Le migliori pratiche per l'utilizzo di AWS CDK in per TypeScript creare progetti IaC

AWS Guida prescrittiva

AWS Guida prescrittiva: Le migliori pratiche per l'utilizzo di AWS CDK in per TypeScript creare progetti IaC

Table of Contents

Le migliori pratiche per utilizzare il CDK AWS TypeScript per creare progetti IaC

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

Febbraio 2024 (cronologia dei documenti)

Questa guida fornisce consigli e best practice per l'utilizzo di AWS Cloud Development Kit (AWS CDK)in TypeScript per creare e implementare progetti Infrastructure as Code (IaC) su larga scala. AWS CDK È un framework per definire l'infrastruttura cloud in codice e fornire tale infrastruttura tramite. AWS CloudFormation Se non disponi di una struttura di progetto ben definita, creare e gestire una AWS CDK base di codice per progetti su larga scala può essere difficile. Per affrontare queste sfide, alcune organizzazioni utilizzano degli anti-pattern per i progetti su larga scala, ma tali pattern possono rallentare il progetto e creare altri problemi che hanno un impatto negativo sull'organizzazione. Ad esempio, gli anti-pattern possono complicare e rallentare l'onboarding degli sviluppatori, la correzione di bug e l'adozione di nuove funzionalità.

Questa guida fornisce un'alternativa all'uso degli anti-pattern e mostra come organizzare il codice a fini di scalabilità, esecuzione di test e allineamento con le best practice in materia di sicurezza. Puoi utilizzare questa guida per migliorare la qualità del codice per i tuoi progetti IaC e massimizzare l'agilità aziendale. Questa guida è destinata agli architetti, ai responsabili tecnici, agli ingegneri delle infrastrutture e a qualsiasi altro ruolo che cerchi di creare un progetto ben architettato AWS CDK per progetti su larga scala.

Obiettivi

Questa guida può aiutarti a raggiungere i seguenti obiettivi aziendali specifici:

- Costi ridotti: è possibile utilizzarli AWS CDK per progettare componenti riutilizzabili personalizzati che soddisfino i requisiti di sicurezza, conformità e governance dell'organizzazione. Puoi anche condividere facilmente i componenti all'interno dell'organizzazione, in modo da poter avviare rapidamente nuovi progetti allineati alle best practice per impostazione predefinita.
- Tempi di commercializzazione più rapidi: sfrutta le funzionalità familiari di AWS CDK per accelerare il processo di sviluppo. Ciò aumenta la riutilizzabilità per l'implementazione e riduce gli sforzi di sviluppo.
- Maggiore produttività degli sviluppatori: gli sviluppatori possono utilizzare linguaggi di programmazione familiari per definire l'infrastruttura. Questo aiuta gli sviluppatori a esprimere e gestire AWS le risorse. Ciò può portare a una maggiore efficienza e collaborazione degli sviluppatori.

Best practice

Questa sezione fornisce una panoramica delle seguenti best practice:

- Organizzazione del codice per progetti su larga scala
- Sviluppo di pattern riutilizzabili
- Creazione o estensione di costrutti
- Segui le TypeScript migliori pratiche
- Scansione per individuare vulnerabilità di sicurezza ed errori di formattazione
- Sviluppo e perfezionamento della documentazione
- Adozione di un approccio di sviluppo basato su test
- Uso del controllo di rilasci e versioni per i costrutti
- Applicazione della gestione delle versioni della libreria

Organizzazione del codice per progetti su larga scala

Perché l'organizzazione del codice è importante

Per i AWS CDK progetti su larga scala è fondamentale avere una struttura ben definita e di alta qualità. Man mano che un progetto si amplia e il numero di funzionalità e costrutti supportati aumenta, la navigazione all'interno del codice diventa più difficile. Tale difficoltà può influire sulla produttività e rallentare l'onboarding degli sviluppatori.

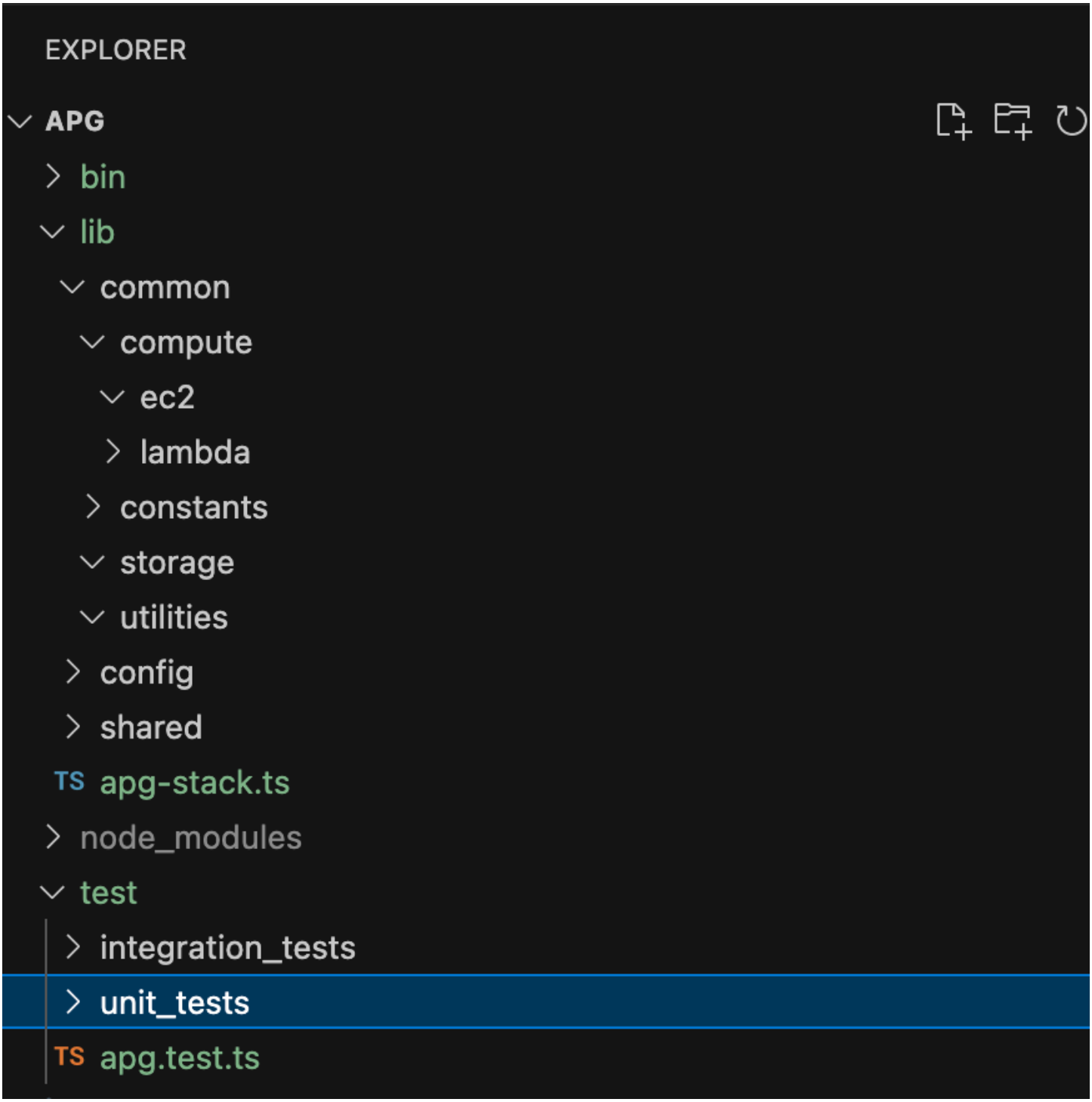
Come organizzare il codice in funzione della scalabilità

Per ottenere un livello elevato di flessibilità e leggibilità del codice, consigliamo di suddividerlo in parti logiche in base alla funzionalità. Tale divisione riflette il fatto che la maggior parte dei costrutti viene utilizzata in diversi domini aziendali. Ad esempio, sia le applicazioni frontend che quelle di backend potrebbero richiedere una AWS Lambda funzione e utilizzare lo stesso codice sorgente. Le factory possono creare oggetti senza esporre la logica di creazione al client e utilizzare un'interfaccia comune per fare riferimento agli oggetti appena creati. Puoi utilizzare una factory come pattern efficace per creare un comportamento coerente nella base di codice. Inoltre, una factory può fungere da unica fonte di verità per evitare che il codice sia ripetitivo e per agevolare la risoluzione dei problemi.

Per meglio comprendere il funzionamento delle factory, consideriamo l'esempio di un produttore di automobili. Un produttore di automobili non deve disporre delle conoscenze e delle infrastrutture necessarie per la produzione di pneumatici. Piuttosto, affida tali competenze a un produttore di pneumatici specializzato e poi si limita a ordinare gli pneumatici da quel produttore a seconda delle esigenze. Lo stesso principio si applica al codice. Ad esempio, puoi creare una factory Lambda in grado di sviluppare funzioni Lambda di alta qualità e quindi effettuare una chiamata alla factory Lambda nel codice ogni volta che devi creare una funzione Lambda. Analogamente, puoi utilizzare il medesimo processo di esternalizzazione per disaccoppiare l'applicazione e realizzare componenti modulari.

Esempio di organizzazione del codice

Il seguente progetto di TypeScript esempio, come mostrato nell'immagine seguente, include una cartella comune in cui è possibile conservare tutti i costrutti o le funzionalità comuni.



Ad esempio, la cartella compute (che si trova nella cartella common) contiene tutta la logica per diversi costrutti di calcolo. I nuovi sviluppatori possono aggiungere facilmente nuovi costrutti di calcolo senza condizionare le altre risorse. Tutti gli altri costrutti non avranno bisogno di creare nuove risorse internamente. Invece, a costrutti è sufficiente effettuare una chiamata alla factory di costrutti comune. Con la medesima procedura è possibile organizzare altri costrutti, come l'archiviazione.

Le configurazioni contengono dati basati sull'ambiente che devi disaccoppiare dalla cartella common in cui conservi la logica. Consigliamo di posizionare i dati config comuni in una cartella condivisa. Consigliamo inoltre di utilizzare la cartella utilities per svolgere tutte le funzioni helper e riordinare gli script.

Sviluppo di pattern riutilizzabili

I pattern di progettazione software sono soluzioni riutilizzabili a problemi comuni nello sviluppo di software. Fungono da guida o paradigma per aiutare gli ingegneri di software a creare prodotti che seguano le best practice. Questa sezione fornisce una panoramica di due modelli riutilizzabili che è possibile utilizzare nella propria AWS CDK codebase: il pattern Abstract Factory e il pattern Chain of Responsibility. Puoi utilizzare ciascun pattern come schema e personalizzarlo per il problema di progettazione specifico del tuo codice. Per ulteriori informazioni sui pattern di progettazione, consulta Pattern di progettazione nella documentazione di Refactoring.Guru.

Abstract Factory

Il pattern Abstract Factory fornisce interfacce per creare famiglie di oggetti correlati o dipendenti senza specificarne le classi concrete. Questo pattern è valido per i seguenti casi d'uso:

- Quando il client è indipendente dal modo in cui crei e componi gli oggetti nel sistema
- Quando il sistema è composto da più famiglie di oggetti progettate per essere utilizzate insieme
- Quando è necessario disporre di un valore di runtime per realizzare una particolare dipendenza

Per ulteriori informazioni sul pattern Abstract Factory, vedete Abstract Factory TypeScript nella documentazione di Refactoring.Guru.

Il seguente esempio di codice mostra come è possibile utilizzare il pattern Abstract Factory per creare una factory di archiviazione 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

Chain of Responsibility è un pattern di progettazione comportamentale che consente di inoltrare una richiesta lungo la catena di potenziali gestori finché uno di essi non la gestirà. Il pattern Chain of Responsibility è valido per i seguenti casi d'uso:

- Quando più oggetti, determinati durante il runtime, sono candidati alla gestione di una richiesta
- Quando non si desidera specificare i gestori in modo esplicito nel codice
- Quando desideri inviare una richiesta a uno di più oggetti senza specificare in modo esplicito il destinatario

Per ulteriori informazioni sul modello Chain of Responsibility, consulta Chain of Responsibility TypeScript nella documentazione di Refactoring.Guru.

Il codice seguente mostra un esempio di come il pattern Chain of Responsibility viene utilizzato per creare una serie di azioni necessarie per completare l'attività.

```
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);
    }
}
```

Creazione o estensione di costrutti

Che cos'è un costrutto

Un costrutto è l'elemento costitutivo di base di un'applicazione. AWS CDK Un costrutto può rappresentare una singola AWS risorsa, ad esempio un bucket Amazon Simple Storage Service (Amazon S3), oppure può essere un'astrazione di livello superiore composta da più risorse correlate. AWS I componenti di un costrutto possono includere una coda di lavoro con la capacità di elaborazione associata o un lavoro pianificato con risorse di monitoraggio e una dashboard. AWS CDK Include una raccolta di costrutti chiamata Construct Library. AWS La libreria contiene costrutti per tutti. Servizio AWS Puoi utilizzare Construct Hub per scoprire costrutti aggiuntivi forniti da AWS terze parti e dalla comunità open source. AWS CDK

Quali sono i diversi tipi di costrutti

Esistono tre diversi tipi di costrutti per: AWS CDK

- Costrutti L1: i costrutti di livello 1 o L1 sono esattamente le risorse definite da CloudFormation — né più né meno. È necessario fornire personalmente le risorse necessarie per la configurazione. Questi costrutti L1 sono molto semplici e devono essere configurati manualmente. I costrutti L1

hanno un `Cfn` prefisso e corrispondono direttamente alle specifiche. CloudFormation Servizi AWS I nuovi vengono supportati non AWS CDK appena CloudFormation viene fornito il supporto per questi servizi. CfnBucketè un buon esempio di costrutto L1. Questa classe rappresenta un bucket S3 in cui è necessario configurare esplicitamente tutte le proprietà. Ti consigliamo di utilizzare un costrutto L1 solo se non riesci a trovare il costrutto L2 o L3 corrispondente.

- Costrutti L2: i costrutti di livello 2, ovvero L2, presentano un codice boilerplate e una logica glue comuni. Questi costrutti sono dotati di comode impostazioni predefinite e riducono la quantità di conoscenze necessarie su di essi. I costrutti L2 utilizzano API basate sugli intenti per costruire le risorse e in genere incapsulano i moduli L1 corrispondenti. Un buon esempio di costrutto L2 è Bucket. Questa classe crea un bucket S3 con proprietà e metodi predefiniti come bucket.add Rule (), che aggiunge una regola del ciclo di vita al LifeCycle bucket.
- Costrutti L3: un costrutto di livello 3, ovvero L3, è detto pattern. I costrutti L3 sono progettati per aiutarti a completare attività comuni in, che spesso coinvolgono più tipi di risorse. AWS Si tratta di costrutti ancora più specifici e rigidi dei costrutti L2 e servono a un caso d'uso specifico. Ad esempio, aws-ecs-patterns. ApplicationLoadBalancedFargateIl costrutto di servizio rappresenta un'architettura che include un cluster di AWS Fargate contenitori che utilizza un Application Load Balancer. Un altro esempio è aws-apigateway. LambdaRestCostrutto Api. Questo rappresenta un'API Gateway Amazon API supportata da una funzione Lambda.

Man mano che i livelli dei costrutti aumentano, si formulano più ipotesi sull'uso che ne verrà fatto. Ciò consente di fornire interfacce con impostazioni predefinite più efficaci per casi d'uso altamente specifici.

Come creare il proprio costrutto

Per definire il proprio costrutto, è necessario seguire un approccio specifico. Questo perché tutti i costrutti estendono la classe `Construct`. La classe `Construct` è l'elemento base dell'albero dei costrutti. I costrutti sono implementati in classi che estendono la classe di base `Construct`. Tutti i costrutti accettano tre parametri al momento dell'inizializzazione:

- Ambito: il genitore o il proprietario di un costrutto, uno stack o un altro costrutto, che ne determina la posizione nell'albero dei costrutti. In genere devi superare `this` (o `self` in Python), che rappresenta l'oggetto corrente, per lo scope.
- id: un identificatore che deve essere univoco all'interno di tale scope. L'identificatore funge da namespace per tutto ciò che è definito all'interno del costrutto corrente e viene utilizzato per allocare identità univoche, ad esempio nomi di risorse e ID logici. CloudFormation

- Props: un insieme di proprietà che definiscono la configurazione iniziale del costrutto.

