Melhores práticas para usar o AWS CDK in TypeScript para criar projetos de IaC

AWS Orientação prescritiva

AWS Orientação prescritiva: Melhores práticas para usar o AWS CDK in TypeScript para criar projetos de IaC

Table of Contents

Melhores práticas para usar o AWS CDK na criação de projetos TypeScript de IaC

Sandeep Gawande, Mason Cahill, Sandip Gangapadhyay, Siamak Heshmati e Rajneesh Tyagi, Amazon Web Services (AWS)

Fevereiro de 2024 (histórico do documento)

Este guia fornece recomendações e melhores práticas para usar o AWS Cloud Development Kit (AWS CDK)in TypeScript para criar e implantar projetos de infraestrutura como código (IaC) em grande escala. AWS CDK É uma estrutura para definir a infraestrutura de nuvem em código e provisionar essa infraestrutura por meio dela. AWS CloudFormation Se você não tem uma estrutura de projeto bem definida, criar e gerenciar uma AWS CDK base de código para projetos de grande escala pode ser um desafio. Para lidar com esses desafios, algumas organizações usam antipadrões para projetos de grande escala, mas esses padrões podem retardar seu projeto e criar outros problemas que afetam negativamente sua organização. Por exemplo, os antipadrões podem complicar e retardar a integração de desenvolvedores, a correção de bugs e a adoção de novos recursos.

Este guia fornece uma alternativa ao uso de antipadrões e mostra como organizar seu código para escalabilidade, testes e alinhamento com as práticas recomendadas de segurança. Este guia pode ser usado para melhorar a qualidade do código para seus projetos de IaC e maximizar a agilidade de seus negócios. Este guia é destinado a arquitetos, líderes técnicos, engenheiros de infraestrutura e qualquer outra função que busque criar um AWS CDK projeto bem arquitetado para projetos de grande escala.

Objetivos

Este guia pode ajudar você a alcançar os seguintes resultados comerciais específicos:

- Custos reduzidos — Você pode usar o AWS CDK para projetar seus próprios componentes reutilizáveis que atendam aos requisitos de segurança, conformidade e governança da sua organização. Você também pode compartilhar facilmente componentes em toda a sua organização a fim de iniciar rapidamente novos projetos que se alinhem às práticas recomendadas por padrão.
- Tempo de comercialização mais rápido — Aproveite os recursos familiares do AWS CDK para acelerar seu processo de desenvolvimento. Isso aumenta a reutilização para implantação e reduz os esforços de desenvolvimento.
- Maior produtividade do desenvolvedor — Os desenvolvedores podem usar linguagens de programação familiares para definir a infraestrutura. Isso ajuda os desenvolvedores a expressar e manter AWS os recursos. Isso pode levar ao aumento da eficiência e colaboração do desenvolvedor.

Práticas recomendadas

Esta seção fornece uma visão geral das seguintes práticas recomendadas:

- Organize o código para projetos de grande escala
- Desenvolva padrões reutilizáveis
- Crie ou estenda estruturas
- Siga as TypeScript melhores práticas
- Verifique se há vulnerabilidades de segurança e erros de formatação
- Desenvolva e refine a documentação
- Adote uma abordagem de desenvolvimento orientado por testes
- Use controle de versão e lançamento para estruturas
- Imponha o gerenciamento de versões da biblioteca

Organize o código para projetos de grande escala

Por que a organização do código é importante

É fundamental que AWS CDK projetos de grande escala tenham uma estrutura bem definida e de alta qualidade. À medida que um projeto cresce e seu número de recursos e estruturas compatíveis aumenta, a navegação pelo código se torna mais difícil. Essa dificuldade pode afetar a produtividade e retardar a integração do desenvolvedor.

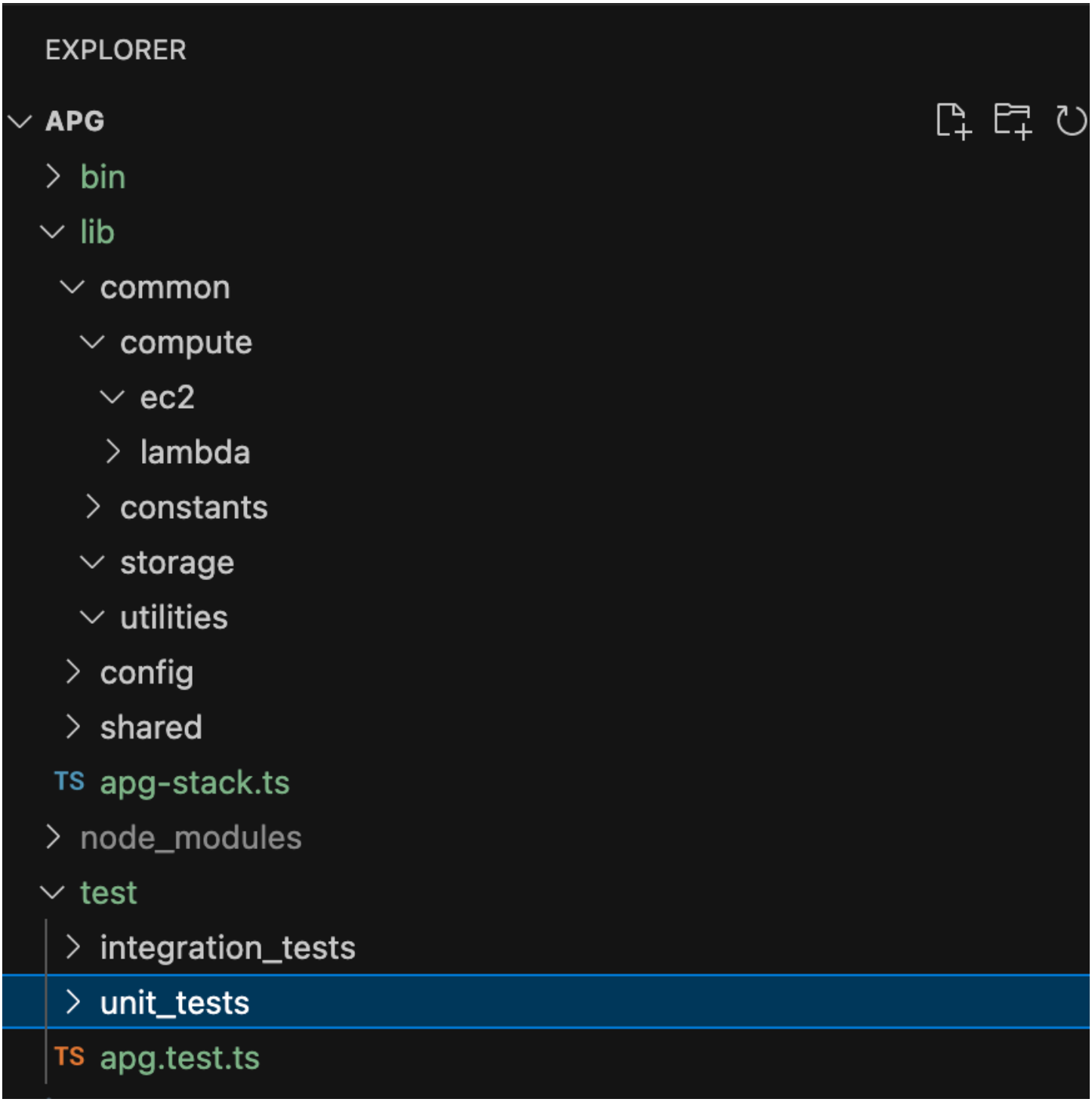
Como organizar seu código para escalar

Para alcançar um alto nível de flexibilidade e legibilidade do código, recomendamos dividir o código em partes lógicas com base na funcionalidade. Essa divisão reflete o fato de que a maioria das suas estruturas é usada em diferentes domínios de negócios. Por exemplo, seus aplicativos de front-end e back-end podem exigir uma AWS Lambda função e consumir o mesmo código-fonte. As fábricas podem criar objetos sem expor a lógica de criação ao cliente e usar uma interface comum para se referir aos objetos recém-criados. Você pode usar uma fábrica como um padrão eficaz para criar um comportamento consistente em sua base de código. Além disso, uma fábrica pode servir como uma única fonte confiável para ajudar a evitar códigos repetitivos e facilitar a solução de problemas.

