvolvimento (DVSM)

Um processo usado para identificar e priorizar restrições que afetam negativamente a velocidade e a qualidade em um ciclo de vida de desenvolvimento de software. O DVSM estende o processo de mapeamento do fluxo de valor originalmente projetado para práticas de manufatura enxuta. Ele se concentra nas etapas e equipes necessárias para criar e movimentar valor por meio do processo de desenvolvimento de software.

gêmeo digital

Uma representação virtual de um sistema real, como um prédio, fábrica, equipamento industrial ou linha de produção. Os gêmeos digitais oferecem suporte à manutenção preditiva, ao monitoramento remoto e à otimização da produção.

tabela de dimensões

Em um esquema em estrela, uma tabela menor que contém atributos de dados sobre dados quantitativos em uma tabela de fatos. Os atributos da tabela de dimensões geralmente são campos de texto ou números discretos que se comportam como texto. Esses atributos são comumente usados para restringir consultas, filtrar e rotular conjuntos de resultados.

desastre

Um evento que impede que uma workload ou sistema cumpra seus objetivos de negócios em seu local principal de implantação. Esses eventos podem ser desastres naturais, falhas técnicas ou o resultado de ações humanas, como configuração incorreta não intencional ou ataque de malware.

Recuperação de desastres (RD)

A estratégia e o processo que você usa para minimizar o tempo de inatividade e a perda de dados causados por um desastre. Para obter mais informações, consulte Recuperação de desastres de cargas de trabalho em AWS: Recuperação na nuvem no AWS Well-Architected Framework.

DML

Consulte linguagem de manipulação de banco de dados.

design orientado por domínio

Uma abordagem ao desenvolvimento de um sistema de software complexo conectando seus componentes aos domínios em evolução, ou principais metas de negócios, atendidos por cada componente. Esse conceito foi introduzido por Eric Evans em seu livro, Design orientado por domínio: lidando com a complexidade no coração do software (Boston: Addison-Wesley Professional, 2003). Para obter informações sobre como usar o design orientado por domínio com o padrão strangler fig, consulte Modernizar incrementalmente os serviços web herdados do Microsoft ASP.NET (ASMX) usando contêineres e o Amazon API Gateway.

DR

Veja a recuperação de desastres.

detecção de deriva

Rastreando desvios de uma configuração básica. Por exemplo, você pode usar AWS CloudFormation para detectar desvios nos recursos do sistema ou AWS Control Tower para detectar mudanças em seu landing zone que possam afetar a conformidade com os requisitos de governança.

DVSM

Veja o mapeamento do fluxo de valor do desenvolvimento.

E

EDA

Veja a análise exploratória de dados.

computação de borda

A tecnologia que aumenta o poder computacional de dispositivos inteligentes nas bordas de uma rede de IoT. Quando comparada à computação em nuvem, a computação de ponta pode reduzir a latência da comunicação e melhorar o tempo de resposta.

Criptografia

Um processo de computação que transforma dados de texto simples, legíveis por humanos, em texto cifrado.

chave de criptografia

Uma sequência criptográfica de bits aleatórios que é gerada por um algoritmo de criptografia. As chaves podem variar em tamanho, e cada chave foi projetada para ser imprevisível e exclusiva.

endianismo

A ordem na qual os bytes são armazenados na memória do computador. Os sistemas big-endian armazenam o byte mais significativo antes. Os sistemas little-endian armazenam o byte menos significativo antes.

endpoint

Veja o endpoint do serviço.

serviço de endpoint

Um serviço que pode ser hospedado em uma nuvem privada virtual (VPC) para ser compartilhado com outros usuários. Você pode criar um serviço de endpoint com AWS PrivateLink e conceder permissões a outros diretores Contas da AWS ou a AWS Identity and Access Management (IAM). Essas contas ou entidades principais podem se conectar ao serviço de endpoint de maneira privada criando endpoints da VPC de interface. Para obter mais informações, consulte Criar um serviço de endpoint na documentação do Amazon Virtual Private Cloud (Amazon VPC).

planejamento de recursos corporativos (ERP)

Um sistema que automatiza e gerencia os principais processos de negócios (como contabilidade, MES e gerenciamento de projetos) para uma empresa.

criptografia envelopada

O processo de criptografar uma chave de criptografia com outra chave de criptografia. Para obter mais informações, consulte Criptografia de envelope na documentação AWS Key Management Service (AWS KMS).

environment (ambiente)

Uma instância de uma aplicação em execução. Estes são tipos comuns de ambientes na computação em nuvem:

- ambiente de desenvolvimento: uma instância de uma aplicação em execução que está disponível somente para a equipe principal responsável pela manutenção da aplicação. Ambientes de desenvolvimento são usados para testar mudanças antes de promovê-las para ambientes superiores. Esse tipo de ambiente às vezes é chamado de ambiente de teste.
- ambientes inferiores: todos os ambientes de desenvolvimento para uma aplicação, como aqueles usados para compilações e testes iniciais.
- ambiente de produção: uma instância de uma aplicação em execução que os usuários finais podem acessar. Em um pipeline de CI/CD, o ambiente de produção é o último ambiente de implantação.
- ambientes superiores: todos os ambientes que podem ser acessados por usuários que não sejam a equipe principal de desenvolvimento. Isso pode incluir um ambiente de produção, ambientes de pré-produção e ambientes para testes de aceitação do usuário.

epic

Em metodologias ágeis, categorias funcionais que ajudam a organizar e priorizar seu trabalho. Os epics fornecem uma descrição de alto nível dos requisitos e das tarefas de implementação. Por exemplo, os épicos de segurança AWS da CAF incluem gerenciamento de identidade e acesso, controles de detetive, segurança de infraestrutura, proteção de dados e resposta a incidentes. Para obter mais informações sobre epics na estratégia de migração da AWS , consulte o guia de implementação do programa.

ERP

Consulte planejamento de recursos corporativos.

análise exploratória de dados (EDA)

O processo de analisar um conjunto de dados para entender suas principais características. Você coleta ou agrega dados e, em seguida, realiza investigações iniciais para encontrar padrões,

detectar anomalias e verificar suposições. O EDA é realizado por meio do cálculo de estatísticas resumidas e da criação de visualizações de dados.

