



Bewährte Methoden für die Verwendung von AWS CDK in TypeScript zur Erstellung von IaC-Projekten

AWS Präskriptive Leitlinien



AWS Präskriptive Leitlinien: Bewährte Methoden für die Verwendung von AWS CDK in TypeScript zur Erstellung von IaC-Projekten

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Handelsmarken und die Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Einführung	1
Ziele	1
Best Practices	3
Code für große Projekte organisieren	3
Warum die Organisation von Code wichtig ist	3
Wie Sie Ihren Code in großem Maßstab organisieren	3
Beispielcode-Organisation	4
Entwickeln Sie wiederverwendbare Muster	6
Abstract Factory	6
Kette der Verantwortung	7
Konstrukte erstellen oder erweitern	8
Was ist ein Konstrukt	8
Was die verschiedenen Arten von Konstrukten sind	9
So erstellen Sie Ihr eigenes Konstrukt	10
Erstellen oder Erweitern eines L2-Konstrukts	10
Erstellen eines L3-Konstrukts	11
Escape-Luke	12
Benutzerdefinierte Ressourcen	14
Folgen Sie TypeScript den bewährten Methoden	17
Beschreiben Ihrer Daten	17
Enums verwenden	17
Schnittstellen verwenden	18
Schnittstellen erweitern	19
Leere Schnittstellen vermeiden	19
Factories verwenden	20
Destrukturierung für Eigenschaften verwenden	20
Standard-Benennungskonventionen definieren	20
Verwenden Sie nicht das Schlüsselwort var	21
Erwägen Sie die Verwendung von ESLint und Prettier	22
Verwenden Sie Zugriffsmodifikatoren	22
Verwenden Sie Dienstprogrammatypen	23
Nach Sicherheits-Schwachstellen und Formatierungsfehlern scannen	24
Sicherheitsansätze und Tools	24
Gängige Entwicklungstools	24

Dokumentation entwickeln und verfeinern	25
Warum ist Codedokumentation für AWS CDK Konstrukte erforderlich	26
Verwendung TypeDoc mit der AWS Construct-Bibliothek	26
Einen testgetriebenen Entwicklungsansatz verwenden	27
Komponententest	28
Integrationstest	32
Release- und Versionskontrolle für Konstrukte verwenden	32
Versionskontrolle für AWS CDK	32
Repository AWS CDK und Paketierung für Konstrukte	33
Erstellen Sie das Releasing für AWS CDK	34
Erzwingen Sie die Versionsverwaltung der Bibliothek	35
Häufig gestellte Fragen	37
Welche Probleme können TypeScript gelöst werden?	37
Warum sollte ich verwenden TypeScript?	37
Sollte ich das AWS CDK oder verwenden CloudFormation?	37
Was ist, wenn das ein neu AWS-Service eingeführtes Programm AWS CDK nicht unterstützt?	37
Welche verschiedenen Programmiersprachen werden von der unterstützt AWS CDK?	38
Wie viel AWS CDK kostet das?	38
Nächste Schritte	39
Ressourcen	40
Dokumentverlauf	41
Glossar	42
#	42
A	43
B	46
C	48
D	52
E	56
F	58
G	60
H	61
I	63
L	66
M	67
O	71

P	74
Q	77
R	78
S	81
T	85
U	87
V	87
W	88
Z	89
.....	XC

Bewährte Methoden für die Verwendung von AWS CDK in TypeScript zur Erstellung von IaC-Projekten

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

Oktober 2025 ([Verlauf der Dokumente](#))

Dieser Leitfaden enthält Empfehlungen und bewährte Verfahren für die Verwendung von [AWS Cloud Development Kit \(AWS CDK\)](#) in für den Aufbau und TypeScript die Bereitstellung umfangreicher Infrastructure-as-Code-Projekte (IaC). Das AWS CDK ist ein Framework für die Definition der Cloud-Infrastruktur im Code und die Bereitstellung dieser Infrastruktur über AWS CloudFormation. Wenn Sie keine klar definierte Projektstruktur haben, kann der Aufbau und die Verwaltung einer AWS CDK Codebasis für Großprojekte eine Herausforderung sein. Um diese Herausforderungen zu bewältigen, verwenden einige Organisationen Anti-Muster für Großprojekte. Diese Muster können Ihr Projekt jedoch verlangsamen und zu anderen Problemen führen, die sich negativ auf Ihre Organisation auswirken. Anti-Muster können beispielsweise das Onboarding von Entwicklern, Fehlerkorrekturen und die Einführung neuer Features erschweren und verlangsamen.

Dieses Handbuch bietet eine Alternative zur Verwendung von Anti-Mustern und zeigt Ihnen, wie Sie Ihren Code so organisieren, dass er skalierbar ist, getestet und an bewährte Sicherheitsmethoden angepasst wird. Sie können diesen Leitfaden verwenden, um die Codequalität für Ihre IaC-Projekte zu verbessern und Ihre geschäftliche Flexibilität zu maximieren. Dieser Leitfaden richtet sich an Architekten, technische Leiter, Infrastrukturingenieure und alle anderen Personen, die ein gut durchdachtes AWS CDK Projekt für Großprojekte erstellen möchten.

Ziele

- **Geringere Kosten** — Sie können den verwenden AWS CDK , um Ihre eigenen wiederverwendbaren Komponenten zu entwerfen, die den Sicherheits-, Compliance- und Governance-Anforderungen Ihres Unternehmens entsprechen. Sie können Komponenten auch problemlos in Ihrer Organisation gemeinsam nutzen, sodass Sie schnell neue Projekte starten können, die standardmäßig den bewährten Methoden entsprechen.
- **Schnellere Markteinführung** — Nutzen Sie vertraute Funktionen von AWS CDK , um Ihren Entwicklungsprozess zu beschleunigen. Dies erhöht die Wiederverwendbarkeit für die Bereitstellung und reduziert den Entwicklungsaufwand.

- **Höhere Entwicklerproduktivität** — Entwickler können vertraute Programmiersprachen verwenden, um die Infrastruktur zu definieren. Dies hilft Entwicklern, AWS Ressourcen bereitzustellen und zu verwalten. Dies kann zu einer erhöhten Effizienz und Zusammenarbeit der Entwickler führen.

Best Practices

Dieser Abschnitt bietet einen Überblick über die folgenden bewährten Methoden:

- [Code für große Projekte organisieren](#)
- [Entwickeln Sie wiederverwendbare Muster](#)
- [Konstrukte erstellen oder erweitern](#)
- [TypeScript Bewährte Methoden befolgen](#)
- [Nach Sicherheits-Schwachstellen und Formatierungsfehlern scannen](#)
- [Dokumentation entwickeln und verfeinern](#)
- [Einen testgetriebenen Entwicklungsansatz verwenden](#)
- [Release- und Versionskontrolle für Konstrukte verwenden](#)
- [Erzwingen Sie die Versionsverwaltung der Bibliothek](#)

Code für große Projekte organisieren

Warum die Organisation von Code wichtig ist

Für AWS CDK Großprojekte ist es von entscheidender Bedeutung, eine qualitativ hochwertige, klar definierte Struktur zu haben. Je größer ein Projekt wird und die Anzahl der unterstützten Features und Konstrukte zunimmt, desto schwieriger wird die Codenavigation. Diese Schwierigkeit kann sich auf die Produktivität auswirken und das Onboarding von Entwicklern verlangsamen.

Wie Sie Ihren Code in großem Maßstab organisieren

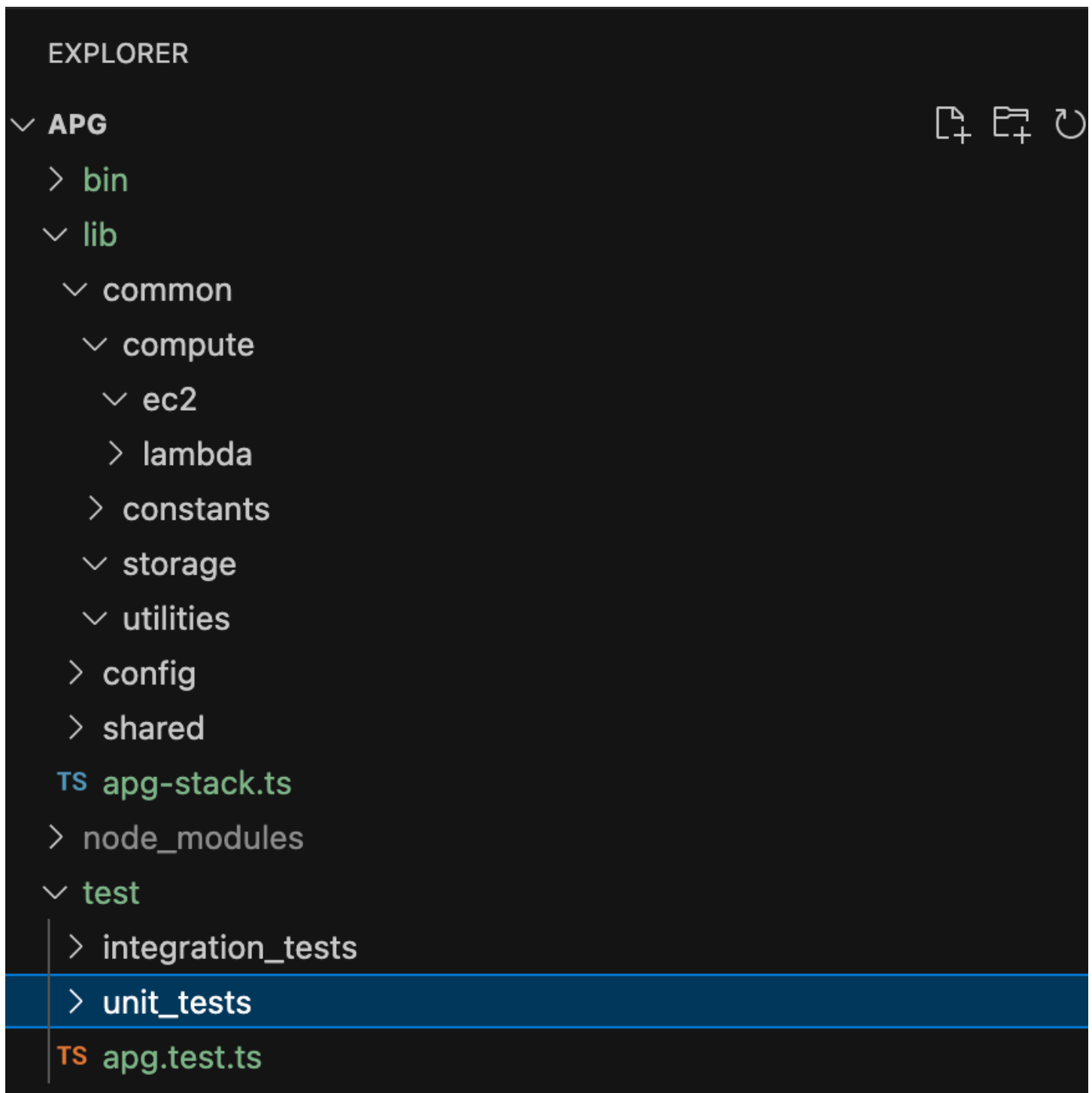
Um ein hohes Maß an Codeflexibilität und Lesbarkeit zu erreichen, empfehlen wir Ihnen, Ihren Code auf der Grundlage der Funktionalität in logische Teile zu unterteilen. Diese Unterteilung spiegelt die Tatsache wider, dass die meisten Ihrer Konstrukte in verschiedenen Geschäftsbereichen verwendet werden. Beispielsweise könnten sowohl Ihre Frontend- als auch Ihre Backend-Anwendungen eine AWS Lambda Funktion benötigen und denselben Quellcode verwenden. Factories können Objekte erstellen, ohne dem Client die Erstellungslogik zugänglich zu machen, und eine gemeinsame Schnittstelle verwenden, um auf neu erstellte Objekte zu verweisen. Sie können eine Factory als effektives Muster verwenden, um ein konsistentes Verhalten in Ihrer Codebasis zu erzeugen. Darüber

hinaus kann eine Factory als zentrale Informationsquelle dienen, um sich wiederholenden Code zu vermeiden und die Fehlerbehebung zu vereinfachen.

Um besser zu verstehen, wie Factories funktionieren, schauen Sie sich das Beispiel eines Automobilherstellers an. Ein Autohersteller muss nicht über das Wissen und die Infrastruktur verfügen, die für die Herstellung von Reifen erforderlich sind. Stattdessen lagert der Autohersteller dieses Fachwissen an einen spezialisierten Reifenhersteller aus und bestellt die Reifen dann einfach bei diesem Hersteller nach Bedarf. Das gleiche Prinzip gilt für Code. Sie können beispielsweise eine Lambda-Factory erstellen, die hochwertige Lambda-Funktionen erstellen kann, und dann die Lambda-Factory in Ihrem Code aufrufen, wann immer Sie eine Lambda-Funktion erstellen müssen. In ähnlicher Weise können Sie denselben Outsourcing-Prozess verwenden, um Ihre Anwendung zu entkoppeln und modulare Komponenten zu erstellen.