Para entender melhor como as fábricas funcionam, considere o exemplo de um fabricante de automóveis. Um fabricante de automóveis não precisa ter o conhecimento e a infraestrutura necessários para fabricar pneus. Em vez disso, o fabricante do carro terceiriza essa experiência para um fabricante especializado de pneus e, em seguida, simplesmente encomenda os pneus desse fabricante conforme necessário. O mesmo princípio se aplica ao código. Por exemplo, é possível criar uma fábrica do Lambda capaz de criar funções do Lambda de alta qualidade e, em seguida, chamar a fábrica do Lambda em seu código sempre que precisar criar uma função do Lambda. Da mesma forma, é possível usar esse mesmo processo de terceirização para desacoplar sua aplicação e criar componentes modulares.

Exemplo de organização de código

O projeto de TypeScript amostra a seguir, conforme mostrado na imagem a seguir, inclui uma pasta comum na qual você pode manter todas as suas construções ou funcionalidades comuns.



Por exemplo, a pasta computar (localizada na pasta common) contém toda a lógica para diferentes estruturas de computação. Novos desenvolvedores podem adicionar facilmente novas estruturas de computação sem afetar os outros recursos. Todas as outras construções não precisarão criar novos recursos internamente. Em vez disso, essas estruturas simplesmente chamam a fábrica de estruturas comuns. Você pode organizar outras estruturas, como armazenamento, da mesma forma.

As configurações contêm dados baseados no ambiente que você deve desacoplar da pasta common em que você guarda a lógica. Recomendamos colocar seus dados de configuração comuns em uma pasta compartilhada. Também recomendamos usar a pasta utilities para servir todas as funções auxiliares e limpar scripts.

Desenvolva padrões reutilizáveis

Os padrões de design de software são soluções reutilizáveis para problemas comuns no desenvolvimento de software. Eles atuam como um guia ou paradigma para ajudar os engenheiros de software a criar produtos que sigam as práticas recomendadas. Esta seção fornece uma visão geral de dois padrões reutilizáveis que você pode usar em sua AWS CDK base de código: o padrão Abstract Factory e o padrão Chain of Responsibility. É possível usar cada padrão como um modelo e personalizá-lo para o problema de design específico no código. Para obter mais informações sobre padrões de design, consulte Padrões de design na documentação do Refactoring.Guru.

Abstract Factory

O padrão Abstract Factory fornece interfaces para a criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas. Esse padrão se aplica aos seguintes casos de uso:

- Quando o cliente é independente de como você cria e compõe os objetos no sistema
- Quando o sistema consiste em várias famílias de objetos e essas famílias são projetadas para serem usadas juntas
- Quando é necessário ter um valor em tempo de execução para construir uma dependência específica

Para obter mais informações sobre o padrão Abstract Factory, consulte Abstract Factory TypeScript na documentação do Refactoring.Guru.

O exemplo de código a seguir mostra como o padrão Abstract Factory pode ser usado para criar uma fábrica de armazenamento do Amazon Elastic Block Store (Amazon EBS).

```
abstract class EBSStorage {
    abstract initialize(): void;
}

class ProductEbs extends EBSStorage{
```

```
    constructor(value: String) {
        super();
        console.log(value);
    }
    initialize(): void {}
}

abstract class AbstractFactory {
    abstract createEbs(): EBSStorage
}

class EbsFactory extends AbstractFactory {
    createEbs(): ProductEbs{
        return new ProductEbs('EBS Created.')
    }
}

const ebs = new EbsFactory();
ebs.createEbs();
```

Chain of Responsibility

O Chain of Responsibility é um padrão de design comportamental que permite passar uma solicitação ao longo da cadeia de manipuladores em potencial até que um deles processe a solicitação. O padrão Chain of Responsibility se aplica aos seguintes casos de uso:

- Quando vários objetos determinados em tempo de execução são candidatos para lidar com uma solicitação
- Quando você não quiser especificar manipuladores explicitamente em seu código
- Quando você deseja emitir uma solicitação para um dos vários objetos sem especificar explicitamente o receptor

Para obter mais informações sobre o padrão da Cadeia de Responsabilidade, consulte Cadeia de Responsabilidade TypeScript na documentação do Refactoring.Guru.

O código a seguir mostra um exemplo de como o padrão Chain of Responsibility é usado para criar uma série de ações necessárias para concluir a tarefa.

```
interface Handler {
    setNext(handler: Handler): Handler;
```

```
    handle(request: string): string;
}
abstract class AbstractHandler implements Handler
{
    private nextHandler: Handler;
    public setNext(handler: Handler): Handler {
        this.nextHandler = handler;
        return handler;
    }

    public handle(request: string): string {
        if (this.nextHandler) {
            return this.nextHandler.handle(request);
        }
        return '';
    }
}

class KMSHandler extends AbstractHandler {
    public handle(request: string): string {
        return super.handle(request);
    }
}
```

Crie ou estenda estruturas

O que é uma estrutura

Uma construção é o alicerce básico de um AWS CDK aplicativo. Uma construção pode representar um único AWS recurso, como um bucket do Amazon Simple Storage Service (Amazon S3), ou pode ser uma abstração de nível superior que consiste em vários recursos relacionados. AWS Os componentes de uma estrutura podem incluir uma fila de trabalho com sua capacidade computacional associada ou um trabalho programado com recursos de monitoramento e um painel. AWS CDK Isso inclui uma coleção de construções chamada AWS Construct Library. A biblioteca contém construções para cada AWS service. Você pode usar o Construct Hub para descobrir construções adicionais AWS de terceiros e da comunidade de código aberto AWS CDK .

Quais são os diferentes tipos de estruturas

Há três tipos diferentes de construções para o AWS CDK:

- Construções L1 — As construções de camada 1, ou L1, são exatamente os recursos definidos por CloudFormation —nem mais, nem menos. Você mesmo deve fornecer os recursos necessários para a configuração. Essas construções L1 são muito básicas e devem ser configuradas manualmente. As construções L1 têm um `Cfn` prefixo e correspondem diretamente às especificações. CloudFormation Novos AWS services são suportados AWS CDK assim CloudFormation que houver suporte para esses serviços. CfnBucketé um bom exemplo de uma construção L1. Essa classe representa um bucket do S3 em que você deve configurar explicitamente todas as propriedades. Recomendamos que você use apenas uma construção L1 se não conseguir encontrar a construção L2 ou L3 para ela.
- Estruturas L2: as estruturas da camada 2, ou L2, têm código boilerplate e lógica de colagem comuns. Essas construções vêm com padrões convenientes e reduzem a quantidade de conhecimento que você precisa saber sobre elas. As construções L2 usam APIs baseadas em intenção para construir seus recursos e normalmente encapsulam seus módulos L1 correspondentes. Um bom exemplo de estrutura L2 é o Bucket. Essa classe cria um bucket S3 com propriedades e métodos padrão, como bucket. addLifeCycleRule (), que adiciona uma regra de ciclo de vida ao bucket.
- Estruturas L3: uma estrutura da camada 3, ou L3, é chamada de padrão. As construções L3 são projetadas para ajudá-lo a concluir tarefas comuns AWS, geralmente envolvendo vários tipos de recursos. Elas são ainda mais específicas e opinativas do que as estruturas L2 e servem a um caso de uso específico. Por exemplo, aws-ecs-patternso. ApplicationLoadBalancedFargateServiceconstruct representa uma arquitetura que inclui um cluster de AWS Fargate contêineres que usa um Application Load Balancer. Outro exemplo é o aws-apigateway. LambdaRestApiconstruir. Essa estrutura representa uma API do Amazon API Gateway que é apoiada por uma função do Lambda.