F

tabela de fatos

A tabela central em um esquema em estrela. Ele armazena dados quantitativos sobre operações comerciais. Normalmente, uma tabela de fatos contém dois tipos de colunas: aquelas que contêm medidas e aquelas que contêm uma chave externa para uma tabela de dimensões.

falham rapidamente

Uma filosofia que usa testes frequentes e incrementais para reduzir o ciclo de vida do desenvolvimento. É uma parte essencial de uma abordagem ágil.

limite de isolamento de falhas

No Nuvem AWS, um limite, como uma zona de disponibilidade, Região da AWS um plano de controle ou um plano de dados, que limita o efeito de uma falha e ajuda a melhorar a resiliência das cargas de trabalho. Para obter mais informações, consulte Limites de isolamento de AWS falhas.

ramificação de recursos

Veja a filial.

recursos

Os dados de entrada usados para fazer uma previsão. Por exemplo, em um contexto de manufatura, os recursos podem ser imagens capturadas periodicamente na linha de fabricação.

importância do recurso

O quanto um recurso é importante para as previsões de um modelo. Isso geralmente é expresso como uma pontuação numérica que pode ser calculada por meio de várias técnicas, como Shapley Additive Explanations (SHAP) e gradientes integrados. Para obter mais informações, consulte Interpretabilidade do modelo de aprendizado de máquina com:AWS.

transformação de recursos

O processo de otimizar dados para o processo de ML, incluindo enriquecer dados com fontes adicionais, escalar valores ou extrair vários conjuntos de informações de um único

campo de dados. Isso permite que o modelo de ML se beneficie dos dados. Por exemplo, se a data “2021-05-27 00:15:37” for dividida em “2021”, “maio”, “quinta” e “15”, isso poderá ajudar o algoritmo de aprendizado a aprender padrões diferenciados associados a diferentes componentes de dados.

FGAC

Veja o controle de acesso refinado.

Controle de acesso refinado (FGAC)

O uso de várias condições para permitir ou negar uma solicitação de acesso.

migração flash-cut

Um método de migração de banco de dados que usa replicação contínua de dados por meio da captura de dados alterados para migrar dados no menor tempo possível, em vez de usar uma abordagem em fases. O objetivo é reduzir ao mínimo o tempo de inatividade.

G

bloqueio geográfico

Veja as restrições geográficas.

restrições geográficas (bloqueio geográfico)

Na Amazon CloudFront, uma opção para impedir que usuários em países específicos acessem distribuições de conteúdo. É possível usar uma lista de permissões ou uma lista de bloqueios para especificar países aprovados e banidos. Para obter mais informações, consulte Restringir a distribuição geográfica do seu conteúdo na CloudFront documentação.

Fluxo de trabalho do GitFlow

Uma abordagem na qual ambientes inferiores e superiores usam ramificações diferentes em um repositório de código-fonte. O fluxo de trabalho do Gitflow é considerado legado, e o fluxo de trabalho baseado em troncos é a abordagem moderna e preferida.

estratégia greenfield

A ausência de infraestrutura existente em um novo ambiente. Ao adotar uma estratégia greenfield para uma arquitetura de sistema, é possível selecionar todas as novas tecnologias sem a

restrição da compatibilidade com a infraestrutura existente, também conhecida como brownfield. Se estiver expandindo a infraestrutura existente, poderá combinar as estratégias brownfield e greenfield.

barreira de proteção

Uma regra de alto nível que ajuda a gerenciar recursos, políticas e conformidade em todas as unidades organizacionais (UOs). Barreiras de proteção preventivas impõem políticas para garantir o alinhamento a padrões de conformidade. Elas são implementadas usando políticas de controle de serviço e limites de permissões do IAM. Barreiras de proteção detectivas detectam violações de políticas e problemas de conformidade e geram alertas para remediação. Eles são implementados usando AWS Config, AWS Security Hub, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector e verificações personalizadas AWS Lambda .

H

HA

Veja a alta disponibilidade.

migração heterogênea de bancos de dados

Migrar seu banco de dados de origem para um banco de dados de destino que usa um mecanismo de banco de dados diferente (por exemplo, Oracle para Amazon Aurora). A migração heterogênea geralmente faz parte de um esforço de redefinição da arquitetura, e converter o esquema pode ser uma tarefa complexa. O AWS fornece o AWS SCT para ajudar nas conversões de esquemas.

alta disponibilidade (HA)

A capacidade de uma workload operar continuamente, sem intervenção, em caso de desafios ou desastres. Os sistemas AH são projetados para realizar o failover automático, oferecer consistentemente desempenho de alta qualidade e lidar com diferentes cargas e falhas com impacto mínimo no desempenho.

modernização de historiador

Uma abordagem usada para modernizar e atualizar os sistemas de tecnologia operacional (OT) para melhor atender às necessidades do setor de manufatura. Um historiador é um tipo de banco de dados usado para coletar e armazenar dados de várias fontes em uma fábrica.

migração homogênea de bancos de dados

Migrar seu banco de dados de origem para um banco de dados de destino que compartilha o mesmo mecanismo de banco de dados (por exemplo, Microsoft SQL Server para Amazon RDS para SQL Server). A migração homogênea geralmente faz parte de um esforço de redefinição da hospedagem ou da plataforma. É possível usar utilitários de banco de dados nativos para migrar o esquema.

dados quentes

Dados acessados com frequência, como dados em tempo real ou dados translacionais recentes. Esses dados normalmente exigem uma camada ou classe de armazenamento de alto desempenho para fornecer respostas rápidas às consultas.

hotfix

Uma correção urgente para um problema crítico em um ambiente de produção. Devido à sua urgência, um hotfix geralmente é feito fora do fluxo de trabalho típico de uma DevOps versão.

período de hipercuidados

Imediatamente após a substituição, o período em que uma equipe de migração gerencia e monitora as aplicações migradas na nuvem para resolver quaisquer problemas. Normalmente, a duração desse período é de 1 a 4 dias. No final do período de hipercuidados, a equipe de migração normalmente transfere a responsabilidade pelas aplicações para a equipe de operações de nuvem.

I

IaC

Veja a infraestrutura como código.

Política baseada em identidade

Uma política anexada a um ou mais diretores do IAM que define suas permissões no Nuvem AWS ambiente.

aplicação ociosa

Uma aplicação que tem um uso médio de CPU e memória entre 5 e 20% em um período de 90 dias. Em um projeto de migração, é comum retirar essas aplicações ou retê-las on-premises.