Beispielcode-Organisation

Das folgende TypeScript Beispielprojekt, wie in der folgenden Abbildung dargestellt, enthält einen gemeinsamen Ordner, in dem Sie alle Ihre Konstrukte oder allgemeinen Funktionen speichern können.



Zum Beispiel enthält der Ordner berechnen (befindet sich im Ordner Gemeinsam) die gesamte Logik für verschiedene Rechenkonstrukte. Neue Entwickler können problemlos neue Rechenkonstrukte hinzufügen, ohne die anderen Ressourcen zu beeinträchtigen. Bei allen anderen Konstrukten müssen keine neuen Ressourcen intern erstellt werden. Stattdessen rufen diese Konstrukte einfach

das gemeinsame Konstrukt `Factory` auf. Sie können andere Konstrukte, wie z. B. Speicher, auf die gleiche Weise organisieren.

Konfigurationen enthalten umgebungsbasierte Daten, die Sie von den Daten im `Gemeinsam-Ordner` entkoppeln müssen, in dem Sie die Logik aufbewahren. Wir empfehlen Ihnen, Ihre gemeinsamen Konfiguration-Daten in einem `gemeinsam` genutzten Ordner platziert. Wir empfehlen Ihnen auch, den `Hilfsprogramme-Ordner` zu verwenden, um alle Hilfsfunktionen bereitzustellen und Skripte zu bereinigen.

Entwickeln Sie wiederverwendbare Muster

Softwaredesignmuster sind wiederverwendbare Lösungen für häufig auftretende Probleme in der Softwareentwicklung. Sie dienen als Leitfaden oder Paradigma, um Softwareingenieuren dabei zu helfen, Produkte zu entwickeln, die bewährten Methoden folgen. Dieser Abschnitt bietet einen Überblick über zwei wiederverwendbare Muster, die Sie in Ihrer AWS CDK Codebasis verwenden können: das `Abstract Factory-Muster` und das `Chain-of-Responsibility-Muster`. Sie können jedes Muster als Vorlage verwenden und es an das jeweilige Designproblem in Ihrem Code anpassen. Weitere Informationen zu Entwurfsmustern finden Sie unter [Entwurfsmuster](#) in der `Refactoring.Guru`-Dokumentation.

Abstract Factory

Das `Abstract-Factory-Muster` bietet Schnittstellen zum Erstellen von Familien verwandter oder abhängiger Objekte, ohne deren konkrete Klassen anzugeben. Dieses Muster gilt für die folgenden Anwendungsfälle:

- Wenn der Client unabhängig davon ist, wie Sie die Objekte im System erstellen und zusammenstellen
- Wenn das System aus mehreren Objektfamilien besteht und diese Familien so konzipiert sind, dass sie zusammen verwendet werden können
- Wenn Sie einen Laufzeitwert benötigen, um eine bestimmte Abhängigkeit zu erstellen

Weitere Informationen zum `Abstract Factory-Muster` finden Sie unter [Abstract Factory TypeScript in](#) der `Refactoring.Guru`-Dokumentation.

Das folgende Codebeispiel zeigt, wie das `Abstract-Factory-Muster` verwendet werden kann, um eine `Amazon-Elastic Block Store (Amazon EBS)-Speicherfabrik` zu erstellen.

```
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();
```

Kette der Verantwortung

Die Kette der Verantwortung ist ein Verhaltensmuster, das es Ihnen ermöglicht, eine Anfrage entlang der Kette potenzieller Bearbeiter weiterzuleiten, bis einer von ihnen die Anfrage bearbeitet. Das Kette-der-Verantwortung-Muster gilt für die folgenden Anwendungsfälle:

- Wenn mehrere Objekte, die zur Laufzeit bestimmt wurden, für die Bearbeitung einer Anfrage in Frage kommen
- Wenn Sie Handler nicht explizit in Ihrem Code angeben möchten
- Wenn Sie eine Anfrage an eines von mehreren Objekten stellen möchten, ohne den Empfänger explizit anzugeben

Weitere Informationen zum Muster der Verantwortungskette finden Sie unter [Verantwortungskette TypeScript in der Refactoring.Guru-Dokumentation](#).

Der folgende Code zeigt ein Beispiel dafür, wie das Kette-der-Verantwortung-Muster verwendet wird, um eine Reihe von Aktionen zu erstellen, die für die Ausführung der Aufgabe erforderlich sind.

```
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);
    }
}
```

Konstrukte erstellen oder erweitern

Was ist ein Konstrukt

Ein Konstrukt ist der Grundbaustein einer Anwendung. AWS CDK Ein Konstrukt kann eine einzelne AWS Ressource darstellen, z. B. einen Amazon Simple Storage Service (Amazon S3) -Bucket, oder es kann sich um eine Abstraktion auf höherer Ebene handeln, die aus mehreren AWS verwandten Ressourcen besteht. Zu den Komponenten eines Konstrukts können eine Worker-Warteschlange mit der zugehörigen Rechenkapazität oder ein geplanter Auftrag mit Überwachungsressourcen und einem Dashboard gehören. Das AWS CDK beinhaltet eine Sammlung von Konstrukten, die so genannte Construct-Bibliothek. AWS Die Bibliothek enthält Konstrukte für jeden. AWS-Service Sie

können den [Construct Hub](#) verwenden, um weitere Konstrukte von AWS Drittanbietern und der AWS CDK Open-Source-Community zu entdecken.

Was die verschiedenen Arten von Konstrukten sind

Es gibt drei verschiedene Arten von Konstrukten für: AWS CDK

- L1-Konstrukte — Layer-1- oder L1-Konstrukte sind genau die Ressourcen, die durch CloudFormation — nicht mehr und nicht weniger definiert sind. Sie müssen die für die Konfiguration erforderlichen Ressourcen selbst bereitstellen. Diese L1-Konstrukte sind sehr einfach und müssen manuell konfiguriert werden. L1-Konstrukte haben ein Cfn Präfix und entsprechen direkt den Spezifikationen. CloudFormation Neue AWS-Services Dienste werden unterstützt AWS CDK , sobald diese CloudFormation Dienste unterstützt werden. [CfnBucket](#) ist ein gutes Beispiel für ein L1-Konstrukt. Diese Klasse stellt einen S3-Bucket dar, in dem Sie alle Eigenschaften explizit konfigurieren müssen. Es wird empfohlen, ein L1-Konstrukt nur zu verwenden, wenn Sie das L2- oder L3-Konstrukt dafür nicht finden können.
- L2-Konstrukte – Layer-2- oder L2-Konstrukte haben einen gemeinsamen Standardcode und eine gemeinsame Glue-Logik. Diese Konstrukte verfügen über praktische Standardwerte und reduzieren den Wissensbedarf, den Sie über sie benötigen. L2-Konstrukte verwenden Intent-based, um Ihre Ressourcen APIs zu erstellen, und kapseln in der Regel die entsprechenden L1-Module. Ein gutes Beispiel für ein L2-Konstrukt ist [Bucket](#). [Diese Klasse erstellt einen S3-Bucket mit Standardeigenschaften und -methoden wie Bucket. addLifecycleRule \(\)](#), die dem Bucket eine Lebenszyklusregel hinzufügt.
- L3-Konstrukte – Ein Layer-3- oder L3-Konstrukt wird als Muster bezeichnet. L3-Konstrukte sollen Ihnen dabei helfen, allgemeine Aufgaben zu erledigen AWS, die häufig mehrere Arten von Ressourcen umfassen. Diese sind noch spezifischer und eigenwilliger als L2-Konstrukte und dienen einem bestimmten Anwendungsfall. [Zum Beispiel die. aws-ecs-patterns ApplicationLoadBalancedFargateService](#) construct stellt eine Architektur dar, die einen AWS Fargate Container-Cluster umfasst, der einen Application Load Balancer verwendet. Ein anderes Beispiel ist das [aws-apigateway. LambdaRestApi](#) konstruieren. Dieses Konstrukt stellt eine Amazon-API-Gateway-API dar, die von einer Lambda-Funktion unterstützt wird.

Je höher die Konstruktebenen werden, desto mehr Annahmen werden darüber getroffen, wie diese Konstrukte verwendet werden sollen. Auf diese Weise können Sie Schnittstellen mit effektiveren Standardeinstellungen für sehr spezifische Anwendungsfälle bereitstellen.

So erstellen Sie Ihr eigenes Konstrukt

Um Ihr eigenes Konstrukt zu definieren, müssen Sie einen bestimmten Ansatz verfolgen. Das liegt daran, dass alle Konstrukte die `Construct`-Klasse erweitern. Die `Construct`-Klasse ist der Baustein des Konstruktbaums. Konstrukte werden in Klassen implementiert, die die `Construct`-Basisklasse erweitern. Alle Konstrukte benötigen drei Parameter, wenn sie initialisiert werden:

- **Geltungsbereich** — Das übergeordnete Objekt oder der Eigentümer eines Konstrukts, entweder ein Stapel oder ein anderes Konstrukt, das seinen Platz im Konstruktbaum bestimmt. Normalerweise müssen Sie `this` (oder `self` in Python), die das aktuelle Objekt darstellt, für den Geltungsbereich übergeben.
- **ID** – Ein Bezeichner, der innerhalb dieses Bereichs eindeutig sein muss. Der Bezeichner dient als Namespace für alles, was im aktuellen Konstrukt definiert ist, und wird verwendet, um eindeutige Identitäten wie Ressourcennamen und logische Identitäten zuzuweisen. CloudFormation IDs
- **Requisiten** — Eine Reihe von Eigenschaften, die die ursprüngliche Konfiguration des Konstrukts definieren.

Das folgende Beispiel zeigt die Definition eines Konstrukts.

```
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
  }
}
```

Erstellen oder Erweitern eines L2-Konstrukts

Ein L2-Konstrukt stellt eine „Cloud-Komponente“ dar und kapselt alles, was zur Erstellung der CloudFormation Komponente erforderlich ist. Ein L2-Konstrukt kann eine oder mehrere AWS Ressourcen enthalten, und es steht Ihnen frei, das Konstrukt selbst anzupassen. Der Vorteil

der Erstellung oder Erweiterung eines L2-Konstrukts besteht darin, dass Sie die Komponenten CloudFormation stapelweise wiederverwenden können, ohne den Code neu definieren zu müssen. Sie können das Konstrukt einfach als Klasse importieren.

Wenn eine Beziehung „ist eine“ zu einer vorhandenen Konstruktion besteht, können Sie eine bestehende Konstruktion erweitern, um zusätzliche Standardfunktionen hinzuzufügen. Es hat sich bewährt, die Eigenschaften des vorhandenen L2-Konstrukts wiederzuverwenden. Sie können Eigenschaften überschreiben, indem Sie die Eigenschaften direkt im Konstruktor ändern.

Das folgende Beispiel zeigt, wie man sich an bewährten Methoden orientiert und ein vorhandenes L2-Konstrukt namens `s3.Bucket` erweitert. Die Erweiterung legt Standardeigenschaften fest, wie z. B. `versioned`, `publicReadAccess`, `blockPublicAccess`, um sicherzustellen, dass für alle Objekte (in diesem Beispiel S3-Buckets), die mit diesem neuen Konstrukt erstellt wurden, immer diese Standardwerte gesetzt sind.

```
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
    });
  }
}
```

Erstellen eines L3-Konstrukts

Die Komposition ist die bessere Wahl, wenn eine Beziehung „hat eine“ zu einer bestehenden Konstruktzusammensetzung besteht. Komposition bedeutet, dass Sie Ihr eigenes Konstrukt auf anderen vorhandenen Konstrukten aufbauen. Sie können Ihr eigenes Muster erstellen, um alle Ressourcen und ihre Standardwerte in einem einzigen übergeordneten L3-Konstrukt zu kapseln, das gemeinsam genutzt werden kann. Der Vorteil der Erstellung eigener L3-Konstrukte (Muster) besteht darin, dass Sie die Komponenten in Stacks wiederverwenden können, ohne den Code neu definieren zu müssen. Sie können das Konstrukt einfach als Klasse importieren. Diese Muster sollen

Verbrauchern helfen, mehrere Ressourcen auf der Grundlage gemeinsamer Muster und mit einem begrenzten Wissensschatz auf präzise Weise bereitzustellen.

Das folgende Codebeispiel erstellt ein AWS CDK Konstrukt namens `ExampleConstruct`. Sie können dieses Konstrukt als Vorlage verwenden, um Ihre Cloud-Komponenten zu definieren.

```
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
  }
}
```

Das folgende Beispiel zeigt, wie Sie das neu erstellte Konstrukt in Ihre AWS CDK Anwendung oder Ihren Stack importieren.

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

Das folgende Beispiel zeigt, wie Sie eine Instance des Konstrukts instanziiieren können, das Sie aus der Basisklasse erweitert haben.