L'esempio seguente mostra come definire un costrutto.

```
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
  }
}
```

Creazione o estensione di un costrutto L2

Un costrutto L2 rappresenta un «componente cloud» e incapsula tutto ciò che CloudFormation deve avere per creare il componente. Un costrutto L2 può contenere una o più AWS risorse e sei libero di personalizzarlo tu stesso. Il vantaggio di creare o estendere un costrutto L2 è che potete riutilizzare i componenti negli stack senza ridefinire il codice. CloudFormation È possibile semplicemente importare il costrutto come classe.

Quando esiste una relazione «is a» con un costrutto esistente, è possibile estendere un costrutto esistente per aggiungere funzionalità predefinite aggiuntive. È consigliabile riutilizzare le proprietà del costrutto L2 esistente. È possibile sovrascrivere le proprietà modificandole direttamente nel costruttore.

L'esempio seguente mostra come allinearsi alle best practice ed estendere un costrutto L2 esistente denominato `s3.Bucket`. L'estensione stabilisce proprietà predefinite, come `versioned`, `publicReadAccess`, `blockPublicAccess`, per assicurarsi che tutti gli oggetti (in questo esempio, i bucket S3) creati a partire da questo nuovo costrutto presentino sempre questi valori predefiniti impostati.

```
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
    });
  }
}
```

Creazione di un costrutto L3

La composizione è la scelta migliore quando esiste una relazione «ha» con una composizione del costrutto esistente. Composizione significa che costruisci il tuo costrutto personale su altri costrutti esistenti. Puoi creare un pattern personalizzato per racchiudere tutte le risorse e i relativi valori predefiniti all'interno di un unico costrutto L3 di livello superiore che può essere condiviso. Il vantaggio di creare i propri costrutti (pattern) L3 è la possibilità di riutilizzare i componenti negli stack senza ridefinire il codice. È possibile semplicemente importare il costrutto come classe. Questi pattern sono progettati per aiutare gli utenti a effettuare in modo conciso il provisioning di più risorse in base a pattern comuni con una quantità limitata di conoscenze.

Il seguente esempio di codice crea un AWS CDK costrutto chiamato. `ExampleConstruct` È possibile utilizzare questo costrutto come modello per definire i propri componenti cloud.

```
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
  }
}
```

L'esempio seguente mostra come importare il costrutto appena creato nell' AWS CDK applicazione o nello stack.

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

L'esempio seguente mostra come creare un'istanza del costrutto esteso a partire dalla classe base.

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

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

Per ulteriori informazioni, consultate AWS CDK Workshop nella documentazione del AWS CDK Workshop.

Escape hatch

È possibile utilizzare una porta di fuga in AWS CDK per salire di un livello di astrazione in modo da poter accedere al livello inferiore dei costrutti. Le botole di fuga vengono utilizzate per estendere il costrutto a funzionalità che non sono esposte nella versione corrente di ma disponibili in. AWS CloudFormation

È consigliabile utilizzare un escape hatch nei seguenti scenari:

- Una Servizio AWS funzionalità è disponibile tramite CloudFormation, ma non ci sono `Construct` costrutti corrispondenti.
- Una Servizio AWS funzionalità è disponibile tramite CloudFormation e sono presenti `Construct` costrutti per il servizio, ma questi non espongono ancora la funzionalità. Poiché `Construct` i costrutti vengono sviluppati «a mano», a volte possono rimanere indietro rispetto ai costrutti delle risorse. CloudFormation

L'esempio di codice seguente mostra un caso d'uso comune per l'uso di un escape hatch. In questo esempio, la funzionalità che non è ancora implementata nel costrutto di livello superiore serve ad aggiungere `httpPutResponseHopLimit` per il dimensionamento automatico di `LaunchConfiguration`.

```
const launchConfig = autoscaling.onDemandASG.node.findChild("LaunchConfig") as
 CfnLaunchConfiguration;
            launchConfig.metadataOptions = {
```

```
                    httpPutResponseHopLimit: autoscalingConfig.httpPutResponseHopLimit||
2
            }
```

L'esempio di codice precedente illustra il seguente flusso di lavoro:

1. Definisci il `AutoScalingGroup` utilizzando un costrutto L2. Il costrutto L2 non supporta l'aggiornamento di`httpPutResponseHopLimit`, quindi è necessario utilizzare una porta di fuga.
2. Accedi alla proprietà `node.defaultChild` sul costrutto `AutoScalingGroup` L2 ed eseguine il casting come risorsa `CfnLaunchConfiguration`.
3. Ora puoi impostare la proprietà `launchConfig.metadataOptions` su `CfnLaunchConfiguration` L1.

Risorse personalizzate

È possibile utilizzare risorse personalizzate per scrivere una logica di provisioning personalizzata in modelli che CloudFormation viene eseguita ogni volta che si creano, aggiornano (se si modifica la risorsa personalizzata) o si eliminano gli stack. Ad esempio, è possibile utilizzare una risorsa personalizzata se si desidera includere risorse che non sono disponibili in. AWS CDK In questo modo puoi comunque gestire tutte le risorse correlate in un singolo stack.

La creazione di una risorsa personalizzata implica la scrittura di una funzione Lambda che risponda agli eventi del ciclo di vita CREATE, UPDATE e DELETE di una risorsa. Se la tua risorsa personalizzata deve effettuare solo una singola chiamata API, prendi in considerazione l'utilizzo del costrutto AwsCustomResource. In questo modo è possibile eseguire chiamate SDK arbitrarie durante una distribuzione. CloudFormation Altrimenti, consigliamo di scrivere una funzione Lambda personalizzata per eseguire il lavoro necessario.

Per ulteriori informazioni sulle risorse personalizzate, consulta Risorse personalizzate nella CloudFormation documentazione. Per un esempio di come utilizzare una risorsa personalizzata, consulta l'archivio Custom Resource su GitHub.

L'esempio seguente mostra come creare una classe di risorse personalizzata per avviare una funzione Lambda e CloudFormation inviare un segnale di successo o di fallimento.

```
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();
    }
}
```

L'esempio seguente mostra la logica principale della risorsa personalizzata.

```
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)
```

L'esempio seguente mostra come lo AWS CDK stack chiama la risorsa personalizzata.

```
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();
```

Segui le TypeScript migliori pratiche

TypeScript è un linguaggio che estende le funzionalità di JavaScript. È un linguaggio fortemente tipizzato e orientato agli oggetti. È possibile TypeScript utilizzarlo per specificare i tipi di dati che vengono trasmessi all'interno del codice e ha la possibilità di segnalare errori quando i tipi non corrispondono. Questa sezione fornisce una panoramica delle TypeScript migliori pratiche.

Descrizione dei dati

È possibile TypeScript utilizzarlo per descrivere la forma degli oggetti e delle funzioni nel codice. Utilizzare il tipo `any` equivale a disattivare il controllo del tipo per una variabile. È preferibile evitare l'uso di `any` nel codice. Ecco un esempio.

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

Uso di enumerazioni

Puoi utilizzare le enumerazioni per definire un insieme di costanti denominate e definire standard che possono essere riutilizzati nella base di codice. Consigliamo di esportare le enumerazioni una volta a livello globale e poi di consentire ad altre classi di importarle e utilizzarle. Supponiamo di voler creare una serie di azioni possibili per acquisire gli eventi nella base di codice. TypeScript

fornisce enumerazioni sia numeriche che basate su stringhe. Nell'esempio seguente viene utilizzata un'enumerazione.

```
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)
```

Uso di interfacce

Un'interfaccia è un contratto per la classe. Se crei un contratto, gli utenti sono tenuti a rispettarlo. Nell'esempio seguente, viene utilizzata un'interfaccia per standardizzare `props` e assicurare che i chiamanti forniscano il parametro previsto quando si utilizza questa 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
})
```

Alcune proprietà possono essere modificate solo al momento della creazione dell'oggetto. Puoi specificarlo inserendo `readonly` prima del nome della proprietà, come illustrato nell'esempio seguente.

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

Estensione delle interfacce

L'estensione delle interfacce riduce la duplicazione, perché non è necessario copiare le proprietà da un'interfaccia all'altra. Inoltre, il lettore del codice può comprendere facilmente le relazioni all'interno dell'applicazione.

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

Evitare le interfacce vuote

Consigliamo di evitare le interfacce vuote a causa dei potenziali rischi che creano. Nell'esempio seguente, c'è un'interfaccia vuota chiamata. `BucketProps` Gli oggetti `myS3Bucket1` e `myS3Bucket2` sono entrambi validi, ma seguono standard diversi perché l'interfaccia non applica

alcun contratto. Il codice seguente compilerà e stamperà le proprietà, ma in tal modo si introdurranno incoerenze nell'applicazione.

```
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",
});
```

Uso di factory

In un pattern Abstract Factory, un'interfaccia è responsabile della creazione di una factory di oggetti correlati senza specificarne esplicitamente le classi. Ad esempio, puoi creare una factory Lambda per la generazione di funzioni Lambda. Invece di creare una nuova funzione Lambda all'interno del tuo costrutto, deleghi il processo di creazione alla fabbrica. Per ulteriori informazioni su questo modello di progettazione, vedete Abstract Factory TypeScript nella documentazione di Refactoring.Guru.

Uso della destrutturazione sulle proprietà

La destrutturazione, introdotta in ECMAScript 6 (ES6), è una JavaScript funzionalità che consente di estrarre più parti di dati da un array o da un oggetto e assegnarle alle proprie variabili.

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

const oName = object.objname;
```

```
const oScop = object.scope;

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

Definizione di convenzioni di denominazione standard

L'applicazione di una convenzione di denominazione mantiene coerente la base di codice e riduce il sovraccarico quando si pensa al nome da assegnare a una variabile. Consigliamo quanto segue:

- Utilizza camelCase per i nomi di variabili e funzioni.
- Da utilizzare PascalCase per i nomi delle classi e dei nomi di interfaccia.
- Utilizza camelCase per i membri delle interfacce.
- Utilizzare PascalCase per i nomi dei tipi e i nomi enum.
- Assegna un nome ai file con camelCase (ad esempio `ebsVolumes.tsx` o `storage.tsb`)

Non utilizzare la parola chiave var

L'`let`istruzione viene utilizzata per dichiarare una variabile locale in TypeScript. È simile alla `var` parola chiave, ma presenta alcune restrizioni nell'ambito rispetto alla `var` parola chiave. Una variabile dichiarata in un blocco con `let` è disponibile per l'uso solo all'interno di quel blocco. La `var` parola chiave non può avere un ambito a blocchi, il che significa che è possibile accedervi al di fuori di un particolare blocco (rappresentato da`{}`) ma non al di fuori della funzione in cui è definita. È possibile dichiarare nuovamente e aggiornare le variabili. `var` È consigliabile evitare di utilizzare la `var` parola chiave.

Possibilità di utilizzare ESLint e Prettier

ESLint analizza staticamente il codice per individuare rapidamente i problemi. Puoi utilizzare ESLint per creare una serie di asserzioni (denominate regole lint) che definiscono come dovrebbe apparire o comportarsi il codice. ESLint offre anche suggerimenti di correzione automatica per aiutarti a migliorare il codice. Infine, ESLint consente di caricare regole lint dai plugin condivisi.

Prettier è un noto formattatore di codice che supporta una varietà di linguaggi di programmazione diversi. Puoi utilizzare Prettier per impostare lo stile del codice in modo da evitarne la formattazione manuale. Dopo l'installazione, puoi aggiornare il file `package.json` ed eseguire i comandi `npm run format` e `npm run lint`.

L'esempio seguente mostra come abilitare ESLint e il formattatore Prettier per il tuo progetto. 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)'"
}
```

Uso di modificatori di accesso

Il modificatore privato in TypeScript limita la visibilità solo alla stessa classe. Quando aggiungi il modificatore privato a una proprietà o a un metodo, puoi accedere a tale proprietà o metodo all'interno della stessa classe.

Il modificatore pubblico consente l'accesso alle proprietà e ai metodi delle classi da tutte le posizioni. Se non specifichi alcun modificatore di accesso per proprietà e metodi, utilizzeranno il modificatore pubblico per impostazione predefinita.

Il modificatore protetto consente l'accesso alle proprietà e ai metodi di una classe all'interno della stessa classe e delle sottoclassi. Utilizzate il modificatore protetto quando pretendete di creare sottoclassi nella vostra applicazione. AWS CDK

Usa tipi di utilità

I tipi di utilità in TypeScript sono funzioni di tipo predefinito che eseguono trasformazioni e operazioni su tipi esistenti. Ciò consente di creare nuovi tipi in base ai tipi esistenti. Ad esempio, è possibile modificare o estrarre proprietà, rendere le proprietà facoltative o obbligatorie o creare versioni immutabili dei tipi. Utilizzando i tipi di utilità, è possibile definire tipi più precisi e rilevare potenziali errori in fase di compilazione.

Parziale <Type>

`Partial`contrassegna tutti i membri di un tipo di input `Type` come facoltativi. Questa utilità restituisce un tipo che rappresenta tutti i sottoinsiemi di un determinato tipo. Di seguito è riportato un esempio di `Partial`.