IIoT

Veja a Internet das Coisas industrial.

infraestrutura imutável

Um modelo que implanta uma nova infraestrutura para cargas de trabalho de produção em vez de atualizar, corrigir ou modificar a infraestrutura existente. Infraestruturas imutáveis são inerentemente mais consistentes, confiáveis e previsíveis do que infraestruturas mutáveis. Para obter mais informações, consulte as melhores práticas de implantação usando infraestrutura imutável no Well-Architected AWS Framework.

VPC de entrada (admissão)

Em uma arquitetura de AWS várias contas, uma VPC que aceita, inspeciona e roteia conexões de rede de fora de um aplicativo. A Arquitetura de referência de segurança da AWS recomenda configurar sua conta de rede com VPCs de entrada, saída e inspeção para proteger a interface bidirecional entre a aplicação e a Internet em geral.

migração incremental

Uma estratégia de substituição na qual você migra a aplicação em pequenas partes, em vez de realizar uma única substituição completa. Por exemplo, é possível mover inicialmente apenas alguns microsserviços ou usuários para o novo sistema. Depois de verificar se tudo está funcionando corretamente, mova os microsserviços ou usuários adicionais de forma incremental até poder descomissionar seu sistema herdado. Essa estratégia reduz os riscos associados a migrações de grande porte.

Indústria 4.0

Um termo que foi introduzido por Klaus Schwab em 2016 para se referir à modernização dos processos de fabricação por meio de avanços em conectividade, dados em tempo real, automação, análise e IA/ML.

infraestrutura

Todos os recursos e ativos contidos no ambiente de uma aplicação.

Infraestrutura como código (IaC)

O processo de provisionamento e gerenciamento da infraestrutura de uma aplicação por meio de um conjunto de arquivos de configuração. A IaC foi projetada para ajudar você a centralizar

o gerenciamento da infraestrutura, padronizar recursos e escalar rapidamente para que novos ambientes sejam reproduzíveis, confiáveis e consistentes.

Internet das Coisas Industrial (IIoT)

O uso de sensores e dispositivos conectados à Internet nos setores industriais, como manufatura, energia, automotivo, saúde, ciências biológicas e agricultura. Para obter mais informações, consulte Construir uma estratégia de transformação digital para a Internet das Coisas Industrial (IIoT).

VPC de inspeção

Em uma arquitetura de AWS várias contas, uma VPC centralizada que gerencia as inspeções do tráfego de rede entre VPCs (na mesma ou em diferentes Regiões da AWS), a Internet e as redes locais. A Arquitetura de referência de segurança da AWS recomenda configurar sua conta de rede com VPCs de entrada, saída e inspeção para proteger a interface bidirecional entre a aplicação e a Internet em geral.

Internet das Coisas (IoT)

A rede de objetos físicos conectados com sensores ou processadores incorporados que se comunicam com outros dispositivos e sistemas pela Internet ou por uma rede de comunicação local. Para obter mais informações, consulte O que é IoT?

interpretabilidade

Uma característica de um modelo de machine learning que descreve o grau em que um ser humano pode entender como as previsões do modelo dependem de suas entradas. Para obter mais informações, consulte Interpretabilidade do modelo de machine learning com a AWS.

IoT

Consulte Internet das Coisas.

Biblioteca de informações de TI (ITIL)

Um conjunto de práticas recomendadas para fornecer serviços de TI e alinhar esses serviços a requisitos de negócios. A ITIL fornece a base para o ITSM.

Gerenciamento de serviços de TI (ITSM)

Atividades associadas a design, implementação, gerenciamento e suporte de serviços de TI para uma organização. Para obter informações sobre a integração de operações em nuvem com ferramentas de ITSM, consulte o guia de integração de operações.

ITIL

Consulte a biblioteca de informações de TI.

ITSM

Veja o gerenciamento de serviços de TI.

L

controle de acesso baseado em etiqueta (LBAC)

Uma implementação do controle de acesso obrigatório (MAC) em que os usuários e os dados em si recebem explicitamente um valor de etiqueta de segurança. A interseção entre a etiqueta de segurança do usuário e a etiqueta de segurança dos dados determina quais linhas e colunas podem ser vistas pelo usuário.

zona de pouso

Uma landing zone é um AWS ambiente bem arquitetado, com várias contas, escalável e seguro. Um ponto a partir do qual suas organizações podem iniciar e implantar rapidamente workloads e aplicações com confiança em seu ambiente de segurança e infraestrutura. Para obter mais informações sobre zonas de pouso, consulte Configurar um ambiente da AWS com várias contas seguro e escalável.

migração de grande porte

Uma migração de 300 servidores ou mais.

LBAC

Veja controle de acesso baseado em etiquetas.

privilégio mínimo

A prática recomendada de segurança de conceder as permissões mínimas necessárias para executar uma tarefa. Para obter mais informações, consulte Aplicar permissões de privilégios mínimos na documentação do IAM.

mover sem alterações (lift-and-shift)

Veja 7 Rs.

sistema little-endian

Um sistema que armazena o byte menos significativo antes. Veja também endianness.

ambientes inferiores

Veja o ambiente.

M

machine learning (ML)

Um tipo de inteligência artificial que usa algoritmos e técnicas para reconhecimento e aprendizado de padrões. O ML analisa e aprende com dados gravados, por exemplo, dados da Internet das Coisas (IoT), para gerar um modelo estatístico baseado em padrões. Para obter mais informações, consulte Machine learning.

ramificação principal

Veja a filial.

malware

Software projetado para comprometer a segurança ou a privacidade do computador. O malware pode interromper os sistemas do computador, vazar informações confidenciais ou obter acesso não autorizado. Exemplos de malware incluem vírus, worms, ransomware, cavalos de Tróia, spyware e keyloggers.

serviços gerenciados

Serviços da AWS para o qual AWS opera a camada de infraestrutura, o sistema operacional e as plataformas, e você acessa os endpoints para armazenar e recuperar dados. O Amazon Simple Storage Service (Amazon S3) e o Amazon DynamoDB são exemplos de serviços gerenciados. Eles também são conhecidos como serviços abstratos.

sistema de execução de manufatura (MES)

Um sistema de software para rastrear, monitorar, documentar e controlar processos de produção que convertem matérias-primas em produtos acabados no chão de fábrica.

MAP

Consulte Migration Acceleration Program.

mecanismo

Um processo completo no qual você cria uma ferramenta, impulsiona a adoção da ferramenta e, em seguida, inspeciona os resultados para fazer ajustes. Um mecanismo é um ciclo que se reforça e se aprimora à medida que opera. Para obter mais informações, consulte Construindo mecanismos no AWS Well-Architected Framework.

conta-membro

Todos, Contas da AWS exceto a conta de gerenciamento, que fazem parte de uma organização em AWS Organizations. Uma conta só pode ser membro de uma organização de cada vez.