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

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

Weitere Informationen finden Sie unter [AWS CDK Workshop](#) in der AWS CDK Workshop-Dokumentation.

Escape-Luke

Sie können eine Notluke in der verwenden, AWS CDK um eine Abstraktionsebene nach oben zu gelangen, sodass Sie auf die untere Ebene der Konstrukte zugreifen können. Fluchtschraffuren

werden verwendet, um das Konstrukt um Objekte zu erweitern, die in der aktuellen Version von AWS nicht verfügbar sind, aber in verfügbar sind. CloudFormation

In den folgenden Szenarien empfehlen wir, einen Notlausstieg zu verwenden:

- Ein AWS-Service Feature ist über verfügbar CloudFormation, es gibt jedoch keine Construct Konstrukte dafür.
- Ein AWS-Service Feature ist über verfügbar CloudFormation und es gibt Construct Konstrukte für den Service, aber diese machen das Feature noch nicht verfügbar. Da Construct Konstrukte „von Hand“ entwickelt werden, können sie manchmal hinter den CloudFormation Ressourcenkonstrukten zurückbleiben.

Der folgende Beispielcode zeigt einen häufigen Anwendungsfall für die Verwendung eines Notlausstiegs. In diesem Beispiel dient die Funktionalität, die im übergeordneten Konstrukt noch nicht implementiert ist, dem Hinzufügen von `httpPutResponseHopLimit` für die automatische Skalierung von `LaunchConfiguration`.

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

Das vorhergehende Beispiel weist folgenden Workflow auf:

1. Sie definieren Ihr `AutoScalingGroup`, indem Sie ein L2-Konstrukt verwenden. Das L2-Konstrukt unterstützt das Aktualisieren von `httpPutResponseHopLimit`, daher müssen Sie eine Notluke verwenden.
2. Sie verwenden die `node.findChild()` Methode, um das spezifische `LaunchConfig` untergeordnete Element des `AutoScalingGroup` L2-Konstrukts zu finden und es als Ressource umzuwandeln. `CfnLaunchConfiguration`
3. Sie können jetzt die `launchConfig.metadataOptions`-Eigenschaft am L1-`CfnLaunchConfiguration` einstellen.

Benutzerdefinierte Ressourcen

Sie können benutzerdefinierte Ressourcen verwenden, um benutzerdefinierte Bereitstellungslogik in Vorlagen zu schreiben, die immer dann CloudFormation ausgeführt wird, wenn Sie Stapel erstellen, aktualisieren (falls Sie die benutzerdefinierte Ressource geändert haben) oder löschen. Sie können beispielsweise eine benutzerdefinierte Ressource verwenden, wenn Sie Ressourcen einbeziehen möchten, die in der nicht verfügbar sind. AWS CDK Auf diese Weise können Sie alle zugehörigen Ressourcen weiterhin in einem einzigen Stack verwalten.

Das Erstellen einer benutzerdefinierten Ressource beinhaltet das Schreiben einer Lambda-Funktion, die auf die Lebenszyklusevents CREATE, UPDATE und DELETE einer Ressource reagiert. Wenn Ihre benutzerdefinierte Ressource nur einen einzigen API-Aufruf tätigen muss, sollten Sie die Verwendung des [AwsCustomResource](#) Konstrukts in Betracht ziehen. Dadurch ist es möglich, während einer CloudFormation Bereitstellung beliebige SDK-Aufrufe durchzuführen. Andernfalls empfehlen wir Ihnen, Ihre eigene Lambda-Funktion zu schreiben, um die Arbeit auszuführen, die Sie erledigen müssen.

Weitere Informationen zu benutzerdefinierten Ressourcen finden Sie in der CloudFormation Dokumentation unter [Benutzerdefinierte Ressourcen](#). Ein Beispiel für die Verwendung einer benutzerdefinierten Ressource finden Sie im Repository für [benutzerdefinierte Ressourcen](#) unter GitHub.

Das folgende Beispiel zeigt, wie Sie eine benutzerdefinierte Ressourcenklasse erstellen, um eine Lambda-Funktion zu initiieren und CloudFormation ein Erfolgs- oder Fehlschlagsignal zu senden.

```
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();
}
}
```

Das folgende Beispiel zeigt die Hauptlogik der benutzerdefinierten Ressource.

```
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)
```

Das folgende Beispiel zeigt, wie der AWS CDK Stack die benutzerdefinierte Ressource aufruft.

```
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();
```

Folgen Sie TypeScript den bewährten Methoden

TypeScript ist eine Sprache, die die Funktionen von erweitert JavaScript. Es ist eine stark typisierte und objektorientierte Sprache. Sie können TypeScript damit die Datentypen angeben, die in Ihrem Code übergeben werden, und haben die Möglichkeit, Fehler zu melden, wenn die Typen nicht übereinstimmen. Dieser Abschnitt bietet einen Überblick über TypeScript bewährte Methoden.

Beschreiben Ihrer Daten

Sie können ihn verwenden TypeScript , um die Form von Objekten und Funktionen in Ihrem Code zu beschreiben. Die Verwendung des Typs any ist gleichbedeutend mit dem Verzicht auf eine Typüberprüfung für eine Variable. Wir empfehlen, dass Sie die Verwendung von any in Ihrem Code vermeiden. Ein Beispiel.

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

Enums verwenden

Sie können Enums verwenden, um eine Reihe benannter Konstanten und Standards zu definieren, die in Ihrer Codebasis wiederverwendet werden können. Wir empfehlen, dass Sie Ihre Enums einmal auf globaler Ebene exportieren und dann andere Klassen die Aufzählungen importieren und verwenden lassen. Gehen Sie davon aus, dass Sie eine Reihe möglicher Aktionen erstellen möchten, um die Ereignisse in Ihrer Codebasis zu erfassen. TypeScript stellt sowohl numerische als auch auf Zeichenketten basierende Aufzählungen bereit. Im folgenden Beispiel wird ein Enum verwendet.

```
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)
```

Schnittstellen verwenden

Eine Schnittstelle ist ein Vertrag für die Klasse. Wenn Sie einen Vertrag erstellen, müssen Ihre Benutzer den Vertrag einhalten. Im folgenden Beispiel wird eine Schnittstelle verwendet, um die props zu standardisieren und sicherzustellen, dass der Aufrufer den erwarteten Parameter bereitstellt, wenn er diese Klasse verwendet.

```
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: "amzn-s3-demo-bucket",
  region: "us-east-1",
  encryption: false
})
```

Einige Eigenschaften können nur geändert werden, wenn ein Objekt zum ersten Mal erstellt wird. Sie können dies angeben, indem Sie `readonly` vor dem Namen der Eigenschaft platzieren, wie das folgende Beispiel zeigt.

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

Schnittstellen erweitern

Durch das Erweitern von Schnittstellen wird die Doppelarbeit reduziert, da Sie die Eigenschaften nicht zwischen Schnittstellen kopieren müssen. Außerdem kann der Leser Ihres Codes die Beziehungen in Ihrer Anwendung leicht verstehen.

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

Leere Schnittstellen vermeiden

Wir empfehlen, leere Schnittstellen aufgrund der damit verbundenen potenziellen Risiken zu vermeiden. Im folgenden Beispiel wird eine leere Schnittstelle aufgerufen. `BucketProps` Die `myS3Bucket1`- und `myS3Bucket2`-Objekte sind beide gültig, aber sie folgen unterschiedlichen Standards, da die Schnittstelle keine Verträge erzwingt. Der folgende Code kompiliert und druckt die Eigenschaften, was jedoch zu Inkonsistenzen in Ihrer Anwendung führt.

```
interface BucketProps {}

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

```
const myS3Bucket1 = new S3Bucket({
  name: "amzn-s3-demo-bucket",
  region: "us-east-1",
  encryption: false,
});

const myS3Bucket2 = new S3Bucket({
  name: "amzn-s3-demo-bucket",
});
```

Factories verwenden

In einem Abstract-Factory-Muster ist eine Schnittstelle dafür verantwortlich, eine Factory verwandter Objekte zu erstellen, ohne deren Klassen explizit anzugeben. Sie können beispielsweise eine Lambda-Factory zum Erstellen von Lambda-Funktionen erstellen. Anstatt eine neue Lambda-Funktion in Ihrem Konstrukt zu erstellen, delegieren Sie den Erstellungsprozess an die Factory. Weitere Informationen zu diesem Entwurfsmuster finden Sie unter [Abstract Factory TypeScript in](#) der Refactoring.Guru-Dokumentation.

Destrukturierung für Eigenschaften verwenden

Die in ECMAScript 6 (ES6) eingeführte Destrukturierung ist eine JavaScript Funktion, mit der Sie mehrere Daten aus einem Array oder Objekt extrahieren und sie ihren eigenen Variablen zuweisen können.

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

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

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

Standard-Benennungskonventionen definieren

Durch die Durchsetzung einer Namenskonvention bleibt die Codebasis konsistent und der Aufwand bei der Benennung einer Variablen wird reduziert. Wir empfehlen Folgendes:

- Verwenden Sie Groß-/Kleinschreibung für Variablen- und Funktionsnamen.
- Verwenden Sie UPPER_CASE für globale Konstanten, um unveränderliche Werte zur Kompilierzeit eindeutig anzugeben.
- Wird für Klassennamen und PascalCase Schnittstellennamen verwendet.
- Verwenden Sie Groß-/Kleinschreibung für Schnittstellenmitglieder.
- Wird PascalCase für Typnamen und Aufzählungsnamen verwendet.
- Benennen Sie Dateien mit CamelCase (zum Beispiel `ebsVolumes.tsx` oder `storage.ts`)

Im Folgenden finden Sie Beispiele für diese empfohlenen Benennungskonventionen:

```
// Variables and functions
const userName = 'john';
function getUserData() { }
```

```
// Global constants
const MAX_RETRY_ATTEMPTS = 3;
const API_BASE_URL = 'https://api.example.com';
```

```
// Classes and interfaces
class DatabaseConnection { }
```

```
interface UserProfile { }
```

```
// Types and enums
type ResponseStatus = 'success' | 'error';
enum HttpStatusCode { }
```

Verwenden Sie nicht das Schlüsselwort var

Die `let` Anweisung wird verwendet, um eine lokale Variable in zu deklarieren TypeScript. Es ähnelt dem `var` Schlüsselwort, weist jedoch im Vergleich zum Schlüsselwort einige Einschränkungen beim Gültigkeitsbereich auf `var`. Eine Variable, die in einem Block mit `let` deklariert ist, ist nur für die Verwendung innerhalb dieses Blocks verfügbar. Das `var` Schlüsselwort kann nicht blockbezogen sein, was bedeutet, dass auf es außerhalb eines bestimmten Blocks (repräsentiert durch `{}`) zugegriffen werden kann, aber nicht außerhalb der Funktion, in der es definiert ist. Sie können Variablen neu deklarieren und aktualisieren. `var` Es hat sich bewährt, die Verwendung des `var` Schlüsselworts zu vermeiden.

Erwägen Sie die Verwendung von ESLint und Prettier

ESLint analysiert Ihren Code statisch, um Probleme schnell zu finden. Sie können ESLint damit eine Reihe von Assertionen (sogenannte Lint-Regeln) erstellen, die definieren, wie Ihr Code aussehen oder sich verhalten soll. ESLint enthält auch Vorschläge für automatische Korrekturen, mit denen Sie Ihren Code verbessern können. Schließlich können Sie ESLint damit Lint-Regeln aus gemeinsam genutzten Plugins laden.

Prettier ist ein bekannter Codeformatierer, der eine Vielzahl verschiedener Programmiersprachen unterstützt. Sie können Prettier verwenden, um Ihren Codestil so festzulegen, dass Sie Ihren Code nicht manuell formatieren müssen. Nach der Installation können Sie Ihre `package.json`-Datei aktualisieren und die Befehle `npm run format` und `npm run lint` ausführen.

Das folgende Beispiel zeigt Ihnen, wie Sie den Prettier-Formatierer für Ihr AWS CDK Projekt aktivieren ESLint und verwenden können.

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

Verwenden Sie Zugriffsmodifikatoren

Der `private` Modifikator TypeScript beschränkt die Sichtbarkeit nur auf dieselbe Klasse. Wenn Sie den privaten Modifikator einer Eigenschaft oder Methode hinzufügen, können Sie innerhalb derselben Klasse auf diese Eigenschaft oder Methode zugreifen.

Der öffentliche Modifikator ermöglicht den Zugriff auf Klasseneigenschaften und Methoden von allen Orten aus. Wenn Sie keine Zugriffsmodifikatoren für Eigenschaften und Methoden angeben, verwenden sie standardmäßig den öffentlichen Modifikator.

Der geschützte Modifikator ermöglicht den Zugriff auf Eigenschaften und Methoden einer Klasse innerhalb derselben Klasse und innerhalb von Unterklassen. Verwenden Sie den Modifikator `protected`, wenn Sie erwarten, Unterklassen in Ihrer Anwendung zu erstellen. AWS CDK

Verwenden Sie Dienstprogrammatypen

Bei den Dienstprogrammtypen TypeScript handelt es sich um vordefinierte Typfunktionen, die Transformationen und Operationen an vorhandenen Typen durchführen. Auf diese Weise können Sie neue Typen auf der Grundlage vorhandener Typen erstellen. Sie können beispielsweise Eigenschaften ändern oder extrahieren, Eigenschaften optional oder erforderlich machen oder unveränderliche Versionen von Typen erstellen. Mithilfe von Dienstprogrammtypen können Sie genauere Typen definieren und potenzielle catch bei der Kompilierung erkennen.

Teilweise <Type>

`Partial` markiert alle Mitglieder eines Eingabetyps `Type` als optional. Dieses Tool gibt einen Typ zurück, der alle Teilmengen eines bestimmten Typs darstellt. Es folgt ein Beispiel für `Partial`.

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

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

Erforderlich <Type>

`Required` macht das Gegenteil von `Partial`. Es macht alle Elemente eines Eingabetyps `Type` nicht optional (mit anderen Worten, erforderlich). Es folgt ein Beispiel für `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
```

```
};
```