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

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

Richiesto <Type>

`Required`fa il contrario di`Partial`. Rende tutti i membri di un tipo di input `Type` non opzionali (in altre parole, obbligatori). Di seguito è riportato un esempio di `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
};
```

Scansione per individuare vulnerabilità di sicurezza ed errori di formattazione

L'Infrastruttura come codice (IaC) e l'automazione sono diventate aspetti essenziali per le imprese. Data la robustezza di IaC, hai una grande responsabilità per quanto riguarda la gestione dei rischi di sicurezza. I rischi relativi alla sicurezza comuni di IaC possono includere quanto segue:

- Privilegi troppo permissivi AWS Identity and Access Management (IAM)
- Gruppi di sicurezza aperti
- Risorse non crittografate

- Log di accesso non attivati

Approcci e strumenti di sicurezza

Consigliamo di implementare i seguenti approcci di sicurezza:

- Rilevamento delle vulnerabilità in fase di sviluppo: la correzione delle vulnerabilità in fase di produzione è costosa e richiede molto tempo a causa della complessità dello sviluppo e della distribuzione delle patch software. Inoltre, le vulnerabilità in fase di produzione comportano il rischio di sfruttamento. Consigliamo di utilizzare la scansione del codice sulle risorse IaC in modo da rilevare e correggere le vulnerabilità prima del rilascio in produzione.
- Conformità e riparazione automatica: AWS Config offre regole AWS gestite. Queste regole ti aiutano a far rispettare la conformità e ti consentono di tentare la riparazione automatica utilizzando l'automazione.AWS Systems Manager È inoltre possibile creare e associare documenti di automazione personalizzati utilizzando le regole. AWS Config

Strumenti di sviluppo comuni

Gli strumenti descritti in questa sezione consentono di estendere le funzionalità integrate con regole personalizzate. Ti consigliamo di allineare le regole personalizzate agli standard della tua organizzazione. Ecco alcuni strumenti di sviluppo comuni da prendere in considerazione:

- Usa cfn-nag per identificare i problemi di sicurezza dell'infrastruttura, come le regole IAM permissive o i valori letterali delle password, nei modelli. CloudFormation Per ulteriori informazioni, consulta il repository cfn-nag di Stelligent. GitHub
- Utilizza cdk-nag, ispirato a cfn-nag, per verificare che i costrutti all'interno di un determinato scope siano conformi a un insieme di regole definito. Puoi anche utilizzare cdk-nag per la soppressione delle regole e la creazione di report sulla conformità. Lo strumento cdk-nag convalida i costrutti estendendo gli aspetti di. AWS CDK Per ulteriori informazioni, consulta Gestire la sicurezza e la conformità delle applicazioni con e cdk-nag nel blog AWS Cloud Development Kit (AWS CDK). AWS DevOps
- Utilizza lo strumento open source Checkov per eseguire analisi statiche sul tuo ambiente IaC. Checkov aiuta a identificare le configurazioni errate del cloud scansionando il codice dell'infrastruttura in Kubernetes, Terraform o. CloudFormation Puoi utilizzare Checkov per ottenere output in diversi formati, tra cui JSON, JUnit XML o CLI. Checkov è in grado di gestire le variabili

in modo efficace creando un grafico che illustra la dipendenza di codice dinamica. Per ulteriori informazioni, consulta il repository Checkov di Bridgecrew. GitHub

- Utilizza TFlint per verificare la presenza di errori e sintassi obsoleta e agevolare l'applicazione delle best practice. Tieni presente che TFlint potrebbe non convalidare problemi specifici del provider. Per ulteriori informazioni su TFlint, consulta il repository TFlint di Terraform Linters. GitHub
- Usa Amazon CodeWhisperer per eseguire scansioni di sicurezza. CodeWhisperer è uno strumento di produttività basato sull'intelligenza artificiale in grado di generare suggerimenti di codice in tempo reale. Può aiutarti a identificare i problemi di sicurezza, seguire le migliori pratiche e aumentare la produttività per l'implementazione del codice.

Sviluppo e perfezionamento della documentazione

La documentazione è fondamentale per il successo del tuo progetto. Non solo spiega come funziona il codice, ma aiuta anche gli sviluppatori a comprendere meglio le caratteristiche e le funzionalità delle applicazioni. Lo sviluppo e il perfezionamento di documentazione di alta qualità possono rafforzare il processo di sviluppo di software, mantenere software eccellenti e favorire il trasferimento delle conoscenze tra sviluppatori.

Esistono due categorie di documentazione: la documentazione all'interno del codice e quella di supporto relativa al codice. La prima si presenta sotto forma di commenti. La seconda può consistere in file README e documenti esterni. Non è raro che gli sviluppatori considerino la documentazione come un sovraccarico, poiché il codice stesso è facile da capire. Ciò potrebbe valere i progetti piccoli, ma la documentazione è fondamentale per i progetti su larga scala che vedono coinvolti più team.

È consigliabile che l'autore del codice scriva la documentazione poiché ne conosce bene le funzionalità. Gli sviluppatori possono avere difficoltà a gestire il sovraccarico aggiuntivo derivante dalla gestione di una documentazione di supporto a sé stante. Per superare questo ostacolo, possono aggiungere all'interno del codice commenti che possono essere estratti in modo automatico, così che tutte le versioni del codice e della documentazione siano sincronizzate.

Esistono diversi strumenti per aiutare gli sviluppatori a estrarre commenti dal codice e generare la relativa documentazione. Questa guida è considerata lo strumento preferito per i AWS CDK costrutti. TypeDoc

Perché è necessaria la documentazione del codice per AWS CDK i costrutti

AWS CDK i costrutti comuni vengono creati da più team di un'organizzazione e condivisi tra diversi team per essere utilizzati. Una buona documentazione aiuta gli utenti della libreria di costrutti a integrarli facilmente e a costruire la propria infrastruttura con il minimo sforzo. Mantenere sincronizzati tutti i documenti non è semplice. Ti consigliamo di mantenere il documento all'interno del codice, che verrà estratto utilizzando la TypeDoc libreria.

Utilizzo TypeDoc con la AWS Construct Library

TypeDoc è un generatore di documenti per TypeScript. È possibile TypeDoc utilizzarlo per leggere i file TypeScript sorgente, analizzare i commenti in tali file e quindi generare un sito statico che contiene la documentazione per il codice.

Il codice seguente mostra come eseguire l'integrazione TypeDoc con la AWS Construct Library e quindi aggiungere i seguenti pacchetti nel `package.json` file in. `devDependencies`

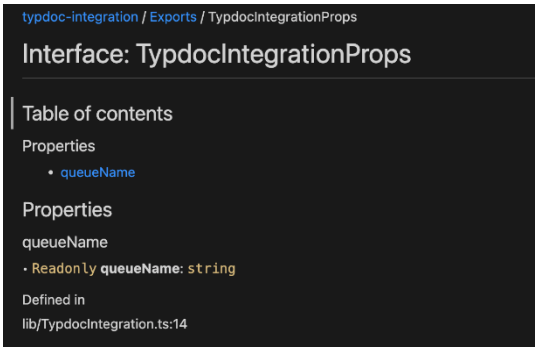
```
{

  "devDependencies": {

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

}
```

Per aggiungere `typedoc.json` nella cartella della libreria CDK, utilizza il codice seguente.

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

Per generare i file README, esegui il `npx typedoc` comando nella directory principale del progetto della libreria di AWS CDK costruzioni.

Il seguente documento di esempio è generato da. TypeDoc



Per ulteriori informazioni sulle opzioni di TypeDoc integrazione, consulta Doc Comments nella TypeDoc documentazione.

Adozione di un approccio di sviluppo basato su test

Ti consigliamo di seguire un approccio di sviluppo basato sui test (TDD) con. AWS CDK TDD è un approccio di sviluppo di software in cui si sviluppano casi di test per specificare e verificare il codice. In parole povere, per prima cosa si creano casi di test per ciascuna funzionalità e, se il test ha esito negativo, si scrive il nuovo codice per superare il test e fare in modo che sia semplice e privo di bug.

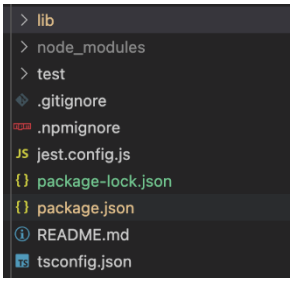
Puoi utilizzare TDD per scrivere prima il caso di test. Ciò consente di verificare l'infrastruttura con diversi vincoli di progettazione in termini di applicazione della politicy di sicurezza per le risorse e di rispetto di una convenzione di denominazione esclusiva per il progetto. L'approccio standard per testare AWS CDK le applicazioni consiste nell'utilizzare il modulo AWS CDKassertions e i framework di test più diffusi, come Jest for e/o pytest for TypeScript JavaScript Python.

Esistono due categorie di test che puoi scrivere per le tue applicazioni: AWS CDK

- Utilizza asserzioni dettagliate per testare un aspetto specifico del CloudFormation modello generato, ad esempio «questa risorsa ha questa proprietà con questo valore». Questi test possono rilevare eventuali regressioni e sono utili anche quando vuoi sviluppare nuove funzionalità con TDD (prima scrivi un test, poi fai in modo che venga superato scrivendo un'implementazione corretta). Le asserzioni granulari sono i test che scriverai più spesso.
- Utilizza i test snapshot per testare il modello sintetizzato CloudFormation rispetto a un modello di base memorizzato in precedenza. I test snapshot consentono di rifattorizzare liberamente, perché hai la certezza che il codice rifattorizzato funziona esattamente allo stesso modo dell'originale. Se le modifiche sono state intenzionali, puoi accettare una nuova base per i test futuri. Tuttavia, AWS CDK gli aggiornamenti possono anche causare modifiche ai modelli sintetizzati, quindi non puoi fare affidamento solo sulle istantanee per assicurarti che l'implementazione sia corretta.

Test unitario

Questa guida si concentra TypeScript specificamente sull'integrazione dei test unitari. Per abilitare i test, verifica che il file `package.json` disponga delle seguenti librerie: `@types/jest`, `jest` `ets-jest` in `devDependencies`. Per aggiungere questi pacchetti, esegui il comando `cdk init lib --language=typescript`. Dopo aver eseguito il comando precedente, visualizzerai la struttura seguente.



Il codice seguente è un esempio di `package.json` file abilitato con la libreria 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",
    ...
  }
}
```

Nella cartella Test, puoi scrivere il caso di test. L'esempio seguente mostra un test case per un AWS CodePipeline costrutto.

```
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
    });
});
```

Per avviare un test, esegui il comando `npm run test` nel progetto. Il test restituisce i risultati seguenti.

```
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
```

Per ulteriori informazioni sui casi di test, consultate Testing constructs nella AWS Cloud Development Kit (AWS CDK) Developer Guide.

Test di integrazione

I test di integrazione per AWS CDK i costrutti possono essere inclusi anche utilizzando un `integ-tests` modulo. Un test di integrazione deve essere definito come un' AWS CDK applicazione. Dovrebbe esserci una one-to-one relazione tra un test di integrazione e un' AWS CDK applicazione. Per ulteriori informazioni, visita il integ-tests-alpha modulo nell'AWS CDK API Reference.

Uso del controllo di rilasci e versioni per i costrutti

Controllo della versione per AWS CDK

AWS CDK i costrutti comuni possono essere creati da più team e condivisi all'interno di un'organizzazione per essere utilizzati. In genere, gli sviluppatori rilasciano nuove funzionalità o correzioni di bug nei loro costrutti comuni AWS CDK . Questi costrutti vengono utilizzati dalle AWS CDK applicazioni o da qualsiasi altro AWS CDK costrutto esistente come parte di una dipendenza. Per questo motivo, è fondamentale che gli sviluppatori aggiornino e rilascino i costrutti con versioni semantiche adeguate in modo indipendente. AWS CDK Le applicazioni downstream o altri AWS CDK costrutti possono aggiornare la propria dipendenza per utilizzare la versione di costruzione appena rilasciata. AWS CDK

Il controllo delle versioni semantico (Semver) è un insieme di regole, o metodo, per fornire numeri software univoci al software informatico. Le versioni sono definite come segue:

- Una versione PRINCIPALE è costituita da modifiche API incompatibili o da una modifica sostanziale.
- Una versione SECONDARIA è costituita da funzionalità aggiunte in modo retrocompatibile.
- Una versione PATCH consiste in correzioni di bug retrocompatibili..

Per ulteriori informazioni sul controllo delle versioni semantiche, vedete Semantic Versioning Specification () nella documentazione del controllo delle versioni semantiche. SemVer

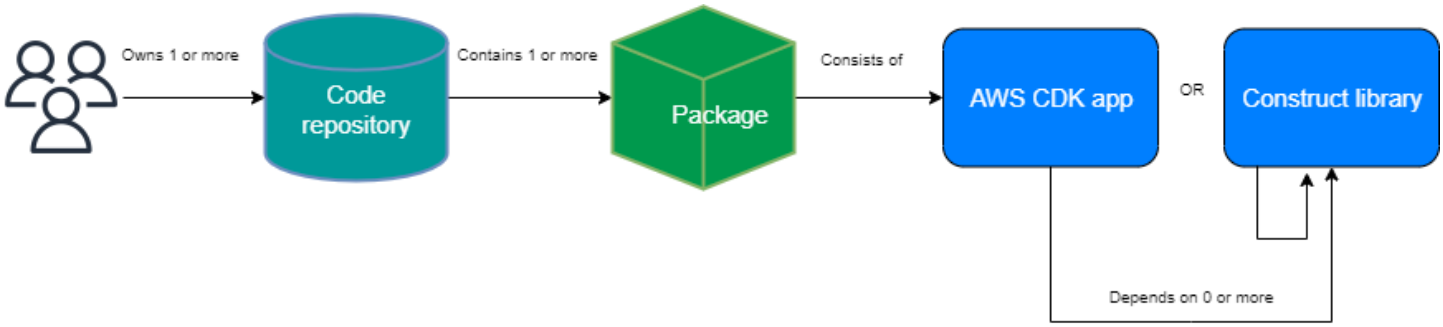
AWS CDK Deposito e imballaggio per costrutti

Poiché AWS CDK i costrutti vengono sviluppati da team diversi e utilizzati da più AWS CDK applicazioni, è possibile utilizzare un repository separato per ogni costrutto. AWS CDK Questo può anche aiutarti ad applicare il controllo degli accessi. Ogni repository può contenere tutto il codice sorgente relativo allo stesso AWS CDK costrutto insieme a tutte le sue dipendenze. Mantenendo una singola applicazione (ovvero un AWS CDK costrutto) in un unico repository, è possibile ridurre l'ambito di impatto delle modifiche durante la distribuzione.

AWS CDK Non solo genera CloudFormation modelli per l'implementazione dell'infrastruttura, ma raggruppa anche risorse di runtime come funzioni Lambda e immagini Docker e le distribuisce insieme all'infrastruttura. Non solo è possibile combinare il codice che definisce l'infrastruttura e il codice che implementa la logica di runtime in un unico costrutto, ma è anche una best practice. Non è necessario che questi due tipi di codice risiedano in repository o addirittura in pacchetti separati.

Per utilizzare i pacchetti oltre i confini del repository, è necessario disporre di un archivio di pacchetti privato, simile a npm o Maven Central PyPi, ma interno all'organizzazione. È inoltre necessario disporre di un processo di rilascio che crei, verifichi e pubblichi il pacchetto nel repository di pacchetti privato. Puoi creare repository privati, ad esempio PyPi server, utilizzando una macchina virtuale locale (VM) o Amazon S3. Quando progetti o crei un registro di pacchetti privato, è fondamentale considerare il rischio di interruzione del servizio a causa dell'elevata disponibilità e scalabilità. Un servizio gestito senza server ospitato nel cloud per archiviare i pacchetti può ridurre notevolmente il sovraccarico di manutenzione. Ad esempio, è possibile utilizzarlo AWS CodeArtifactper ospitare pacchetti per i linguaggi di programmazione più diffusi. È inoltre possibile CodeArtifact utilizzarlo per impostare connessioni a repository esterni e replicarle all'interno. CodeArtifact

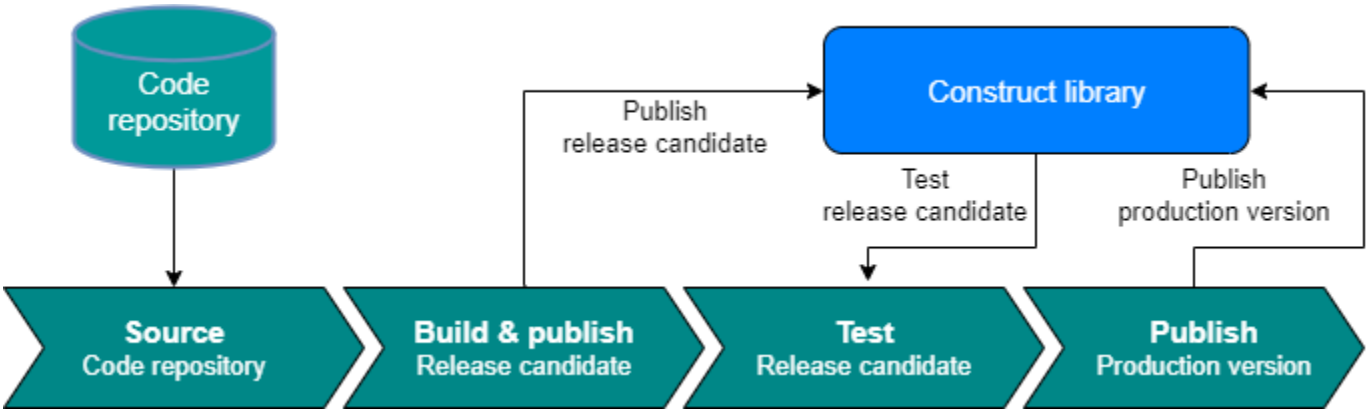
Le dipendenze dai pacchetti nel repository dei pacchetti sono gestite dal gestore di pacchetti della tua lingua (ad esempio, npm for or applications). TypeScript JavaScript Il gestore di pacchetti si assicura che le build siano ripetibili registrando le versioni specifiche di ciascun pacchetto da cui dipende l'applicazione e quindi ti permette di aggiornare tali dipendenze in modo controllato, come illustra il diagramma seguente.



Construct Releasing per AWS CDK

Ti consigliamo di creare la tua pipeline automatizzata per creare e rilasciare nuove AWS CDK versioni di build. Se disponi di un adeguato processo di approvazione delle richieste pull, dopo aver eseguito il commit e inserito il codice di origine nel ramo principale del repository, la pipeline può sviluppare e creare una versione release candidate. Tale versione può essere installata CodeArtifact e testata prima di rilasciare la versione pronta per la produzione. Facoltativamente, puoi testare la tua nuova AWS CDK versione di build localmente prima di unire il codice con il ramo principale. In tal modo, la pipeline esegue il rilascio della versione pronta alla produzione. Tieni presente che i costrutti e i pacchetti condivisi devono essere testati a prescindere dall'applicazione che li utilizza, come se fossero rilasciati al pubblico.

Il diagramma seguente mostra una pipeline di rilascio della versione di esempio AWS CDK .