À medida que os níveis de estrutura aumentam, mais suposições são feitas sobre como essas estruturas serão usadas. Isso permite que você forneça interfaces com padrões mais eficazes para casos de uso altamente específicos.

Como criar sua própria estrutura

Para definir sua própria estrutura, é necessário seguir uma abordagem específica. Isso ocorre porque todas as estruturas estendem a classe `Construct`. A classe `Construct` é o alicerce da árvore de estruturas. As estruturas são implementadas em classes que estendem a classe base `Construct`. Todas as estruturas usam três parâmetros ao serem inicializadas:

- Escopo — O pai ou proprietário de uma construção, seja uma pilha ou outra construção, que determina seu lugar na árvore de construção. Em geral, é necessário passar `this` (ou `self` em Python), que representa o objeto atual, para o escopo.
- id: um identificador que deve ser exclusivo dentro desse escopo. O identificador serve como um namespace para tudo o que está definido na construção atual e é usado para alocar identidades exclusivas, como nomes de recursos e IDs lógicos. CloudFormation
- Props — Um conjunto de propriedades que definem a configuração inicial da construção.

O exemplo a seguir mostra como definir uma estrutura.

```
import { Construct } from 'constructs';

export interface CustomProps {
  // List all the properties
  Name: string;
}
export class MyConstruct extends Construct {
  constructor(scope: Construct, id: string, props: CustomProps) {
    super(scope, id);

    // TODO
  }
}
```

Criar ou estender uma estrutura L2

Uma construção L2 representa um “componente de nuvem” e encapsula tudo o que é CloudFormation necessário para criar o componente. Uma construção L2 pode conter um ou mais AWS recursos, e você mesmo pode personalizar a construção. A vantagem de criar ou estender uma construção L2 é que você pode reutilizar os componentes em CloudFormation pilhas sem redefinir o código. Você pode simplesmente importar a estrutura como uma classe.

Quando há uma relação “é a” com uma construção existente, você pode estender uma construção existente para adicionar recursos padrão adicionais. É uma prática recomendada reutilizar as propriedades da construção L2 existente. É possível sobrescrever propriedades modificando as propriedades diretamente no construtor.

O exemplo a seguir mostra como se alinhar às práticas recomendadas e estender uma estrutura L2 existente chamada `s3.Bucket`. A extensão estabelece propriedades padrão, como `versioned`,

`publicReadAccess`, `blockPublicAccess`, para garantir que todos os objetos (neste exemplo, buckets do S3) criados a partir dessa nova estrutura tenham sempre esses valores padrão definidos.

```
import * as s3 from 'aws-cdk-lib/aws-s3';
import { Construct } from 'constructs';
export class MySecureBucket extends s3.Bucket {
  constructor(scope: Construct, id: string, props?: s3.BucketProps) {

    super(scope, id, {
      ...props,
      versioned: true,
      publicReadAccess: false,
      blockPublicAccess: s3.BlockPublicAccess.BLOCK_ALL
    });
  }
}
```

Criar uma estrutura L3

A composição é a melhor escolha quando há uma relação de “tem uma” com uma composição de construção existente. Composição significa que você constrói sua própria estrutura em cima de outras estruturas existentes. Você pode criar seu próprio padrão para encapsular todos os recursos e seus valores padrão em uma única estrutura L3 de nível superior que pode ser compartilhada. A vantagem de criar suas próprias estruturas L3 (padrões) é a possibilidade de reutilizar os componentes em pilhas sem redefinir o código. Você pode simplesmente importar a estrutura como uma classe. Esses padrões são projetados para ajudar os consumidores a provisionar vários recursos com base em padrões comuns com uma quantidade limitada de conhecimento de forma concisa.

O exemplo de código a seguir cria uma AWS CDK construção chamada`ExampleConstruct`. É possível usar essa estrutura como modelo para definir seus componentes de nuvem.

```
import * as cdk from 'aws-cdk-lib';
import { Construct } from 'constructs';

export interface ExampleConstructProps {
  //insert properties you wish to expose
}

export class ExampleConstruct extends Construct {
```

```
  constructor(scope: Construct, id: string, props: ExampleConstructProps) {
    super(scope, id);
    //Insert the AWS components you wish to integrate
  }
}
```

O exemplo a seguir mostra como importar a construção recém-criada em seu AWS CDK aplicativo ou pilha.

```
import { ExampleConstruct } from './lib/construct-name';
```

O exemplo a seguir mostra como você é possível instanciar uma instância da estrutura que foi estendida da classe base.

```
import { ExampleConstruct } from './lib/construct-name';

new ExampleConstruct(this, 'newConstruct', {
  //insert props which you exposed in the interface `ExampleConstructProps`
});
```

Para obter mais informações, consulte AWS CDK Workshop na documentação do AWS CDK Workshop.

Escotilha de emergência

Você pode usar uma saída de emergência no AWS CDK para subir um nível de abstração para poder acessar o nível inferior das construções. As escotilhas de escape são usadas para estender a construção de recursos que não estão expostos na versão atual do AWS , mas estão disponíveis em CloudFormation.

Recomenda-se usar uma escotilha de emergência nos seguintes cenários:

- Um AWS service recurso está disponível por meio dele CloudFormation, mas não há `Construct` construções para ele.
- Um AWS service recurso está disponível CloudFormation e existem `Construct` construções para o serviço, mas elas ainda não expõem o recurso. Como `Construct` as construções são desenvolvidas “à mão”, às vezes elas podem ficar aquém das construções de CloudFormation recursos.

O código de exemplo a seguir mostra um caso de uso comum para o uso de uma escotilha de emergência. Neste exemplo, a funcionalidade que ainda não foi implementada na estrutura de nível superior destina-se a adicionar `httpPutResponseHopLimit` para escalonamento automático de `LaunchConfiguration`.

```
const launchConfig = autoscaling.onDemandASG.node.findChild("LaunchConfig") as
 CfnLaunchConfiguration;
            launchConfig.metadataOptions = {
                   httpPutResponseHopLimit: autoscalingConfig.httpPutResponseHopLimit||
 2
            }
```

O exemplo de código acima mostra o seguinte fluxo de trabalho:

1. Você define o seu `AutoScalingGroup` usando uma estrutura L2. A construção L2 não suporta a atualização do`httpPutResponseHopLimit`, então você deve usar uma escotilha de escape.
2. Você acessa a propriedade `node.defaultChild` na estrutura `AutoScalingGroup` L2 e a molda como o recurso `CfnLaunchConfiguration`.
3. Agora você pode definir a propriedade `launchConfig.metadataOptions` na `CfnLaunchConfiguration` L1.

Recursos personalizados

Você pode usar recursos personalizados para escrever uma lógica de provisionamento personalizada em modelos que são CloudFormation executados sempre que você cria, atualiza (se você alterou o recurso personalizado) ou exclui pilhas. Por exemplo, você pode usar um recurso personalizado se quiser incluir recursos que não estão disponíveis no AWS CDK. Dessa forma, você ainda pode gerenciar todos os recursos relacionados em uma única pilha.

A criação de um recurso personalizado envolve escrever uma função do Lambda que responde aos eventos de ciclo de vida CREATE, UPDATE e DELETE de um recurso. Se seu recurso personalizado precisar fazer apenas uma única chamada de API, considere usar a AwsCustomResourceconstrução. Isso possibilita realizar chamadas arbitrárias do SDK durante uma CloudFormation implantação. Caso contrário, sugerimos escrever sua própria função do Lambda para realizar o trabalho que precisa ser feito.