Nach Sicherheits-Schwachstellen und Formatierungsfehlern scannen

Infrastructure as Code (IaC) und Automatisierung sind für Unternehmen unverzichtbar geworden. Da IaC so robust ist, haben Sie eine große Verantwortung für den Umgang mit Sicherheitsrisiken. Zu den häufigsten IaC-Sicherheitsrisiken können Folgende gehören:

- Zu freigiebigere Rechte AWS Identity and Access Management (IAM)
- Offene Sicherheitsgruppen
- Unverschlüsselte Ressourcen
- Die Zugriffsprotokolle sind nicht aktiviert

Sicherheitsansätze und Tools

Wir empfehlen die Implementierung der folgenden Sicherheitsansätze:

- Erkennung von Sicherheits-Schwachstellen in der Entwicklung – Die Behebung von Sicherheitslücken in der Produktion ist aufgrund der Komplexität der Entwicklung und Verteilung von Softwarepatches teuer und zeitaufwändig. Darüber hinaus bergen Schwachstellen in der Produktion das Risiko, ausgenutzt zu werden. Wir empfehlen Ihnen, Codescanning auf Ihren IaC-Ressourcen zu verwenden, damit Schwachstellen erkannt und behoben werden können, bevor sie für die Produktion freigegeben werden.
- Konformität und automatische Korrektur — AWS Config bietet AWS verwaltete Regeln. [Diese Regeln helfen Ihnen dabei, die Einhaltung von Vorschriften durchzusetzen, und ermöglichen es Ihnen, mithilfe AWS Systems Manager von Automatisierung eine automatische Korrektur zu versuchen.](#) Mithilfe AWS Config von Regeln können Sie auch benutzerdefinierte Automatisierungsdokumente erstellen und verknüpfen.

Gängige Entwicklungstools

Die in diesem Abschnitt behandelten Tools helfen Ihnen dabei, ihre integrierten Funktionen mit Ihren eigenen benutzerdefinierten Regeln zu erweitern. Wir empfehlen Ihnen, Ihre benutzerdefinierten

Regeln an den Standards Ihrer Organisation auszurichten. Hier sind einige gängige zu erwägende Entwicklungstools:

- Verwenden Sie `cdk-nag`, um zu überprüfen, ob Konstrukte innerhalb eines bestimmten Bereichs einem definierten Regelsatz entsprechen. Sie können `cdk-nag` auch zur Unterdrückung von Regeln und zur Berichterstattung über die Einhaltung von Vorschriften verwenden. [Das Tool `cdk-nag` validiert Konstrukte, indem es Aspekte in der erweitert.](#) AWS CDK Weitere Informationen finden Sie im Blog unter [Anwendungssicherheit und Konformität mit dem AWS Cloud Development Kit \(AWS CDK\) und `cdk-nag` verwalten](#). AWS DevOps
- Verwenden Sie das Open-Source-Tool Checkov, um statische Analysen in Ihrer IaC-Umgebung durchzuführen. Checkov hilft bei der Identifizierung von Cloud-Fehlkonfigurationen, indem es Ihren Infrastrukturcode in Kubernetes, Terraform oder scannt. CloudFormation Sie können Checkov verwenden, um Ausgaben in verschiedenen Formaten zu erhalten, einschließlich JSON, JUnit XML oder CLI. Checkov kann effektiv mit Variablen umgehen, indem es ein Diagramm erstellt, das die dynamische Codeabhängigkeit zeigt. Weitere Informationen finden Sie im GitHub [Checkov-Repository von Bridgecrew](#).
- Wird verwendet TFLint , um nach Fehlern und veralteter Syntax zu suchen und um bewährte Methoden durchzusetzen. Beachten Sie, dass TFLint anbieterspezifische Probleme möglicherweise nicht bestätigt werden. Weitere Informationen zu finden Sie im TFLint GitHub [TFLint](#)Repository von Terraform Linters.
- Verwenden Sie Amazon Q Developer, um [Sicherheitsscans](#) durchzuführen. Bei Verwendung in einer integrierten Entwicklungsumgebung (IDE) bietet Amazon Q Developer KI-gestützte Unterstützung bei der Softwareentwicklung. Es kann über Code chatten, Inline-Code-Vervollständigungen bereitstellen, neuen Code generieren, Ihren Code auf Sicherheitslücken scannen und Code-Upgrades und -Verbesserungen vornehmen.

Dokumentation entwickeln und verfeinern

Die Dokumentation ist entscheidend für den Erfolg Ihres Projekts. Die Dokumentation erklärt nicht nur, wie Ihr Code funktioniert, sondern hilft Entwicklern auch, die Merkmale und Feature Ihrer Anwendungen besser zu verstehen. Durch die Entwicklung und Verfeinerung hochwertiger Dokumentation kann der Softwareentwicklungsprozess gestärkt, die Qualität der Software aufrechterhalten und der Wissenstransfer zwischen Entwicklern unterstützt werden.

Es gibt zwei Kategorien von Dokumentation: Dokumentation innerhalb des Codes und unterstützende Dokumentation zum Code. Die Dokumentation innerhalb des Codes erfolgt in Form von

Kommentaren. Unterstützende Dokumentation zum Code können README-Dateien und externe Dokumente sein. Es ist nicht ungewöhnlich, dass Entwickler Dokumentation als Mehraufwand betrachten, da der Code selbst leicht zu verstehen ist. Dies könnte für kleine Projekte zutreffen, aber die Dokumentation ist für große Projekte, an denen mehrere Teams beteiligt sind, von entscheidender Bedeutung.

Es ist eine bewährte Methode für den Autor des Codes, die Dokumentation zu schreiben, da er die Funktionen gut kennt. Entwickler können mit dem zusätzlichen Aufwand zu kämpfen haben, der mit der Pflege einer separaten unterstützenden Dokumentation verbunden ist. Um diese Herausforderung zu bewältigen, können Entwickler die Kommentare zum Code hinzufügen und diese Kommentare können automatisch extrahiert werden, sodass jede Version von Code und Dokumentation synchronisiert ist.

Es gibt eine Vielzahl verschiedener Tools, mit denen Entwickler Kommentare aus dem Code extrahieren und die Dokumentation dafür generieren können. Dieser Leitfaden konzentriert sich auf TypeDoc das bevorzugte Tool für AWS CDK Konstrukte.

Warum ist Codedokumentation für AWS CDK Konstrukte erforderlich

AWS CDK Allgemeine Konstrukte werden von mehreren Teams in einer Organisation erstellt und von verschiedenen Teams zur Nutzung gemeinsam genutzt. Eine gute Dokumentation hilft den Benutzern der Konstrukt-Bibliothek dabei, Konstrukte einfach zu integrieren und ihre Infrastruktur mit minimalem Aufwand aufzubauen. Alle Dokumente synchron zu halten ist eine große Aufgabe. Wir empfehlen, dass Sie das Dokument innerhalb des Codes verwalten, der mithilfe der TypeDoc Bibliothek extrahiert wird.

Verwendung TypeDoc mit der AWS Construct-Bibliothek

TypeDoc ist ein Dokumentengenerator für TypeScript. Sie können TypeDoc es verwenden, um Ihre TypeScript Quelldateien zu lesen, die Kommentare in diesen Dateien zu analysieren und dann eine statische Site zu generieren, die Dokumentation für Ihren Code enthält.

Der folgende Code zeigt Ihnen, wie Sie die TypeDoc Integration in die AWS Construct-Bibliothek durchführen und dann die folgenden Pakete zu Ihrer `package.json` Datei hinzufügen können.

```
devDependencies
```

```
{  
  
  "devDependencies": {
```

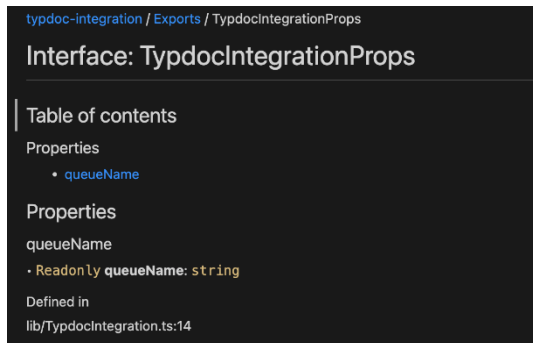
```
"typedoc-plugin-markdown": "^3.11.7",
"typescript": "~3.9.7"
},
}
```

Verwenden Sie zum Hinzufügen von `typedoc.json` im CDK-Bibliotheksordner den folgenden Code.

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

Um die README-Dateien zu generieren, führen Sie den `npx typedoc` Befehl im Stammverzeichnis des AWS CDK Construct-Library-Projekts aus.

Das folgende Beispieldokument wurde von TypeDoc generiert.



Weitere Informationen zu TypeDoc Integrationsoptionen finden Sie in der TypeDoc Dokumentation unter [Kommentare zu](#) Dokumenten.

Einen testgetriebenen Entwicklungsansatz verwenden

Wir empfehlen Ihnen, mit dem einen Test-Driven-Driven-Development-Ansatz (TDD) zu verfolgen. AWS CDK TDD ist ein Softwareentwicklungsansatz, bei dem Sie Testfälle entwickeln, um Ihren Code zu spezifizieren und zu validieren. Einfach ausgedrückt, erstellen Sie zunächst Testfälle für jede Funktionalität. Wenn der Test fehlschlägt, schreiben Sie dann den neuen Code, um den Test zu bestehen und den Code einfach und fehlerfrei zu gestalten.

Sie können TDD verwenden, um zuerst den Testfall zu schreiben. Dies hilft Ihnen bei der Validierung der Infrastruktur mit verschiedenen Design-Einschränkungen in Bezug auf die Durchsetzung von

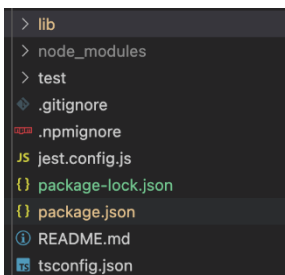
Sicherheitsrichtlinien für die Ressourcen und die Einhaltung einer eindeutigen Namenskonvention für das Projekt. Der Standardansatz zum Testen von AWS CDK Anwendungen besteht darin, das AWS CDK [Assertions-Modul](#) und beliebige Test-Frameworks wie [Jest](#) for TypeScript und JavaScript oder [pytest](#) for Python zu verwenden.

Es gibt zwei Kategorien von Tests, die Sie für Ihre Anwendungen schreiben können: AWS CDK

- Verwenden Sie detaillierte Assertionen, um einen bestimmten Aspekt der generierten CloudFormation Vorlage zu testen, z. B. „Diese Ressource hat diese Eigenschaft mit diesem Wert“. Diese Tests können Regressionen erkennen und sind auch nützlich, wenn Sie neue Feature mit TDD entwickeln (schreiben Sie zuerst einen Test, dann bestehen Sie ihn, indem Sie eine korrekte Implementierung schreiben). Fine-grained assertions sind die Tests, die Sie am häufigsten schreiben werden.
- Verwenden Sie Snapshot-Tests, um die synthetisierte CloudFormation Vorlage anhand einer zuvor gespeicherten Basisvorlage zu testen. Snapshot-Tests ermöglichen einen freien Faktorwechsel, da Sie sicher sein können, dass der umgestaltete Code genauso funktioniert wie das Original. Wenn die Änderungen beabsichtigt waren, können Sie eine neue Ausgangsbasis für zukünftige Tests akzeptieren. AWS CDK Upgrades können jedoch auch dazu führen, dass sich synthetisierte Vorlagen ändern, sodass Sie sich nicht nur auf Snapshots verlassen können, um sicherzustellen, dass Ihre Implementierung korrekt ist.

