Aufbau sechseckiger Architekturen auf AWS

AWS Präskriptive Leitlinien

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Präskriptive Leitlinien: Aufbau sechseckiger Architekturen auf AWS

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Handelsmarken und 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
Übersicht ... 3
Domänengesteuertes Design (DDD) ... 3
Sechseckige Architektur ... 3
Gezielte Geschäftsergebnisse ... 5
Verbesserung des Entwicklungszyklus ... 6
Testen in der Cloud ... 6
Lokales Testen ... 6
Parallelisierung der Entwicklung ... 7
Zeit bis zur Markteinführung des Produkts ... 7
Qualität durch Design ... 8
Lokalisierte Änderungen und verbesserte Lesbarkeit ... 8
Zuerst die Geschäftslogik testen ... 8
Wartbarkeit ... 9
Anpassung an Veränderungen ... 11
Anpassung an neue, nicht funktionale Anforderungen mithilfe von Anschlüssen und Adaptern ... 11
Anpassung an neue Geschäftsanforderungen mithilfe von Befehlen und Befehlshandlern ... 11
Entkopplung von Komponenten mithilfe der Servicefassade oder des CQRS-Musters ... 12
Organisatorische Skalierung ... 13
Bewährte Methoden ... 15
Modellieren Sie den Geschäftsbereich ... 15
Schreiben und führen Sie Tests von Anfang an durch ... 15
Definieren Sie das Verhalten der Domäne ... 16
Automatisieren Sie Tests und Bereitstellung ... 16
Skalieren Sie Ihr Produkt mithilfe von Microservices und CQRS ... 16
Entwerfen Sie eine Projektstruktur, die den Konzepten der hexagonalen Architektur entspricht ... 17
Beispiele für Infrastruktur ... 19
Fangen Sie einfach an ... 19
Wenden Sie das CQRS-Muster an ... 20
Entwickeln Sie die Architektur weiter, indem Sie Container, eine relationale Datenbank und eine externe API hinzufügen ... 21
Fügen Sie weitere Domänen hinzu (verkleinern) ... 22
Häufig gestellte Fragen ... 24

Warum sollte ich eine sechseckige Architektur verwenden? ... 24
Warum sollte ich domänengesteuertes Design verwenden? ... 24
Kann ich testgetriebene Entwicklung ohne hexagonale Architektur praktizieren? ... 24
Kann ich mein Produkt ohne hexagonale Architektur und domänengesteuertes Design skalieren? ... 24
Welche Technologien sollte ich verwenden, um eine hexagonale Architektur zu implementieren? ... 24
Ich entwickle ein Produkt, das am wenigsten praktikabel ist. Macht es Sinn, Zeit damit zu verbringen, über Softwarearchitektur nachzudenken? ... 25
Ich entwickle ein Produkt, das nur minimal realisierbar ist, und habe keine Zeit, Tests zu schreiben. ... 25
Welche zusätzlichen Entwurfsmuster kann ich bei der hexagonalen Architektur verwenden? ... 25
Nächste Schritte ... 26
Ressourcen ... 27
Dokumentverlauf ... 29
Glossar ... 30
... 30
A ... 31
B ... 34
C ... 36
D ... 39
E ... 44
F ... 46
G ... 48
H ... 49
I ... 51
L ... 53
M ... 54
O ... 59
P ... 62
Q ... 65
R ... 65
S ... 68
T ... 72
U ... 74
V ... 75

W 75
Z 76
..... lxxviii

Aufbau sechseckiger Architekturen auf AWS

Furkan Oruc, Dominik Goby, Darius Kunce und Michal Ploski, Amazon Web Services ()AWS

Juni 2022 (Dokumentverlauf)

Dieser Leitfaden beschreibt ein mentales Modell und eine Sammlung von Mustern für die Entwicklung von Softwarearchitekturen. Diese Architekturen lassen sich leicht verwalten, erweitern und im gesamten Unternehmen skalieren, wenn die Produktakzeptanz zunimmt. Cloud-Hyperscaler wie Amazon Web Services (AWS) bieten Bausteine für kleine und große Unternehmen, um Innovationen zu entwickeln und neue Softwareprodukte zu entwickeln. Das schnelle Tempo, mit dem diese neuen Dienste und Funktionen eingeführt werden, führt dazu, dass Geschäftsbeteiligte erwarten, dass ihre Entwicklungsteams schneller Prototypen neuer Produkte (MVPs) entwickeln, damit neue Ideen so schnell wie möglich getestet und verifiziert werden können. Oft MVPs werden diese übernommen und werden Teil des Unternehmenssoftware-Ökosystems. Bei der Erstellung dieser MVPs Programme verzichten Teams manchmal auf Regeln und bewährte Verfahren für die Softwareentwicklung, wie z. B. SOLID-Prinzipien und Unit-Tests. Sie gehen davon aus, dass dieser Ansatz die Entwicklung beschleunigen und die Markteinführungszeit verkürzen wird. Wenn es ihnen jedoch nicht gelingt, ein grundlegendes Modell und ein Framework für die Softwarearchitektur auf allen Ebenen zu erstellen, wird es schwierig oder sogar unmöglich sein, neue Funktionen für das Produkt zu entwickeln. Mangelnde Sicherheit und sich ändernde Anforderungen können das Team auch während des Entwicklungsprozesses verlangsamen.

In diesem Leitfaden wird eine vorgeschlagene Softwarearchitektur vorgestellt, von einer sechseckigen Architektur auf niedriger Ebene bis hin zu einer Struktur- und Organisationszerlegung auf hoher Ebene, bei der Domain-Driven Design (DDD) zur Bewältigung dieser Herausforderungen verwendet wird. DDD hilft dabei, die Komplexität des Unternehmens zu bewältigen und das Entwicklungsteam bei der Entwicklung neuer Funktionen zu skalieren. Es bringt geschäftliche und technische Interessengruppen mithilfe einer allgegenwärtigen Sprache auf die Geschäftsprobleme, die sogenannten Domänen, ein. Die hexagonale Architektur ist eine technische Grundlage für diesen Ansatz in einem sehr spezifischen Bereich, der als begrenzter Kontext bezeichnet wird. Bei einem begrenzten Kontext handelt es sich um einen stark zusammenhängenden und lose miteinander verknüpften Teilbereich des Geschäftsproblems. Wir empfehlen Ihnen, für alle Ihre Unternehmenssoftwareprojekte unabhängig von deren Komplexität eine hexagonale Architektur zu verwenden.

Die hexagonale Architektur ermutigt das Entwicklungsteam, zuerst das Geschäftsproblem zu lösen, wohingegen bei der klassischen mehrschichtigen Architektur der technische Fokus

weg von der Domäne auf die Lösung technischer Probleme zuerst verlagert wird. Wenn Software einer sechseckigen Architektur folgt, ist es außerdem einfacher, einen testgetriebenen Entwicklungsansatz zu verfolgen, der die Rückkopplungsschleife reduziert, die Entwickler zum Testen von Geschäftsanforderungen benötigen. Und schließlich ist die Verwendung von Befehlen und Befehlshandlern eine Möglichkeit, die Prinzipien der Einzelverantwortung und der offenen und geschlossenen Lösung von SOLID anzuwenden. Durch die Einhaltung dieser Prinzipien entsteht eine Codebasis, in der Entwickler und Architekten, die an dem Projekt arbeiten, leicht navigieren und sie verstehen können. Außerdem wird das Risiko verringert, dass grundlegende Änderungen an bestehenden Funktionen vorgenommen werden.

Dieser Leitfaden richtet sich an Softwarearchitekten und Entwickler, die daran interessiert sind, die Vorteile einer hexagonalen Architektur und DDD für ihre Softwareentwicklungsprojekte zu verstehen. Es enthält ein Beispiel für den Entwurf einer Infrastruktur für Ihre Anwendung, die eine hexagonale AWS Architektur unterstützt. Eine Beispielimplementierung finden Sie unter Strukturieren eines Python-Projekts in hexagonaler Architektur AWS Lambda auf der AWS Prescriptive Guidance Guidance-Website.