MES

Veja o sistema de execução de manufatura.

Transporte de telemetria de enfileiramento de mensagens (MQTT)

Um protocolo de comunicação leve machine-to-machine (M2M), baseado no padrão de publicação/assinatura, para dispositivos de IoT com recursos limitados.

microsserviço

Um serviço pequeno e independente que se comunica por meio de APIs bem definidas e normalmente pertence a equipes pequenas e autônomas. Por exemplo, um sistema de seguradora pode incluir microsserviços que mapeiam as capacidades comerciais, como vendas ou marketing, ou subdomínios, como compras, reclamações ou análises. Os benefícios dos microsserviços incluem agilidade, escalabilidade flexível, fácil implantação, código reutilizável e resiliência. Para obter mais informações, consulte Integração de microsserviços usando serviços sem AWS servidor.

arquitetura de microsserviços

Uma abordagem à criação de aplicações com componentes independentes que executam cada processo de aplicação como um microsserviço. Esses microsserviços se comunicam por meio de uma interface bem definida usando APIs leves. Cada microsserviço nessa arquitetura pode ser atualizado, implantado e escalado para atender à demanda por funções específicas de uma aplicação. Para obter mais informações, consulte Implementação de microsserviços em. AWS

Programa de Aceleração da Migração (MAP)

Um AWS programa que fornece suporte de consultoria, treinamento e serviços para ajudar as organizações a criar uma base operacional sólida para migrar para a nuvem e ajudar a

compensar o custo inicial das migrações. O MAP inclui uma metodologia de migração para executar migrações legadas de forma metódica e um conjunto de ferramentas para automatizar e acelerar cenários comuns de migração.

migração em escala

O processo de mover a maior parte do portfólio de aplicações para a nuvem em ondas, com mais aplicações sendo movidas em um ritmo mais rápido a cada onda. Essa fase usa as práticas recomendadas e lições aprendidas nas fases anteriores para implementar uma fábrica de migração de equipes, ferramentas e processos para agilizar a migração de workloads por meio de automação e entrega ágeis. Esta é a terceira fase da estratégia de migração para a AWS.

fábrica de migração

Equipes multifuncionais que simplificam a migração de workloads por meio de abordagens automatizadas e ágeis. As equipes da fábrica de migração geralmente incluem operações, analistas e proprietários de negócios, engenheiros de migração, desenvolvedores e DevOps profissionais que trabalham em sprints. Entre 20 e 50% de um portfólio de aplicações corporativas consiste em padrões repetidos que podem ser otimizados por meio de uma abordagem de fábrica. Para obter mais informações, consulte discussão sobre fábricas de migração e o guia do Cloud Migration Factory neste conjunto de conteúdo.

metadados de migração

As informações sobre a aplicação e o servidor necessárias para concluir a migração. Cada padrão de migração exige um conjunto de metadados de migração diferente. Exemplos de metadados de migração incluem a sub-rede, o grupo de segurança e AWS a conta de destino.

padrão de migração

Uma tarefa de migração repetível que detalha a estratégia de migração, o destino da migração e a aplicação ou o serviço de migração usado. Exemplo: rehospede a migração para o Amazon EC2 AWS com o Application Migration Service.

Avaliação de Portfólio para Migração (MPA)

Uma ferramenta on-line que fornece informações para validar o caso de negócios para migrar para o. Nuvem AWS O MPA fornece avaliação detalhada do portfólio (dimensionamento correto do servidor, preços, comparações de TCO, análise de custos de migração), bem como planejamento de migração (análise e coleta de dados de aplicações, agrupamento de aplicações, priorização de migração e planejamento de ondas). A ferramenta MPA (requer login) está disponível gratuitamente para todos os AWS consultores e consultores parceiros da APN.

Avaliação de Preparação para Migração (MRA)

O processo de obter insights sobre o status de prontidão de uma organização para a nuvem, identificar pontos fortes e fracos e criar um plano de ação para fechar as lacunas identificadas, usando o CAF. AWS Para mais informações, consulte o guia de preparação para migração. A MRA é a primeira fase da estratégia de migração para a AWS.

estratégia de migração

A abordagem usada para migrar uma carga de trabalho para o. Nuvem AWS Para obter mais informações, consulte a entrada de 7 Rs neste glossário e consulte Mobilize sua organização para acelerar migrações em grande escala.

ML

Veja o aprendizado de máquina.

modernização

Transformar uma aplicação desatualizada (herdada ou monolítica) e sua infraestrutura em um sistema ágil, elástico e altamente disponível na nuvem para reduzir custos, ganhar eficiência e aproveitar as inovações. Para obter mais informações, consulte Estratégia para modernizar aplicativos no Nuvem AWS.

avaliação de preparação para modernização

Uma avaliação que ajuda a determinar a preparação para modernização das aplicações de uma organização. Ela identifica benefícios, riscos e dependências e determina o quão bem a organização pode acomodar o estado futuro dessas aplicações. O resultado da avaliação é um esquema da arquitetura de destino, um roteiro que detalha as fases de desenvolvimento e os marcos do processo de modernização e um plano de ação para abordar as lacunas identificadas. Para obter mais informações, consulte Avaliação da prontidão para modernização de aplicativos no. Nuvem AWS

aplicações monolíticas (monólitos)

Aplicações que são executadas como um único serviço com processos fortemente acoplados. As aplicações monolíticas apresentam várias desvantagens. Se um recurso da aplicação apresentar um aumento na demanda, toda a arquitetura deverá ser escalada. Adicionar ou melhorar os recursos de uma aplicação monolítica também se torna mais complexo quando a base de código cresce. Para resolver esses problemas, é possível criar uma arquitetura de microsserviços. Para obter mais informações, consulte Decompor monólitos em microsserviços.

MAPA

Consulte Avaliação do portfólio de migração.

MQTT

Consulte Transporte de telemetria de enfileiramento de mensagens.

classificação multiclasse

Um processo que ajuda a gerar previsões para várias classes (prevendo um ou mais de dois resultados). Por exemplo, um modelo de ML pode perguntar “Este produto é um livro, um carro ou um telefone?” ou “Qual categoria de produtos é mais interessante para este cliente?”

infraestrutura mutável

Um modelo que atualiza e modifica a infraestrutura existente para cargas de trabalho de produção. Para melhorar a consistência, confiabilidade e previsibilidade, o AWS Well-Architected Framework recomenda o uso de infraestrutura imutável como uma prática recomendada.