Komponententest

Dieser Leitfaden konzentriert sich TypeScript speziell auf die Integration von Komponententests. Um Tests zu ermöglichen, stellen Sie sicher, dass Ihre `package.json` Datei die folgenden Bibliotheken enthält: `@types/jest`, `jest`, und `ts-jest` in `devDependencies`. Um diese Pakete hinzuzufügen, führen Sie den `cdk init lib --language=typescript`-Befehl aus. Nachdem Sie den vorherigen Befehl ausgeführt haben, sehen Sie die folgende Struktur.



Der folgende Code ist ein Beispiel für eine `package.json` Datei, die mit der Jest-Bibliothek aktiviert wurde.

```
{
  ...
  "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",
    ...
  }
}
```

Unter dem Ordner Test können Sie den Testfall schreiben. Das folgende Beispiel zeigt einen Testfall für ein AWS CodePipeline Konstrukt.

```
import { Stack } from 'aws-cdk-lib';
import { Template } from 'aws-cdk-lib/assertions';
import * as CodePipeline from 'aws-cdk-lib/aws-codepipeline';
import * as CodePipelineActions from 'aws-cdk-lib/aws-codepipeline-actions';
import { MyPipelineStack } from '../lib/my-pipeline-stack';
test('Pipeline Created with GitHub Source', () => {
  // ARRANGE
  const stack = new Stack();
  // ACT
  new MyPipelineStack(stack, 'MyTestStack');
  // ASSERT
  const template = Template.fromStack(stack);
  // Verify that the pipeline resource is created
  template.resourceCountIs('AWS::CodePipeline::Pipeline', 1);
  // Verify that the pipeline has the expected stages with GitHub source
  template.hasResourceProperties('AWS::CodePipeline::Pipeline', {
    Stages: [
      {
        Name: 'Source',
        Actions: [
          {
            Name: 'SourceAction',
            ActionTypeId: {
```

```
    Category: 'Source',
    Owner: 'ThirdParty',
    Provider: 'GitHub',
    Version: '1'
  },
  Configuration: {
    Owner: {
      'Fn::Join': [
        '',
        [
          '{{resolve:secretsmanager:',
          {
            Ref: 'GitHubTokenSecret'
          },
          ':SecretString:owner}}'
        ]
      ]
    },
    Repo: {
      'Fn::Join': [
        '',
        [
          '{{resolve:secretsmanager:',
          {
            Ref: 'GitHubTokenSecret'
          },
          ':SecretString:repo}}'
        ]
      ]
    },
    Branch: 'main',
    OAuthToken: {
      'Fn::Join': [
        '',
        [
          '{{resolve:secretsmanager:',
          {
            Ref: 'GitHubTokenSecret'
          },
          ':SecretString:token}}'
        ]
      ]
    }
  ]
}
```

```
    },
    OutputArtifacts: [
      {
        Name: 'SourceOutput'
      }
    ],
    RunOrder: 1
  }
]
},
{
  Name: 'Build',
  Actions: [
    {
      Name: 'BuildAction',
      ActionTypeId: {
        Category: 'Build',
        Owner: 'AWS',
        Provider: 'CodeBuild',
        Version: '1'
      },
      InputArtifacts: [
        {
          Name: 'SourceOutput'
        }
      ],
      OutputArtifacts: [
        {
          Name: 'BuildOutput'
        }
      ],
      RunOrder: 1
    }
  ]
}
// Add more stage checks as needed
]);
// Verify that a GitHub token secret is created
template.resourceCountIs('AWS::SecretsManager::Secret', 1);
});
);
```

Um einen Test auszuführen, führen Sie den `npm run test`-Befehl in Ihrem Projekt aus. Der Test gibt die folgenden Ergebnisse zurück.

```
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
```

Weitere Informationen zu Testfällen finden Sie unter [Testen von Konstrukten](#) im AWS Cloud Development Kit (AWS CDK) Entwicklerhandbuch.

Integrationstest

Integrationstests für AWS CDK Konstrukte können auch mithilfe eines `integ-tests` Moduls eingebunden werden. Ein Integrationstest sollte als AWS CDK Anwendung definiert werden. Es sollte eine one-to-one Beziehung zwischen einem Integrationstest und einer AWS CDK Anwendung bestehen. Weitere Informationen finden Sie im [integ-tests-alpha Modul](#) in der AWS CDK API-Referenz.

Release- und Versionskontrolle für Konstrukte verwenden

Versionskontrolle für AWS CDK

AWS CDK Gemeinsame Konstrukte können von mehreren Teams erstellt und von der gesamten Organisation gemeinsam genutzt werden. In der Regel veröffentlichen Entwickler neue Funktionen oder Fehlerkorrekturen in ihren gemeinsamen AWS CDK Konstrukten. Diese Konstrukte werden von AWS CDK Anwendungen oder anderen vorhandenen AWS CDK Konstrukten als Teil einer Abhängigkeit verwendet. Aus diesem Grund ist es wichtig, dass Entwickler ihr Konstrukt unabhängig voneinander aktualisieren und mit den richtigen semantischen Versionen veröffentlichen. AWS CDK Downstream-Anwendungen oder andere AWS CDK Konstrukte können ihre Abhängigkeit aktualisieren, um die neu veröffentlichte AWS CDK Konstruktversion zu verwenden.

Semantische Versionsverwaltung (Semver) ist ein Regelwerk oder eine Methode zur Bereitstellung eindeutiger Softwarenummern für Computersoftware. Versionen sind wie folgt definiert:

- Eine HAUPT-Version besteht aus inkompatiblen API-Änderungen oder einer bahnbrechenden Änderung.

- Eine MINOR-Version besteht aus Funktionen, die abwärtskompatibel hinzugefügt wurden.
- Eine PATCH-Version besteht aus abwärtskompatiblen Bugfixes.

Weitere Informationen zur semantischen Versionierung finden Sie unter [Semantic Versioning Specification \(SemVer\) in der Semantic Versioning-Dokumentation](#).

Repository AWS CDK und Paketierung für Konstrukte

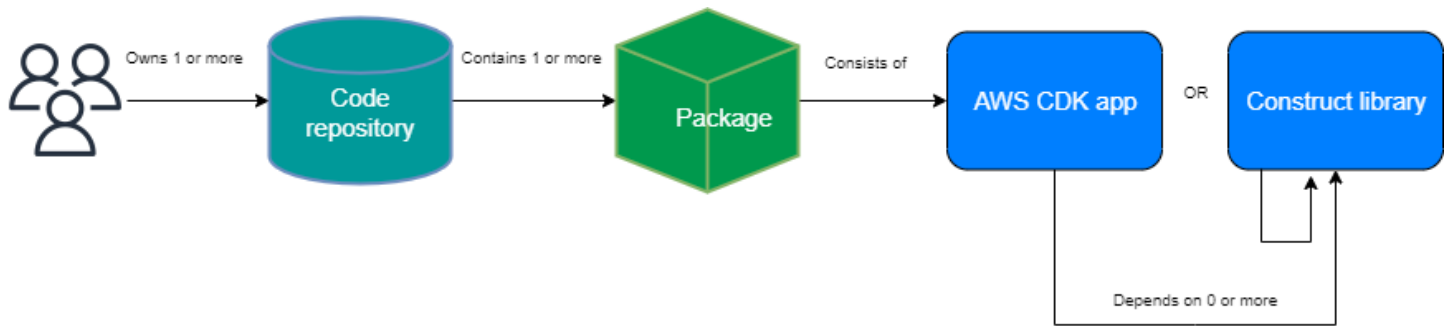
Da AWS CDK Konstrukte von verschiedenen Teams entwickelt und von mehreren AWS CDK Anwendungen verwendet werden, können Sie für jedes AWS CDK Konstrukt ein separates Repository verwenden. Dies kann Ihnen auch dabei helfen, die Zugriffskontrolle durchzusetzen. Jedes Repository könnte den gesamten Quellcode enthalten, der sich auf dasselbe AWS CDK Konstrukt bezieht, zusammen mit all seinen Abhängigkeiten. Indem Sie eine einzelne Anwendung (d. h. ein AWS CDK Konstrukt) in einem einzigen Repository speichern, können Sie den Umfang der Auswirkungen von Änderungen während der Bereitstellung verringern.

Das generiert AWS CDK nicht nur CloudFormation Vorlagen für die Bereitstellung der Infrastruktur, sondern bündelt auch Laufzeitressourcen wie Lambda-Funktionen und Docker-Images und stellt sie zusammen mit Ihrer Infrastruktur bereit. Es ist nicht nur möglich, den Code, der Ihre Infrastruktur definiert, und den Code, der Ihre Laufzeitlogik implementiert, in einem einzigen Konstrukt zu kombinieren — das ist eine bewährte Methode. Diese beiden Arten von Code müssen sich nicht in separaten Repositories oder sogar in separaten Paketen befinden.

Um Pakete über Repository-Grenzen hinweg nutzen zu können, benötigen Sie ein privates Paket-Repository — ähnlich wie npm oder Maven Central PyPi, aber unternehmensintern. Sie benötigen außerdem einen Release-Prozess, der das Paket erstellt, testet und im privaten Paket-Repository veröffentlicht. Sie können private Repositories, z. B. PyPi Server, mithilfe einer lokalen virtuellen Maschine (VM) oder Amazon S3 erstellen. Wenn Sie eine private Paketregistrierung entwerfen oder erstellen, müssen Sie unbedingt das Risiko einer Serviceunterbrechung aufgrund der hohen Verfügbarkeit und Skalierbarkeit berücksichtigen. Ein serverloser verwalteter Service, der zum Speichern von Paketen in der Cloud gehostet wird, kann den Wartungsaufwand erheblich reduzieren. Sie können ihn beispielsweise [AWS CodeArtifact](#) zum Hosten von Paketen für die gängigsten Programmiersprachen verwenden. Sie können es auch verwenden CodeArtifact, um externe Repository-Verbindungen einzurichten und diese innerhalb CodeArtifact zu replizieren.

Abhängigkeiten von Paketen im Paket-Repository werden vom Paketmanager Ihrer Sprache verwaltet (z. B. npm for TypeScript oder JavaScript applications). Ihr Paketmanager stellt sicher, dass Builds wiederholbar sind, indem er die spezifischen Versionen jedes Pakets aufzeichnet, von

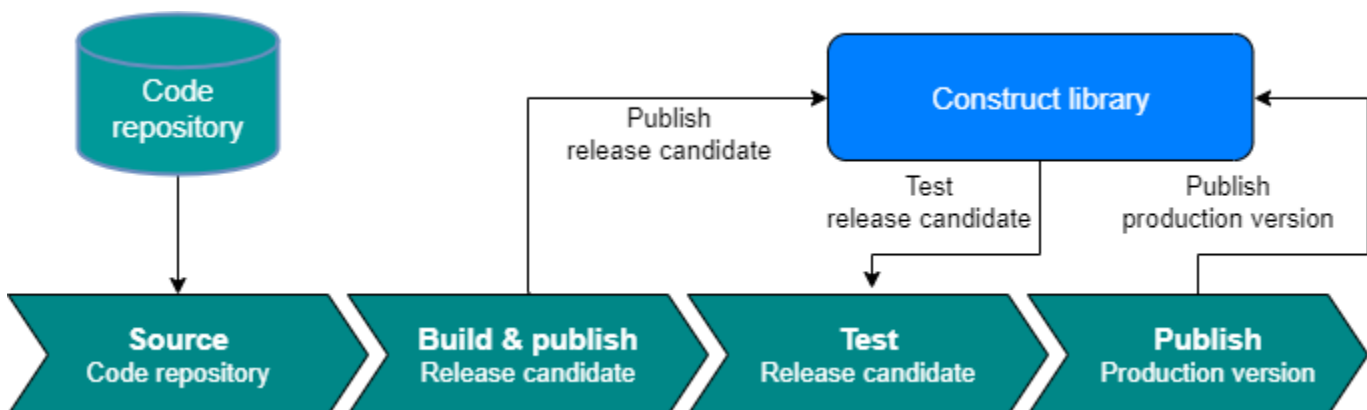
dem Ihre Anwendung abhängt, und Ihnen dann ermöglicht, diese Abhängigkeiten kontrolliert zu aktualisieren, wie das folgende Diagramm zeigt.



Erstellen Sie das Releasing für AWS CDK

Wir empfehlen Ihnen, Ihre eigene automatisierte Pipeline zu erstellen, um neue AWS CDK Construct-Versionen zu erstellen und zu veröffentlichen. Wenn Sie ein geeignetes Genehmigungsverfahren für Pull-Requests eingerichtet haben, kann die Pipeline, sobald Sie Ihren Quellcode festgeschrieben und in den Hauptzweig des Repositorys übertragen haben, eine Release-Candidate-Version erstellen und veröffentlichen. Auf diese Version kann gepusht CodeArtifact und getestet werden, bevor die produktionsreife Version veröffentlicht wird. Optional können Sie Ihre neue AWS CDK Construct-Version lokal testen, bevor Sie den Code mit dem Hauptzweig zusammenführen. Dies veranlasst die Pipeline, die produktionsreife Version zu veröffentlichen. Beachten Sie, dass gemeinsam genutzte Konstrukte und Pakete unabhängig von der nutzenden Anwendung getestet werden müssen, als ob sie der Öffentlichkeit zugänglich gemacht würden.