Puoi utilizzare i seguenti comandi di esempio per creare, testare e pubblicare pacchetti npm. Innanzitutto, accedi al repository di artefatti eseguendo il comando seguente.

```
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>
```

Quindi, completa questa procedura:

1. Installa i pacchetti necessari in base al file `package.json`: `npm install`
2. Crea la versione release candidate: `npm version prerelease --preid rc`
3. Crea il pacchetto npm: `npm run build`
4. Testa il pacchetto npm: `npm run test`
5. Pubblica il pacchetto npm: `npm publish`

Applicazione della gestione delle versioni della libreria

La gestione del ciclo di vita è una sfida importante quando si gestiscono basi di codice. AWS CDK Ad esempio, supponiamo di iniziare un AWS CDK progetto con la versione 1.97 e che la versione 1.169 diventi disponibile in un secondo momento. La versione 1.169 offre nuove funzionalità e correzioni di bug, ma l'infrastruttura è stata implementata utilizzando la versione precedente. Ora, con l'aumento del divario, l'aggiornamento dei costrutti diventa difficile a causa delle modifiche sostanziali che potrebbero essere introdotte nelle nuove versioni. Ciò può rappresentare un ostacolo se l'ambiente dispone di molte risorse. Lo schema introdotto in questa sezione può aiutarvi a gestire la versione della AWS CDK libreria utilizzando l'automazione. Ecco il flusso di lavoro di questo pattern:

1. Quando si avvia un nuovo prodotto CodeArtifact Service Catalog, le versioni della AWS CDK libreria e le relative dipendenze vengono archiviate nel `package.json` file.
2. Implementa una pipeline comune che tenga traccia di tutti i repository, in modo da poter applicare a questi ultimi aggiornamenti automatici se non ci sono modifiche sostanziali.
3. Una AWS CodeBuild fase verifica la presenza dell'albero delle dipendenze e cerca le ultime modifiche.
4. La pipeline crea un ramo di funzionalità e quindi esegue `cdk synth` con la nuova versione per confermare l'assenza di errori.
5. La nuova versione viene implementata nell'ambiente di test e infine esegue un test di integrazione per assicurarsi che l'implementazione sia corretta.
6. Puoi utilizzare due code Amazon Simple Queue Service (Amazon SQS) per monitorare gli stack. Gli utenti possono esaminare manualmente gli stack nella coda delle eccezioni e gestire le modifiche sostanziali. Gli elementi che superano il test di integrazione possono essere uniti e rilasciati.

Domande frequenti

Quali problemi possono essere TypeScript risolti?

In genere, puoi eliminare i bug nel codice scrivendo test automatici, verificando manualmente che il codice funzioni come previsto e infine chiedendo a un'altra persona di convalidare il codice. La convalida delle connessioni tra ogni parte del progetto richiede molto tempo. Per velocizzare il processo di convalida, puoi utilizzare un linguaggio a controllo tipografico, TypeScript ad esempio automatizzare la convalida del codice e fornire un feedback immediato durante lo sviluppo.

Perché dovrei usare? TypeScript

TypeScript è un linguaggio open source che semplifica il JavaScript codice, facilitandone la lettura e il debug. TypeScript fornisce inoltre strumenti di sviluppo altamente produttivi per JavaScript IDE e pratiche, come il controllo statico. Inoltre, TypeScript offre i vantaggi di ECMAScript 6 (ES6) e può aumentare la produttività. Infine, TypeScript può aiutarvi a evitare i dolorosi bug che gli sviluppatori incontrano di solito quando scrivono per tipo JavaScript controllando il codice.

Devo usare AWS CDK o CloudFormation?

Ti consigliamo di utilizzare AWS Cloud Development Kit (AWS CDK) invece diAWS CloudFormation, se la tua organizzazione ha le competenze di sviluppo necessarie per sfruttare ilAWS CDK. Questo perché AWS CDK è più flessibile di CloudFormation, poiché è possibile utilizzare un linguaggio di programmazione e concetti OOP. Tieni presente che puoi utilizzare CloudFormation per creare AWS risorse in modo ordinato e prevedibile. Nel CloudFormation, le risorse vengono scritte in file di testo utilizzando il formato JSON o YAML.

Cosa succede se AWS CDK non supporta una versione appena lanciata? Servizio AWS

Puoi usare un override non elaborato o una risorsa CloudFormation personalizzata.

Quali sono i diversi linguaggi di programmazione supportati da? AWS CDK

AWS CDKè generalmente disponibile in JavaScript, Python TypeScript, Java, C# e Go (in Developer Preview).

Quanto costa? AWS CDK

Non è previsto alcun costo aggiuntivo per ilAWS CDK. Paghi per le AWS risorse (come le istanze Amazon EC2 o i sistemi di bilanciamento del carico Elastic Load Balancing) che vengono create quando le AWS CDK utilizzi nello stesso modo in cui le paghi se le creassi manualmente. I prezzi sono calcolati in base all'uso effettivo. Non sono previste tariffe minime né sono richiesti impegni anticipati.

Passaggi successivi

Ti consigliamo di iniziare a creare con AWS Cloud Development Kit (AWS CDK) in TypeScript. Per ulteriori informazioni, consulta l'AWS CDK Immersion Day Workshop.

Risorse

Riferimenti

- AWS Costrutti di soluzione (AWS soluzioni)
- AWS Kit di sviluppo cloud (AWS CDK) () GitHub
- AWS Riferimento all'API Construct Library (documentazione diAWS CDK riferimento)
- AWS CDK Documentazione di riferimento (documentazioneAWS CDK di riferimento)
- AWS CDK Workshop di un giorno di immersione (AWS Workshop Studio)

Strumenti

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

Guide e pattern

- AWS Modelli di Solutions Constructs (documentazione)AWS

Cronologia dei documenti

La tabella seguente descrive le modifiche significative apportate a questa guida. Per ricevere notifiche sugli aggiornamenti futuri, puoi abbonarti a un feed RSS.

Modifica	Descrizione	Data
Aggiorna il codice	Abbiamo aggiornato gli esempi di codice nella sezione Segui le TypeScript migliori pratiche.	16 febbraio 2024
Aggiungi sezioni	Abbiamo aggiunto le sezioni Use utility types e Integration test.	10 gennaio 2024
Aggiornamento secondario	Esempio di codice aggiornato per la creazione di un costrutto L3.	16 giugno 2023
Pubblicazione iniziale	—	8 dicembre 2022

AWS Glossario delle linee guida prescrittive

I seguenti sono termini comunemente usati nelle strategie, nelle guide e nei modelli forniti da AWS Prescriptive Guidance. Per suggerire voci, utilizza il link Fornisci feedback alla fine del glossario.

Numeri

7 R

Sette strategie di migrazione comuni per trasferire le applicazioni sul cloud. Queste strategie si basano sulle 5 R identificate da Gartner nel 2011 e sono le seguenti:

- Rifattorizzare/riprogettare: trasferisci un'applicazione e modifica la sua architettura sfruttando appieno le funzionalità native del cloud per migliorare l'agilità, le prestazioni e la scalabilità. Ciò comporta in genere la portabilità del sistema operativo e del database. Esempio: esegui la migrazione del database Oracle on-premise ad Amazon Aurora edizione compatibile con PostgreSQL.
- Ridefinire la piattaforma (lift and reshape): trasferisci un'applicazione nel cloud e introduci un certo livello di ottimizzazione per sfruttare le funzionalità del cloud. Esempio: migra il tuo database Oracle locale su Amazon Relational Database Service (Amazon RDS) per Oracle nel cloud. AWS
- Riacquistare (drop and shop): passa a un prodotto diverso, in genere effettuando la transizione da una licenza tradizionale a un modello SaaS. Esempio: esegui la migrazione del tuo sistema di gestione delle relazioni con i clienti (CRM) su Salesforce.com.
- Eseguire il rehosting (lift and shift): trasferisci un'applicazione sul cloud senza apportare modifiche per sfruttare le funzionalità del cloud. Esempio: migra il tuo database Oracle locale su Oracle su un'istanza EC2 nel cloud. AWS
- Trasferire (eseguire il rehosting a livello hypervisor): trasferisci l'infrastruttura sul cloud senza acquistare nuovo hardware, riscrivere le applicazioni o modificare le operazioni esistenti. Questo scenario di migrazione è specifico di VMware Cloud on AWS, che supporta la compatibilità delle macchine virtuali (VM) e la portabilità del carico di lavoro tra l'ambiente locale e. AWSÈ possibile utilizzare le tecnologie VMware Cloud Foundation dai data center on-premise durante la migrazione dell'infrastruttura a VMware Cloud su AWS. Esempio: trasferisci l'hypervisor che ospita il database Oracle su VMware Cloud on. AWS
- Riesaminare (mantenere): mantieni le applicazioni nell'ambiente di origine. Queste potrebbero includere applicazioni che richiedono una rifattorizzazione significativa che desideri rimandare a

un momento successivo e applicazioni legacy che desideri mantenere, perché non vi è alcuna giustificazione aziendale per effettuarne la migrazione.

- Ritirare: disattiva o rimuovi le applicazioni che non sono più necessarie nell'ambiente di origine.

A

ABAC

Vedi controllo degli accessi basato sugli attributi.

servizi astratti

Vedi servizi gestiti.

ACIDO

Vedi atomicità, consistenza, isolamento, durata.

migrazione attiva-attiva

Un metodo di migrazione del database in cui i database di origine e di destinazione vengono mantenuti sincronizzati (utilizzando uno strumento di replica bidirezionale o operazioni di doppia scrittura) ed entrambi i database gestiscono le transazioni provenienti dalle applicazioni di connessione durante la migrazione. Questo metodo supporta la migrazione in piccoli batch controllati anziché richiedere una conversione una tantum. È più flessibile ma richiede più lavoro rispetto alla migrazione attiva-passiva.

migrazione attiva-passiva

Un metodo di migrazione di database in cui i database di origine e di destinazione vengono mantenuti sincronizzati, ma solo il database di origine gestisce le transazioni provenienti dalle applicazioni di connessione mentre i dati vengono replicati nel database di destinazione. Il database di destinazione non accetta alcuna transazione durante la migrazione.

funzione aggregata

Una funzione SQL che opera su un gruppo di righe e calcola un singolo valore restituito per il gruppo. Esempi di funzioni aggregate includono `SUM` e. `MAX`

Intelligenza artificiale

Vedi intelligenza artificiale.

AIOps

Guarda le operazioni di intelligenza artificiale.

anonimizzazione

Il processo di eliminazione permanente delle informazioni personali in un set di dati. L'anonimizzazione può aiutare a proteggere la privacy personale. I dati anonimi non sono più considerati dati personali.

anti-modello

Una soluzione utilizzata di frequente per un problema ricorrente in cui la soluzione è controproducente, inefficace o meno efficace di un'alternativa.

controllo delle applicazioni

Un approccio alla sicurezza che consente l'uso solo di applicazioni approvate per proteggere un sistema dal malware.

portfolio di applicazioni

Una raccolta di informazioni dettagliate su ogni applicazione utilizzata da un'organizzazione, compresi i costi di creazione e manutenzione dell'applicazione e il relativo valore aziendale. Queste informazioni sono fondamentali per il processo di scoperta e analisi del portfolio e aiutano a identificare e ad assegnare la priorità alle applicazioni da migrare, modernizzare e ottimizzare.

intelligenza artificiale (IA)

Il campo dell'informatica dedicato all'uso delle tecnologie informatiche per svolgere funzioni cognitive tipicamente associate agli esseri umani, come l'apprendimento, la risoluzione di problemi e il riconoscimento di schemi. Per ulteriori informazioni, consulta la sezione Che cos'è l'intelligenza artificiale?

operazioni di intelligenza artificiale (AIOps)

Il processo di utilizzo delle tecniche di machine learning per risolvere problemi operativi, ridurre gli incidenti operativi e l'intervento umano e aumentare la qualità del servizio. Per ulteriori informazioni su come viene utilizzato AIOps nella strategia di migrazione AWS , consulta la guida all'integrazione delle operazioni.

crittografia asimmetrica

Un algoritmo di crittografia che utilizza una coppia di chiavi, una chiave pubblica per la crittografia e una chiave privata per la decrittografia. Puoi condividere la chiave pubblica perché non viene utilizzata per la decrittografia, ma l'accesso alla chiave privata deve essere altamente limitato.

atomicità, consistenza, isolamento, durabilità (ACID)

Un insieme di proprietà del software che garantiscono la validità dei dati e l'affidabilità operativa di un database, anche in caso di errori, interruzioni di corrente o altri problemi.

Controllo degli accessi basato su attributi (ABAC)

La pratica di creare autorizzazioni dettagliate basate su attributi utente, come reparto, ruolo professionale e nome del team. Per ulteriori informazioni, consulta ABAC for AWS nella documentazione AWS Identity and Access Management (IAM).

fonte di dati autorevole

Una posizione in cui è archiviata la versione principale dei dati, considerata la fonte di informazioni più affidabile. È possibile copiare i dati dalla fonte di dati autorevole in altre posizioni allo scopo di elaborarli o modificarli, ad esempio anonimizzandoli, oscurandoli o pseudonimizzandoli.

Zona di disponibilità

Una posizione distinta all'interno di un edificio Regione AWS che è isolata dai guasti in altre zone di disponibilità e offre una connettività di rete economica e a bassa latenza verso altre zone di disponibilità nella stessa regione.

AWS Cloud Adoption Framework (CAF)AWS

Un framework di linee guida e best practice AWS per aiutare le organizzazioni a sviluppare un piano efficiente ed efficace per passare con successo al cloud. AWS CAF organizza le linee guida in sei aree di interesse chiamate prospettive: business, persone, governance, piattaforma, sicurezza e operazioni. Le prospettive relative ad azienda, persone e governance si concentrano sulle competenze e sui processi aziendali; le prospettive relative alla piattaforma, alla sicurezza e alle operazioni si concentrano sulle competenze e sui processi tecnici. Ad esempio, la prospettiva relativa alle persone si rivolge alle parti interessate che gestiscono le risorse umane (HR), le funzioni del personale e la gestione del personale. In questa prospettiva, AWS CAF fornisce linee guida per lo sviluppo delle persone, la formazione e le comunicazioni per aiutare a preparare l'organizzazione all'adozione del cloud di successo. Per ulteriori informazioni, consulta il sito web di AWS CAF e il white paper AWS CAF.

AWS Workload Qualification Framework (WQF)AWS

Uno strumento che valuta i carichi di lavoro di migrazione dei database, consiglia strategie di migrazione e fornisce stime del lavoro. AWS WQF è incluso in (). AWS Schema Conversion Tool AWS SCT Analizza gli schemi di database e gli oggetti di codice, il codice dell'applicazione, le dipendenze e le caratteristiche delle prestazioni e fornisce report di valutazione.

B

bot difettoso

Un bot che ha lo scopo di interrompere o causare danni a individui o organizzazioni.

BCP

Vedi la pianificazione della continuità operativa.

grafico comportamentale

Una vista unificata, interattiva dei comportamenti delle risorse e delle interazioni nel tempo. Puoi utilizzare un grafico comportamentale con Amazon Detective per esaminare tentativi di accesso non riusciti, chiamate API sospette e azioni simili. Per ulteriori informazioni, consulta Dati in un grafico comportamentale nella documentazione di Detective.

sistema big-endian

Un sistema che memorizza per primo il byte più importante. Vedi anche endianness.

Classificazione binaria

Un processo che prevede un risultato binario (una delle due classi possibili). Ad esempio, il modello di machine learning potrebbe dover prevedere problemi come "Questa e-mail è spam o non è spam?" o "Questo prodotto è un libro o un'auto?"

filtro Bloom

Una struttura di dati probabilistica ed efficiente in termini di memoria che viene utilizzata per verificare se un elemento fa parte di un set.

distribuzioni blu/verdi

Una strategia di implementazione in cui si creano due ambienti separati ma identici. La versione corrente dell'applicazione viene eseguita in un ambiente (blu) e la nuova versione

dell'applicazione nell'altro ambiente (verde). Questa strategia consente di ripristinare rapidamente il sistema con un impatto minimo.

bot

Un'applicazione software che esegue attività automatizzate su Internet e simula l'attività o l'interazione umana. Alcuni bot sono utili o utili, come i web crawler che indicizzano le informazioni su Internet. Alcuni altri bot, noti come bot dannosi, hanno lo scopo di disturbare o causare danni a individui o organizzazioni.

botnet

Reti di bot infettate da malware e controllate da un'unica parte, nota come bot herder o bot operator. Le botnet sono il meccanismo più noto per scalare i bot e il loro impatto.

ramo

Un'area contenuta di un repository di codice. Il primo ramo creato in un repository è il ramo principale. È possibile creare un nuovo ramo a partire da un ramo esistente e quindi sviluppare funzionalità o correggere bug al suo interno. Un ramo creato per sviluppare una funzionalità viene comunemente detto ramo di funzionalità. Quando la funzionalità è pronta per il rilascio, il ramo di funzionalità viene ricongiunto al ramo principale. Per ulteriori informazioni, consulta Informazioni sulle filiali (documentazione). GitHub

accesso break-glass

In circostanze eccezionali e tramite una procedura approvata, un mezzo rapido per consentire a un utente di accedere a un sito a Account AWS cui in genere non dispone delle autorizzazioni necessarie. Per ulteriori informazioni, vedere l'indicatore Implementate break-glass procedures nella guida Well-Architected AWS .

strategia brownfield

L'infrastruttura esistente nell'ambiente. Quando si adotta una strategia brownfield per un'architettura di sistema, si progetta l'architettura in base ai vincoli dei sistemi e dell'infrastruttura attuali. Per l'espansione dell'infrastruttura esistente, è possibile combinare strategie brownfield e greenfield.

cache del buffer

L'area di memoria in cui sono archiviati i dati a cui si accede con maggiore frequenza.

capacità di business

Azioni intraprese da un'azienda per generare valore (ad esempio vendite, assistenza clienti o marketing). Le architetture dei microservizi e le decisioni di sviluppo possono essere guidate dalle capacità aziendali. Per ulteriori informazioni, consulta la sezione Organizzazione in base alle funzionalità aziendali del whitepaper Esecuzione di microservizi containerizzati su AWS.

pianificazione della continuità operativa (BCP)

Un piano che affronta il potenziale impatto di un evento che comporta l'interruzione dell'attività, come una migrazione su larga scala, sulle operazioni e consente a un'azienda di riprendere rapidamente le operazioni.