O

OAC

Veja o controle de acesso de origem.

CARVALHO

Veja a identidade de acesso de origem.

OCM

Veja o gerenciamento de mudanças organizacionais.

migração offline

Um método de migração no qual a workload de origem é desativada durante o processo de migração. Esse método envolve tempo de inatividade prolongado e geralmente é usado para workloads pequenas e não críticas.

OI

Veja a integração de operações.

OLA

Veja o contrato em nível operacional.

migração online

Um método de migração no qual a workload de origem é copiada para o sistema de destino sem ser colocada offline. As aplicações conectadas à workload podem continuar funcionando durante a migração. Esse método envolve um tempo de inatividade nulo ou mínimo e normalmente é usado para workloads essenciais para a produção.

OPC-UA

Consulte Comunicação de processo aberto — Arquitetura unificada.

Comunicação de processo aberto — Arquitetura unificada (OPC-UA)

Um protocolo de comunicação machine-to-machine (M2M) para automação industrial. O OPC-UA fornece um padrão de interoperabilidade com esquemas de criptografia, autenticação e autorização de dados.

acordo de nível operacional (OLA)

Um acordo que esclarece o que os grupos funcionais de TI prometem oferecer uns aos outros para apoiar um acordo de serviço (SLA).

análise de prontidão operacional (ORR)

Uma lista de verificação de perguntas e melhores práticas associadas que ajudam você a entender, avaliar, prevenir ou reduzir o escopo de incidentes e possíveis falhas. Para obter mais informações, consulte Operational Readiness Reviews (ORR) no Well-Architected AWS Framework.

tecnologia operacional (OT)

Sistemas de hardware e software que funcionam com o ambiente físico para controlar operações, equipamentos e infraestrutura industriais. Na manufatura, a integração dos sistemas OT e de tecnologia da informação (TI) é o foco principal das transformações da Indústria 4.0.

integração de operações (OI)

O processo de modernização das operações na nuvem, que envolve planejamento de preparação, automação e integração. Para obter mais informações, consulte o guia de integração de operações.

trilha organizacional

Uma trilha criada por ela AWS CloudTrail registra todos os eventos de todos Contas da AWS em uma organização em AWS Organizations. Essa trilha é criada em cada Conta da AWS que faz parte da organização e monitora a atividade em cada conta. Para obter mais informações, consulte Criação de uma trilha para uma organização na CloudTrail documentação.

gerenciamento de alterações organizacionais (OCM)

Uma estrutura para gerenciar grandes transformações de negócios disruptivas de uma perspectiva de pessoas, cultura e liderança. O OCM ajuda as organizações a se prepararem e fazerem a transição para novos sistemas e estratégias, acelerando a adoção de alterações, abordando questões de transição e promovendo mudanças culturais e organizacionais. Na estratégia de AWS migração, essa estrutura é chamada de aceleração de pessoas, devido à velocidade de mudança exigida nos projetos de adoção da nuvem. Para obter mais informações, consulte o guia do OCM.

controle de acesso de origem (OAC)

Em CloudFront, uma opção aprimorada para restringir o acesso para proteger seu conteúdo do Amazon Simple Storage Service (Amazon S3). O OAC oferece suporte a todos os buckets S3 Regiões da AWS, criptografia do lado do servidor com AWS KMS (SSE-KMS) e solicitações dinâmicas ao bucket S3. `PUT` `DELETE`

Identidade do acesso de origem (OAI)

Em CloudFront, uma opção para restringir o acesso para proteger seu conteúdo do Amazon S3. Quando você usa o OAI, CloudFront cria um principal com o qual o Amazon S3 pode se autenticar. Os diretores autenticados podem acessar o conteúdo em um bucket do S3 somente por meio de uma distribuição específica. CloudFront Veja também OAC, que fornece um controle de acesso mais granular e aprimorado.

OU

Veja a análise de prontidão operacional.

NÃO

Veja a tecnologia operacional.

VPC de saída (egresso)

Em uma arquitetura de AWS várias contas, uma VPC que gerencia conexões de rede que são iniciadas de dentro de um aplicativo. A Arquitetura de referência de segurança da AWS

recomenda configurar sua conta de rede com VPCs de entrada, saída e inspeção para proteger a interface bidirecional entre a aplicação e a Internet em geral.

P

limite de permissões

Uma política de gerenciamento do IAM anexada a entidades principais do IAM para definir as permissões máximas que o usuário ou perfil podem ter. Para obter mais informações, consulte Limites de permissões na documentação do IAM.

Informações de identificação pessoal (PII)

Informações que, quando visualizadas diretamente ou combinadas com outros dados relacionados, podem ser usadas para inferir razoavelmente a identidade de um indivíduo. Exemplos de PII incluem nomes, endereços e informações de contato.

PII

Veja informações de identificação pessoal.

manual

Um conjunto de etapas predefinidas que capturam o trabalho associado às migrações, como a entrega das principais funções operacionais na nuvem. Um manual pode assumir a forma de scripts, runbooks automatizados ou um resumo dos processos ou etapas necessários para operar seu ambiente modernizado.

PLC

Consulte controlador lógico programável.

AMEIXA

Veja o gerenciamento do ciclo de vida do produto.

política

Um objeto que pode definir permissões (consulte a política baseada em identidade), especificar as condições de acesso (consulte a política baseada em recursos) ou definir as permissões máximas para todas as contas em uma organização em AWS Organizations (consulte a política de controle de serviços).

persistência poliglota

Escolher de forma independente a tecnologia de armazenamento de dados de um microsserviço com base em padrões de acesso a dados e outros requisitos. Se seus microsserviços tiverem a mesma tecnologia de armazenamento de dados, eles poderão enfrentar desafios de implementação ou apresentar baixa performance. Os microsserviços serão implementados com mais facilidade e alcançarão performance e escalabilidade melhores se usarem o armazenamento de dados mais bem adaptado às suas necessidades. Para obter mais informações, consulte Habilitar a persistência de dados em microsserviços.

avaliação do portfólio

Um processo de descobrir, analisar e priorizar o portfólio de aplicações para planejar a migração. Para obter mais informações, consulte Avaliar a preparação para a migração.

predicado

Uma condição de consulta que retorna `true` ou`false`, normalmente localizada em uma `WHERE` cláusula.

pressão de predicados

Uma técnica de otimização de consulta de banco de dados que filtra os dados na consulta antes da transferência. Isso reduz a quantidade de dados que devem ser recuperados e processados do banco de dados relacional e melhora o desempenho das consultas.

controle preventivo

Um controle de segurança projetado para evitar que um evento ocorra. Esses controles são a primeira linha de defesa para ajudar a evitar acesso não autorizado ou alterações indesejadas em sua rede. Para obter mais informações, consulte Controles preventivos em Como implementar controles de segurança na AWS.

principal (entidade principal)

Uma entidade AWS que pode realizar ações e acessar recursos. Essa entidade geralmente é um usuário raiz para um Conta da AWS, uma função do IAM ou um usuário. Para obter mais informações, consulte Entidade principal em Termos e conceitos de perfis na documentação do IAM.