Übersicht

Domänengesteuertes Design (DDD)

Beim domänengesteuerten Design (DDD) ist eine Domäne der Kern des Softwaresystems. Das Domänenmodell wird zuerst definiert, bevor Sie ein anderes Modul entwickeln, und es hängt nicht von anderen Low-Level-Modulen ab. Stattdessen hängen Module wie Datenbanken, die Darstellungsschicht und externe Module APIs alle von der Domäne ab.

In DDD zerlegen Architekten die Lösung in begrenzte Kontexte, indem sie statt der technischen Zerlegung eine auf Geschäftslogik basierende Zerlegung verwenden. Die Vorteile dieses Ansatzes werden in diesem Abschnitt erörtert. Gezielte Geschäftsergebnisse

DDD ist einfacher zu implementieren, wenn Teams eine hexagonale Architektur verwenden. Bei der hexagonalen Architektur ist der Anwendungskern das Zentrum der Anwendung. Es ist durch Ports und Adapter von anderen Modulen entkoppelt und hat keine Abhängigkeiten von anderen Modulen. Dies passt perfekt zu DDD, wo eine Domäne der Kern der Anwendung ist, die ein Geschäftsproblem löst. In diesem Leitfaden wird ein Ansatz vorgeschlagen, bei dem Sie den Kern der hexagonalen Architektur als Domänenmodell eines begrenzten Kontextes modellieren. Im nächsten Abschnitt wird die hexagonale Architektur ausführlicher beschrieben.

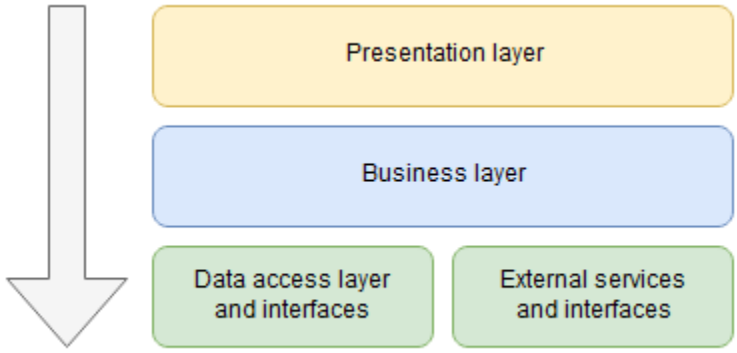
Dieser Leitfaden behandelt nicht alle Aspekte von DDD, einem sehr weit gefassten Thema. Zum besseren Verständnis können Sie sich die DDD-Ressourcen ansehen, die auf der Domain Language-Website aufgeführt sind.

Sechseckige Architektur

Die sechseckige Architektur, auch bekannt als Ports und Adapter oder Onion-Architektur, ist ein Prinzip zur Verwaltung der Abhängigkeitsinversion in Softwareprojekten. Die hexagonale Architektur fördert eine starke Fokussierung auf die Geschäftslogik der Kerndomäne bei der Softwareentwicklung und behandelt externe Integrationspunkte als zweitrangig. Die hexagonale Architektur unterstützt Softwareingenieure bei der Einführung bewährter Verfahren wie Test-Driven Development (TDD), was wiederum die Weiterentwicklung der Architektur fördert und Ihnen hilft, komplexe Bereiche langfristig zu verwalten.

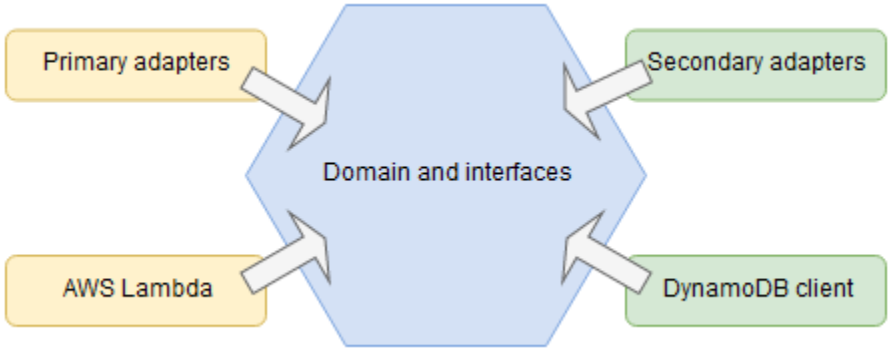
Vergleichen wir die hexagonale Architektur mit der klassischen mehrschichtigen Architektur, der beliebtesten Wahl für die Modellierung strukturierter Softwareprojekte. Es gibt subtile, aber starke Unterschiede zwischen den beiden Ansätzen.

In einer mehrschichtigen Architektur sind Softwareprojekte in Stufen strukturiert, die allgemeine Aspekte wie Geschäftslogik oder Präsentationslogik beinhalten. Diese Architektur verwendet eine Abhängigkeitshierarchie, bei der die obersten Schichten Abhängigkeiten von den darunterliegenden Ebenen haben, aber nicht umgekehrt. In der folgenden Abbildung ist die Darstellungsschicht für Benutzerinteraktionen verantwortlich. Sie umfasst also die Benutzeroberfläche APIs, Befehlszeilenschnittstellen und ähnliche Komponenten. Die Darstellungsschicht ist von der Geschäftsebene abhängig, die die Domänenlogik implementiert. Die Geschäftsebene wiederum ist von der Datenzugriffsebene und von mehreren externen Diensten abhängig.



Der Hauptnachteil dieser Konfiguration ist die Abhängigkeitsstruktur. Wenn sich beispielsweise das Modell zum Speichern von Daten in der Datenbank ändert, wirkt sich dies auf die Datenzugriffsschnittstelle aus. Jede Änderung des Datenmodells wirkt sich auch auf die Geschäftsebene aus, die von der Datenzugriffsschnittstelle abhängig ist. Daher können Softwareingenieure keine Änderungen an der Infrastruktur vornehmen, ohne die Domänenlogik zu beeinträchtigen. Dies wiederum erhöht die Wahrscheinlichkeit von Regressionsfehlern.

Die hexagonale Architektur definiert Abhängigkeitsbeziehungen auf andere Weise, wie in der folgenden Abbildung dargestellt. Sie konzentriert die Entscheidungsfindung auf die Geschäftslogik der Domäne, die alle Schnittstellen definiert. Externe Komponenten interagieren mit der Geschäftslogik über Schnittstellen, die als Ports bezeichnet werden. Ports sind Abstraktionen, die die Interaktionen der Domäne mit der Außenwelt definieren. Jede Infrastrukturkomponente muss diese Ports implementieren, sodass sich Änderungen an diesen Komponenten nicht mehr auf die Kernlogik der Domäne auswirken.



Die umgebenden Komponenten werden Adapter genannt. Ein Adapter ist ein Proxy zwischen der Außenwelt und der Innenwelt und implementiert einen in der Domäne definierten Port. Adapter können in zwei Gruppen eingeteilt werden: primäre und sekundäre. Primäre Adapter sind die Einstiegspunkte zur Softwarekomponente. Sie ermöglichen externen Akteuren, Benutzern und Diensten die Interaktion mit der Kernlogik. AWS Lambda ist ein gutes Beispiel für einen Primäradapter. Es lässt sich in mehrere AWS Dienste integrieren, die die Lambda-Funktionen als Einstiegspunkte aufrufen können. Sekundäre Adapter sind Wrapper für externe Servicebibliotheken, die die Kommunikation mit der Außenwelt übernehmen. Ein gutes Beispiel für einen sekundären Adapter ist ein Amazon DynamoDB-Client für den Datenzugriff.

Gezielte Geschäftsergebnisse

Die in diesem Handbuch beschriebene hexagonale Architektur hilft Ihnen dabei, die folgenden Ziele zu erreichen:

- Verkürzen Sie die Markteinführungszeit, indem Sie den Entwicklungszyklus verbessern
- Verbessern Sie die Softwarequalität
- Passen Sie sich leichter an Veränderungen an

Diese Prozesse werden in den folgenden Abschnitten ausführlich behandelt.