Para obter mais informações sobre recursos personalizados, consulte Recursos personalizados na CloudFormation documentação. Para ver um exemplo de como usar um recurso personalizado, consulte o repositório de recursos personalizados em GitHub.

O exemplo a seguir mostra como criar uma classe de recurso personalizada para iniciar uma função Lambda e CloudFormation enviar um sinal de sucesso ou falha.

```
import cdk = require('aws-cdk-lib');
import customResources = require('aws-cdk-lib/custom-resources');
import lambda = require('aws-cdk-lib/aws-lambda');
import { Construct } from 'constructs';

import fs = require('fs');

export interface MyCustomResourceProps {
  /**
   * Message to echo
   */
  message: string;
}

export class MyCustomResource extends Construct {
  public readonly response: string;

  constructor(scope: Construct, id: string, props: MyCustomResourceProps) {
    super(scope, id);

    const fn = new lambda.SingletonFunction(this, 'Singleton', {
      uuid: 'f7d4f730-4ee1-11e8-9c2d-fa7ae01bbebc',
      code: new lambda.InlineCode(fs.readFileSync('custom-resource-handler.py',
 { encoding: 'utf-8' })),
      handler: 'index.main',
      timeout: cdk.Duration.seconds(300),
      runtime: lambda.Runtime.PYTHON_3_6,
    });

    const provider = new customResources.Provider(this, 'Provider', {
      onEventHandler: fn,
    });

    const resource = new cdk.CustomResource(this, 'Resource', {
      serviceToken: provider.serviceToken,
      properties: props,
```

```
    });

    this.response = resource.getAtt('Response').toString();
    }
}
```

O exemplo a seguir mostra a lógica principal do recurso personalizado.

```
def main(event, context):
    import logging as log
    import cfnresponse
    log.getLogger().setLevel(log.INFO)

    # This needs to change if there are to be multiple resources in the same stack
    physical_id = 'TheOnlyCustomResource'

    try:
        log.info('Input event: %s', event)

        # Check if this is a Create and we're failing Creates
        if event['RequestType'] == 'Create' and
 event['ResourceProperties'].get('FailCreate', False):
            raise RuntimeError('Create failure requested')

        # Do the thing
        message = event['ResourceProperties']['Message']
        attributes = {
            'Response': 'You said "%s"' % message
        }

        cfnresponse.send(event, context, cfnresponse.SUCCESS, attributes, physical_id)
    except Exception as e:
        log.exception(e)
        # cfnresponse's error message is always "see CloudWatch"
        cfnresponse.send(event, context, cfnresponse.FAILED, {}, physical_id)
```

O exemplo a seguir mostra como a AWS CDK pilha chama o recurso personalizado.

```
import cdk = require('aws-cdk-lib');
import { MyCustomResource } from './my-custom-resource';

/**
```

```
 * A stack that sets up MyCustomResource and shows how to get an attribute from it
 */
class MyStack extends cdk.Stack {
  constructor(scope: cdk.App, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    const resource = new MyCustomResource(this, 'DemoResource', {
      message: 'CustomResource says hello',
    });

    // Publish the custom resource output
    new cdk.CfnOutput(this, 'ResponseMessage', {
      description: 'The message that came back from the Custom Resource',
      value: resource.response
    });
  }
}

const app = new cdk.App();
new MyStack(app, 'CustomResourceDemoStack');
app.synth();
```

Siga as TypeScript melhores práticas

TypeScript é uma linguagem que amplia os recursos do JavaScript. É uma linguagem fortemente digitada e orientada a objetos. Você pode usar TypeScript para especificar os tipos de dados que estão sendo transmitidos em seu código e tem a capacidade de relatar erros quando os tipos não coincidem. Esta seção fornece uma visão geral das TypeScript melhores práticas.

Descreva seus dados

Você pode usar TypeScript para descrever a forma dos objetos e funções em seu código. Usar o tipo `any` é equivalente a optar por não verificar o tipo de uma variável. Recomendamos evitar usar `any`em seu código. Aqui está um exemplo.

```
type Result = "success" | "failure"
function verifyResult(result: Result) {
    if (result === "success") {
        console.log("Passed");
    } else {
        console.log("Failed")
```

```
    }
}
```

Use enumerações

É possível usar enumerações para definir um conjunto de constantes nomeadas e definir padrões que podem ser reutilizados em sua base de código. Recomendamos exportar suas enumerações uma vez em nível global e depois permitir que outras classes importem e usem as enumerações. Suponha que você queira criar um conjunto de ações possíveis para capturar os eventos em sua base de código. TypeScript fornece enumerações numéricas e baseadas em seqüências de caracteres. O exemplo a seguir usa uma enumeração.

```
enum EventType {
    Create,
    Delete,
    Update
}

class InfraEvent {
    constructor(event: EventType) {
        if (event === EventType.Create) {
            // Call for other function
            console.log(`Event Captured :${event}`);
        }
    }
}

let eventSource: EventType = EventType.Create;
const eventExample = new InfraEvent(eventSource)
```

Use interfaces do

Uma interface é um contrato para a classe. Se você criar um contrato, os usuários deverão cumpri-lo. No exemplo a seguir, uma interface é usada para padronizar `props` e garantir que os chamadores forneçam o parâmetro esperado ao usar essa classe.

```
import { Stack, App } from "aws-cdk-lib";
import { Construct } from "constructs";

interface BucketProps {
    name: string;
```

```
    region: string;
    encryption: boolean;

}

class S3Bucket extends Stack {
    constructor(scope: Construct, props: BucketProps) {
        super(scope);
        console.log(props.name);

    }
}
const app = App();
const myS3Bucket = new S3Bucket(app, {
    name: "my-bucket",
    region: "us-east-1",
    encryption: false
})
```

Algumas propriedades só podem ser modificadas quando um objeto é criado pela primeira vez. É possível especificar isso colocando `readonly` antes do nome da propriedade, conforme mostrado no exemplo a seguir.

```
interface Position {
    readonly latitude: number;
    readonly longitute: number;
}
```

Estenda interfaces

A extensão de interfaces reduz a duplicação, pois não é necessário copiar as propriedades entre as interfaces. Além disso, o leitor do seu código pode entender facilmente os relacionamentos em sua aplicação.

```
 interface BaseInterface{
    name: string;
  }
  interface EncryptedVolume extends BaseInterface{
      keyName: string;
  }
  interface UnencryptedVolume extends BaseInterface {
```

```
      tags: string[];
  }
```

Evite interfaces vazias

Recomendamos evitar interfaces vazias devido aos riscos potenciais criados por elas. No exemplo a seguir, há uma interface vazia chamada`BucketProps`. Os objetos `myS3Bucket1` e `myS3Bucket2` são ambos válidos, mas seguem padrões diferentes porque a interface não impõe nenhum contrato. O código a seguir compilará e imprimirá as propriedades, mas isso introduz inconsistências na aplicação.

```
interface BucketProps {}

class S3Bucket implements BucketProps {
    constructor(props: BucketProps){
        console.log(props);
    }
}

const myS3Bucket1 = new S3Bucket({
    name: "my-bucket",
    region: "us-east-1",
    encryption: false,
});

const myS3Bucket2 = new S3Bucket({
    name: "my-bucket",
});
```

Use fábricas

Em um padrão Abstract Factory, uma interface é responsável por criar uma fábrica de objetos relacionados sem especificar explicitamente suas classes. Por exemplo, você pode criar uma fábrica do Lambda para criar funções do Lambda. Em vez de criar uma nova função Lambda em sua construção, você está delegando o processo de criação à fábrica. Para obter mais informações sobre esse padrão de design, consulte Abstract Factory TypeScript na documentação do Refactoring.Guru.