Privacidade por design

Uma abordagem em engenharia de sistemas que leva em consideração a privacidade em todo o processo de engenharia.

zonas hospedadas privadas

Um contêiner que armazena informações sobre como você quer que o Amazon Route 53 responda a consultas ao DNS para um domínio e seus subdomínios dentro de uma ou mais VPCs. Para obter mais informações, consulte Como trabalhar com zonas hospedadas privadas na documentação do Route 53.

controle proativo

Um controle de segurança projetado para impedir a implantação de recursos não compatíveis. Esses controles examinam os recursos antes de serem provisionados. Se o recurso não estiver em conformidade com o controle, ele não será provisionado. Para obter mais informações, consulte o guia de referência de controles na AWS Control Tower documentação e consulte Controles proativos em Implementação de controles de segurança em AWS.

gerenciamento do ciclo de vida do produto (PLM)

O gerenciamento de dados e processos de um produto em todo o seu ciclo de vida, desde o design, desenvolvimento e lançamento, passando pelo crescimento e maturidade, até o declínio e a remoção.

ambiente de produção

Veja o ambiente.

controlador lógico programável (PLC)

Na fabricação, um computador altamente confiável e adaptável que monitora as máquinas e automatiza os processos de fabricação.

pseudonimização

O processo de substituir identificadores pessoais em um conjunto de dados por valores de espaço reservado. A pseudonimização pode ajudar a proteger a privacidade pessoal. Os dados pseudonimizados ainda são considerados dados pessoais.

publicar/assinar (pub/sub)

Um padrão que permite comunicações assíncronas entre microsserviços para melhorar a escalabilidade e a capacidade de resposta. Por exemplo, em um MES baseado em microsserviços, um microsserviço pode publicar mensagens de eventos em um canal no qual outros microsserviços possam se inscrever. O sistema pode adicionar novos microsserviços sem alterar o serviço de publicação.

Q

plano de consulta

Uma série de etapas, como instruções, usadas para acessar os dados em um sistema de banco de dados relacional SQL.

regressão de planos de consultas

Quando um otimizador de serviço de banco de dados escolhe um plano menos adequado do que escolhia antes de uma determinada alteração no ambiente de banco de dados ocorrer. Isso pode ser causado por alterações em estatísticas, restrições, configurações do ambiente, associações de parâmetros de consulta e atualizações do mecanismo de banco de dados.

R

Matriz RACI

Veja responsável, responsável, consultado, informado (RACI).

ransomware

Um software mal-intencionado desenvolvido para bloquear o acesso a um sistema ou dados de computador até que um pagamento seja feito.

Matriz RASCI

Veja responsável, responsável, consultado, informado (RACI).

RCAC

Veja o controle de acesso por linha e coluna.

réplica de leitura

Uma cópia de um banco de dados usada somente para leitura. É possível encaminhar consultas para a réplica de leitura e reduzir a carga no banco de dados principal.

rearquiteta

Veja 7 Rs.

objetivo de ponto de recuperação (RPO).

O máximo período de tempo aceitável desde o último ponto de recuperação de dados. Isso determina o que é considerado uma perda aceitável de dados entre o último ponto de recuperação e a interrupção do serviço.

objetivo de tempo de recuperação (RTO)

O máximo atraso aceitável entre a interrupção e a restauração do serviço.

refatorar

Veja 7 Rs.

Região

Uma coleção de AWS recursos em uma área geográfica. Cada um Região da AWS é isolado e independente dos outros para fornecer tolerância a falhas, estabilidade e resiliência. Para obter mais informações, consulte Especificar o que Regiões da AWS sua conta pode usar.

regressão

Uma técnica de ML que prevê um valor numérico. Por exemplo, para resolver o problema de “Por qual preço esta casa será vendida?” um modelo de ML pode usar um modelo de regressão linear para prever o preço de venda de uma casa com base em fatos conhecidos sobre a casa (por exemplo, a metragem quadrada).

redefinir a hospedagem

Veja 7 Rs.

versão

Em um processo de implantação, o ato de promover mudanças em um ambiente de produção.

realocar

Veja 7 Rs.

redefinir a plataforma

Veja 7 Rs.

recomprar

Veja 7 Rs.

resiliência

A capacidade de um aplicativo de resistir ou se recuperar de interrupções. Alta disponibilidade e recuperação de desastres são considerações comuns ao planejar a resiliência no. Nuvem AWS Para obter mais informações, consulte Nuvem AWS Resiliência.

política baseada em recurso

Uma política associada a um recurso, como um bucket do Amazon S3, um endpoint ou uma chave de criptografia. Esse tipo de política especifica quais entidades principais têm acesso permitido, ações válidas e quaisquer outras condições que devem ser atendidas.

matriz responsável, accountable, consultada, informada (RACI)

Uma matriz que define as funções e responsabilidades de todas as partes envolvidas nas atividades de migração e nas operações de nuvem. O nome da matriz é derivado dos tipos de responsabilidade definidos na matriz: responsável (R), responsabilizável (A), consultado (C) e informado (I). O tipo de suporte (S) é opcional. Se você incluir suporte, a matriz será chamada de matriz RASCI e, se excluir, será chamada de matriz RACI.

controle responsivo

Um controle de segurança desenvolvido para conduzir a remediação de eventos adversos ou desvios em relação à linha de base de segurança. Para obter mais informações, consulte Controles responsivos em Como implementar controles de segurança na AWS.

reter

Veja 7 Rs.

aposentar-se

Veja 7 Rs.

rotação

O processo de atualizar periodicamente um segredo para dificultar o acesso das credenciais por um invasor.

controle de acesso por linha e coluna (RCAC)

O uso de expressões SQL básicas e flexíveis que tenham regras de acesso definidas. O RCAC consiste em permissões de linha e máscaras de coluna.