Verbesserung des Entwicklungszyklus

Die Entwicklung von Software für die Cloud stellt Softwareingenieure vor neue Herausforderungen, da es sehr schwierig ist, die Laufzeitumgebung lokal auf dem Entwicklungscomputer zu replizieren. Eine einfache Möglichkeit, die Software zu validieren, besteht darin, sie in der Cloud bereitzustellen und dort zu testen. Dieser Ansatz beinhaltet jedoch einen langen Feedback-Zyklus, insbesondere wenn die Softwarearchitektur mehrere serverlose Bereitstellungen umfasst. Durch die Verbesserung dieses Feedback-Zyklus wird die Zeit bis zur Entwicklung von Funktionen verkürzt, was die Markteinführungszeit erheblich verkürzt.

Testen in der Cloud

Nur durch direkte Tests in der Cloud können Sie sicherstellen, dass Ihre Architekturkomponenten wie Gateways in Amazon API Gateway, AWS Lambda Funktionen, Amazon DynamoDB-Tabellen und AWS Identity and Access Management (IAM-) Berechtigungen korrekt konfiguriert sind. Dies ist möglicherweise auch die einzig zuverlässige Methode, um Komponentenintegrationen zu testen. Obwohl einige AWS Dienste (wie DynamoDB) lokal bereitgestellt werden können, können die meisten von ihnen nicht in einer lokalen Konfiguration repliziert werden. Gleichzeitig geben Drittanbieter-Tools wie Moto und solche, LocalStackdie AWS Dienste zu Testzwecken simulieren, möglicherweise nicht die tatsächlichen API-Verträge an, oder die Anzahl der Funktionen ist möglicherweise begrenzt.

Der komplexeste Teil der Unternehmenssoftware liegt jedoch in der Geschäftslogik, nicht in der Cloud-Architektur. Die Architektur ändert sich seltener als die Domäne, die neuen Geschäftsanforderungen gerecht werden muss. Daher wird das Testen der Geschäftslogik in der Cloud zu einem intensiven Prozess, bei dem eine Änderung am Code vorgenommen, eine Bereitstellung initiiert, darauf gewartet wird, dass die Umgebung bereit ist, und die Änderung validiert wird. Wenn eine Bereitstellung nur 5 Minuten dauert, dauert das Vornehmen und Testen von 10 Änderungen an der Geschäftslogik eine Stunde oder länger. Wenn die Geschäftslogik komplexer ist, kann das Testen tagelang dauern, bis die Bereitstellung abgeschlossen ist. Wenn Sie mehrere Funktionen und Techniker im Team haben, macht sich der längere Zeitraum für das Unternehmen schnell bemerkbar.

Lokales Testen

Eine sechseckige Architektur hilft Entwicklern, sich auf die Domäne zu konzentrieren, anstatt sich auf technische Fragen der Infrastruktur zu konzentrieren. Bei diesem Ansatz werden lokale

Tests (die Unit-Test-Tools in Ihrem ausgewählten Entwicklungsframework) verwendet, um die Anforderungen an die Domänenlogik abzudecken. Sie müssen keine Zeit damit verbringen, technische Integrationsprobleme zu lösen oder Ihre Software in der Cloud bereitzustellen, um die Geschäftslogik zu testen. Sie können Komponententests lokal ausführen und die Feedback-Schleife von Minuten auf Sekunden reduzieren. Wenn eine Bereitstellung 5 Minuten dauert, Komponententests jedoch innerhalb von 5 Sekunden abgeschlossen sind, bedeutet das eine deutliche Verkürzung der Zeit, die zur Erkennung von Fehlern benötigt wird. Im Zuerst die Geschäftslogik testen Abschnitt weiter unten in diesem Handbuch wird dieser Ansatz ausführlicher behandelt.

Parallelisierung der Entwicklung

Der sechseckige Architekturansatz ermöglicht es Entwicklungsteams, die Entwicklungsbemühungen zu parallelisieren. Entwickler können verschiedene Komponenten des Dienstes individuell entwerfen und implementieren. Diese Parallelisierung ist durch die Isolierung jeder Komponente und die definierten Schnittstellen zwischen den einzelnen Komponenten möglich.

Zeit bis zur Markteinführung des Produkts

Tests vor Ort verbessern den Feedback-Zyklus bei der Entwicklung und verkürzen die Markteinführungszeit für neue Produkte oder Funktionen, insbesondere wenn diese, wie bereits erläutert, komplexe Geschäftslogik beinhalten. Darüber hinaus verringert eine erhöhte Codeabdeckung durch Komponententests das Risiko, dass Regressionsfehler auftreten, wenn Sie die Codebasis aktualisieren oder umgestalten, erheblich. Die Abdeckung durch Unit-Tests ermöglicht es Ihnen auch, die Codebasis kontinuierlich umzugestalten, um sie gut organisiert zu halten, was den Onboarding-Prozess für neue Ingenieure beschleunigt. Darauf wird in diesem Abschnitt näher eingegangen. Qualität durch Design Und wenn die Geschäftslogik gut isoliert und getestet ist, können Entwickler sich schnell an sich ändernde funktionale und nichtfunktionale Anforderungen anpassen. Dies wird im Anpassung an Veränderungen Abschnitt näher erläutert.

Qualität durch Design

Die Einführung einer hexagonalen Architektur trägt dazu bei, die Qualität Ihrer Codebasis von Beginn Ihres Projekts an zu verbessern. Es ist wichtig, einen Prozess zu entwickeln, der Ihnen hilft, die erwarteten Qualitätsanforderungen von Anfang an zu erfüllen, ohne den Entwicklungsprozess zu verlangsamen.

Lokalisierte Änderungen und verbesserte Lesbarkeit

Mithilfe des hexagonalen Architekturansatzes können Entwickler Code in einer Klasse oder Komponente ändern, ohne andere Klassen oder Komponenten zu beeinflussen. Dieses Design fördert den Zusammenhalt der entwickelten Komponenten. Indem Sie die Domäne von Adaptern entkoppeln und bekannte Schnittstellen verwenden, können Sie die Lesbarkeit des Codes erhöhen. Es wird einfacher, Probleme und Sonderfälle zu identifizieren.

Dieser Ansatz erleichtert auch die Überprüfung des Codes während der Entwicklung und begrenzt die Einführung unentdeckter Änderungen oder technischer Probleme.

Zuerst die Geschäftslogik testen

Lokale Tests können durch Einführung end-to-end, Integration und Komponententests in das Projekt durchgeführt werden. End-to-endTests decken den gesamten Lebenszyklus eingehender Anfragen ab. In der Regel rufen sie einen Einstiegspunkt für eine Anwendung auf und testen, ob die Geschäftsanforderungen erfüllt wurden. Jedes Softwareprojekt sollte über mindestens ein Testszenario verfügen, das bekannte Eingaben verwendet und erwartete Ergebnisse liefert. Das Hinzufügen weiterer Eckfallszenarien kann jedoch komplex werden, da jeder Test so konfiguriert sein muss, dass er eine Anfrage über einen Einstiegspunkt sendet (z. B. über eine REST-API oder Warteschlangen), alle Integrationspunkte durchläuft, die für die Geschäftsaktion erforderlich sind, und dann das Ergebnis bestätigt. Die Einrichtung der Umgebung für das Testszenario und die Bestätigung der Ergebnisse können viel Zeit in Anspruch nehmen.

In der hexagonalen Architektur testen Sie die Geschäftslogik isoliert und verwenden Integrationstests, um sekundäre Adapter zu testen. Sie können in Ihren Geschäftslogiktests simulierte oder gefälschte Adapter verwenden. Sie können die Tests für geschäftliche Anwendungsfälle auch mit Komponententests für Ihr Domänenmodell kombinieren, um eine hohe Abdeckung bei geringer Kopplung aufrechtzuerhalten. Es hat sich bewährt, dass Integrationstests die Geschäftslogik nicht

validieren sollten. Stattdessen sollten sie überprüfen, ob der sekundäre Adapter die externen Dienste korrekt aufruft.