C

CAF

Vedi AWS Cloud Adoption Framework.

implementazione canaria

Il rilascio lento e incrementale di una versione agli utenti finali. Quando sei sicuro, distribuisci la nuova versione e sostituisci la versione corrente nella sua interezza.

CoE

Vedi Cloud Center of Excellence.

CDC

Vedi Change Data Capture.

Change Data Capture (CDC)

Il processo di tracciamento delle modifiche a un'origine dati, ad esempio una tabella di database, e di registrazione dei metadati relativi alla modifica. È possibile utilizzare CDC per vari scopi, ad esempio il controllo o la replica delle modifiche in un sistema di destinazione per mantenere la sincronizzazione.

ingegneria del caos

Introduzione intenzionale di guasti o eventi dirompenti per testare la resilienza di un sistema. Puoi usare AWS Fault Injection Service (AWS FIS) per eseguire esperimenti che stressano i tuoi AWS carichi di lavoro e valutarne la risposta.

CI/CD

Vedi integrazione continua e distribuzione continua.

classificazione

Un processo di categorizzazione che aiuta a generare previsioni. I modelli di ML per problemi di classificazione prevedono un valore discreto. I valori discreti sono sempre distinti l'uno dall'altro. Ad esempio, un modello potrebbe dover valutare se in un'immagine è presente o meno un'auto.

crittografia lato client

Crittografia dei dati a livello locale, prima che il destinatario li Servizio AWS riceva.

centro di eccellenza del cloud (CCoE)

Un team multidisciplinare che guida le iniziative di adozione del cloud in tutta l'organizzazione, tra cui lo sviluppo di best practice per il cloud, la mobilitazione delle risorse, la definizione delle tempistiche di migrazione e la guida dell'organizzazione attraverso trasformazioni su larga scala. Per ulteriori informazioni, consulta i post di CCoE sul blog AWS Cloud Enterprise Strategy.

cloud computing

La tecnologia cloud generalmente utilizzata per l'archiviazione remota di dati e la gestione dei dispositivi IoT. Il cloud computing è generalmente connesso alla tecnologia di edge computing.

modello operativo cloud

In un'organizzazione IT, il modello operativo utilizzato per creare, maturare e ottimizzare uno o più ambienti cloud. Per ulteriori informazioni, consulta Building your Cloud Operating Model.

fasi di adozione del cloud

Le quattro fasi che le organizzazioni in genere attraversano quando migrano al AWS cloud:

- Progetto: esecuzione di alcuni progetti relativi al cloud per scopi di dimostrazione e apprendimento
- Fondamento: effettuare investimenti fondamentali per dimensionare l'adozione del cloud (ad esempio, creazione di una zona di destinazione, definizione di un CCoE, definizione di un modello operativo)
- Migrazione: migrazione di singole applicazioni
- Reinvenzione: ottimizzazione di prodotti e servizi e innovazione nel cloud

Queste fasi sono state definite da Stephen Orban nel post sul blog The Journey Toward Cloud-First & the Stages of Adoption on the AWS Cloud Enterprise Strategy. Per informazioni su come si relazionano alla strategia di AWS migrazione, consulta la guida alla preparazione alla migrazione.

CMDB

Vedi database di gestione della configurazione.

repository di codice

Una posizione in cui il codice di origine e altri asset, come documentazione, esempi e script, vengono archiviati e aggiornati attraverso processi di controllo delle versioni. Gli archivi cloud più comuni includono GitHub o AWS CodeCommit. Ogni versione del codice è denominata ramo. In una struttura a microservizi, ogni repository è dedicato a una singola funzionalità. Una singola pipeline CI/CD può utilizzare più repository.

cache fredda

Una cache del buffer vuota, non ben popolata o contenente dati obsoleti o irrilevanti. Ciò influisce sulle prestazioni perché l'istanza di database deve leggere dalla memoria o dal disco principale, il che richiede più tempo rispetto alla lettura dalla cache del buffer.

dati freddi

Dati a cui si accede raramente e che in genere sono storici. Quando si eseguono interrogazioni di questo tipo di dati, le interrogazioni lente sono in genere accettabili. Lo spostamento di questi dati su livelli o classi di storage meno costosi e con prestazioni inferiori può ridurre i costi.

visione artificiale (CV)

Un campo dell'intelligenza artificiale che utilizza l'apprendimento automatico per analizzare ed estrarre informazioni da formati visivi come immagini e video digitali. Ad esempio, AWS Panorama offre dispositivi che aggiungono CV alle reti di telecamere locali e Amazon SageMaker fornisce algoritmi di elaborazione delle immagini per CV.

deriva della configurazione

Per un carico di lavoro, una modifica della configurazione rispetto allo stato previsto. Potrebbe causare la non conformità del carico di lavoro e in genere è graduale e involontaria.

database di gestione della configurazione (CMDB)

Un repository che archivia e gestisce le informazioni su un database e il relativo ambiente IT, inclusi i componenti hardware e software e le relative configurazioni. In genere si utilizzano i dati di un CMDB nella fase di individuazione e analisi del portafoglio della migrazione.

Pacchetto di conformità

Una raccolta di AWS Config regole e azioni correttive che puoi assemblare per personalizzare i controlli di conformità e sicurezza. È possibile distribuire un pacchetto di conformità come singola entità in una regione Account AWS and o all'interno di un'organizzazione utilizzando un modello YAML. Per ulteriori informazioni, consulta i Conformance Pack nella documentazione. AWS Config

integrazione e distribuzione continua (continuous integration and continuous delivery, CI/CD)

Il processo di automazione delle fasi di origine, creazione, test, gestione temporanea e produzione del processo di rilascio del software. Il processo CI/CD è comunemente descritto come una pipeline. CI/CD può aiutare ad automatizzare i processi, migliorare la produttività, migliorare la qualità del codice e velocizzare le distribuzioni. Per ulteriori informazioni, consulta Vantaggi della distribuzione continua. CD può anche significare continuous deployment (implementazione continua). Per ulteriori informazioni, consulta Distribuzione continua e implementazione continua a confronto.

CV

Vedi visione artificiale.

D

dati a riposo

Dati stazionari nella rete, ad esempio i dati archiviati.

classificazione dei dati

Un processo per identificare e classificare i dati nella rete in base alla loro criticità e sensibilità. È un componente fondamentale di qualsiasi strategia di gestione dei rischi di sicurezza informatica perché consente di determinare i controlli di protezione e conservazione appropriati per i dati. La classificazione dei dati è un componente del pilastro della sicurezza nel AWS Well-Architected Framework. Per ulteriori informazioni, consulta Classificazione dei dati.

deriva dei dati

Una variazione significativa tra i dati di produzione e i dati utilizzati per addestrare un modello di machine learning o una modifica significativa dei dati di input nel tempo. La deriva dei dati può ridurre la qualità, l'accuratezza e l'equità complessive nelle previsioni dei modelli ML.

dati in transito

Dati che si spostano attivamente attraverso la rete, ad esempio tra le risorse di rete.

rete di dati

Un framework architettonico che fornisce la proprietà distribuita e decentralizzata dei dati con gestione e governance centralizzate.

riduzione al minimo dei dati

Il principio della raccolta e del trattamento dei soli dati strettamente necessari. Praticare la riduzione al minimo dei dati in the Cloud AWS può ridurre i rischi per la privacy, i costi e l'impronta di carbonio delle analisi.

perimetro dei dati

Una serie di barriere preventive nell' AWS ambiente che aiutano a garantire che solo le identità attendibili accedano alle risorse attendibili delle reti previste. Per ulteriori informazioni, consulta Building a data perimeter on. AWS

pre-elaborazione dei dati

Trasformare i dati grezzi in un formato che possa essere facilmente analizzato dal modello di ML. La pre-elaborazione dei dati può comportare la rimozione di determinate colonne o righe e l'eliminazione di valori mancanti, incoerenti o duplicati.

provenienza dei dati

Il processo di tracciamento dell'origine e della cronologia dei dati durante il loro ciclo di vita, ad esempio il modo in cui i dati sono stati generati, trasmessi e archiviati.

soggetto dei dati

Un individuo i cui dati vengono raccolti ed elaborati.

data warehouse

Un sistema di gestione dei dati che supporta la business intelligence, come l'analisi. I data warehouse contengono in genere grandi quantità di dati storici e vengono generalmente utilizzati per interrogazioni e analisi.

linguaggio di definizione del database (DDL)

Istruzioni o comandi per creare o modificare la struttura di tabelle e oggetti in un database.

linguaggio di manipolazione del database (DML)

Istruzioni o comandi per modificare (inserire, aggiornare ed eliminare) informazioni in un database.

DDL

Vedi linguaggio di definizione del database.

deep ensemble

Combinare più modelli di deep learning per la previsione. È possibile utilizzare i deep ensemble per ottenere una previsione più accurata o per stimare l'incertezza nelle previsioni.

deep learning

Un sottocampo del ML che utilizza più livelli di reti neurali artificiali per identificare la mappatura tra i dati di input e le variabili target di interesse.

defense-in-depth

Un approccio alla sicurezza delle informazioni in cui una serie di meccanismi e controlli di sicurezza sono accuratamente stratificati su una rete di computer per proteggere la riservatezza, l'integrità e la disponibilità della rete e dei dati al suo interno. Quando si adotta questa strategia AWS, si aggiungono più controlli a diversi livelli della AWS Organizations struttura per proteggere le risorse. Ad esempio, un defense-in-depth approccio potrebbe combinare l'autenticazione a più fattori, la segmentazione della rete e la crittografia.

amministratore delegato

In AWS Organizations, un servizio compatibile può registrare un account AWS membro per amministrare gli account dell'organizzazione e gestire le autorizzazioni per quel servizio. Questo account è denominato amministratore delegato per quel servizio specifico. Per ulteriori informazioni e un elenco di servizi compatibili, consulta Servizi che funzionano con AWS Organizations nella documentazione di AWS Organizations .

implementazione

Il processo di creazione di un'applicazione, di nuove funzionalità o di correzioni di codice disponibili nell'ambiente di destinazione. L'implementazione prevede l'applicazione di modifiche in una base di codice, seguita dalla creazione e dall'esecuzione di tale base di codice negli ambienti applicativi.

Ambiente di sviluppo

> Vedi ambiente.

controllo di rilevamento

> Un controllo di sicurezza progettato per rilevare, registrare e avvisare dopo che si è verificato un evento. Questi controlli rappresentano una seconda linea di difesa e avvisano l'utente in caso di eventi di sicurezza che aggirano i controlli preventivi in vigore. Per ulteriori informazioni, consulta Controlli di rilevamento in Implementazione dei controlli di sicurezza in AWS.

mappatura del flusso di valore dello sviluppo (DVSM)

> Un processo utilizzato per identificare e dare priorità ai vincoli che influiscono negativamente sulla velocità e sulla qualità nel ciclo di vita dello sviluppo del software. DVSM estende il processo di mappatura del flusso di valore originariamente progettato per pratiche di produzione snella. Si concentra sulle fasi e sui team necessari per creare e trasferire valore attraverso il processo di sviluppo del software.

gemello digitale

> Una rappresentazione virtuale di un sistema reale, ad esempio un edificio, una fabbrica, un'attrezzatura industriale o una linea di produzione. I gemelli digitali supportano la manutenzione predittiva, il monitoraggio remoto e l'ottimizzazione della produzione.

tabella delle dimensioni

> In uno schema a stella, una tabella più piccola che contiene gli attributi dei dati quantitativi in una tabella dei fatti. Gli attributi della tabella delle dimensioni sono in genere campi di testo o numeri discreti che si comportano come testo. Questi attributi vengono comunemente utilizzati per il vincolo delle query, il filtraggio e l'etichettatura dei set di risultati.

disastro

> Un evento che impedisce a un carico di lavoro o a un sistema di raggiungere gli obiettivi aziendali nella sua sede principale di implementazione. Questi eventi possono essere disastri naturali, guasti tecnici o il risultato di azioni umane, come errori di configurazione involontari o attacchi di malware.

disaster recovery (DR)

> La strategia e il processo utilizzati per ridurre al minimo i tempi di inattività e la perdita di dati causati da un disastro. Per ulteriori informazioni, consulta Disaster Recovery of Workloads su AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

Vedi linguaggio di manipolazione del database.

progettazione basata sul dominio

Un approccio allo sviluppo di un sistema software complesso collegandone i componenti a domini in evoluzione, o obiettivi aziendali principali, perseguiti da ciascun componente. Questo concetto è stato introdotto da Eric Evans nel suo libro,Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Per informazioni su come utilizzare la progettazione basata sul dominio con il modello del fico strangolatore (Strangler Fig), consulta la sezione Modernizzazione incrementale dei servizi Web Microsoft ASP.NET (ASMX) legacy utilizzando container e il Gateway Amazon API.

DOTT.

Vedi disaster recovery.

rilevamento della deriva

Tracciamento delle deviazioni da una configurazione di base. Ad esempio, puoi utilizzarlo AWS CloudFormation per rilevare la deriva nelle risorse di sistema oppure puoi usarlo AWS Control Tower per rilevare cambiamenti nella tua landing zone che potrebbero influire sulla conformità ai requisiti di governance.

DVSM

Vedi la mappatura del flusso di valore dello sviluppo.

E

EDA

Vedi analisi esplorativa dei dati.

edge computing

La tecnologia che aumenta la potenza di calcolo per i dispositivi intelligenti all'edge di una rete IoT. Rispetto al cloud computing, l'edge computing può ridurre la latenza di comunicazione e migliorare i tempi di risposta.

crittografia

Un processo di elaborazione che trasforma i dati in chiaro, leggibili dall'uomo, in testo cifrato.

chiave crittografica

Una stringa crittografica di bit randomizzati generata da un algoritmo di crittografia. Le chiavi possono variare di lunghezza e ogni chiave è progettata per essere imprevedibile e univoca.

endianità

L'ordine in cui i byte vengono archiviati nella memoria del computer. I sistemi big-endian memorizzano per primo il byte più importante. I sistemi little-endian memorizzano per primo il byte meno importante.

endpoint

Vedi service endpoint.

servizio endpoint

Un servizio che puoi ospitare in un cloud privato virtuale (VPC) da condividere con altri utenti. Puoi creare un servizio endpoint con AWS PrivateLink e concedere autorizzazioni ad altri Account AWS o a AWS Identity and Access Management (IAM) principali. Questi account o principali possono connettersi al servizio endpoint in privato creando endpoint VPC di interfaccia. Per ulteriori informazioni, consulta Creazione di un servizio endpoint nella documentazione di Amazon Virtual Private Cloud (Amazon VPC).

pianificazione delle risorse aziendali (ERP)

Un sistema che automatizza e gestisce i processi aziendali chiave (come contabilità, MES e gestione dei progetti) per un'azienda.

crittografia envelope

Il processo di crittografia di una chiave di crittografia con un'altra chiave di crittografia. Per ulteriori informazioni, vedete Envelope encryption nella documentazione AWS Key Management Service (AWS KMS).

ambiente

Un'istanza di un'applicazione in esecuzione. Di seguito sono riportati i tipi di ambiente più comuni nel cloud computing:

- ambiente di sviluppo: un'istanza di un'applicazione in esecuzione disponibile solo per il team principale responsabile della manutenzione dell'applicazione. Gli ambienti di sviluppo vengono utilizzati per testare le modifiche prima di promuoverle negli ambienti superiori. Questo tipo di ambiente viene talvolta definito ambiente di test.