Use desestruturação em propriedades

A desestruturação, introduzida no ECMAScript 6 (ES6), é um JavaScript recurso que permite extrair vários dados de uma matriz ou objeto e atribuí-los às suas próprias variáveis.

```
const object = {
    objname: "obj",
    scope: "this",
};

const oName = object.objname;
const oScop = object.scope;

const { objname, scope } = object;
```

Defina convenções de nomenclatura padrão

A aplicação de uma convenção de nomenclatura mantém a base de código consistente e reduz a sobrecarga representada por pensar em como nomear uma variável. Recomendamos o seguinte:

- Use camelCase para nomes de variáveis e funções.
- Use PascalCase para nomes de classes e nomes de interface.
- Use camelCase para membros da interface.
- Use PascalCase para nomes de tipos e nomes de enumeração.
- Nomeie arquivos com camelCase (por exemplo, `ebsVolumes.tsx` ou `storage.tsb`)

Não use a palavra-chave var

A `let` instrução é usada para declarar uma variável local em TypeScript. É semelhante à `var` palavra-chave, mas tem algumas restrições no escopo em comparação com a `var` palavra-chave. Uma variável declarada em um bloco com `let` só está disponível para uso dentro desse bloco. A `var` palavra-chave não pode ter escopo de bloco, o que significa que ela pode ser acessada fora de um bloco específico (representado por`{}`), mas não fora da função em que está definida. Você pode redeclarar e atualizar `var` variáveis. É uma prática recomendada evitar o uso da `var` palavra-chave.

Considere usar ESLint e Prettier

O ESLint analisa estaticamente seu código para encontrar problemas rapidamente. O ESLint pode ser usado para criar uma série de afirmações (chamadas regras de lint) que definem como seu código deve parecer ou se comportar. O ESLint também tem sugestões de correção automática para ajudar você a melhorar seu código. Finalmente, você pode usar o ESLint para carregar regras de lint de plug-ins compartilhados.

O Prettier é um formatador de código conhecido compatível com toda uma variedade de linguagens de programação. O Prettier pode ser usado para definir seu estilo de código a fim de evitar sua formatação manual. Após a instalação, você pode atualizar seu arquivo `package.json` e executar os comandos `npm run format` e `npm run lint`.

O exemplo a seguir mostra como habilitar o ESLint e o formatador Prettier para seu projeto. AWS CDK

```
"scripts": {
    "build": "tsc",
    "watch": "tsc -w",
    "test": "jest",
    "cdk": "cdk",
    "lint": "eslint --ext .js,.ts .",
    "format": "prettier --ignore-path .gitignore --write '**/*.+(js|ts|json)'"
}
```

Use modificadores de acesso

O modificador privado em TypeScript limita a visibilidade somente para a mesma classe. Ao adicionar o modificador privado a uma propriedade ou método, é possível acessar essa propriedade ou método dentro da mesma classe.

O modificador público permite que propriedades e métodos de classe sejam acessíveis de todos os locais. Se você não especificar nenhum modificador de acesso para propriedades e métodos, eles usarão o modificador público por padrão.

O modificador protegido permite que propriedades e métodos de uma classe sejam acessíveis dentro da mesma classe e dentro de subclasses. Use o modificador protegido quando você espera criar subclasses em seu AWS CDK aplicativo.

Use tipos de utilitários

Os tipos de utilitários em TypeScript são funções de tipo predefinidas que realizam transformações e operações em tipos existentes. Isso ajuda você a criar novos tipos com base nos tipos existentes. Por exemplo, você pode alterar ou extrair propriedades, tornar as propriedades opcionais ou obrigatórias ou criar versões imutáveis de tipos. Ao usar tipos de utilitários, você pode definir tipos mais precisos e capturar possíveis erros em tempo de compilação.

Parcial <Type>

`Partial`marca todos os membros de um tipo de entrada `Type` como opcionais. Esse utilitário retorna um tipo que representa todos os subconjuntos de um determinado tipo. Veja a seguir um exemplo de `Partial`.

```
interface Dog {
  name: string;
  age: number;
  breed: string;
  weight: number;
}

let partialDog: Partial<Dog> = {};
```

Obrigatório <Type>

`Required`faz o oposto de`Partial`. Isso torna todos os membros de um tipo de entrada `Type` não opcionais (em outras palavras, obrigatórios). Veja a seguir um exemplo de `Required`.

```
interface Dog {
  name: string;
  age: number;
  breed: string;
  weight?: number;
}

let dog: Required<Dog> = {
  name: "scruffy",
  age: 5,
  breed: "labrador",
  weight 55 // "Required" forces weight to be defined
```

```
};
```

Verifique se há vulnerabilidades de segurança e erros de formatação

A infraestrutura como código (IaC) e a automação se tornaram essenciais para as empresas. Com a IaC sendo tão robusta, você tem uma grande responsabilidade de gerenciar os riscos de segurança. Os riscos comuns de segurança da IaC podem incluir:

- Privilégios superpermissivos AWS Identity and Access Management (IAM)
- Escopo dos grupos de segurança
- Recursos não criptografados
- Logs de acesso não ativados

Abordagens e ferramentas de segurança

Recomendamos implementar as seguintes abordagens de segurança:

- Detecção de vulnerabilidades no desenvolvimento: a correção de vulnerabilidades na produção é cara e demorada devido à complexidade de desenvolver e distribuir patches de software. Além disso, as vulnerabilidades na produção causam o risco de exploração. Recomendamos usar a varredura de código em seus recursos de IaC para que as vulnerabilidades possam ser detectadas e corrigidas antes da liberação para produção.
- Conformidade e remediação automática — AWS Config oferece regras AWS gerenciadas. Essas regras ajudam a impor a conformidade e permitem que você tente a remediação automática usando a automação.AWS Systems Manager Você também pode criar e associar documentos de automação personalizados usando AWS Config regras.

Ferramentas de desenvolvimento comuns

As ferramentas abordadas nesta seção ajudam você a estender suas funcionalidades integradas com suas próprias regras personalizadas. Recomendamos que você alinhe suas regras personalizadas com os padrões da sua organização. Alguns fatores comuns a considerar são:

- Use cfn-nag para identificar problemas de segurança da infraestrutura, como regras permissivas do IAM ou literais de senha, em modelos. CloudFormation Para obter mais informações, consulte o repositório GitHub cfn-nag da Stelligent.
- Use cdk-nag, inspirado no cfn-nag, para validar se as estruturas dentro de um determinado escopo estão em conformidade com um conjunto definido de regras. Você também pode usar o cdk-nag para suprimir regras e relatórios de conformidade. A ferramenta cdk-nag valida construções estendendo aspectos no. AWS CDK Para obter mais informações, consulte Gerenciar a segurança e a conformidade de aplicativos com o AWS Cloud Development Kit (AWS CDK) e cdk-nag no blog. AWS DevOps
- Use a ferramenta de código aberto Checkov para realizar análises estáticas em seu ambiente de IaC. O Checkov ajuda a identificar configurações incorretas na nuvem escaneando seu código de infraestrutura em Kubernetes, Terraform ou. CloudFormation Você pode usar o Checkov para obter saídas em diferentes formatos, incluindo JSON, JUnit XML ou CLI. O Checkov pode lidar com variáveis de forma eficaz criando um gráfico que mostra a dependência dinâmica do código. Para obter mais informações, consulte o repositório GitHub Checkov da Bridgecrew.
- Use o TFlint para verificar erros e sintaxe obsoleta e para obter ajuda para aplicar as práticas recomendadas. Observe que o TFlint pode não validar problemas específicos de provedor. Para obter mais informações sobre o TFlint, consulte o repositório GitHub TFlint do Terraform Linters.
- Use o Amazon Q Developer para realizar verificações de segurança. Quando usado em um ambiente de desenvolvimento integrado (IDE), o Amazon Q Developer fornece assistência de desenvolvimento de software com inteligência artificial. Ele pode conversar sobre código, fornecer preenchimentos de código em linha, gerar novos códigos, escanear seu código em busca de vulnerabilidades de segurança e fazer atualizações e melhorias no código.