Im Idealfall können Sie Test-Driven Development (TDD) verwenden und gleich zu Beginn der Entwicklung mit der Definition von Domänenentitäten oder Geschäftsanwendungsfällen mit geeigneten Tests beginnen. Wenn Sie zuerst die Tests schreiben, können Sie Scheinimplementierungen der von der Domain benötigten Schnittstellen erstellen. Wenn die Tests erfolgreich sind und die Regeln der Domänenlogik erfüllt sind, können Sie die eigentlichen Adapter implementieren und Software in der Testumgebung bereitstellen. Zu diesem Zeitpunkt ist Ihre Implementierung der Domänenlogik möglicherweise nicht ideal. Anschließend können Sie daran arbeiten, die bestehende Architektur umzugestalten, um sie weiterzuentwickeln, indem Sie Entwurfsmuster einführen oder Code generell neu anordnen. Mit diesem Ansatz können Sie die Einführung von Regressionsfehlern vermeiden und die Architektur verbessern, wenn das Projekt wächst. Durch die Kombination dieses Ansatzes mit den automatischen Tests, die Sie im Rahmen Ihres kontinuierlichen Integrationsprozesses ausführen, können Sie die Anzahl potenzieller Fehler verringern, bevor sie produktiv werden.

Wenn Sie serverlose Bereitstellungen verwenden, können Sie schnell eine Instanz der Anwendung in Ihrem AWS Konto für die manuelle Integration und end-to-end das Testen bereitstellen. Nach diesen Implementierungsschritten empfehlen wir Ihnen, die Tests zu automatisieren, wobei jede neue Änderung in das Repository übertragen wird.

Wartbarkeit

Wartbarkeit bezieht sich auf den Betrieb und die Überwachung einer Anwendung, um sicherzustellen, dass sie alle Anforderungen erfüllt, und um die Wahrscheinlichkeit eines Systemausfalls zu minimieren. Um das System betriebsbereit zu machen, müssen Sie es an future Verkehrs- oder Betriebsanforderungen anpassen. Sie müssen außerdem sicherstellen, dass es verfügbar und einfach zu implementieren ist und dass es nur minimale oder keine Auswirkungen auf die Kunden hat.

Um den aktuellen und historischen Zustand Ihres Systems zu verstehen, müssen Sie es beobachtbar machen. Sie können dies tun, indem Sie spezifische Metriken, Protokolle und Traces bereitstellen, anhand derer die Bediener sicherstellen können, dass das System wie erwartet funktioniert, und um Fehler zu verfolgen. Diese Mechanismen sollten es den Bedienern auch ermöglichen, Ursachenanalysen durchzuführen, ohne sich am Computer anmelden und den Code lesen zu müssen.

Eine sechseckige Architektur zielt darauf ab, die Wartbarkeit Ihrer Webanwendungen zu verbessern, sodass Ihr Code insgesamt weniger Arbeit erfordert. Durch die Trennung von Modulen, die Lokalisierung von Änderungen und die Entkopplung der Anwendungslogik von der Adapterimplementierung können Sie Metriken und Protokolle erstellen, die den Bedienern helfen, ein tiefes Verständnis des Systems zu erlangen und den Umfang bestimmter Änderungen zu verstehen, die an den primären oder sekundären Adaptern vorgenommen wurden.

Anpassung an Veränderungen

Softwaresysteme neigen dazu, kompliziert zu werden. Ein Grund dafür könnten häufige Änderungen der Geschäftsanforderungen und wenig Zeit sein, um die Softwarearchitektur entsprechend anzupassen. Ein weiterer Grund könnten unzureichende Investitionen für die Einrichtung der Softwarearchitektur zu Beginn des Projekts sein, um sie an häufige Änderungen anpassen zu können. Was auch immer der Grund sein mag, ein Softwaresystem könnte so kompliziert werden, dass es fast unmöglich ist, eine Änderung vorzunehmen. Daher ist es wichtig, von Beginn des Projekts an eine wartbare Softwarearchitektur aufzubauen. Eine gute Softwarearchitektur kann sich leicht an Änderungen anpassen.

In diesem Abschnitt wird erklärt, wie verwaltbare Anwendungen mithilfe einer hexagonalen Architektur entworfen werden können, die sich leicht an nicht funktionale oder geschäftliche Anforderungen anpassen lässt.

Anpassung an neue, nicht funktionale Anforderungen mithilfe von Anschlüssen und Adaptern

Als Kern der Anwendung definiert das Domänenmodell die Aktionen, die von außen erforderlich sind, um die Geschäftsanforderungen zu erfüllen. Diese Aktionen werden durch Abstraktionen definiert, die als Ports bezeichnet werden. Diese Ports werden durch separate Adapter implementiert. Jeder Adapter ist für die Interaktion mit einem anderen System verantwortlich. Beispielsweise haben Sie möglicherweise einen Adapter für das Datenbank-Repository und einen anderen Adapter für die Interaktion mit einer Drittanbieter-API. Die Domäne kennt die Adapterimplementierung nicht, sodass es einfach ist, einen Adapter durch einen anderen zu ersetzen. Beispielsweise könnte die Anwendung von einer SQL-Datenbank zu einer NoSQL-Datenbank wechseln. In diesem Fall muss ein neuer Adapter entwickelt werden, um die durch das Domänenmodell definierten Ports zu implementieren. Die Domäne hat keine Abhängigkeiten vom Datenbank-Repository und verwendet Abstraktionen für die Interaktion, sodass am Domänenmodell nichts geändert werden müsste. Daher passt sich die hexagonale Architektur mühelos an nichtfunktionale Anforderungen an.

Anpassung an neue Geschäftsanforderungen mithilfe von Befehlen und Befehlshandlern

In der klassischen Schichtarchitektur hängt die Domäne von der Persistenzschicht ab. Wenn Sie die Domain ändern möchten, müssten Sie auch die Persistenzschicht ändern. Im Vergleich dazu hängt

die Domäne bei einer hexagonalen Architektur nicht von anderen Modulen in der Software ab. Die Domäne ist der Kern der Anwendung, und alle anderen Module (Ports und Adapter) hängen vom Domänenmodell ab. Die Domäne verwendet das Prinzip der Umkehrung von Abhängigkeiten, um über Ports mit der Außenwelt zu kommunizieren. Der Vorteil der Abhängigkeitsumkehrung besteht darin, dass Sie das Domänenmodell frei ändern können, ohne Angst haben zu müssen, andere Teile des Codes zu beschädigen. Da das Domänenmodell das Geschäftsproblem widerspiegelt, das Sie zu lösen versuchen, ist es kein Problem, das Domänenmodell zu aktualisieren, um es an sich ändernde Geschäftsanforderungen anzupassen.

Bei der Entwicklung von Software ist die Trennung von Unternehmensbereichen ein wichtiger Grundsatz, den es zu beachten gilt. Um diese Trennung zu erreichen, können Sie ein leicht modifiziertes Befehlsmuster verwenden. Dabei handelt es sich um ein Verhaltensmuster, bei dem alle für den Abschluss eines Vorgangs erforderlichen Informationen in einem Befehlsobjekt zusammengefasst sind. Diese Operationen werden dann von Befehlshandlern verarbeitet. Befehlshandler sind Methoden, die einen Befehl empfangen, den Status der Domäne ändern und dann eine Antwort an den Aufrufer zurückgeben. Sie können verschiedene Clients verwenden, z. B. synchrone APIs oder asynchrone Warteschlangen, um Befehle auszuführen. Wir empfehlen, für jeden Vorgang in der Domäne Befehle und Befehlshandler zu verwenden. Mit diesem Ansatz können Sie neue Funktionen hinzufügen, indem Sie neue Befehle und Befehlshandler einführen, ohne Ihre bestehende Geschäftslogik zu ändern. Die Verwendung eines Befehlsmusters erleichtert somit die Anpassung an neue Geschäftsanforderungen.

Entkopplung von Komponenten mithilfe der Servicefassade oder des CQRS-Musters

In der hexagonalen Architektur sind Primäradapter dafür verantwortlich, eingehende Lese- und Schreibanforderungen von Clients lose an die Domäne zu koppeln. Es gibt zwei Möglichkeiten, diese lose Kopplung zu erreichen: durch die Verwendung eines Dienstfassadenmusters oder durch die Verwendung des CQRS-Musters (Command Query Responsibility Segregation).