- ambienti inferiori: tutti gli ambienti di sviluppo di un'applicazione, ad esempio quelli utilizzati per le build e i test iniziali.
- ambiente di produzione: un'istanza di un'applicazione in esecuzione a cui gli utenti finali possono accedere. In una pipeline CI/CD, l'ambiente di produzione è l'ultimo ambiente di implementazione.
- ambienti superiori: tutti gli ambienti a cui possono accedere utenti diversi dal team di sviluppo principale. Si può trattare di un ambiente di produzione, ambienti di preproduzione e ambienti per i test di accettazione da parte degli utenti.

epica

Nelle metodologie agili, categorie funzionali che aiutano a organizzare e dare priorità al lavoro. Le epiche forniscono una descrizione di alto livello dei requisiti e delle attività di implementazione. Ad esempio, le epopee della sicurezza AWS CAF includono la gestione delle identità e degli accessi, i controlli investigativi, la sicurezza dell'infrastruttura, la protezione dei dati e la risposta agli incidenti. Per ulteriori informazioni sulle epiche, consulta la strategia di migrazione AWS , consulta la guida all'implementazione del programma.

ERP

Vedi pianificazione delle risorse aziendali.

analisi esplorativa dei dati (EDA)

Il processo di analisi di un set di dati per comprenderne le caratteristiche principali. Si raccolgono o si aggregano dati e quindi si eseguono indagini iniziali per trovare modelli, rilevare anomalie e verificare ipotesi. L'EDA viene eseguita calcolando statistiche di riepilogo e creando visualizzazioni di dati.

F

tabella dei fatti

Il tavolo centrale in uno schema a stella. Memorizza dati quantitativi sulle operazioni aziendali. In genere, una tabella dei fatti contiene due tipi di colonne: quelle che contengono misure e quelle che contengono una chiave esterna per una tabella di dimensioni.

fallire velocemente

Una filosofia che utilizza test frequenti e incrementali per ridurre il ciclo di vita dello sviluppo. È una parte fondamentale di un approccio agile.

limite di isolamento dei guasti

Nel Cloud AWS, un limite come una zona di disponibilità Regione AWS, un piano di controllo o un piano dati che limita l'effetto di un errore e aiuta a migliorare la resilienza dei carichi di lavoro. Per ulteriori informazioni, consulta AWS Fault Isolation Boundaries.

ramo di funzionalità

Vedi filiale.

caratteristiche

I dati di input che usi per fare una previsione. Ad esempio, in un contesto di produzione, le caratteristiche potrebbero essere immagini acquisite periodicamente dalla linea di produzione.

importanza delle caratteristiche

Quanto è importante una caratteristica per le previsioni di un modello. Di solito viene espresso come punteggio numerico che può essere calcolato con varie tecniche, come Shapley Additive Explanations (SHAP) e gradienti integrati. Per ulteriori informazioni, vedere Interpretabilità del modello di machine learning con:AWS.

trasformazione delle funzionalità

Per ottimizzare i dati per il processo di machine learning, incluso l'arricchimento dei dati con fonti aggiuntive, il dimensionamento dei valori o l'estrazione di più set di informazioni da un singolo campo di dati. Ciò consente al modello di ML di trarre vantaggio dai dati. Ad esempio, se suddividi la data "2021-05-27 00:15:37" in "2021", "maggio", "giovedì" e "15", puoi aiutare l'algoritmo di apprendimento ad apprendere modelli sfumati associati a diversi componenti dei dati.

FGAC

Vedi il controllo granulare degli accessi.

controllo granulare degli accessi (FGAC)

L'uso di più condizioni per consentire o rifiutare una richiesta di accesso.

migrazione flash-cut

Un metodo di migrazione del database che utilizza la replica continua dei dati tramite l'acquisizione dei dati delle modifiche per migrare i dati nel più breve tempo possibile, anziché utilizzare un approccio graduale. L'obiettivo è ridurre al minimo i tempi di inattività.

G

blocco geografico

Vedi restrizioni geografiche.

limitazioni geografiche (blocco geografico)

In Amazon CloudFront, un'opzione per impedire agli utenti di determinati paesi di accedere alle distribuzioni di contenuti. Puoi utilizzare un elenco consentito o un elenco di blocco per specificare i paesi approvati e vietati. Per ulteriori informazioni, consulta Limitare la distribuzione geografica dei contenuti nella CloudFront documentazione.

Flusso di lavoro di GitFlow

Un approccio in cui gli ambienti inferiori e superiori utilizzano rami diversi in un repository di codice di origine. Il flusso di lavoro Gitflow è considerato obsoleto e il flusso di lavoro basato su trunk è l'approccio moderno e preferito.

strategia greenfield

L'assenza di infrastrutture esistenti in un nuovo ambiente. Quando si adotta una strategia greenfield per un'architettura di sistema, è possibile selezionare tutte le nuove tecnologie senza il vincolo della compatibilità con l'infrastruttura esistente, nota anche come brownfield. Per l'espansione dell'infrastruttura esistente, è possibile combinare strategie brownfield e greenfield.

guardrail

Una regola di livello elevato che consente di governare risorse, policy e conformità tra le unità organizzative (OU). I guardrail preventivi applicano le policy per garantire l'allineamento agli standard di conformità. Vengono implementati utilizzando le policy di controllo dei servizi e i limiti delle autorizzazioni IAM. I guardrail di rilevamento rilevano le violazioni delle policy e i problemi di conformità e generano avvisi per porvi rimedio. Sono implementati utilizzando Amazon AWS Config AWS Security Hub GuardDuty AWS Trusted Advisor, Amazon Inspector e controlli personalizzati AWS Lambda .

H

AH

Vedi disponibilità elevata.

migrazione di database eterogenea

Migrazione del database di origine in un database di destinazione che utilizza un motore di database diverso (ad esempio, da Oracle ad Amazon Aurora). La migrazione eterogenea fa in genere parte di uno sforzo di riprogettazione e la conversione dello schema può essere un'attività complessa. AWS offre AWS SCT che aiuta con le conversioni dello schema.

alta disponibilità (HA)

La capacità di un carico di lavoro di funzionare in modo continuo, senza intervento, in caso di sfide o disastri. I sistemi HA sono progettati per il failover automatico, fornire costantemente prestazioni di alta qualità e gestire carichi e guasti diversi con un impatto minimo sulle prestazioni.

modernizzazione storica

Un approccio utilizzato per modernizzare e aggiornare i sistemi di tecnologia operativa (OT) per soddisfare meglio le esigenze dell'industria manifatturiera. Uno storico è un tipo di database utilizzato per raccogliere e archiviare dati da varie fonti in una fabbrica.

migrazione di database omogenea

Migrazione del database di origine in un database di destinazione che condivide lo stesso motore di database (ad esempio, da Microsoft SQL Server ad Amazon RDS per SQL Server). La migrazione omogenea fa in genere parte di un'operazione di rehosting o ridefinizione della piattaforma. Per migrare lo schema è possibile utilizzare le utilità native del database.

dati caldi

Dati a cui si accede frequentemente, come dati in tempo reale o dati di traduzione recenti. Questi dati richiedono in genere un livello o una classe di storage ad alte prestazioni per fornire risposte rapide alle query.

hotfix

Una soluzione urgente per un problema critico in un ambiente di produzione. A causa della sua urgenza, un hotfix viene in genere creato al di fuori del tipico DevOps flusso di lavoro di rilascio.

periodo di hypercare

Subito dopo la conversione, il periodo di tempo in cui un team di migrazione gestisce e monitora le applicazioni migrate nel cloud per risolvere eventuali problemi. In genere, questo periodo dura da 1 a 4 giorni. Al termine del periodo di hypercare, il team addetto alla migrazione in genere trasferisce la responsabilità delle applicazioni al team addetto alle operazioni cloud.

I

IaC

Considera l'infrastruttura come codice.

Policy basata su identità

Una policy associata a uno o più principi IAM che definisce le relative autorizzazioni all'interno dell' Cloud AWS ambiente.

applicazione inattiva

Un'applicazione che prevede un uso di CPU e memoria medio compreso tra il 5% e il 20% in un periodo di 90 giorni. In un progetto di migrazione, è normale ritirare queste applicazioni o mantenerle on-premise.

IIoT

Vedi Industrial Internet of Things.

infrastruttura immutabile

Un modello che implementa una nuova infrastruttura per i carichi di lavoro di produzione anziché aggiornare, applicare patch o modificare l'infrastruttura esistente. Le infrastrutture immutabili sono intrinsecamente più coerenti, affidabili e prevedibili delle infrastrutture mutabili. Per ulteriori informazioni, consulta la best practice Deploy using immutable infrastructure in Well-Architected AWS Framework.

VPC in ingresso (ingress)

In un'architettura AWS multi-account, un VPC che accetta, ispeziona e indirizza le connessioni di rete dall'esterno di un'applicazione. Nel documento Architettura di riferimento per la sicurezza di AWS si consiglia di configurare l'account di rete con VPC in entrata, in uscita e di ispezione per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

migrazione incrementale

Una strategia di conversione in cui si esegue la migrazione dell'applicazione in piccole parti anziché eseguire una conversione singola e completa. Ad esempio, inizialmente potresti spostare solo alcuni microservizi o utenti nel nuovo sistema. Dopo aver verificato che tutto funzioni correttamente, puoi spostare in modo incrementale microservizi o utenti aggiuntivi fino alla disattivazione del sistema legacy. Questa strategia riduce i rischi associati alle migrazioni di grandi dimensioni.

Industria 4.0

Un termine introdotto da Klaus Schwab nel 2016 per riferirsi alla modernizzazione dei processi di produzione attraverso progressi in termini di connettività, dati in tempo reale, automazione, analisi e AI/ML.

infrastruttura

Tutte le risorse e gli asset contenuti nell'ambiente di un'applicazione.

infrastruttura come codice (IaC)

Il processo di provisioning e gestione dell'infrastruttura di un'applicazione tramite un insieme di file di configurazione. Il processo IaC è progettato per aiutarti a centralizzare la gestione dell'infrastruttura, a standardizzare le risorse e a dimensionare rapidamente, in modo che i nuovi ambienti siano ripetibili, affidabili e coerenti.

Internet delle cose industriale (IIoT)

L'uso di sensori e dispositivi connessi a Internet nei settori industriali, come quello manifatturiero, energetico, automobilistico, sanitario, delle scienze della vita e dell'agricoltura. Per ulteriori informazioni, consulta Creazione di una strategia di trasformazione digitale dell'Internet delle cose industriale (IIoT).

VPC di ispezione

In un'architettura AWS multi-account, un VPC centralizzato che gestisce le ispezioni del traffico di rete tra VPC (uguali o diversi Regioni AWS), Internet e reti locali. Nel documento Architettura di riferimento per la sicurezza di AWS si consiglia di configurare l'account di rete con VPC in entrata, in uscita e di ispezione per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

Internet of Things (IoT)

La rete di oggetti fisici connessi con sensori o processori incorporati che comunicano con altri dispositivi e sistemi tramite Internet o una rete di comunicazione locale. Per ulteriori informazioni, consulta Cos'è l'IoT?

interpretabilità

Una caratteristica di un modello di machine learning che descrive il grado in cui un essere umano è in grado di comprendere in che modo le previsioni del modello dipendono dai suoi input. Per ulteriori informazioni, consulta la sezione Interpretabilità dei modelli di machine learning con AWS.

IoT

Vedi Internet of Things.

libreria di informazioni IT (ITIL)

Una serie di best practice per offrire servizi IT e allinearli ai requisiti aziendali. ITIL fornisce le basi per ITSM.

gestione dei servizi IT (ITSM)

Attività associate alla progettazione, implementazione, gestione e supporto dei servizi IT per un'organizzazione. Per informazioni sull'integrazione delle operazioni cloud con gli strumenti ITSM, consulta la guida all'integrazione delle operazioni.

ITIL

Vedi la libreria di informazioni IT.

ITSM

Vedi Gestione dei servizi IT.

L

controllo degli accessi basato su etichette (LBAC)

Un'implementazione del controllo di accesso obbligatorio (MAC) in cui agli utenti e ai dati stessi viene assegnato esplicitamente un valore di etichetta di sicurezza. L'intersezione tra l'etichetta di sicurezza utente e l'etichetta di sicurezza dei dati determina quali righe e colonne possono essere visualizzate dall'utente.

zona di destinazione

Una landing zone è un AWS ambiente multi-account ben progettato, scalabile e sicuro. Questo è un punto di partenza dal quale le organizzazioni possono avviare e distribuire rapidamente carichi di lavoro e applicazioni con fiducia nel loro ambiente di sicurezza e infrastruttura. Per ulteriori informazioni sulle zone di destinazione, consulta la sezione Configurazione di un ambiente AWS multi-account sicuro e scalabile.

migrazione su larga scala

Una migrazione di 300 o più server.

BIANCO

Vedi controllo degli accessi basato su etichette.

Privilegio minimo

La best practice di sicurezza per la concessione delle autorizzazioni minime richieste per eseguire un'attività. Per ulteriori informazioni, consulta Applicazione delle autorizzazioni del privilegio minimo nella documentazione di IAM.

eseguire il rehosting (lift and shift)

Vedi 7 R.

sistema little-endian

Un sistema che memorizza per primo il byte meno importante. Vedi anche endianità.

ambienti inferiori

Vedi ambiente.

M

machine learning (ML)

Un tipo di intelligenza artificiale che utilizza algoritmi e tecniche per il riconoscimento e l'apprendimento di schemi. Il machine learning analizza e apprende dai dati registrati, come i dati dell'Internet delle cose (IoT), per generare un modello statistico basato su modelli. Per ulteriori informazioni, consulta la sezione Machine learning.

ramo principale

Vedi filiale.

malware

Software progettato per compromettere la sicurezza o la privacy del computer. Il malware potrebbe interrompere i sistemi informatici, divulgare informazioni sensibili o ottenere accessi non autorizzati. Esempi di malware includono virus, worm, ransomware, trojan horse, spyware e keylogger.

servizi gestiti

Servizi AWS per cui AWS gestisce il livello di infrastruttura, il sistema operativo e le piattaforme e si accede agli endpoint per archiviare e recuperare i dati. Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) e Amazon DynamoDB sono esempi di servizi gestiti. Questi sono noti anche come servizi astratti.

sistema di esecuzione della produzione (MES)

Un sistema software per tracciare, monitorare, documentare e controllare i processi di produzione che convertono le materie prime in prodotti finiti in officina.

MAP

Vedi Migration Acceleration Program.

meccanismo

Un processo completo in cui si crea uno strumento, si promuove l'adozione dello strumento e quindi si esaminano i risultati per apportare le modifiche. Un meccanismo è un ciclo che si rafforza e si migliora man mano che funziona. Per ulteriori informazioni, consulta Creazione di meccanismi nel AWS Well-Architected Framework.

account membro

Tutti gli account Account AWS diversi dall'account di gestione che fanno parte di un'organizzazione in. AWS Organizations Un account può essere membro di una sola organizzazione alla volta.

MEH

Vedi sistema di esecuzione della produzione.