Desenvolva e refine a documentação

A documentação é fundamental para o sucesso do seu projeto. A documentação não apenas explica como o código funciona, mas também ajuda os desenvolvedores a entender melhor os recursos e a funcionalidade das aplicações. Desenvolver e refinar documentação de alta qualidade pode fortalecer o processo de desenvolvimento de software, manter um software de alta qualidade e ajudar na transferência de conhecimento entre desenvolvedores.

Há duas categorias de documentação: documentação dentro do código e documentação de apoio sobre o código. A documentação dentro do código é fornecida na forma de comentários. A documentação de apoio sobre o código pode ser feita por arquivos README e documentos

externos. Não é incomum que os desenvolvedores pensem na documentação como uma sobrecarga, pois o código em si é fácil de entender. Isso pode ser verdade para projetos pequenos, mas a documentação é crucial para projetos de grande escala em que várias equipes estão envolvidas.

É uma prática recomendada para o autor do código escrever a documentação, pois ele tem um bom entendimento de suas funcionalidades. Os desenvolvedores podem enfrentar dificuldades com a sobrecarga adicional de manter uma documentação de suporte separada. Para superar esse desafio, os desenvolvedores podem adicionar os comentários no código e esses comentários podem ser extraídos automaticamente para que todas as versões do código e da documentação sejam sincronizadas.

Existem diversas ferramentas diferentes para ajudar os desenvolvedores a extrair comentários do código e gerar a documentação para ele. Este guia se concentra em TypeDoc ser a ferramenta preferida para AWS CDK construções.

Por que a documentação do código é necessária para AWS CDK construções

AWS CDK construções comuns são criadas por várias equipes em uma organização e compartilhadas entre diferentes equipes para consumo. Uma boa documentação ajuda os consumidores da biblioteca de estruturas a integrar facilmente estruturas e construir sua infraestrutura com o mínimo esforço. Manter todos os documentos sincronizados é uma tarefa árdua. Recomendamos que você mantenha o documento dentro do código, que será extraído usando a TypeDoc biblioteca.

Usando TypeDoc com a AWS Construct Library

TypeDoc é um gerador de documentos para TypeScript. Você pode usar TypeDoc para ler seus arquivos de TypeScript origem, analisar os comentários nesses arquivos e, em seguida, gerar um site estático que contém documentação para seu código.

O código a seguir mostra como fazer a integração TypeDoc com a AWS Construct Library e, em seguida, adicionar os seguintes pacotes ao seu `package.json` arquivo em`devDependencies`.

```
{

  "devDependencies": {
```

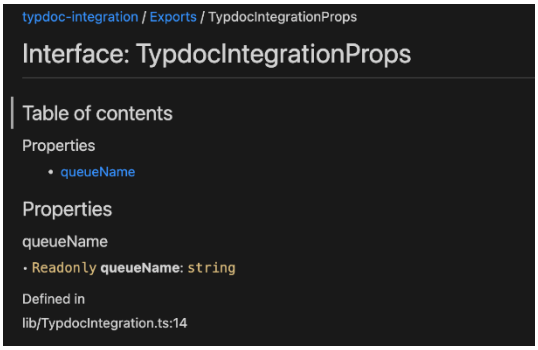
```
    "typedoc-plugin-markdown": "^3.11.7",
    "typescript": "~3.9.7"
  },

}
```

Para adicionar `typedoc.json` na pasta da biblioteca do CDK, use o código a seguir.

```
{
    "$schema": "https://typedoc.org/schema.json",
    "entryPoints": ["./lib"],
}
```

Para gerar os arquivos README, execute o `npx typedoc` comando no diretório raiz do projeto da biblioteca de AWS CDK construção.

O documento de amostra a seguir é gerado por TypeDoc.



Para obter mais informações sobre as opções de TypeDoc integração, consulte Comentários do documento na TypeDoc documentação.

Adote uma abordagem de desenvolvimento orientado por testes

Recomendamos que você siga uma abordagem de desenvolvimento orientado a testes (TDD) com o. AWS CDK O TDD é uma abordagem de desenvolvimento de software em que você desenvolve casos de teste para especificar e validar seu código. Em termos simples, primeiro você cria casos de teste para cada funcionalidade e, se o teste falhar, então você escreverá um novo código para passar no teste e tornar o código simples e livre de erros.

É possível usar o TDD para escrever o caso de teste primeiro. Isso ajuda a validar a infraestrutura com diferentes restrições de design em termos de aplicar a política de segurança para os recursos

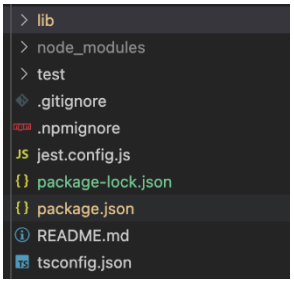
e seguir uma convenção de nomenclatura exclusiva para o projeto. A abordagem padrão para testar AWS CDK aplicativos é usar o módulo de AWS CDK asserções e estruturas de teste populares, como Jest para JavaScript e/ou pytest para TypeScript Python.

Há duas categorias de testes que você pode escrever para seus AWS CDK aplicativos:

- Use afirmações refinadas para testar um aspecto específico do CloudFormation modelo gerado, como “esse recurso tem essa propriedade com esse valor”. Esses testes podem detectar regressões e também são úteis quando você está desenvolvendo novos recursos usando o TDD (escreva um teste primeiro e depois faça ele ser aprovado escrevendo uma implementação correta). Afirmações minuciosas são os testes que você mais escreverá.
- Use testes instantâneos para testar o modelo sintetizado em relação a um CloudFormation modelo de linha de base armazenado anteriormente. Os testes de snapshots possibilitam refatorar livremente, pois você pode ter certeza de que o código refatorado funciona exatamente da mesma forma que o original. Se as alterações foram intencionais, é possível aceitar uma nova linha de base para futuros testes. No entanto, AWS CDK as atualizações também podem fazer com que os modelos sintetizados sejam alterados, portanto, você não pode confiar apenas em instantâneos para garantir que sua implementação esteja correta.

Teste unitário

Este guia se concentra TypeScript especificamente na integração de testes unitários. Para habilitar o teste, certifique-se de que o arquivo `package.json` tenha as seguintes bibliotecas: `@types/jest`, `jest` e `ts-jest` em `devDependencies`. Para adicionar esses pacotes, execute o comando `cdk init lib --language=typescript`. É possível ver a estrutura a seguir após executar o comando anterior.



O código a seguir é um exemplo de um `package.json` arquivo habilitado com a biblioteca Jest.

```
{
  ...
```

```
  "scripts": {
    "build": "npm run lint && tsc",
    "watch": "tsc -w",
    "test": "jest",
  },
  "devDependencies": {
    ...
    "@types/jest": "27.5.2",
    "jest": "27.5.1",
    "ts-jest": "27.1.5",
    ...
  }
}
```

Escreva o caso de teste na pasta Test. O exemplo a seguir mostra um caso de teste para uma AWS CodePipeline construção.

```
import {App,Stack} from 'aws-cdk-lib';
import { Template } from 'aws-cdk-lib/assertions';
import * as CodepipelineModule from '../lib/index';
import { Role, ServicePrincipal } from 'aws-cdk-lib/aws-iam';
import { Repository } from 'aws-cdk-lib/aws-codecommit';
import { PipelineProject } from 'aws-cdk-lib/aws-codebuild';

const testData:CodepipelineModule.CodepipelineModuleProps = {

    pipelineName: "validate-test-pipeline",
    serviceRoleARN: "",
    codeCommitRepositoryARN: "",
    branchName: "master",
    buildStages:[]
}


test('Code Pipeline Created', () => {
    const app = new App();
    const stack = new Stack(app, "TestStack");
    // WHEN
    const serviceRole = new Role(stack, "testRole", { assumedBy: new
 ServicePrincipal('codepipeline.amazonaws.com') })
    const codeCommit = new Repository(stack, "testRepo", {
        repositoryName: "validate-codeCommit-repo"
    });
```

```
    const codeBuildProject=new PipelineProject(stack,"TestCodeBuildProject",{});
    testData.serviceRoleARN = serviceRole.roleArn;
    testData.codeCommitRepositoryARN = codeCommit.repositoryArn;
    testData["buildStages"].push({
        stageName:"Deploy",
        codeBuildProject:codeBuildProject
    })
    new CodepipelineModule.CodepipelineModule(stack, 'MyTestConstruct', testData);
    // THEN
    const template = Template.fromStack(stack);

    template.hasResourceProperties('AWS::CodePipeline::Pipeline', {
        Name:testData.pipelineName
    });
});
```

Para executar um teste, execute o comando `npm run test` no projeto. A consulta retorna os resultados a seguir.

```
PASS  test/codepipeline-module.test.ts (5.972 s)
  # Code Pipeline Created (97 ms)
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        6.142 s, estimated 9 s
```

Para obter mais informações sobre casos de teste, consulte Construções de teste no Guia do AWS Cloud Development Kit (AWS CDK) desenvolvedor.