Das folgende Diagramm zeigt ein Beispiel für eine AWS CDK Versions-Release-Pipeline.



Sie können die folgenden Beispielbefehle verwenden, um npm-Pakete zu erstellen, zu testen und zu veröffentlichen. Melden Sie sich zunächst beim Artefakt-Repository an, indem Sie den folgenden Befehl ausführen.

```
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>
```

Führen Sie dann die folgenden Schritte aus:

1. Installieren Sie die erforderlichen Pakete auf der `package.json` Datei: `npm install`
2. Erstellen Sie die Release-Candidate-Version: `npm version prerelease --preid rc`
3. Erstellen Sie das npm-Paket: `npm run build`
4. Testen Sie das npm-Paket: `npm run test`
5. Veröffentlichen Sie das npm-Paket: `npm publish`

Erzwingen Sie die Versionsverwaltung der Bibliothek

Das Lebenszyklusmanagement ist eine große Herausforderung bei der Pflege von AWS CDK Codebasen. Nehmen wir zum Beispiel an, dass Sie ein AWS CDK Projekt mit Version 1.97 beginnen und Version 1.169 dann später verfügbar wird. Version 1.169 bietet neue Features und Bugfixes, aber Sie haben Ihre Infrastruktur mit der alten Version bereitgestellt. Jetzt wird es schwierig, die Konstrukte zu aktualisieren, da diese Lücke aufgrund der bahnbrechenden Änderungen, die in neuen Versionen eingeführt werden könnten, immer größer wird. Dies kann eine Herausforderung sein, wenn Sie viele Ressourcen in Ihrer Umgebung haben. Das in diesem Abschnitt vorgestellte Muster kann Ihnen helfen, Ihre AWS CDK Bibliotheksversion mithilfe von Automatisierung zu verwalten. Hier ist der Arbeitsablauf dieses Musters:

1. Wenn Sie ein neues CodeArtifact Service Catalog-Produkt starten, werden die AWS CDK Bibliotheksversionen und ihre Abhängigkeiten in der `package.json` Datei gespeichert.
2. Sie stellen eine gemeinsame Pipeline bereit, die alle Repositories verfolgt, sodass Sie automatische Upgrades auf sie anwenden können, falls es keine grundlegenden Änderungen gibt.
3. AWS CodeBuild In einer Phase wird nach der Abhängigkeitsstruktur gesucht und nach den wichtigsten Änderungen gesucht.
4. Die Pipeline erstellt einen Feature-Zweig und führt dann `cdk synth` mit der neuen Version aus, um zu bestätigen, dass keine Fehler vorliegen.
5. Die neue Version wird in der Testumgebung bereitgestellt und führt abschließend einen Integrationstest durch, um sicherzustellen, dass die Bereitstellung fehlerfrei ist.

6. Sie können zwei Amazon Simple Queue Service (Amazon SQS)-Warteschlangen verwenden, um den Überblick über die Stacks zu behalten. Benutzer können die Stacks in der Ausnahmewarteschlange manuell überprüfen und wichtige Änderungen korrigieren. Elemente, die den Integrationstest bestehen, dürfen zusammengeführt und veröffentlicht werden.

Häufig gestellte Fragen

Welche Probleme können TypeScript gelöst werden?

In der Regel können Sie Fehler in Ihrem Code beheben, indem Sie automatisierte Tests schreiben, manuell überprüfen, ob der Code erwartungsgemäß funktioniert, und schließlich Ihren Code von einer anderen Person validieren lassen. Die Überprüfung der Verbindungen zwischen den einzelnen Teilen Ihres Projekts ist zeitaufwändig. Um den Validierungsprozess zu beschleunigen, können Sie eine typgeprüfte Sprache verwenden, z. B. TypeScript um die Codevalidierung zu automatisieren und während der Entwicklung sofortiges Feedback zu geben.

Warum sollte ich verwenden TypeScript?

TypeScript ist eine Open-Source-Sprache, die JavaScript Code vereinfacht und so das Lesen und Debuggen erleichtert. TypeScript bietet außerdem hochproduktive Entwicklungstools JavaScript IDEs und -praktiken, wie z. B. statische Prüfungen. Darüber hinaus TypeScript bietet es die Vorteile von ECMAScript 6 (ES6) und kann Ihre Produktivität steigern. Schließlich TypeScript kann es Ihnen helfen, schmerzhafte Fehler zu vermeiden, auf die Entwickler beim Schreiben JavaScript durch Typprüfung des Codes häufig stoßen.

Sollte ich das AWS CDK oder verwenden CloudFormation?

Wir empfehlen Ihnen, das AWS Cloud Development Kit (AWS CDK) anstelle von zu verwenden AWS CloudFormation, wenn Ihre Organisation über die erforderliche Entwicklungskompetenz verfügt, um die Vorteile von zu nutzen AWS CDK. Das liegt daran, dass das flexibler AWS CDK ist als CloudFormation, da Sie eine Programmiersprache und OOP-Konzepte verwenden können. Denken Sie daran, dass Sie CloudFormation damit AWS Ressourcen auf geordnete und vorhersehbare Weise erstellen können. CloudFormationIn werden Ressourcen im JSON- oder YAML-Format in Textdateien geschrieben.

Was ist, wenn das ein neu AWS-Service eingeführtes Programm AWS CDK nicht unterstützt?

Sie können eine [unformatierte Version](#) oder eine CloudFormation [benutzerdefinierte Ressource](#) verwenden.

Welche verschiedenen Programmiersprachen werden von der unterstützt AWS CDK?

AWS CDK ist allgemein in JavaScript, Python TypeScript, Java, C# und Go (in Developer Preview) verfügbar.

Wie viel AWS CDK kostet das?

Es fallen keine zusätzlichen Gebühren für die an AWS CDK. Sie zahlen für die AWS Ressourcen (wie Amazon EC2 EC2-Instances oder Elastic Load Balancing Load Balancer), die AWS CDK bei der Verwendung erstellt werden, genauso wie wenn Sie sie manuell erstellt hätten. Sie zahlen nur für das, was Sie nutzen, wenn Sie es nutzen. Es werden keine Mindestgebühren oder Vorauszahlungen benötigt.

Nächste Schritte

Wir empfehlen, dass Sie mit dem Bauen mit AWS Cloud Development Kit (AWS CDK) in beginnen TypeScript. Weitere Informationen finden Sie im [AWS CDK Immersion Day Workshop](#).

Ressourcen

Referenzen

- [AWS Lösungskonstrukte](#) (AWS Lösungen)
- [AWS Cloud Development Kit \(AWS CDK\)](#) (GitHub)
- [AWS API-Referenz zur Construct-Bibliothek](#) (AWS CDK Referenzdokumentation)
- [AWS CDK Referenzdokumentation](#) (AWS CDK Referenzdokumentation)
- AWS CDK Workshop zum [Immersionstag \(AWS Workshop-Studio\)](#)

Tools

- [cdk-nag](#) () GitHub
- [TypeScript ESLint](#)(Dokumentation) TypeScript ESLint

Anleitungen und Muster

- [AWS Lösungen konstruiert Muster](#) (AWS Dokumentation)

Dokumentverlauf

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

Änderung	Beschreibung	Datum
Bewährte Methoden wurden aktualisiert	Wir haben die Empfehlung, cfn-nag zu verwenden, im Abschnitt Allgemeine Entwicklungstools entfernt. Im Abschnitt Standardbenennungskonventionen definieren haben wir Standardbenennungskonventionen für globale Konstanten und Benennungsbispiele hinzugefügt.	23. Oktober 2025
Code aktualisieren	Wir haben die Codebeispiele im Abschnitt „ TypeScript Bewährte Methoden befolgen “ aktualisiert.	16. Februar 2024
Abschnitte hinzufügen	Wir haben die Abschnitte Use utility types und Integrationstest hinzugefügt.	10. Januar 2024
Kleines Update	Das Codebeispiel für die Erstellung eines L3-Konstrukts wurde aktualisiert.	16. Juni 2023
Erste Veröffentlichung	—	8. Dezember 2022

AWS Glossar zu präskriptiven Leitlinien

Die folgenden Begriffe werden häufig in Strategien, Leitfäden und Mustern von AWS Prescriptive Guidance verwendet. Um Einträge vorzuschlagen, verwenden Sie bitte den Link Feedback geben am Ende des Glossars.

Zahlen

7 Rs

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

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

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

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

A

ABAC

Siehe [attributbasierte](#) Zugriffskontrolle.

abstrahierte Dienste

Siehe [Managed Services](#).

ACID

Siehe [Atomarität, Konsistenz, Isolierung und Haltbarkeit](#).

Aktiv-Aktiv-Migration

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

Aktiv-Passiv-Migration

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

Aggregatfunktion

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

AI

Siehe [künstliche Intelligenz](#).

AIOps

Siehe [Operationen im Bereich künstliche Intelligenz](#).

Anonymisierung

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

Anti-Muster

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

Anwendungssteuerung

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

Anwendungsportfolio

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

künstliche Intelligenz (KI)

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

Operationen mit künstlicher Intelligenz (AIOps)

Der Prozess des Einsatzes von Techniken des Machine Learning zur Lösung betrieblicher Probleme, zur Reduzierung betrieblicher Zwischenfälle und menschlicher Eingriffe sowie zur Steigerung der Servicequalität. Weitere Informationen zur Verwendung in der AWS Migrationsstrategie finden Sie im [Operations Integration Guide](#). AIOps

Asymmetrische Verschlüsselung

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

Atomizität, Konsistenz, Isolierung, Haltbarkeit (ACID)

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

Attributbasierte Zugriffskontrolle (ABAC)

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

maßgebliche Datenquelle

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

Availability Zone

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

AWS Framework für die Einführung der Cloud (AWS CAF)

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

Cloud-Einführung vorzubereiten. Weitere Informationen finden Sie auf der [AWS -CAF-Webseite](#) und dem [AWS -CAF-Whitepaper](#).

AWS Workload-Qualifizierungsrahmen (AWS WQF)

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

B

schlechter Bot

Ein [Bot](#), der Einzelpersonen oder Organisationen stören oder ihnen Schaden zufügen soll.

BCP

Siehe [Planung der Geschäftskontinuität](#).

Verhaltensdiagramm

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

Big-Endian-System

Ein System, welches das höchstwertige Byte zuerst speichert. Siehe auch [Endianness](#).

Binäre Klassifikation

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

Bloom-Filter

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

Blau/Grün-Bereitstellung

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

Bot

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

Botnetz

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

branch

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

Zugang durch Glasbruch

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

Brownfield-Strategie

Die bestehende Infrastruktur in Ihrer Umgebung. Wenn Sie eine Brownfield-Strategie für eine Systemarchitektur anwenden, richten Sie sich bei der Gestaltung der Architektur nach den Einschränkungen der aktuellen Systeme und Infrastruktur. Wenn Sie die bestehende Infrastruktur erweitern, könnten Sie Brownfield- und [Greenfield](#)-Strategien mischen.

Puffer-Cache

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

Geschäftsfähigkeit

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

Planung der Geschäftskontinuität (BCP)

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

C

CAF

[Weitere Informationen finden Sie unter Framework AWS für die Cloud-Einführung.](#)

Bereitstellung auf Kanaren

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

CCoE

Weitere Informationen finden Sie [im Cloud Center of Excellence](#).

CDC

Siehe [Erfassung von Änderungsdaten](#).

Erfassung von Datenänderungen (CDC)

Der Prozess der Nachverfolgung von Änderungen an einer Datenquelle, z. B. einer Datenbanktabelle, und der Aufzeichnung von Metadaten zu der Änderung. Sie können CDC für verschiedene Zwecke verwenden, z. B. für die Prüfung oder Replikation von Änderungen in einem Zielsystem, um die Synchronisation aufrechtzuerhalten.

Chaos-Technik

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

CI/CD

Siehe [Continuous Integration und Continuous Delivery](#).

Klassifizierung

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

clientseitige Verschlüsselung

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

Cloud-Exzellenzzentrum (CCoE)

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

Cloud Computing

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

Cloud-Betriebsmodell

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

Phasen der Einführung der Cloud

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

- Projekt – Durchführung einiger Cloud-bezogener Projekte zu Machbarkeitsnachweisen und zu Lernzwecken
- Fundament — Tätigen Sie grundlegende Investitionen, um Ihre Cloud-Einführung zu skalieren (z. B. Einrichtung einer landing zone, Definition eines CCo E, Einrichtung eines Betriebsmodells)
- Migration – Migrieren einzelner Anwendungen
- Neuentwicklung – Optimierung von Produkten und Services und Innovation in der Cloud

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

CMDB

Siehe [Datenbank für das Konfigurationsmanagement](#).