Message Queuing Telemetry Transport (MQTT)

Un protocollo di comunicazione machine-to-machine (M2M) leggero, basato sul modello di pubblicazione/sottoscrizione, per dispositivi IoT con risorse limitate.

microservizio

Un piccolo servizio indipendente che comunica tramite API ben definite ed è in genere di proprietà di piccoli team autonomi. Ad esempio, un sistema assicurativo potrebbe includere microservizi che si riferiscono a funzionalità aziendali, come vendite o marketing, o sottodomini, come acquisti, reclami o analisi. I vantaggi dei microservizi includono agilità, dimensionamento flessibile,

facilità di implementazione, codice riutilizzabile e resilienza. Per ulteriori informazioni, consulta Integrazione dei microservizi utilizzando servizi serverless. AWS

architettura di microservizi

Un approccio alla creazione di un'applicazione con componenti indipendenti che eseguono ogni processo applicativo come microservizio. Questi microservizi comunicano tramite un'interfaccia ben definita utilizzando API leggere. Ogni microservizio in questa architettura può essere aggiornato, distribuito e dimensionato per soddisfare la richiesta di funzioni specifiche di un'applicazione. Per ulteriori informazioni, vedere Implementazione dei microservizi su. AWS

Programma di accelerazione della migrazione (MAP)

Un AWS programma che fornisce consulenza, supporto, formazione e servizi per aiutare le organizzazioni a costruire una solida base operativa per il passaggio al cloud e per contribuire a compensare il costo iniziale delle migrazioni. MAP include una metodologia di migrazione per eseguire le migrazioni precedenti in modo metodico e un set di strumenti per automatizzare e accelerare gli scenari di migrazione comuni.

migrazione su larga scala

Il processo di trasferimento della maggior parte del portfolio di applicazioni sul cloud avviene a ondate, con più applicazioni trasferite a una velocità maggiore in ogni ondata. Questa fase utilizza le migliori pratiche e le lezioni apprese nelle fasi precedenti per implementare una fabbrica di migrazione di team, strumenti e processi per semplificare la migrazione dei carichi di lavoro attraverso l'automazione e la distribuzione agile. Questa è la terza fase della strategia di migrazione AWS.

fabbrica di migrazione

Team interfunzionali che semplificano la migrazione dei carichi di lavoro attraverso approcci automatizzati e agili. I team di Migration Factory includono in genere operazioni, analisti e proprietari aziendali, ingegneri addetti alla migrazione, sviluppatori e DevOps professionisti che lavorano nell'ambito degli sprint. Tra il 20% e il 50% di un portfolio di applicazioni aziendali è costituito da schemi ripetuti che possono essere ottimizzati con un approccio di fabbrica. Per ulteriori informazioni, consulta la discussione sulle fabbriche di migrazione e la Guida alla fabbrica di migrazione al cloud in questo set di contenuti.

metadati di migrazione

Le informazioni sull'applicazione e sul server necessarie per completare la migrazione. Ogni modello di migrazione richiede un set diverso di metadati di migrazione. Esempi di metadati di migrazione includono la sottorete, il gruppo di sicurezza e l'account di destinazione. AWS

modello di migrazione

Un'attività di migrazione ripetibile che descrive in dettaglio la strategia di migrazione, la destinazione della migrazione e l'applicazione o il servizio di migrazione utilizzati. Esempio: riorganizza la migrazione su Amazon EC2 AWS con Application Migration Service.

Valutazione del portfolio di migrazione (MPA)

Uno strumento online che fornisce informazioni per la convalida del business case per la migrazione al Cloud. AWS MPA offre una valutazione dettagliata del portfolio (dimensionamento corretto dei server, prezzi, confronto del TCO, analisi dei costi di migrazione) e pianificazione della migrazione (analisi e raccolta dei dati delle applicazioni, raggruppamento delle applicazioni, prioritizzazione delle migrazioni e pianificazione delle ondate). Lo strumento MPA (richiede l'accesso) è disponibile gratuitamente per tutti i AWS consulenti e i consulenti dei partner APN.

valutazione della preparazione alla migrazione (MRA)

Il processo di acquisizione di informazioni sullo stato di preparazione al cloud di un'organizzazione, l'identificazione dei punti di forza e di debolezza e la creazione di un piano d'azione per colmare le lacune identificate, utilizzando il CAF. AWS Per ulteriori informazioni, consulta la guida di preparazione alla migrazione. MRA è la prima fase della strategia di migrazione AWS.

strategia di migrazione

L'approccio utilizzato per migrare un carico di lavoro nel cloud. AWS Per ulteriori informazioni, consulta la voce 7 R in questo glossario e consulta Mobilita la tua organizzazione per accelerare le migrazioni su larga scala.

ML

Vedi machine learning.

modernizzazione

Trasformazione di un'applicazione obsoleta (legacy o monolitica) e della relativa infrastruttura in un sistema agile, elastico e altamente disponibile nel cloud per ridurre i costi, aumentare

l'efficienza e sfruttare le innovazioni. Per ulteriori informazioni, vedere Strategia per la modernizzazione delle applicazioni in. Cloud AWS

valutazione della preparazione alla modernizzazione

Una valutazione che aiuta a determinare la preparazione alla modernizzazione delle applicazioni di un'organizzazione, identifica vantaggi, rischi e dipendenze e determina in che misura l'organizzazione può supportare lo stato futuro di tali applicazioni. Il risultato della valutazione è uno schema dell'architettura di destinazione, una tabella di marcia che descrive in dettaglio le fasi di sviluppo e le tappe fondamentali del processo di modernizzazione e un piano d'azione per colmare le lacune identificate. Per ulteriori informazioni, consulta la sezione Valutazione della preparazione alla modernizzazione per le applicazioni nel cloud AWS.

applicazioni monolitiche (monoliti)

Applicazioni eseguite come un unico servizio con processi strettamente collegati. Le applicazioni monolitiche presentano diversi inconvenienti. Se una funzionalità dell'applicazione registra un picco di domanda, l'intera architettura deve essere dimensionata. L'aggiunta o il miglioramento delle funzionalità di un'applicazione monolitica diventa inoltre più complessa man mano che la base di codice cresce. Per risolvere questi problemi, puoi utilizzare un'architettura di microservizi. Per ulteriori informazioni, consulta la sezione Scomposizione dei monoliti in microservizi.

MAPPA

Vedi Migration Portfolio Assessment.

MQTT

Vedi Message Queuing Telemetry Transport.

classificazione multiclasse

Un processo che aiuta a generare previsioni per più classi (prevedendo uno o più di due risultati). Ad esempio, un modello di machine learning potrebbe chiedere "Questo prodotto è un libro, un'auto o un telefono?" oppure "Quale categoria di prodotti è più interessante per questo cliente?"

infrastruttura mutabile

Un modello che aggiorna e modifica l'infrastruttura esistente per i carichi di lavoro di produzione. Per migliorare la coerenza, l'affidabilità e la prevedibilità, il AWS Well-Architected Framework consiglia l'uso di un'infrastruttura immutabile come best practice.

O

OAC

Vedi Origin Access Control.

QUERCIA

Vedi Origin Access Identity.

OCM

Vedi gestione delle modifiche organizzative.

migrazione offline

Un metodo di migrazione in cui il carico di lavoro di origine viene eliminato durante il processo di migrazione. Questo metodo prevede tempi di inattività prolungati e viene in genere utilizzato per carichi di lavoro piccoli e non critici.

OI

Vedi l'integrazione delle operazioni.

OLA

Vedi accordo a livello operativo.

migrazione online

Un metodo di migrazione in cui il carico di lavoro di origine viene copiato sul sistema di destinazione senza essere messo offline. Le applicazioni connesse al carico di lavoro possono continuare a funzionare durante la migrazione. Questo metodo comporta tempi di inattività pari a zero o comunque minimi e viene in genere utilizzato per carichi di lavoro di produzione critici.

OPC-UA

Vedi Open Process Communications - Unified Architecture.

Comunicazioni a processo aperto - Architettura unificata (OPC-UA)

Un protocollo di comunicazione machine-to-machine (M2M) per l'automazione industriale. OPC-UA fornisce uno standard di interoperabilità con schemi di crittografia, autenticazione e autorizzazione dei dati.

accordo a livello operativo (OLA)

Un accordo che chiarisce quali sono gli impegni reciproci tra i gruppi IT funzionali, a supporto di un accordo sul livello di servizio (SLA).

revisione della prontezza operativa (ORR)

Un elenco di domande e best practice associate che aiutano a comprendere, valutare, prevenire o ridurre la portata degli incidenti e dei possibili guasti. Per ulteriori informazioni, vedere Operational Readiness Reviews (ORR) nel Well-Architected AWS Framework.

tecnologia operativa (OT)

Sistemi hardware e software che interagiscono con l'ambiente fisico per controllare le operazioni, le apparecchiature e le infrastrutture industriali. Nella produzione, l'integrazione di sistemi OT e di tecnologia dell'informazione (IT) è un obiettivo chiave per le trasformazioni dell'Industria 4.0.

integrazione delle operazioni (OI)

Il processo di modernizzazione delle operazioni nel cloud, che prevede la pianificazione, l'automazione e l'integrazione della disponibilità. Per ulteriori informazioni, consulta la guida all'integrazione delle operazioni.

trail organizzativo

Un percorso creato da noi AWS CloudTrail che registra tutti gli eventi di un'organizzazione per tutti Account AWS . AWS Organizations Questo percorso viene creato in ogni Account AWS che fa parte dell'organizzazione e tiene traccia dell'attività in ogni account. Per ulteriori informazioni, consulta Creazione di un percorso per un'organizzazione nella CloudTrail documentazione.

gestione del cambiamento organizzativo (OCM)

Un framework per la gestione di trasformazioni aziendali importanti e che comportano l'interruzione delle attività dal punto di vista delle persone, della cultura e della leadership. OCM aiuta le organizzazioni a prepararsi e passare a nuovi sistemi e strategie accelerando l'adozione del cambiamento, affrontando i problemi di transizione e promuovendo cambiamenti culturali e organizzativi. Nella strategia di AWS migrazione, questo framework si chiama accelerazione delle persone, a causa della velocità di cambiamento richiesta nei progetti di adozione del cloud. Per ulteriori informazioni, consultare la Guida OCM.

controllo dell'accesso all'origine (OAC)

In CloudFront, un'opzione avanzata per limitare l'accesso per proteggere i contenuti di Amazon Simple Storage Service (Amazon S3). OAC supporta tutti i bucket S3 in generale Regioni AWS, la

crittografia lato server con AWS KMS (SSE-KMS) e le richieste dinamiche e dirette al bucket S3. `PUT` `DELETE`

identità di accesso origine (OAI)

Nel CloudFront, un'opzione per limitare l'accesso per proteggere i tuoi contenuti Amazon S3. Quando usi OAI, CloudFront crea un principale con cui Amazon S3 può autenticarsi. I principali autenticati possono accedere ai contenuti in un bucket S3 solo tramite una distribuzione specifica. CloudFront Vedi anche OAC, che fornisce un controllo degli accessi più granulare e avanzato.

O

Vedi la revisione della prontezza operativa.

- NON

Vedi la tecnologia operativa.

VPC in uscita (egress)

In un'architettura AWS multi-account, un VPC che gestisce le connessioni di rete avviate dall'interno di un'applicazione. Nel documento Architettura di riferimento per la sicurezza di AWS si consiglia di configurare l'account di rete con VPC in entrata, in uscita e di ispezione per proteggere l'interfaccia bidirezionale tra l'applicazione e Internet in generale.

P

limite delle autorizzazioni

Una policy di gestione IAM collegata ai principali IAM per impostare le autorizzazioni massime che l'utente o il ruolo possono avere. Per ulteriori informazioni, consulta Limiti delle autorizzazioni nella documentazione di IAM.

informazioni di identificazione personale (PII)

Informazioni che, se visualizzate direttamente o abbinate ad altri dati correlati, possono essere utilizzate per dedurre ragionevolmente l'identità di un individuo. Esempi di informazioni personali includono nomi, indirizzi e informazioni di contatto.

Informazioni che consentono l'identificazione personale degli utenti

Visualizza le informazioni di identificazione personale.

playbook

Una serie di passaggi predefiniti che raccolgono il lavoro associato alle migrazioni, come l'erogazione delle funzioni operative principali nel cloud. Un playbook può assumere la forma di script, runbook automatici o un riepilogo dei processi o dei passaggi necessari per gestire un ambiente modernizzato.

PLC

Vedi controllore logico programmabile.

PLM

Vedi la gestione del ciclo di vita del prodotto.

policy

Un oggetto in grado di definire le autorizzazioni (vedi politica basata sull'identità), specificare le condizioni di accesso (vedi politicabasata sulle risorse) o definire le autorizzazioni massime per tutti gli account di un'organizzazione in (vedi politica di controllo dei servizi). AWS Organizations

persistenza poliglotta

Scelta indipendente della tecnologia di archiviazione di dati di un microservizio in base ai modelli di accesso ai dati e ad altri requisiti. Se i microservizi utilizzano la stessa tecnologia di archiviazione di dati, possono incontrare problemi di implementazione o registrare prestazioni scadenti. I microservizi vengono implementati più facilmente e ottengono prestazioni e scalabilità migliori se utilizzano l'archivio dati più adatto alle loro esigenze. Per ulteriori informazioni, consulta la sezione Abilitazione della persistenza dei dati nei microservizi.

valutazione del portfolio

Un processo di scoperta, analisi e definizione delle priorità del portfolio di applicazioni per pianificare la migrazione. Per ulteriori informazioni, consulta la pagina Valutazione della preparazione alla migrazione.

predicate

Una condizione di interrogazione che restituisce o, in genere, si trova in una clausola`true`. `false` `WHERE`

predicato pushdown

Una tecnica di ottimizzazione delle query del database che filtra i dati della query prima del trasferimento. Ciò riduce la quantità di dati che devono essere recuperati ed elaborati dal database relazionale e migliora le prestazioni delle query.

controllo preventivo

Un controllo di sicurezza progettato per impedire il verificarsi di un evento. Questi controlli sono la prima linea di difesa per impedire accessi non autorizzati o modifiche indesiderate alla rete. Per ulteriori informazioni, consulta Controlli preventivi in Implementazione dei controlli di sicurezza in AWS.

principale

Un'entità in AWS grado di eseguire azioni e accedere alle risorse. Questa entità è in genere un utente root per un Account AWS ruolo IAM o un utente. Per ulteriori informazioni, consulta Principali in Termini e concetti dei ruoli nella documentazione di IAM.

Privacy fin dalla progettazione

Un approccio all'ingegneria dei sistemi che tiene conto della privacy durante l'intero processo di progettazione.

zone ospitate private

Un container che contiene informazioni su come si desidera che Amazon Route 53 risponda alle query DNS per un dominio e i relativi sottodomini all'interno di uno o più VPC. Per ulteriori informazioni, consulta Utilizzo delle zone ospitate private nella documentazione di Route 53.

controllo proattivo

Un controllo di sicurezza progettato per impedire l'implementazione di risorse non conformi. Questi controlli analizzano le risorse prima del loro provisioning. Se la risorsa non è conforme al controllo, non viene fornita. Per ulteriori informazioni, consulta la guida di riferimento sui controlli nella AWS Control Tower documentazione e consulta Controlli proattivi in Implementazione dei controlli di sicurezza su. AWS

gestione del ciclo di vita del prodotto (PLM)

La gestione dei dati e dei processi di un prodotto durante l'intero ciclo di vita, dalla progettazione, sviluppo e lancio, attraverso la crescita e la maturità, fino al declino e alla rimozione.

Ambiente di produzione

Vedi ambiente.

controllore logico programmabile (PLC)

Nella produzione, un computer altamente affidabile e adattabile che monitora le macchine e automatizza i processi di produzione.

pseudonimizzazione

Il processo di sostituzione degli identificatori personali in un set di dati con valori segnaposto. La pseudonimizzazione può aiutare a proteggere la privacy personale. I dati pseudonimizzati sono ancora considerati dati personali.

pubblica/iscriviti (pub/sub)

Un pattern che consente comunicazioni asincrone tra microservizi per migliorare la scalabilità e la reattività. Ad esempio, in un MES basato su microservizi, un microservizio può pubblicare messaggi di eventi su un canale a cui altri microservizi possono abbonarsi. Il sistema può aggiungere nuovi microservizi senza modificare il servizio di pubblicazione.

Q

Piano di query

Una serie di passaggi, come le istruzioni, utilizzati per accedere ai dati in un sistema di database relazionale SQL.

regressione del piano di query

Quando un ottimizzatore del servizio di database sceglie un piano non ottimale rispetto a prima di una determinata modifica all'ambiente di database. Questo può essere causato da modifiche a statistiche, vincoli, impostazioni dell'ambiente, associazioni dei parametri di query e aggiornamenti al motore di database.

R

Matrice RACI

Vedi responsabile, responsabile, consultato, informato (RACI).

ransomware

Un software dannoso progettato per bloccare l'accesso a un sistema informatico o ai dati fino a quando non viene effettuato un pagamento.

Matrice RASCI

Vedi responsabile, responsabile, consultato, informato (RACI).