Das Service-Fassadenmuster bietet eine nach vorne gerichtete Oberfläche für die Bedienung von Clients, z. B. der Präsentationsebene oder einem Microservice. Eine Servicefassade bietet Kunden mehrere Lese- und Schreiboperationen. Sie ist dafür verantwortlich, eingehende Anfragen an die Domain zu übertragen und die von der Domain empfangene Antwort den Clients zuzuordnen. Die Verwendung einer Service-Fassade ist für Microservices, die eine einzige Verantwortung mit mehreren Vorgängen haben, einfach. Bei der Nutzung der Servicefassade ist es jedoch schwieriger, den Prinzipien der Einzelverantwortung und der offenen und geschlossenen Struktur zu folgen. Das

Prinzip der Alleinverantwortung besagt, dass jedes Modul nur für eine einzige Funktionalität der Software verantwortlich sein sollte. Das Prinzip der offenen und geschlossenen Version besagt, dass Code für Erweiterungen offen und für Änderungen geschlossen sein sollte. Mit der Erweiterung der Service-Fassade werden alle Operationen in einer Oberfläche zusammengefasst, mehr Abhängigkeiten werden darin zusammengefasst und mehr Entwickler beginnen, dieselbe Fassade zu modifizieren. Daher empfehlen wir, eine Service-Fassade nur dann zu verwenden, wenn klar ist, dass der Service während der Entwicklung nicht wesentlich erweitert werden würde.

Eine weitere Möglichkeit, Primäradapter in einer hexagonalen Architektur zu implementieren, ist die Verwendung des CQRS-Musters, das Lese- und Schreibvorgänge mithilfe von Abfragen und Befehlen voneinander trennt. Wie bereits erläutert, handelt es sich bei Befehlen um Objekte, die alle Informationen enthalten, die zur Änderung des Status der Domäne erforderlich sind. Befehle werden durch Befehlshandler-Methoden ausgeführt. Abfragen ändern dagegen nicht den Zustand des Systems. Ihr einziger Zweck besteht darin, Daten an Kunden zurückzugeben. Im CQRS-Muster werden Befehle und Abfragen in separaten Modulen implementiert. Dies ist besonders vorteilhaft für Projekte, die einer ereignisgesteuerten Architektur folgen, da ein Befehl als Ereignis implementiert werden könnte, das asynchron verarbeitet wird, wohingegen eine Abfrage mithilfe einer API synchron ausgeführt werden kann. Eine Abfrage kann auch eine andere Datenbank verwenden, die für sie optimiert ist. Der Nachteil des CQRS-Musters besteht darin, dass die Implementierung mehr Zeit in Anspruch nimmt als die Implementierung einer Servicefassade. Wir empfehlen, das CQRS-Muster für Projekte zu verwenden, die Sie langfristig skalieren und verwalten möchten. Befehle und Abfragen bieten einen effektiven Mechanismus zur Anwendung des Prinzips der alleinigen Verantwortung und zur Entwicklung lose gekoppelter Software, insbesondere bei Großprojekten.

CQRS hat auf lange Sicht große Vorteile, erfordert jedoch eine Anfangsinvestition. Aus diesem Grund empfehlen wir Ihnen, Ihr Projekt sorgfältig zu prüfen, bevor Sie sich für die Verwendung des CQRS-Musters entscheiden. Sie können Ihre Anwendung jedoch von Anfang an strukturieren, indem Sie Befehle und Befehlshandler verwenden, ohne Lese- und Schreibvorgänge zu trennen. Auf diese Weise können Sie Ihr Projekt problemlos für CQRS umgestalten, falls Sie sich später für diesen Ansatz entscheiden.

Organisatorische Skalierung

Eine Kombination aus sechseckiger Architektur, domänengesteuertem Design und (optional) CQRS ermöglicht es Ihrem Unternehmen, Ihr Produkt schnell zu skalieren. Gemäß dem Gesetz von Conway entwickeln sich Softwarearchitekturen in der Regel so, dass sie die Kommunikationsstrukturen eines Unternehmens widerspiegeln. Diese Beobachtung hatte in der Vergangenheit negative

Assoziationen, da große Unternehmen ihre Teams häufig auf der Grundlage von technischem Fachwissen wie Datenbanken, Enterprise Service Bus usw. strukturieren. Das Problem bei diesem Ansatz besteht darin, dass die Entwicklung von Produkten und Funktionen immer mit Querschnittsfragen wie Sicherheit und Skalierbarkeit einhergeht, die eine ständige Kommunikation zwischen den Teams erfordern. Die Strukturierung von Teams auf der Grundlage technischer Merkmale führt zu unnötigen Silos in der Organisation, die zu schlechter Kommunikation, mangelnder Eigenverantwortung und dem Verlust des Gesamtbildes führen. Letztlich spiegeln sich diese organisatorischen Probleme in der Softwarearchitektur wider.

Das Inverse Conway-Manöver hingegen definiert die Organisationsstruktur auf der Grundlage von Bereichen, die die Softwarearchitektur fördern. Funktionsübergreifenden Teams wird beispielsweise die Verantwortung für eine bestimmte Gruppe von begrenzten Kontexten übertragen, die mithilfe von DDD und Event Storming identifiziert werden. Diese begrenzten Kontexte können sehr spezifische Merkmale des Produkts widerspiegeln. Beispielsweise könnte das Account-Team für den Zahlungskontext verantwortlich sein. Jede neue Funktion wird einem neuen Team zugewiesen, das sehr kohärente und lose miteinander verknüpfte Verantwortlichkeiten hat, sodass es sich nur auf die Bereitstellung dieser Funktion konzentrieren und die Markteinführungszeit verkürzen kann. Teams können entsprechend der Komplexität der Funktionen skaliert werden, sodass komplexe Funktionen mehr Technikern zugewiesen werden können.

Bewährte Methoden

Modellieren Sie den Geschäftsbereich

Gehen Sie von der Geschäftsdomäne zum Softwaredesign zurück, um sicherzustellen, dass die Software, die Sie schreiben, den Geschäftsanforderungen entspricht.

Verwenden Sie DDD-Methoden (Domain-Driven Design) wie Event Storming, um die Geschäftsdomäne zu modellieren. Event Storming bietet ein flexibles Workshop-Format. Während des Workshops untersuchen Domänen- und Softwareexperten gemeinsam die Komplexität des Geschäftsbereichs. Softwareexperten nutzen die Ergebnisse des Workshops, um den Entwurfs- und Entwicklungsprozess für Softwarekomponenten zu starten.

Schreiben und führen Sie Tests von Anfang an durch

Verwenden Sie Test-Driven Development (TDD), um die Richtigkeit der Software, die Sie entwickeln, zu überprüfen. TDD funktioniert am besten auf Unit-Test-Ebene. Der Entwickler entwirft eine Softwarekomponente, indem er zuerst einen Test schreibt, der diese Komponente aufruft. Diese Komponente ist zu Beginn nicht implementiert, daher schlägt der Test fehl. Als nächsten Schritt implementiert der Entwickler die Funktionalität der Komponente, indem er Testgeräte mit Scheinobjekten verwendet, um das Verhalten externer Abhängigkeiten oder Ports zu simulieren. Wenn der Test erfolgreich ist, kann der Entwickler weitermachen, indem er echte Adapter implementiert. Dieser Ansatz verbessert die Softwarequalität und führt zu besser lesbarem Code, da Entwickler verstehen, wie Benutzer die Komponenten verwenden würden. Die hexagonale Architektur unterstützt die TDD-Methodik, indem sie den Anwendungskern voneinander trennt. Entwickler schreiben Komponententests, die sich auf das Verhalten des Domänenkerns konzentrieren. Sie müssen keine komplexen Adapter schreiben, um ihre Tests durchzuführen. Stattdessen können sie einfache Scheinobjekte und Fixtures verwenden.