RPO

Veja o objetivo do ponto de recuperação.

RTO

Veja o objetivo do tempo de recuperação.

runbook

Um conjunto de procedimentos manuais ou automatizados necessários para realizar uma tarefa específica. Eles são normalmente criados para agilizar operações ou procedimentos repetitivos com altas taxas de erro.

S

SAML 2.0

Um padrão aberto que muitos provedores de identidade (IdPs) usam. Esse recurso permite o login único federado (SSO), para que os usuários possam fazer login AWS Management Console ou chamar as operações da AWS API sem que você precise criar um usuário no IAM para todos em sua organização. Para obter mais informações sobre a federação baseada em SAML 2.0, consulte Sobre a federação baseada em SAML 2.0 na documentação do IAM.

SCADA

Veja controle de supervisão e aquisição de dados.

SCP

Veja a política de controle de serviços.

secret

Em AWS Secrets Manager, informações confidenciais ou restritas, como uma senha ou credenciais de usuário, que você armazena de forma criptografada. Ele consiste no valor secreto e em seus metadados. O valor secreto pode ser binário, uma única string ou várias strings. Para obter mais informações, consulte O que há em um segredo do Secrets Manager? na documentação do Secrets Manager.

controle de segurança

Uma barreira de proteção técnica ou administrativa que impede, detecta ou reduz a capacidade de uma ameaça explorar uma vulnerabilidade de segurança. Existem quatro tipos principais de controles de segurança: preventivos, detectivos, responsivos e proativos.

fortalecimento da segurança

O processo de reduzir a superfície de ataque para torná-la mais resistente a ataques. Isso pode incluir ações como remover recursos que não são mais necessários, implementar a prática recomendada de segurança de conceder privilégios mínimos ou desativar recursos desnecessários em arquivos de configuração.

sistema de gerenciamento de eventos e informações de segurança (SIEM)

Ferramentas e serviços que combinam sistemas de gerenciamento de informações de segurança (SIM) e gerenciamento de eventos de segurança (SEM). Um sistema SIEM coleta, monitora e analisa dados de servidores, redes, dispositivos e outras fontes para detectar ameaças e violações de segurança e gerar alertas.

automação de resposta de segurança

Uma ação predefinida e programada projetada para responder ou remediar automaticamente um evento de segurança. Essas automações servem como controles de segurança responsivos ou detectivos que ajudam você a implementar as melhores práticas AWS de segurança. Exemplos de ações de resposta automatizada incluem a modificação de um grupo de segurança da VPC, a correção de uma instância do Amazon EC2 ou a rotação de credenciais.

Criptografia do lado do servidor

Criptografia dos dados em seu destino, por AWS service (Serviço da AWS) quem os recebe.

política de controle de serviços (SCP)

Uma política que fornece controle centralizado sobre as permissões de todas as contas em uma organização no AWS Organizations. As SCPs definem barreiras de proteção ou estabelecem limites para as ações que um administrador pode delegar a usuários ou perfis. É possível usar SCPs como listas de permissão ou de negação para especificar quais serviços ou ações são permitidos ou proibidos. Para obter mais informações, consulte Políticas de controle de serviço na AWS Organizations documentação.

service endpoint (endpoint de serviço)

O URL do ponto de entrada para um AWS service (Serviço da AWS). Você pode usar o endpoint para se conectar programaticamente ao serviço de destino. Para obter mais informações, consulte Endpoints do AWS service (Serviço da AWS) na Referência geral da AWS.

acordo de serviço (SLA)

Um acordo que esclarece o que uma equipe de TI promete fornecer aos clientes, como tempo de atividade e performance do serviço.

indicador de nível de serviço (SLI)

Uma medida de um aspecto de desempenho de um serviço, como taxa de erro, disponibilidade ou taxa de transferência.

objetivo de nível de serviço (SLO)

Uma métrica alvo que representa a integridade de um serviço, conforme medida por um indicador de nível de serviço.

modelo de responsabilidade compartilhada

Um modelo que descreve a responsabilidade com a qual você compartilha AWS pela segurança e conformidade na nuvem. AWS é responsável pela segurança da nuvem, enquanto você é responsável pela segurança na nuvem. Para obter mais informações, consulte o Modelo de responsabilidade compartilhada.

SIEM

Veja informações de segurança e sistema de gerenciamento de eventos.

ponto único de falha (SPOF)

Uma falha em um único componente crítico de um aplicativo que pode interromper o sistema.

SLA

Veja o contrato de nível de serviço.

ESGUIO

Veja o indicador de nível de serviço.

SLO

Veja o objetivo do nível de serviço.

split-and-seed modelo

Um padrão para escalar e acelerar projetos de modernização. À medida que novos recursos e lançamentos de produtos são definidos, a equipe principal se divide para criar novas equipes

de produtos. Isso ajuda a escalar os recursos e os serviços da sua organização, melhora a produtividade do desenvolvedor e possibilita inovações rápidas. Para obter mais informações, consulte Abordagem em fases para modernizar aplicativos no. Nuvem AWS

CUSPE

Veja um único ponto de falha.

esquema de estrelas

Uma estrutura organizacional de banco de dados que usa uma grande tabela de fatos para armazenar dados transacionais ou medidos e usa uma ou mais tabelas dimensionais menores para armazenar atributos de dados. Essa estrutura foi projetada para uso em um data warehouse ou para fins de inteligência comercial.

padrão strangler fig

Uma abordagem à modernização de sistemas monolíticos que consiste em reescrever e substituir incrementalmente a funcionalidade do sistema até que o sistema herdado possa ser desativado. Esse padrão usa a analogia de uma videira que cresce e se torna uma árvore estabelecida e, eventualmente, supera e substitui sua hospedeira. O padrão foi apresentado por Martin Fowler como forma de gerenciar riscos ao reescrever sistemas monolíticos. Para ver um exemplo de como aplicar esse padrão, consulte Modernizar incrementalmente os serviços Web herdados do Microsoft ASP.NET (ASMX) usando contêineres e o Amazon API Gateway.

sub-rede

Um intervalo de endereços IP na VPC. Cada sub-rede fica alocada em uma única zona de disponibilidade.

controle de supervisão e aquisição de dados (SCADA)

Na manufatura, um sistema que usa hardware e software para monitorar ativos físicos e operações de produção.

symmetric encryption (criptografia simétrica)

Um algoritmo de criptografia que usa a mesma chave para criptografar e descriptografar dados.

testes sintéticos

Testar um sistema de forma que simule as interações do usuário para detectar possíveis problemas ou monitorar o desempenho. Você pode usar o Amazon CloudWatch Synthetics para criar esses testes.