Teste de integração

Os testes de integração para AWS CDK construções também podem ser incluídos usando um `integ-tests` módulo. Um teste de integração deve ser definido como um AWS CDK aplicativo. Deve haver uma one-to-one relação entre um teste de integração e um AWS CDK aplicativo. Para obter mais informações, visite o integ-tests-alpha módulo na Referência AWS CDK da API.

Use controle de versão e lançamento para estruturas

Controle de versão para o AWS CDK

AWS CDK construções comuns podem ser criadas por várias equipes e compartilhadas em uma organização para consumo. Normalmente, os desenvolvedores lançam novos recursos ou correções de erros em suas AWS CDK construções comuns. Essas construções são usadas por AWS CDK aplicativos ou quaisquer outras AWS CDK construções existentes como parte de uma dependência. Por esse motivo, é crucial que os desenvolvedores atualizem e liberem suas estruturas com versões semânticas adequadas de forma independente. AWS CDK Aplicativos downstream ou outras AWS CDK construções podem atualizar sua dependência para usar a versão de construção recém-lançada AWS CDK .

Versionamento semântico (Semver) é um conjunto de regras, ou método, para fornecer números de software exclusivos ao software de computador. As versões são definidas da seguinte forma:

- Uma versão PRINCIPAL consiste em alterações de API incompatíveis ou uma alteração significativa.
- Uma versão SECUNDÁRIA consiste em funcionalidades adicionadas de maneira compatível com versões anteriores.
- Uma versão PATCH consiste em correções de bugs compatíveis com versões anteriores.

Para obter mais informações sobre controle de versão semântica, consulte Especificação de versão semântica (SemVer) na documentação de controle de versão semântica.

Repositório e embalagem para construções AWS CDK

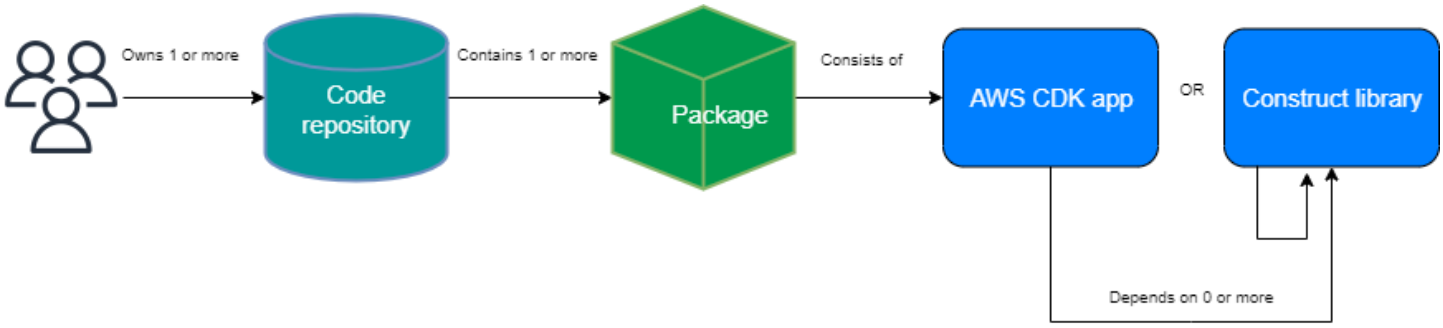
Como as AWS CDK construções são desenvolvidas por equipes diferentes e usadas por vários AWS CDK aplicativos, você pode usar um repositório separado para cada AWS CDK construção. Isso também pode ajudar na aplicação do controle de acesso. Cada repositório pode conter todo o código-fonte relacionado à mesma AWS CDK construção junto com todas as suas dependências. Ao manter um único aplicativo (ou seja, uma AWS CDK construção) em um único repositório, você pode diminuir o escopo do impacto das alterações durante a implantação.

Ele AWS CDK não apenas gera CloudFormation modelos para a implantação da infraestrutura, mas também agrupa ativos de tempo de execução, como funções Lambda e imagens do Docker, e os implanta junto com sua infraestrutura. Além de ser possível combinar o código que define

sua infraestrutura e o código que implementa sua lógica de tempo de execução em uma única construção, é uma prática recomendada. Esses dois tipos de código não precisam estar em repositórios separados ou mesmo em pacotes separados.

Para consumir pacotes além dos limites do repositório, você deve ter um repositório de pacotes privado, semelhante ao npm ou ao Maven Central PyPi, mas interno à sua organização. Também é necessário ter um processo de lançamento que compile, teste e publique o pacote no repositório de pacotes privado. Você pode criar repositórios privados, como PyPi servidores, usando uma máquina virtual (VM) local ou o Amazon S3. Ao projetar ou criar um registro de pacotes privado, é fundamental considerar o risco de interrupção do serviço devido a alta disponibilidade e escalabilidade. Um serviço gerenciado sem servidor hospedado na nuvem para armazenar pacotes pode diminuir consideravelmente a sobrecarga de manutenção. Por exemplo, você pode usar AWS CodeArtifactpara hospedar pacotes para as linguagens de programação mais populares. Você também pode usar CodeArtifact para definir conexões de repositórios externos e replicá-las internamente. CodeArtifact

As dependências dos pacotes no repositório de pacotes são gerenciadas pelo gerenciador de pacotes da sua linguagem (por exemplo, npm for TypeScript or JavaScript applications). Seu gerenciador de pacotes garante que as compilações sejam repetíveis gravando as versões específicas de cada pacote do qual a aplicação depende e, em seguida, permite que você atualize essas dependências de maneira controlada, conforme mostrado no diagrama a seguir.