Verwenden Sie verhaltensgesteuerte Entwicklung (BDD), um die end-to-end Akzeptanz auf Funktionsebene sicherzustellen. In BDD definieren Entwickler Szenarien für Funktionen und verifizieren sie mit Geschäftspartnern. BDD-Tests verwenden so viel natürliche Sprache wie möglich, um dies zu erreichen. Die hexagonale Architektur unterstützt die BDD-Methodik mit ihrem Konzept von Primär- und Sekundäradaptern. Entwickler können primäre und sekundäre Adapter erstellen, die lokal ausgeführt werden können, ohne externe Dienste aufrufen zu müssen. Sie konfigurieren die BDD-Testsuite so, dass sie den lokalen Primäradapter zum Ausführen der Anwendung verwendet.

Führen Sie jeden Test in der Continuous-Integration-Pipeline automatisch aus, um die Qualität des Systems ständig zu bewerten.

Definieren Sie das Verhalten der Domäne

Zerlegen Sie die Domäne in Entitäten, Wertobjekte und Aggregate (lesen Sie mehr über die Implementierung eines domänengesteuerten Designs) und definieren Sie deren Verhalten. Implementieren Sie das Verhalten der Domäne, sodass Tests, die zu Beginn des Projekts geschrieben wurden, erfolgreich sind. Definieren Sie Befehle, die das Verhalten von Domänenobjekten aufrufen. Definieren Sie Ereignisse, die von den Domänenobjekten ausgelöst werden, nachdem sie ein Verhalten abgeschlossen haben.

Definieren Sie Schnittstellen, über die Adapter mit der Domäne interagieren können.

Automatisieren Sie Tests und Bereitstellung

Nach einem ersten Machbarkeitsnachweis empfehlen wir Ihnen, Zeit für die Implementierung von DevOps Praktiken zu investieren. Beispielsweise helfen Ihnen CI/CD-Pipelines (Continuous Integration and Continuous Delivery) sowie dynamische Testumgebungen dabei, die Qualität des Codes aufrechtzuerhalten und Fehler bei der Bereitstellung zu vermeiden.

- Führen Sie Ihre Komponententests innerhalb Ihres CI-Prozesses aus und testen Sie Ihren Code, bevor er zusammengeführt wird.
- Erstellen Sie einen CD-Prozess, um Ihre Anwendung in einer statischen Entwicklungs-/Testumgebung oder in dynamisch erstellten Umgebungen bereitzustellen, die automatische Integration und end-to-end Tests unterstützen.
- Automatisieren Sie den Bereitstellungsprozess für dedizierte Umgebungen.

Skalieren Sie Ihr Produkt mithilfe von Microservices und CQRS

Wenn Ihr Produkt erfolgreich ist, skalieren Sie Ihr Produkt, indem Sie Ihr Softwareprojekt in Microservices zerlegen. Nutzen Sie die Portabilität, die die hexagonale Architektur bietet, um die Leistung zu verbessern. Teilen Sie Abfragedienste und Befehlshandler in separate synchrone und asynchrone Dienste auf. APIs Erwägen Sie die Einführung des CQRS-Musters (Command Query Responsibility Segregation) und der ereignisgesteuerten Architektur.

Wenn Sie viele Anfragen nach neuen Funktionen erhalten, sollten Sie erwägen, Ihre Organisation auf der Grundlage von DDD-Mustern zu skalieren. Strukturieren Sie Ihre Teams so, dass sie über eine oder mehrere Funktionen in begrenzten Kontexten verfügen, wie bereits in diesem Abschnitt beschrieben. Organisatorische Skalierung Diese Teams können dann Geschäftslogik mithilfe einer hexagonalen Architektur implementieren.

Entwerfen Sie eine Projektstruktur, die den Konzepten der hexagonalen Architektur entspricht

Infrastructure as Code (IaC) ist eine weit verbreitete Praxis in der Cloud-Entwicklung. Damit können Sie Ihre Infrastrukturressourcen (wie Netzwerke, Load Balancer, virtuelle Maschinen und Gateways) als Quellcode definieren und verwalten. Auf diese Weise können Sie alle Änderungen an Ihrer Architektur mithilfe eines Versionskontrollsystems verfolgen. Darüber hinaus können Sie die Infrastruktur zu Testzwecken einfach erstellen und verschieben. Wir empfehlen Ihnen, Ihren Anwendungscode und Infrastrukturcode bei der Entwicklung Ihrer Cloud-Anwendungen im selben Repository aufzubewahren. Dieser Ansatz erleichtert die Wartung der Infrastruktur für Ihre Anwendung.

Wir empfehlen, dass Sie Ihre Anwendung in drei Ordner oder Projekte aufteilen, die den Konzepten der hexagonalen Architektur entsprechen: `entrypoints` (Primäradapter), `domain` (Domäne und Schnittstellen) und `adapters` (Sekundäradapter).

Die folgende Projektstruktur bietet ein Beispiel für diesen Ansatz beim Entwerfen einer API auf. AWS Das Projekt verwaltet Anwendungscode (`app`) und Infrastrukturcode (`infra`) im selben Repository, wie zuvor empfohlen.

```
app/  # application code
|--- adapters/  # implementation of the ports defined in the domain
     |--- tests/  # adapter unit tests
|--- entrypoints/  # primary adapters, entry points
     |--- api/  # api entry point
          |--- model/  # api model
          |--- tests/  # end to end api tests
|--- domain/  # domain to implement business logic using hexagonal architecture
     |--- command_handlers/  # handlers used to run commands on the domain
     |--- commands/  # commands on the domain
     |--- events/  # events emitted by the domain
     |--- exceptions/  # exceptions defined on the domain
     |--- model/  # domain model
```

```
     |--- ports/  # abstractions used for external communication
     |--- tests/  # domain tests
infra/  # infrastructure code
```

Wie bereits erwähnt, ist die Domäne der Kern der Anwendung und hängt von keinem anderen Modul ab. Es wird empfohlen, den `domain` Ordner so zu strukturieren, dass er die folgenden Unterordner enthält:

- `command handlers`enthält die Methoden oder Klassen, die Befehle in der Domäne ausführen.
- `commands`enthält die Befehlsobjekte, die die Informationen definieren, die für die Ausführung eines Vorgangs in der Domäne erforderlich sind.
- `events`enthält die Ereignisse, die über die Domäne ausgelöst und dann an andere Microservices weitergeleitet werden.
- `exceptions`enthält die bekannten Fehler, die innerhalb der Domäne definiert sind.
- `model`enthält die Domänenentitäten, Wertobjekte und Domänendienste.
- `ports`enthält die Abstraktionen, über die die Domain mit Datenbanken oder anderen externen Komponenten kommuniziert. APIs
- `tests`enthält die Testmethoden (z. B. Geschäftslogiktests), die in der Domäne ausgeführt werden.

Die primären Adapter sind die Einstiegspunkte für die Anwendung, wie sie durch den `entrypoints` Ordner dargestellt werden. In diesem Beispiel wird der `api` Ordner als primärer Adapter verwendet. Dieser Ordner enthält eine API`model`, die die Schnittstelle definiert, die der Primäradapter für die Kommunikation mit Clients benötigt. Der `tests` Ordner enthält end-to-end Tests für die API. Dabei handelt es sich um oberflächliche Tests, mit denen bestätigt wird, dass die Komponenten der Anwendung integriert sind und miteinander harmonieren.

Die sekundären Adapter, wie sie im `adapters` Ordner dargestellt werden, implementieren die externen Integrationen, die für die Domänenports erforderlich sind. Ein Datenbank-Repository ist ein hervorragendes Beispiel für einen sekundären Adapter. Wenn sich das Datenbanksystem ändert, können Sie einen neuen Adapter schreiben, indem Sie die Implementierung verwenden, die durch die Domäne definiert ist. Es ist nicht erforderlich, die Domäne oder die Geschäftslogik zu ändern. Der `tests` Unterordner enthält externe Integrationstests für jeden Adapter.