Code-Repository

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

Kalter Cache

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

Kalte Daten

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

Computer Vision (CV)

Ein Bereich der [KI](#), der maschinelles Lernen nutzt, um Informationen aus visuellen Formaten wie digitalen Bildern und Videos zu analysieren und zu extrahieren. Amazon SageMaker AI bietet beispielsweise Bildverarbeitungsalgorithmen für CV.

Drift in der Konfiguration

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

Verwaltung der Datenbankkonfiguration (CMDB)

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

Konformitätspaket

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

Kontinuierliche Bereitstellung und kontinuierliche Integration (CI/CD)

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

CV

Siehe [Computer Vision](#).

D

Daten im Ruhezustand

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

Datenklassifizierung

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

Datendrift

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

Daten während der Übertragung

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

Datennetz

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

Datenminimierung

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

Datenperimeter

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

Vorverarbeitung der Daten

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

Herkunft der Daten

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

betreffene Person

Eine Person, deren Daten gesammelt und verarbeitet werden.

Data Warehouse

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

Datenbankdefinitionssprache (DDL)

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

Datenbankmanipulationssprache (DML)

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

DDL

Siehe [Datenbankdefinitionssprache](#).

Deep-Ensemble

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

Deep Learning

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

defense-in-depth

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

delegierter Administrator

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

Einsatz

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

Entwicklungsumgebung

Siehe [Umgebung](#).

Detektivische Kontrolle

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

Abbildung des Wertstroms in der Entwicklung (DVSM)

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

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

digitaler Zwilling

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

Maßtabelle

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

Katastrophe

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

Notfallwiederherstellung (DR)

Die Strategie und der Prozess, mit denen Sie Ausfallzeiten und Datenverluste aufgrund einer [Katastrophe](#) minimieren. Weitere Informationen finden Sie unter [Disaster Recovery von Workloads unter AWS: Wiederherstellung in der Cloud im AWS Well-Architected Framework](#).

DML

Siehe Sprache zur [Datenbankmanipulation](#).

Domainorientiertes Design

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

DR

Siehe [Disaster Recovery](#).

Erkennung von Driften

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

DVSM

Siehe [Abbildung des Wertstroms in der Entwicklung](#).

E

EDA

Siehe [explorative Datenanalyse](#).

EDI

Siehe [elektronischer Datenaustausch](#).

Edge-Computing

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

elektronischer Datenaustausch (EDI)

Der automatisierte Austausch von Geschäftsdokumenten zwischen Organisationen. Weitere Informationen finden Sie unter [Was ist elektronischer Datenaustausch](#).

Verschlüsselung

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

Verschlüsselungsschlüssel

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

Endianismus

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

Endpunkt

[Siehe](#) Service-Endpunkt.

Endpunkt-Services

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

Unternehmensressourcenplanung (ERP)

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

Envelope-Verschlüsselung

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

Umgebung

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

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

- Produktionsumgebung – Eine Instance einer laufenden Anwendung, auf die Endbenutzer zugreifen können. In einer CI/CD Pipeline ist die Produktionsumgebung die letzte Bereitstellungsumgebung.
- Höhere Umgebungen – Alle Umgebungen, auf die auch andere Benutzer als das Kernentwicklungsteam zugreifen können. Dies kann eine Produktionsumgebung, Vorproduktionsumgebungen und Umgebungen für Benutzerakzeptanztests umfassen.

Epics

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

ERP

Siehe [Enterprise Resource Planning](#).

Explorative Datenanalyse (EDA)

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

F

Faktentabelle

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

schnell scheitern

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

Grenze zur Fehlerisolierung

Dabei handelt es sich um eine Grenze AWS Cloud, z. B. eine Availability Zone AWS-Region, eine Steuerungsebene oder eine Datenebene, die die Auswirkungen eines Fehlers begrenzt und die Widerstandsfähigkeit von Workloads verbessert. Weitere Informationen finden Sie unter [Grenzen zur AWS Fehlerisolierung](#).

Feature-Zweig

Siehe [Zweig](#).

Features

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

Bedeutung der Feature

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

Featuretransformation

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

Eingabeaufforderung mit wenigen Klicks

Bereitstellung einer kleinen Anzahl von Beispielen, die die Aufgabe und das gewünschte Ergebnis veranschaulichen, bevor das [LLM](#) aufgefordert wird, eine ähnliche Aufgabe auszuführen. Bei dieser Technik handelt es sich um eine Anwendung des kontextbezogenen Lernens, bei der Modelle anhand von Beispielen (Aufnahmen) lernen, die in Eingabeaufforderungen eingebettet sind. Bei Aufgaben, die spezifische Formatierungs-, Argumentations- oder Fachkenntnisse erfordern, kann die Eingabeaufforderung mit wenigen Handgriffen effektiv sein. [Siehe auch Zero-Shot Prompting](#).

FGAC

Siehe [detaillierte Zugriffskontrolle](#).

Feinkörnige Zugriffskontrolle (FGAC)

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

Flash-Cut-Migration

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

FM

Siehe [Fundamentmodell](#).

Fundamentmodell (FM)

Ein großes neuronales Deep-Learning-Netzwerk, das mit riesigen Datensätzen generalisierter und unbeschrifteter Daten trainiert wurde. FMs sind in der Lage, eine Vielzahl allgemeiner Aufgaben zu erfüllen, z. B. Sprache zu verstehen, Text und Bilder zu generieren und Konversationen in natürlicher Sprache zu führen. Weitere Informationen finden Sie unter [Was sind Foundation-Modelle](#).

G

Generative KI

Eine Untergruppe von [KI-Modellen](#), die mit großen Datenmengen trainiert wurden und mit einer einfachen Textaufforderung neue Inhalte und Artefakte wie Bilder, Videos, Text und Audio erstellen können. Weitere Informationen finden Sie unter [Was ist Generative KI](#).

Geoblocking

Siehe [geografische Einschränkungen](#).

Geografische Einschränkungen (Geoblocking)

Bei Amazon eine Option CloudFront, um zu verhindern, dass Benutzer in bestimmten Ländern auf Inhaltsverteilungen zugreifen. Sie können eine Zulassungsliste oder eine Sperrliste verwenden,

um zugelassene und gesperrte Länder anzugeben. Weitere Informationen finden Sie in [der Dokumentation unter Beschränkung der geografischen Verteilung Ihrer Inhalte](#). CloudFront

Gitflow-Workflow

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

goldenes Bild

Ein Snapshot eines Systems oder einer Software, der als Vorlage für die Bereitstellung neuer Instanzen dieses Systems oder dieser Software verwendet wird. In der Fertigung kann ein Golden Image beispielsweise zur Bereitstellung von Software auf mehreren Geräten verwendet werden und trägt zur Verbesserung der Geschwindigkeit, Skalierbarkeit und Produktivität bei der Geräteherstellung bei.

Greenfield-Strategie

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

Integritätsschutz

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

H

HEKTAR

Siehe [Hochverfügbarkeit](#).

Heterogene Datenbankmigration

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

hohe Verfügbarkeit (HA)

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

historische Modernisierung

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

Daten zurückhalten

Ein Teil historischer, beschrifteter Daten, der aus einem Datensatz zurückgehalten wird, der zum Trainieren eines Modells für [maschinelles](#) Lernen verwendet wird. Sie können Holdout-Daten verwenden, um die Modellleistung zu bewerten, indem Sie die Modellvorhersagen mit den Holdout-Daten vergleichen.

Homogene Datenbankmigration

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

heiße Daten

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

Hotfix

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

Hypercare-Phase

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

I

IaC

Sehen Sie [Infrastruktur als Code](#).

Identitätsbasierte Richtlinie

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

Leerlaufanwendung

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

IIoT

Siehe [Industrielles Internet der Dinge](#).

unveränderliche Infrastruktur

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

Eingehende (ingress) VPC

In einer Architektur AWS mit mehreren Konten ist dies eine VPC, die Netzwerkverbindungen von außerhalb einer Anwendung akzeptiert, überprüft und weiterleitet. Die [AWS Security Reference Architecture](#) empfiehlt, Ihr Netzwerkkonto mit eingehendem und ausgehendem Datenverkehr und Inspektion einzurichten, VPCs um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet im weiteren Sinne zu schützen.

Inkrementelle Migration

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

Industrie 4.0

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

Infrastruktur

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

Infrastructure as Code (IaC)

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

industrielles Internet der Dinge (T) Ilo

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

Inspektions-VPC

In einer Architektur AWS mit mehreren Konten eine zentralisierte VPC, die Inspektionen des Netzwerkverkehrs zwischen VPCs (in demselben oder unterschiedlichen AWS-Regionen), dem Internet und lokalen Netzwerken verwaltet. In der [AWS Security Reference Architecture](#) wird empfohlen, Ihr Netzwerkkonto mit eingehendem und ausgehendem Datenverkehr sowie Inspektionen einzurichten, VPCs um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet im weiteren Sinne zu schützen.

Internet of Things (IoT)

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

Interpretierbarkeit

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

IoT

Siehe [Internet der Dinge](#).

IT information library (ITIL, IT-Informationsbibliothek)

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

T service management (ITSM, IT-Service-Management)

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

BIS

Siehe [IT-Informationsbibliothek](#).

ITSM

Siehe [IT-Service-Management](#).

L

Labelbasierte Zugangskontrolle (LBAC)

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

Landing Zone

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

großes Sprachmodell (LLM)

Ein [Deep-Learning-KI-Modell](#), das anhand einer riesigen Datenmenge vorab trainiert wurde. Ein LLM kann mehrere Aufgaben ausführen, z. B. Fragen beantworten, Dokumente zusammenfassen, Text in andere Sprachen übersetzen und Sätze vervollständigen. [Weitere Informationen finden Sie unter Was sind LLMs](#)

Große Migration

Eine Migration von 300 oder mehr Servern.

SCHWARZ

Siehe [Labelbasierte Zugriffskontrolle](#).

Geringste Berechtigung

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

Lift and Shift

Siehe [7 Rs](#).

Little-Endian-System

Ein System, welches das niedrigwertigste Byte zuerst speichert. Siehe auch [Endianness](#).

LLM

Siehe [großes Sprachmodell](#).

Niedrigere Umgebungen

Siehe [Umgebung](#).

M

Machine Learning (ML)

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

Hauptzweig

Siehe [Filiale](#).

Malware

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

verwaltete Dienste

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

Manufacturing Execution System (MES)

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

MAP

Siehe [Migration Acceleration Program](#).

Mechanismus

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

Mitgliedskonto

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

MES

Siehe [Manufacturing Execution System](#).

Message Queuing-Telemetrietransport (MQTT)

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

Microservice

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

Microservices-Architekturen

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

Migration Acceleration Program (MAP)

Ein AWS Programm, das Beratung, Unterstützung, Schulungen und Services bietet, um Unternehmen dabei zu unterstützen, eine solide betriebliche Grundlage für die Umstellung auf

die Cloud zu schaffen und die anfänglichen Kosten von Migrationen auszugleichen. MAP umfasst eine Migrationsmethode für die methodische Durchführung von Legacy-Migrationen sowie eine Reihe von Tools zur Automatisierung und Beschleunigung gängiger Migrationsszenarien.

Migration in großem Maßstab

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

Migrationsfabrik

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

Migrationsmetadaten

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

Migrationsmuster

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

Migration Portfolio Assessment (MPA)

Ein Online-Tool, das Informationen zur Validierung des Geschäftsszenarios für die Migration auf das bereitstellt. AWS Cloud MPA bietet eine detaillierte Portfoliobewertung (richtige Servergröße, Preisgestaltung, Gesamtbetriebskostenanalyse, Migrationskostenanalyse) sowie Migrationsplanung (Anwendungsdatenanalyse und Datenerfassung, Anwendungsgruppierung,

Migrationspriorisierung und Wellenplanung). Das [MPA-Tool](#) (Anmeldung erforderlich) steht allen AWS Beratern und APN-Partnerberatern kostenlos zur Verfügung.

Migration Readiness Assessment (MRA)

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

Migrationsstrategie

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

ML

[Siehe maschinelles Lernen.](#)

Modernisierung

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

Bewertung der Modernisierungsfähigkeit

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

Monolithische Anwendungen (Monolithen)

Anwendungen, die als ein einziger Service mit eng gekoppelten Prozessen ausgeführt werden. Monolithische Anwendungen haben verschiedene Nachteile. Wenn ein Anwendungs-Feature stark nachgefragt wird, muss die gesamte Architektur skaliert werden. Das Hinzufügen oder

Verbessern der Feature einer monolithischen Anwendung wird ebenfalls komplexer, wenn die Codebasis wächst. Um diese Probleme zu beheben, können Sie eine Microservices-Architektur verwenden. Weitere Informationen finden Sie unter [Zerlegen von Monolithen in Microservices](#).

MPA

Siehe [Bewertung des Migrationsportfolios](#).

MQTT

Siehe [Message Queuing-Telemetrietransport](#).