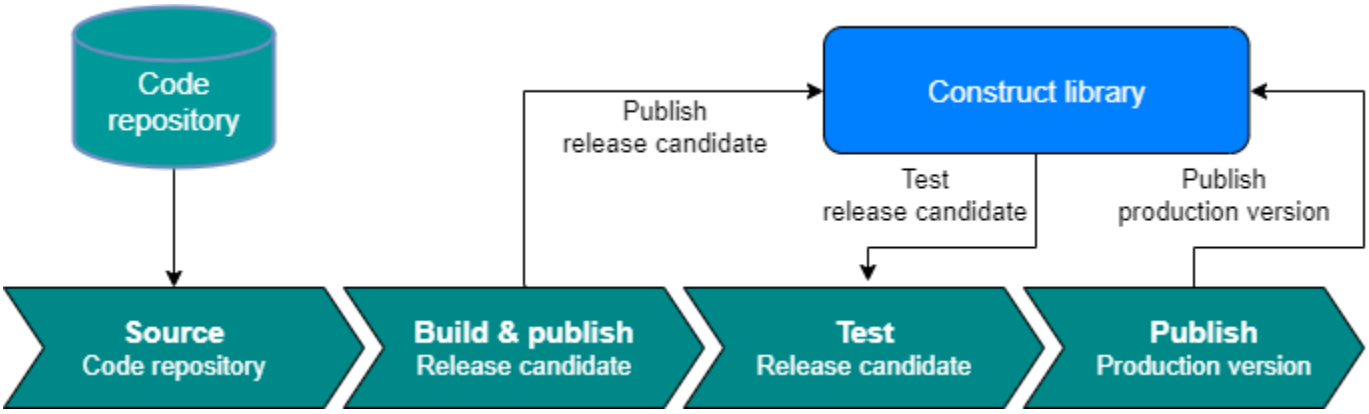


Construa a liberação para o AWS CDK

Recomendamos que você crie seu próprio pipeline automatizado para criar e lançar novas versões de AWS CDK construção. Se você implementar um processo adequado de aprovação por solicitações pull, depois de confirmar e enviar seu código-fonte para a ramificação principal do repositório, o pipeline poderá criar e construir uma versão candidata a lançamento. Essa versão pode ser enviada CodeArtifact e testada antes do lançamento da versão pronta para produção. Opcionalmente, você pode testar sua nova versão de AWS CDK construção localmente antes de

mesclar o código com a ramificação principal. Isso faz com que o pipeline lance a versão pronta para produção. Leve em consideração que estruturas e pacotes compartilhados devem ser testados independentemente da aplicação consumidora, como se estivessem sendo lançados para o público.

O diagrama a seguir mostra um exemplo de pipeline de lançamento de AWS CDK versão.



É possível usar os seguintes exemplos de comandos para compilar, testar e publicar pacotes npm. Primeiro, faça login no repositório de artefatos executando o comando a seguir.

```
aws codeartifact login --tool npm --domain <Domain Name> --domain-owner $(aws sts get-
caller-identity --output text --query 'Account')  \
--repository <Repository Name> --region <AWS Region Name>
```

Depois, execute as etapas a seguir:

1. Instale os pacotes necessários com base no arquivo `package.json`: `npm install`
2. Crie a versão candidata ao lançamento: `npm version prerelease --preid rc`
3. Compile o pacote npm:`npm run build`
4. Teste o pacote npm:`npm run test`
5. Publique o pacote npm:`npm publish`

Imponha o gerenciamento de versões da biblioteca

O gerenciamento do ciclo de vida é um desafio significativo quando você mantém bases de AWS CDK código. Por exemplo, suponha que você inicie um AWS CDK projeto com a versão 1.97 e, em seguida, a versão 1.169 fique disponível posteriormente. A versão 1.169 oferece novos recursos e correções de erros, mas você implantou sua infraestrutura usando a versão antiga. Agora, atualizar as estruturas se torna um desafio à medida que essa lacuna aumenta devido às alterações

significativas que podem ser introduzidas em novas versões. Isso poderá ser um desafio se houver muitos recursos em seu ambiente. O padrão apresentado nesta seção pode ajudá-lo a gerenciar a versão da sua AWS CDK biblioteca usando a automação. Aqui está o fluxo de trabalho desse padrão:

1. Quando você inicia um novo produto do CodeArtifact Service Catalog, as versões da AWS CDK biblioteca e suas dependências são armazenadas no `package.json` arquivo.
2. Você implanta um pipeline comum que monitora todos os repositórios para que você possa aplicar atualizações automáticas a eles se não houver alterações significativas.
3. Um AWS CodeBuild estágio verifica a árvore de dependências e procura as alterações significativas.
4. O pipeline cria uma ramificação de recursos e, em seguida, executa `cdk synth` com a nova versão para confirmar que não há erros.
5. A nova versão é implantada no ambiente de teste e, finalmente, executa um teste de integração para garantir que a implantação esteja íntegra.
6. É possível usar duas filas do Amazon Simple Queue Service (Amazon SQS) para acompanhar as pilhas. Os usuários podem revisar as pilhas manualmente na fila de exceções e resolver as alterações mais importantes. Os itens que forem aprovados no teste de integração terão permissão para ser mesclados e lançados.

Perguntas frequentes

Quais problemas podem ser TypeScript resolvidos?

Normalmente, você pode eliminar bugs em seu código escrevendo testes automatizados, verificando manualmente se o código funciona conforme o esperado e, finalmente, fazendo com que outra pessoa valide seu código. Validar as conexões entre cada parte de um projeto é demorado. Para acelerar o processo de validação, você pode usar uma linguagem de verificação de tipo, como TypeScript para automatizar a validação do código e fornecer feedback instantâneo durante o desenvolvimento.

Por que eu deveria usar TypeScript?

TypeScript é uma linguagem de código aberto que simplifica o JavaScript código, facilitando a leitura e a depuração. TypeScript também fornece ferramentas de desenvolvimento altamente produtivas para JavaScript IDEs e práticas, como verificação estática. Além disso, TypeScript oferece os benefícios do ECMAScript 6 (ES6) e pode aumentar sua produtividade. Por fim, TypeScript pode ajudá-lo a evitar bugs dolorosos que os desenvolvedores geralmente encontram ao escrever JavaScript , verificando o tipo do código.

Devo usar o AWS CDK ou CloudFormation?

Recomendamos que você use o AWS Cloud Development Kit (AWS CDK) em vez deAWS CloudFormation, se sua organização tiver a experiência em desenvolvimento para tirar proveito doAWS CDK. Isso ocorre porque AWS CDK é mais flexível do que CloudFormation, já que você pode usar uma linguagem de programação e conceitos de OOP. Lembre-se de que você pode usar CloudFormation para criar AWS recursos de maneira ordenada e previsível. Em CloudFormation, os recursos são gravados em arquivos de texto usando o formato JSON ou YAML.

E se o AWS CDK não oferecer suporte a um recém-lançadoAWS service?

Você pode usar uma substituição bruta ou um recurso CloudFormation personalizado.

Quais são as diferentes linguagens de programação suportadas peloAWS CDK?

AWS CDKgeralmente está disponível em JavaScript, Python TypeScript, Java, C# e Go (no Developer Preview).

Quanto AWS CDK custa?

Não há cobrança adicional para AWS CDK o. Você paga pelos AWS recursos (como instâncias do Amazon EC2 ou balanceadores de carga do Elastic Load Balancing) que são criados quando você os AWS CDK usa da mesma forma como se os tivesse criado manualmente. Você só paga pelo que usa à medida que usa. Não há taxas mínimas nem compromissos antecipados.

Próximas etapas

Recomendamos que você comece a construir com o AWS Cloud Development Kit (AWS CDK) in TypeScript. Para obter mais informações, consulte o Workshop do Dia AWS CDK de Imersão.

Recursos

Referências

- AWS Construções de AWS soluções (soluções)
- AWS Cloud Development Kit (AWS CDK) (GitHub)
- AWS Referência da API Construct Library (documentação de AWS CDK referência)
- AWS CDK Documentação de AWS CDK referência (documentação de referência)
- AWS CDK Workshop do Dia de Imersão (AWS Workshop Studio)

Ferramentas

- cdk-bag () GitHub
- TypeScript ESLint (documentação TypeScript ESLint)

Guias e padrões

- AWS Soluções e constrói padrões (AWS documentação)

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
Código de atualização	Atualizamos os exemplos de código na seção Siga as TypeScript melhores práticas.	16 de fevereiro de 2024
Adicionar seções	Adicionamos as seções Usar tipos de utilitários e Teste de integração.	10 de janeiro de 2024
Atualização secundária	Exemplo de código atualizado para criar uma estrutura L3.	16 de junho de 2023
Publicação inicial	—	08 de dezembro de 2022

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 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

AWS services 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 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. Você pode usar o endpoint para se conectar programaticamente ao serviço de destino. Para obter mais informações, consulte Endpoints do AWS service 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. Uma sub-rede deve residir 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.