Infrastrukturbeispiele auf AWS

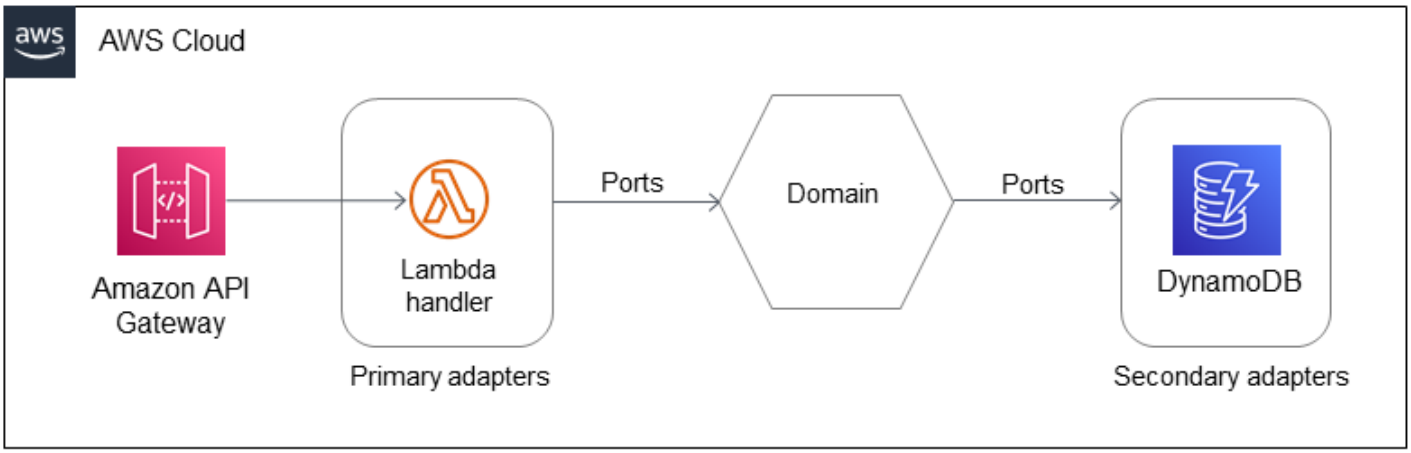
Dieser Abschnitt enthält Beispiele für den Entwurf einer Infrastruktur für Ihre Anwendung AWS , auf der Sie eine hexagonale Architektur implementieren können. Wir empfehlen, dass Sie mit einer einfachen Architektur beginnen, um ein Minimum Viable Product (MVP) zu erstellen. Die meisten Microservices benötigen einen einzigen Einstiegspunkt für die Bearbeitung von Client-Anfragen, eine Rechenebene zum Ausführen des Codes und eine Persistenzschicht zum Speichern von Daten. Die folgenden AWS Dienste eignen sich hervorragend für den Einsatz als Clients, Primäradapter und Sekundäradapter in hexagonaler Architektur:

- Kunden: Amazon API Gateway, Amazon Simple Queue Service (Amazon SQS), Elastic Load Balancing, Amazon EventBridge
- Primäre Adapter: AWS Lambda Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), Amazon Elastic Compute Cloud (Amazon) EC2
- Sekundäre Adapter: Amazon DynamoDB, Amazon Relational Database Service (Amazon RDS), Amazon Aurora, API Gateway, Amazon SQS, Elastic Load Balancing EventBridge, Amazon Simple Notification Service (Amazon SNS)

In den folgenden Abschnitten werden diese Dienste im Kontext der hexagonalen Architektur ausführlicher behandelt.

Fangen Sie einfach an

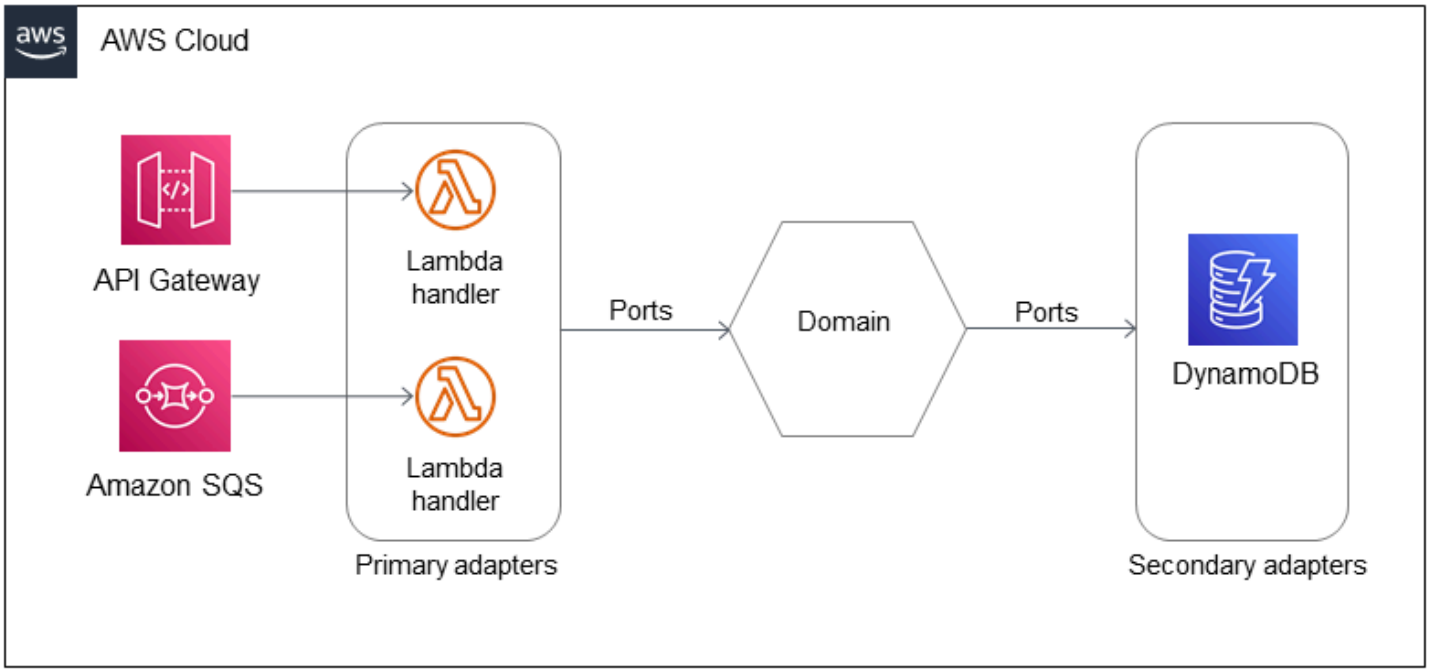
Wir empfehlen Ihnen, einfach zu beginnen, wenn Sie eine Anwendung mithilfe einer hexagonalen Architektur entwerfen. In diesem Beispiel wird API Gateway als Client (REST-API), Lambda als primärer Adapter (Compute) und DynamoDB als sekundärer Adapter (Persistenz) verwendet. Der Gateway-Client ruft den Einstiegspunkt auf, der in diesem Fall ein Lambda-Handler ist.



Diese Architektur ist vollständig serverlos und bietet dem Architekten einen guten Ausgangspunkt. Wir empfehlen, das Befehlsmuster in der Domäne zu verwenden, da es die Wartung des Codes erleichtert und sich an neue geschäftliche und nicht funktionale Anforderungen anpasst. Diese Architektur könnte ausreichen, um einfache Microservices mit wenigen Vorgängen zu erstellen.