RCAC

Vedi controllo dell'accesso a righe e colonne.

replica di lettura

Una copia di un database utilizzata per scopi di sola lettura. È possibile indirizzare le query alla replica di lettura per ridurre il carico sul database principale.

riprogettare

Vedi 7 Rs.

obiettivo del punto di ripristino (RPO)

Il periodo di tempo massimo accettabile dall'ultimo punto di ripristino dei dati. Ciò determina quella che viene considerata una perdita di dati accettabile tra l'ultimo punto di ripristino e l'interruzione del servizio.

obiettivo del tempo di ripristino (RTO)

Il ritardo massimo accettabile tra l'interruzione del servizio e il ripristino del servizio.

rifattorizzare

Vedi 7 R.

Regione

Una raccolta di AWS risorse in un'area geografica. Ciascuna Regione AWS è isolata e indipendente dalle altre per fornire tolleranza agli errori, stabilità e resilienza. Per ulteriori informazioni, consulta Specificare cosa può usare Regioni AWS il tuo account.

regressione

Una tecnica di ML che prevede un valore numerico. Ad esempio, per risolvere il problema "A che prezzo verrà venduta questa casa?" un modello di ML potrebbe utilizzare un modello di regressione lineare per prevedere il prezzo di vendita di una casa sulla base di dati noti sulla casa (ad esempio, la metratura).

riospitare

Vedi 7 R.

rilascio

In un processo di implementazione, l'atto di promuovere modifiche a un ambiente di produzione.

trasferisco

Vedi 7 Rs.

ripiattaforma

Vedi 7 Rs.

riacquisto

Vedi 7 Rs.

resilienza

La capacità di un'applicazione di resistere o ripristinare le interruzioni. L'elevata disponibilità e il disaster recovery sono considerazioni comuni quando si pianifica la resilienza in. Cloud AWSPer ulteriori informazioni, vedere Cloud AWS Resilience.

policy basata su risorse

Una policy associata a una risorsa, ad esempio un bucket Amazon S3, un endpoint o una chiave di crittografia. Questo tipo di policy specifica a quali principali è consentito l'accesso, le azioni supportate e qualsiasi altra condizione che deve essere soddisfatta.

matrice di assegnazione di responsabilità (RACI)

Una matrice che definisce i ruoli e le responsabilità di tutte le parti coinvolte nelle attività di migrazione e nelle operazioni cloud. Il nome della matrice deriva dai tipi di responsabilità definiti nella matrice: responsabile (R), responsabile (A), consultato (C) e informato (I). Il tipo di supporto (S) è facoltativo. Se includi il supporto, la matrice viene chiamata matrice RASCI e, se la escludi, viene chiamata matrice RACI.

controllo reattivo

Un controllo di sicurezza progettato per favorire la correzione di eventi avversi o deviazioni dalla baseline di sicurezza. Per ulteriori informazioni, consulta Controlli reattivi in Implementazione dei controlli di sicurezza in AWS.

retain

Vedi 7 R.

andare in pensione

Vedi 7 Rs.

rotazione

Processo di aggiornamento periodico di un segreto per rendere più difficile l'accesso alle credenziali da parte di un utente malintenzionato.

controllo dell'accesso a righe e colonne (RCAC)

L'uso di espressioni SQL di base e flessibili con regole di accesso definite. RCAC è costituito da autorizzazioni di riga e maschere di colonna.

RPO

Vedi l'obiettivo del punto di ripristino.

RTO

Vedi l'obiettivo del tempo di ripristino.

runbook

Un insieme di procedure manuali o automatizzate necessarie per eseguire un'attività specifica. In genere sono progettati per semplificare operazioni o procedure ripetitive con tassi di errore elevati.

S

SAML 2.0

Uno standard aperto utilizzato da molti provider di identità (IdPs). Questa funzionalità abilita il single sign-on (SSO) federato, in modo che gli utenti possano accedere AWS Management Console o chiamare le operazioni AWS API senza che tu debba creare un utente in IAM per tutti i membri dell'organizzazione. Per ulteriori informazioni sulla federazione basata su SAML 2.0, consulta Informazioni sulla federazione basata su SAML 2.0 nella documentazione di IAM.

SCADA

Vedi controllo di supervisione e acquisizione dati.

SCP

Vedi la politica di controllo del servizio.

Secret

In AWS Secrets Manager, informazioni riservate o riservate, come una password o le credenziali utente, archiviate in forma crittografata. È costituito dal valore segreto e dai relativi metadati. Il valore segreto può essere binario, una stringa singola o più stringhe. Per ulteriori informazioni, consulta Secret nella documentazione di Secrets Manager.

controllo di sicurezza

Un guardrail tecnico o amministrativo che impedisce, rileva o riduce la capacità di un autore di minacce di sfruttare una vulnerabilità di sicurezza. Esistono quattro tipi principali di controlli di sicurezza: preventivi, investigativi, reattivi e proattivi.

rafforzamento della sicurezza

Il processo di riduzione della superficie di attacco per renderla più resistente agli attacchi. Può includere azioni come la rimozione di risorse che non sono più necessarie, l'implementazione di best practice di sicurezza che prevedono la concessione del privilegio minimo o la disattivazione di funzionalità non necessarie nei file di configurazione.

sistema di gestione delle informazioni e degli eventi di sicurezza (SIEM)

Strumenti e servizi che combinano sistemi di gestione delle informazioni di sicurezza (SIM) e sistemi di gestione degli eventi di sicurezza (SEM). Un sistema SIEM raccoglie, monitora e analizza i dati da server, reti, dispositivi e altre fonti per rilevare minacce e violazioni della sicurezza e generare avvisi.

automazione della risposta alla sicurezza

Un'azione predefinita e programmata progettata per rispondere o porre rimedio automaticamente a un evento di sicurezza. Queste automazioni fungono da controlli di sicurezza investigativi o reattivi che aiutano a implementare le migliori pratiche di sicurezza. AWS Esempi di azioni di risposta automatizzate includono la modifica di un gruppo di sicurezza VPC, l'applicazione di patch a un'istanza Amazon EC2 o la rotazione delle credenziali.

Crittografia lato server

Crittografia dei dati a destinazione, da parte di chi li riceve. Servizio AWS

Policy di controllo dei servizi (SCP)

Una policy che fornisce il controllo centralizzato sulle autorizzazioni per tutti gli account di un'organizzazione in AWS Organizations. Le SCP definiscono i guardrail o fissano i limiti alle azioni che un amministratore può delegare a utenti o ruoli. Puoi utilizzare le SCP come elenchi consentiti o elenchi di rifiuto, per specificare quali servizi o azioni sono consentiti o proibiti. Per ulteriori informazioni, consulta le politiche di controllo del servizio nella AWS Organizations documentazione.

endpoint del servizio

L'URL del punto di ingresso per un Servizio AWS. Puoi utilizzare l'endpoint per connetterti a livello di programmazione al servizio di destinazione. Per ulteriori informazioni, consulta Endpoint del Servizio AWS nei Riferimenti generali di AWS.

accordo sul livello di servizio (SLA)

Un accordo che chiarisce ciò che un team IT promette di offrire ai propri clienti, ad esempio l'operatività e le prestazioni del servizio.

indicatore del livello di servizio (SLI)

Misurazione di un aspetto prestazionale di un servizio, ad esempio il tasso di errore, la disponibilità o la velocità effettiva.

obiettivo a livello di servizio (SLO)

Una metrica target che rappresenta lo stato di un servizio, misurato da un indicatore del livello di servizio.

Modello di responsabilità condivisa

Un modello che descrive la responsabilità condivisa AWS per la sicurezza e la conformità del cloud. AWS è responsabile della sicurezza del cloud, mentre tu sei responsabile della sicurezza nel cloud. Per ulteriori informazioni, consulta Modello di responsabilità condivisa.

SIEM

Vedi il sistema di gestione delle informazioni e degli eventi sulla sicurezza.

punto di errore singolo (SPOF)

Un guasto in un singolo componente critico di un'applicazione che può disturbare il sistema.

SLAM

Vedi il contratto sul livello di servizio.

SLI

Vedi l'indicatore del livello di servizio.

LENTA

Vedi obiettivo del livello di servizio.

split-and-seed modello

Un modello per dimensionare e accelerare i progetti di modernizzazione. Man mano che vengono definite nuove funzionalità e versioni dei prodotti, il team principale si divide per creare nuovi team di prodotto. Questo aiuta a dimensionare le capacità e i servizi dell'organizzazione, migliora la produttività degli sviluppatori e supporta una rapida innovazione. Per ulteriori informazioni, vedere Approccio graduale alla modernizzazione delle applicazioni in. Cloud AWS

SPOF

Vedi punto di errore singolo.

schema a stella

Una struttura organizzativa di database che utilizza un'unica tabella dei fatti di grandi dimensioni per archiviare i dati transazionali o misurati e utilizza una o più tabelle dimensionali più piccole per memorizzare gli attributi dei dati. Questa struttura è progettata per l'uso in un data warehouse o per scopi di business intelligence.

modello del fico strangolatore

Un approccio alla modernizzazione dei sistemi monolitici mediante la riscrittura e la sostituzione incrementali delle funzionalità del sistema fino alla disattivazione del sistema legacy. Questo modello utilizza l'analogia di una pianta di fico che cresce fino a diventare un albero robusto e alla fine annienta e sostituisce il suo ospite. Il modello è stato introdotto da Martin Fowler come metodo per gestire il rischio durante la riscrittura di sistemi monolitici. Per un esempio di come applicare questo modello, consulta Modernizzazione incrementale dei servizi Web legacy di Microsoft ASP.NET (ASMX) mediante container e Gateway Amazon API.

sottorete

Un intervallo di indirizzi IP nel VPC. Una sottorete deve risiedere in una singola zona di disponibilità.

controllo di supervisione e acquisizione dati (SCADA)

Nella produzione, un sistema che utilizza hardware e software per monitorare gli asset fisici e le operazioni di produzione.

crittografia simmetrica

Un algoritmo di crittografia che utilizza la stessa chiave per crittografare e decrittografare i dati.

test sintetici

Test di un sistema in modo da simulare le interazioni degli utenti per rilevare potenziali problemi o monitorare le prestazioni. Puoi usare Amazon CloudWatch Synthetics per creare questi test.

T

tags

Coppie chiave-valore che fungono da metadati per l'organizzazione delle risorse. AWS Con i tag è possibile a gestire, identificare, organizzare, cercare e filtrare le risorse. Per ulteriori informazioni, consulta Tagging delle risorse AWS.

variabile di destinazione

Il valore che stai cercando di prevedere nel machine learning supervisionato. Questo è indicato anche come variabile di risultato. Ad esempio, in un ambiente di produzione la variabile di destinazione potrebbe essere un difetto del prodotto.

elenco di attività

Uno strumento che viene utilizzato per tenere traccia dei progressi tramite un runbook. Un elenco di attività contiene una panoramica del runbook e un elenco di attività generali da completare. Per ogni attività generale, include la quantità stimata di tempo richiesta, il proprietario e lo stato di avanzamento.

Ambiente di test

Vedi ambiente.

training

Fornire dati da cui trarre ispirazione dal modello di machine learning. I dati di training devono contenere la risposta corretta. L'algoritmo di apprendimento trova nei dati di addestramento i pattern che mappano gli attributi dei dati di input al target (la risposta che si desidera prevedere).

Produce un modello di ML che acquisisce questi modelli. Puoi quindi utilizzare il modello di ML per creare previsioni su nuovi dati di cui non si conosce il target.

Transit Gateway

Un hub di transito di rete che è possibile utilizzare per collegare i VPC e le reti on-premise. Per ulteriori informazioni, consulta Cos'è un gateway di transito nella AWS Transit Gateway documentazione.

flusso di lavoro basato su trunk

Un approccio in cui gli sviluppatori creano e testano le funzionalità localmente in un ramo di funzionalità e quindi uniscono tali modifiche al ramo principale. Il ramo principale viene quindi integrato negli ambienti di sviluppo, preproduzione e produzione, in sequenza.

Accesso attendibile

Concessione delle autorizzazioni a un servizio specificato dall'utente per eseguire attività all'interno dell'organizzazione AWS Organizations e nei suoi account per conto dell'utente. Il servizio attendibile crea un ruolo collegato al servizio in ogni account, quando tale ruolo è necessario, per eseguire attività di gestione per conto dell'utente. Per ulteriori informazioni, consulta Utilizzo AWS Organizations con altri AWS servizi nella AWS Organizations documentazione.

regolazione

Modificare alcuni aspetti del processo di training per migliorare la precisione del modello di ML. Ad esempio, puoi addestrare il modello di ML generando un set di etichette, aggiungendo etichette e quindi ripetendo questi passaggi più volte con impostazioni diverse per ottimizzare il modello.

team da due pizze

Una piccola DevOps squadra che puoi sfamare con due pizze. Un team composto da due persone garantisce la migliore opportunità possibile di collaborazione nello sviluppo del software.

U

incertezza

Un concetto che si riferisce a informazioni imprecise, incomplete o sconosciute che possono minare l'affidabilità dei modelli di machine learning predittivi. Esistono due tipi di incertezza: l'incertezza epistemica, che è causata da dati limitati e incompleti, mentre l'incertezza aleatoria

è causata dal rumore e dalla casualità insiti nei dati. Per ulteriori informazioni, consulta la guida Quantificazione dell'incertezza nei sistemi di deep learning.

compiti indifferenziati

Conosciuto anche come sollevamento di carichi pesanti, è un lavoro necessario per creare e far funzionare un'applicazione, ma che non apporta valore diretto all'utente finale né offre vantaggi competitivi. Esempi di attività indifferenziate includono l'approvvigionamento, la manutenzione e la pianificazione della capacità.

ambienti superiori

Vedi ambiente.

V

vacuum

Un'operazione di manutenzione del database che prevede la pulizia dopo aggiornamenti incrementali per recuperare lo spazio di archiviazione e migliorare le prestazioni.

controllo delle versioni

Processi e strumenti che tengono traccia delle modifiche, ad esempio le modifiche al codice di origine in un repository.

Peering VPC

Una connessione tra due VPC che consente di instradare il traffico tramite indirizzi IP privati. Per ulteriori informazioni, consulta Che cos'è il peering VPC? nella documentazione di Amazon VPC.

vulnerabilità

Un difetto software o hardware che compromette la sicurezza del sistema.

W

cache calda

Una cache del buffer che contiene dati correnti e pertinenti a cui si accede frequentemente. L'istanza di database può leggere dalla cache del buffer, il che richiede meno tempo rispetto alla lettura dalla memoria dal disco principale.

dati caldi

Dati a cui si accede raramente. Quando si eseguono interrogazioni di questo tipo di dati, in genere sono accettabili query moderatamente lente.

funzione finestra

Una funzione SQL che esegue un calcolo su un gruppo di righe che si riferiscono in qualche modo al record corrente. Le funzioni della finestra sono utili per l'elaborazione di attività, come il calcolo di una media mobile o l'accesso al valore delle righe in base alla posizione relativa della riga corrente.

Carico di lavoro

Una raccolta di risorse e codice che fornisce valore aziendale, ad esempio un'applicazione rivolta ai clienti o un processo back-end.

flusso di lavoro

Gruppi funzionali in un progetto di migrazione responsabili di una serie specifica di attività. Ogni flusso di lavoro è indipendente ma supporta gli altri flussi di lavoro del progetto. Ad esempio, il flusso di lavoro del portfolio è responsabile della definizione delle priorità delle applicazioni, della pianificazione delle ondate e della raccolta dei metadati di migrazione. Il flusso di lavoro del portfolio fornisce queste risorse al flusso di lavoro di migrazione, che quindi migra i server e le applicazioni.

VERME

Vedi scrivere una volta, leggere molti.

WQF

Vedi AWS Workload Qualification Framework.

scrivi una volta, leggi molte (WORM)

Un modello di storage che scrive i dati una sola volta e ne impedisce l'eliminazione o la modifica. Gli utenti autorizzati possono leggere i dati tutte le volte che è necessario, ma non possono modificarli. Questa infrastruttura di archiviazione dei dati è considerata immutabile.

Z

exploit zero-day

Un attacco, in genere malware, che sfrutta una vulnerabilità zero-day.

vulnerabilità zero-day

Un difetto o una vulnerabilità assoluta in un sistema di produzione. Gli autori delle minacce possono utilizzare questo tipo di vulnerabilità per attaccare il sistema. Gli sviluppatori vengono spesso a conoscenza della vulnerabilità causata dall'attacco.

applicazione zombie

Un'applicazione che prevede un utilizzo CPU e memoria inferiore al 5%. In un progetto di migrazione, è normale ritirare queste applicazioni.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.