Mehrklassen-Klassifizierung

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

veränderbare Infrastruktur

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

O

OAC

Siehe [Origin Access Control](#).

EICHE

Siehe [Zugriffsidentität von Origin](#).

COM

Siehe [organisatorisches Change-Management](#).

Offline-Migration

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

OI

Siehe [Betriebsintegration](#).

OLA

Siehe Vereinbarung auf [operativer Ebene](#).

Online-Migration

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

OPC-UA

Siehe [Open Process Communications — Unified Architecture](#).

Offene Prozesskommunikation — Einheitliche Architektur (OPC-UA)

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

Vereinbarung auf Betriebsebene (OLA)

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

Überprüfung der Betriebsbereitschaft (ORR)

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

Betriebstechnologie (OT)

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

Betriebsintegration (OI)

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

Organisationspfad

Ein Pfad, der von erstellt wird und in AWS CloudTrail dem alle Ereignisse für alle AWS-Konten in einer Organisation protokolliert werden. AWS Organizations Diese Spur wird in jedem AWS-Konto, der Teil der Organisation ist, erstellt und verfolgt die Aktivität in jedem Konto. Weitere Informationen finden Sie in der CloudTrail Dokumentation unter [Einen Trail für eine Organisation erstellen](#).

Organisatorisches Veränderungsmanagement (OCM)

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

Ursprungszugriffskontrolle (OAC)

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

Ursprungszugriffsidentität (OAI)

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

ORR

Weitere Informationen finden Sie unter [Überprüfung der Betriebsbereitschaft](#).

NICHT

Siehe [Betriebstechnologie](#).

Ausgehende (egress) VPC

In einer Architektur AWS mit mehreren Konten eine VPC, die Netzwerkverbindungen verarbeitet, die von einer Anwendung aus initiiert werden. Die [AWS Security Reference Architecture](#) empfiehlt die Einrichtung Ihres Netzwerkkontos mit eingehendem und ausgehendem Datenverkehr sowie Inspektion, VPCs um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet im weiteren Sinne zu schützen.

P

Berechtigungsgrenze

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

persönlich identifizierbare Informationen (PII)

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

Personenbezogene Daten

Siehe [persönlich identifizierbare Informationen](#).

Playbook

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

PLC

Siehe [programmierbare Logiksteuerung](#).

PLM

Siehe [Produktlebenszyklusmanagement](#).

policy

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

Polyglotte Beharrlichkeit

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

Portfoliobewertung

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

predicate

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

Prädikat Pushdown

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

Präventive Kontrolle

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

Prinzipal

Eine Entität AWS, die Aktionen ausführen und auf Ressourcen zugreifen kann. Diese Entität ist in der Regel ein Root-Benutzer für eine AWS-Konto, eine IAM-Rolle oder einen Benutzer. Weitere Informationen finden Sie unter Prinzipal in [Rollenbegriffe und -konzepte](#) in der IAM-Dokumentation.

Datenschutz von Natur aus

Ein systemtechnischer Ansatz, der den Datenschutz während des gesamten Entwicklungsprozesses berücksichtigt.

Privat gehostete Zonen

Ein Container, der Informationen darüber enthält, wie Amazon Route 53 auf DNS-Abfragen für eine Domain und deren Subdomains innerhalb einer oder mehrerer VPCs Domains antworten soll. Weitere Informationen finden Sie unter [Arbeiten mit privat gehosteten Zonen](#) in der Route-53-Dokumentation.

proaktive Steuerung

Eine [Sicherheitskontrolle](#), die den Einsatz nicht richtlinienkonformer Ressourcen verhindern soll. Diese Steuerelemente scannen Ressourcen, bevor sie bereitgestellt werden. Wenn die Ressource nicht der Kontrolle entspricht, wird sie nicht bereitgestellt. Weitere Informationen finden Sie im [Referenzhandbuch zu Kontrollen](#) in der AWS Control Tower Dokumentation und unter [Proaktive Kontrollen](#) unter Implementierung von Sicherheitskontrollen am AWS.

Produktlebenszyklusmanagement (PLM)

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

Produktionsumgebung

Siehe [Umgebung](#).

Speicherprogrammierbare Steuerung (SPS)

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

schnelle Verkettung

Verwendung der Ausgabe einer [LLM-Eingabeaufforderung](#) als Eingabe für die nächste Aufforderung, um bessere Antworten zu generieren. Diese Technik wird verwendet, um eine komplexe Aufgabe in Unteraufgaben zu unterteilen oder um eine vorläufige Antwort iterativ zu verfeinern oder zu erweitern. Sie trägt dazu bei, die Genauigkeit und Relevanz der Antworten eines Modells zu verbessern und ermöglicht detailliertere, personalisierte Ergebnisse.

Pseudonymisierung

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

publish/subscribe (pub/sub)

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

Q

Abfrageplan

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

Abfrageplanregression

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

R

RACI-Matrix

Siehe [verantwortlich, rechenschaftspflichtig, konsultiert, informiert \(RACI\)](#).

RAG

Siehe Erweiterte [Generierung beim Abrufen](#).

Ransomware

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

RASCI-Matrix

Siehe [verantwortlich, rechenschaftspflichtig, konsultiert, informiert \(RACI\)](#).

RCAC

Siehe [Zugriffskontrolle für Zeilen und Spalten](#).

Read Replica

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

neu strukturieren

Siehe [7 Rs](#).

Recovery Point Objective (RPO)

Die maximal zulässige Zeitspanne seit dem letzten Datenwiederherstellungspunkt. Damit wird festgelegt, was als akzeptabler Datenverlust zwischen dem letzten Wiederherstellungspunkt und der Serviceunterbrechung gilt.

Wiederherstellungszeitziel (RTO)

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

Refaktorisierung

Siehe [7 Rs](#).

Region

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

Regression

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

rehosten

Siehe [7 Rs](#).

Veröffentlichung

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

umziehen

Siehe [7 Rs](#).

neue Plattform

Siehe [7 Rs](#).

Rückkauf

Siehe [7 Rs](#).

Ausfallsicherheit

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

Ressourcenbasierte Richtlinie

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

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

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

Reaktive Kontrolle

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

Beibehaltung

Siehe [7 Rs](#).

zurückziehen

Siehe [7 Rs](#).

Retrieval Augmented Generation (RAG)

Eine [generative KI-Technologie](#), bei der ein [LLM](#) auf eine maßgebliche Datenquelle verweist, die sich außerhalb seiner Trainingsdatenquellen befindet, bevor eine Antwort generiert wird. Ein RAG-Modell könnte beispielsweise eine semantische Suche in der Wissensdatenbank oder in benutzerdefinierten Daten einer Organisation durchführen. Weitere Informationen finden Sie unter [Was ist RAG](#).

Drehung

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

Zugriffskontrolle für Zeilen und Spalten (RCAC)

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

RPO

Siehe [Recovery Point Objective](#).

RTO

Siehe [Ziel für die Erholungszeit](#).

Runbook

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

S

SAML 2.0

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

SCADA

Siehe [Aufsichtskontrolle und Datenerfassung](#).

SCP

Siehe [Richtlinie zur Dienstkontrolle](#).

Secret

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

Sicherheit durch Design

Ein systemtechnischer Ansatz, der die Sicherheit während des gesamten Entwicklungsprozesses berücksichtigt.

Sicherheitskontrolle

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

Härtung der Sicherheit

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

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

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

Automatisierung von Sicherheitsreaktionen

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

Serverseitige Verschlüsselung

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

Service-Kontrollrichtlinie (SCP)

Eine Richtlinie, die eine zentrale Steuerung der Berechtigungen für alle Konten in einer Organisation in ermöglicht AWS Organizations. SCPs Definieren Sie Leitplanken oder legen Sie Grenzwerte für Aktionen fest, die ein Administrator an Benutzer oder Rollen delegieren kann. Sie können sie SCPs als Zulassungs- oder Ablehnungslisten verwenden, um festzulegen, welche Dienste oder Aktionen zulässig oder verboten sind. Weitere Informationen finden Sie in der AWS Organizations Dokumentation unter [Richtlinien zur Dienststeuerung](#).

Service-Endpoint

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

Service Level Agreement (SLA)

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

Service-Level-Indikator (SLI)

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

Service-Level-Ziel (SLO)

Eine Zielkennzahl, die den Zustand eines Dienstes darstellt, gemessen anhand eines [Service-Level-Indikators](#).

Modell der geteilten Verantwortung

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

SIEM

Siehe [Sicherheitsinformations- und Event-Management-System](#).

Single Point of Failure (SPOF)

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

SLA

Siehe [Service Level Agreement](#).

SLI

Siehe [Service-Level-Indikator](#).

ALSO

Siehe [Service-Level-Ziel](#).

split-and-seed Modell

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

SPOTTEN

Siehe [Single Point of Failure](#).

Sternschema

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

Strangler-Fig-Muster

Ein Ansatz zur Modernisierung monolithischer Systeme, bei dem die Systemfunktionen schrittweise umgeschrieben und ersetzt werden, bis das Legacy-System außer Betrieb genommen werden kann. Dieses Muster verwendet die Analogie einer Feigenrebe, die zu einem etablierten Baum heranwächst und schließlich ihren Wirt überwindet und ersetzt. Das Muster wurde [eingeführt von Martin Fowler](#) als Möglichkeit, Risiken beim Umschreiben monolithischer Systeme zu managen. Ein Beispiel für die Anwendung dieses Musters finden Sie unter [Schrittweises Modernisieren älterer Microsoft ASP.NET \(ASMX\)-Webservices mithilfe von Containern und Amazon API Gateway](#).

Subnetz

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

Aufsichtskontrolle und Datenerfassung (SCADA)

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

Symmetrische Verschlüsselung

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

synthetisches Testen

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

Systemaufforderung

Eine Technik, mit der einem [LLM](#) Kontext, Anweisungen oder Richtlinien zur Verfügung gestellt werden, um sein Verhalten zu steuern. Systemaufforderungen helfen dabei, den Kontext festzulegen und Regeln für Interaktionen mit Benutzern festzulegen.

T

tags

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

Zielvariable

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

Aufgabenliste

Ein Tool, das verwendet wird, um den Fortschritt anhand eines Runbooks zu verfolgen. Eine Aufgabenliste enthält eine Übersicht über das Runbook und eine Liste mit allgemeinen Aufgaben, die erledigt werden müssen. Für jede allgemeine Aufgabe werden der geschätzte Zeitaufwand, der Eigentümer und der Fortschritt angegeben.

Testumgebungen

[Siehe Umgebung.](#)

Training

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

Transit-Gateway

Ein Netzwerk-Transit-Hub, über den Sie Ihre Netzwerke VPCs und Ihre lokalen Netzwerke miteinander verbinden können. Weitere Informationen finden Sie in der Dokumentation unter [Was ist ein Transit-Gateway](#). AWS Transit Gateway

Stammbasierter Workflow

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

Vertrauenswürdiger Zugriff

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

Optimieren

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

Zwei-Pizzen-Team

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

U

Unsicherheit

Ein Konzept, das sich auf ungenaue, unvollständige oder unbekanntere Informationen bezieht, die die Zuverlässigkeit von prädiktiven ML-Modellen untergraben können. Es gibt zwei Arten von Unsicherheit: Epistemische Unsicherheit wird durch begrenzte, unvollständige Daten verursacht, wohingegen aleatorische Unsicherheit durch Rauschen und Randomisierung verursacht wird, die in den Daten liegt.

undifferenzierte Aufgaben

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

höhere Umgebungen

Siehe [Umgebung](#).

V

Vacuuming

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

Versionskontrolle

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

VPC-Peering

Eine Verbindung zwischen zwei VPCs, die es Ihnen ermöglicht, den Verkehr mithilfe privater IP-Adressen weiterzuleiten. Weitere Informationen finden Sie unter [Was ist VPC-Peering?](#) in der Amazon-VPC-Dokumentation.

Schwachstelle

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

W

Warmer Cache

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

warme Daten

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

Fensterfunktion

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

Workload

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

Workstream

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

WURM

Sehen [Sie einmal schreiben, viele lesen](#).

WQF

Siehe [AWS Workload-Qualifizierungsrahmen](#).

einmal schreiben, viele lesen (WORM)

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

Z

Zero-Day-Exploit

Ein Angriff, in der Regel Malware, der eine [Zero-Day-Sicherheitslücke](#) ausnutzt.

Zero-Day-Sicherheitslücke

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

Eingabeaufforderung ohne Zwischenfälle

Bereitstellung von Anweisungen für die Ausführung einer Aufgabe an einen [LLM](#), jedoch ohne Beispiele (Schnappschüsse), die ihm als Orientierungshilfe dienen könnten. Der LLM muss sein vortrainiertes Wissen einsetzen, um die Aufgabe zu bewältigen. Die Effektivität von Zero-Shot Prompting hängt von der Komplexität der Aufgabe und der Qualität der Aufforderung ab. [Siehe auch Few-Shot-Prompting](#).

Zombie-Anwendung

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

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