T

tags

Pares de valores-chave que atuam como metadados para organizar seus recursos. AWS As tags podem ajudar você a gerenciar, identificar, organizar, pesquisar e filtrar recursos. Para obter mais informações, consulte Marcar seus recursos do AWS.

variável-alvo

O valor que você está tentando prever no ML supervisionado. Ela também é conhecida como variável de resultado. Por exemplo, em uma configuração de fabricação, a variável-alvo pode ser um defeito do produto.

lista de tarefas

Uma ferramenta usada para monitorar o progresso por meio de um runbook. Uma lista de tarefas contém uma visão geral do runbook e uma lista de tarefas gerais a serem concluídas. Para cada tarefa geral, ela inclui o tempo estimado necessário, o proprietário e o progresso.

ambiente de teste

Veja o ambiente.

treinamento

O processo de fornecer dados para que seu modelo de ML aprenda. Os dados de treinamento devem conter a resposta correta. O algoritmo de aprendizado descobre padrões nos dados de treinamento que mapeiam os atributos dos dados de entrada no destino (a resposta que você deseja prever). Ele gera um modelo de ML que captura esses padrões. Você pode usar o modelo de ML para obter previsões de novos dados cujo destino você não conhece.

gateway de trânsito

Um hub de trânsito de rede que pode ser usado para interconectar as VPCs e as redes on-premises. Para obter mais informações, consulte O que é um gateway de trânsito na AWS Transit Gateway documentação.

fluxo de trabalho baseado em troncos

Uma abordagem na qual os desenvolvedores criam e testam recursos localmente em uma ramificação de recursos e, em seguida, mesclam essas alterações na ramificação principal. A

ramificação principal é então criada para os ambientes de desenvolvimento, pré-produção e produção, sequencialmente.

Acesso confiável

Conceder permissões a um serviço que você especifica para realizar tarefas em sua organização AWS Organizations e em suas contas em seu nome. O serviço confiável cria um perfil vinculado ao serviço em cada conta, quando esse perfil é necessário, para realizar tarefas de gerenciamento para você. Para obter mais informações, consulte Usando AWS Organizations com outros AWS serviços na AWS Organizations documentação.

tuning (ajustar)

Alterar aspectos do processo de treinamento para melhorar a precisão do modelo de ML. Por exemplo, você pode treinar o modelo de ML gerando um conjunto de rótulos, adicionando rótulos e repetindo essas etapas várias vezes em configurações diferentes para otimizar o modelo.

equipe de duas pizzas

Uma pequena DevOps equipe que você pode alimentar com duas pizzas. Uma equipe de duas pizzas garante a melhor oportunidade possível de colaboração no desenvolvimento de software.

U

incerteza

Um conceito que se refere a informações imprecisas, incompletas ou desconhecidas que podem minar a confiabilidade dos modelos preditivos de ML. Há dois tipos de incertezas: a incerteza epistêmica é causada por dados limitados e incompletos, enquanto a incerteza aleatória é causada pelo ruído e pela aleatoriedade inerentes aos dados. Para obter mais informações, consulte o guia Como quantificar a incerteza em sistemas de aprendizado profundo.

tarefas indiferenciadas

Também conhecido como trabalho pesado, trabalho necessário para criar e operar um aplicativo, mas que não fornece valor direto ao usuário final nem oferece vantagem competitiva. Exemplos de tarefas indiferenciadas incluem aquisição, manutenção e planejamento de capacidade.

ambientes superiores

Veja o ambiente.

V

aspiração

Uma operação de manutenção de banco de dados que envolve limpeza após atualizações incrementais para recuperar armazenamento e melhorar a performance.

controle de versões

Processos e ferramentas que rastreiam mudanças, como alterações no código-fonte em um repositório.

emparelhamento de VPC

Uma conexão entre duas VPCs que permite rotear tráfego usando endereços IP privados. Para ter mais informações, consulte O que é emparelhamento de VPC? na documentação da Amazon VPC.

Vulnerabilidade

Uma falha de software ou hardware que compromete a segurança do sistema.

W

cache quente

Um cache de buffer que contém dados atuais e relevantes que são acessados com frequência. A instância do banco de dados pode ler do cache do buffer, o que é mais rápido do que ler da memória principal ou do disco.

dados mornos

Dados acessados raramente. Ao consultar esse tipo de dados, consultas moderadamente lentas geralmente são aceitáveis.

função de janela

Uma função SQL que executa um cálculo em um grupo de linhas que se relacionam de alguma forma com o registro atual. As funções de janela são úteis para processar tarefas, como calcular uma média móvel ou acessar o valor das linhas com base na posição relativa da linha atual.

workload

Uma coleção de códigos e recursos que geram valor empresarial, como uma aplicação voltada para o cliente ou um processo de back-end.

workstreams

Grupos funcionais em um projeto de migração que são responsáveis por um conjunto específico de tarefas. Cada workstream é independente, mas oferece suporte aos outros workstreams do projeto. Por exemplo, o workstream de portfólio é responsável por priorizar aplicações, planejar ondas e coletar metadados de migração. O workstream de portfólio entrega esses ativos ao workstream de migração, que então migra os servidores e as aplicações.

MINHOCA

Veja escrever uma vez, ler muitas.

WQF

Consulte o AWS Workload Qualification Framework.

escreva uma vez, leia muitas (WORM)

Um modelo de armazenamento que grava dados uma única vez e evita que os dados sejam excluídos ou modificados. Os usuários autorizados podem ler os dados quantas vezes forem necessárias, mas não podem alterá-los. Essa infraestrutura de armazenamento de dados é considerada imutável.

Z

exploração de dia zero

Um ataque, geralmente malware, que tira proveito de uma vulnerabilidade de dia zero.

vulnerabilidade de dia zero

Uma falha ou vulnerabilidade não mitigada em um sistema de produção. Os agentes de ameaças podem usar esse tipo de vulnerabilidade para atacar o sistema. Os desenvolvedores frequentemente ficam cientes da vulnerabilidade como resultado do ataque.

aplicação zumbi

Uma aplicação que tem um uso médio de CPU e memória inferior a 5%. Em um projeto de migração, é comum retirar essas aplicações.

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.