Wenden Sie das CQRS-Muster an

Wir empfehlen, zum CQRS-Muster zu wechseln, wenn die Anzahl der Operationen auf der Domain skaliert werden soll. Mithilfe des folgenden Beispiels können Sie das CQRS-Muster als vollständig serverlose Architektur anwenden. AWS

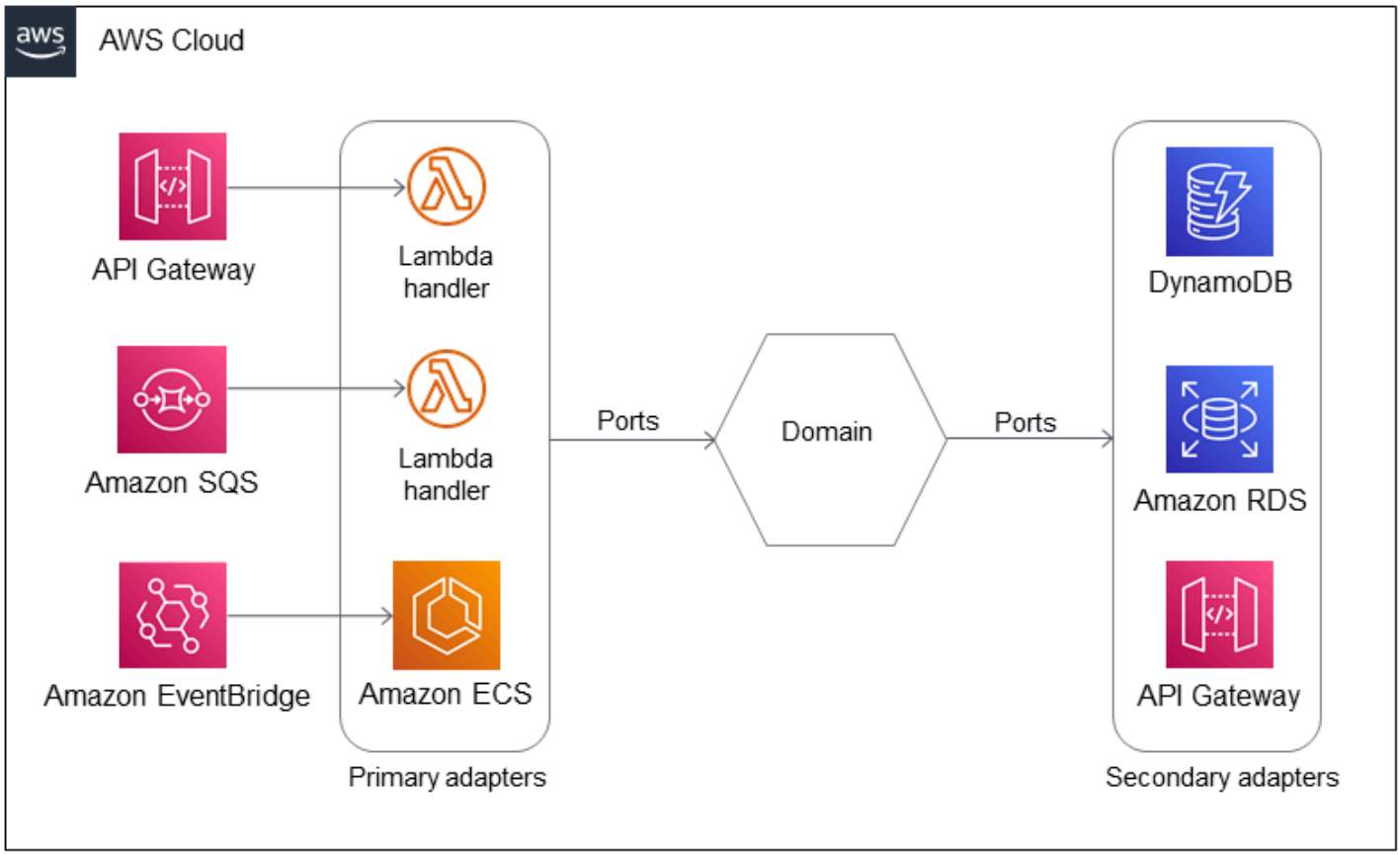


In diesem Beispiel werden zwei Lambda-Handler verwendet, einer für Abfragen und einer für Befehle. Abfragen werden synchron ausgeführt, wobei ein API-Gateway als Client verwendet wird. Befehle werden asynchron ausgeführt, indem Amazon SQS als Client verwendet wird.

Diese Architektur umfasst mehrere Clients (API Gateway und Amazon SQS) und mehrere Primäradapter (Lambda), die von ihren entsprechenden Einstiegspunkten (Lambda-Handlern) aufgerufen werden. Alle Komponenten gehören zu demselben begrenzten Kontext, befinden sich also innerhalb derselben Domäne.

Entwickeln Sie die Architektur weiter, indem Sie Container, eine relationale Datenbank und eine externe API hinzufügen

Container sind eine gute Option für lang andauernde Aufgaben. Möglicherweise möchten Sie auch eine relationale Datenbank verwenden, wenn Sie über ein vordefiniertes Datenschema verfügen und von der Leistungsfähigkeit der SQL-Sprache profitieren möchten. Darüber hinaus müsste die Domäne mit externen APIs kommunizieren. Sie können das Architekturbeispiel weiterentwickeln, um diese Anforderungen zu unterstützen, wie in der folgenden Abbildung dargestellt.

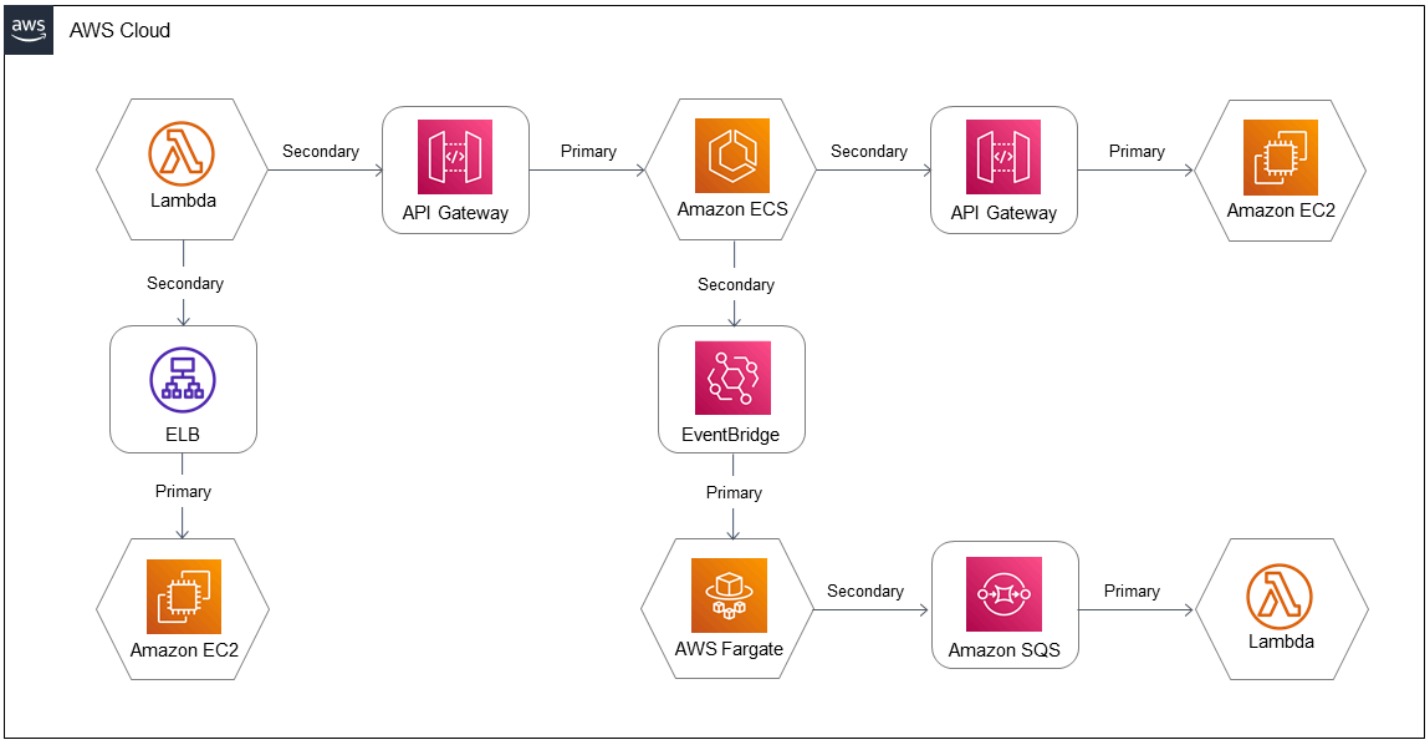


In diesem Beispiel wird Amazon ECS als primärer Adapter für das Starten von Aufgaben mit langer Laufzeit in der Domain verwendet. Amazon EventBridge (Kunde) initiiert eine Amazon ECS-Aufgabe (Einstiegspunkt), wenn ein bestimmtes Ereignis eintritt. Die Architektur beinhaltet Amazon RDS als weiteren sekundären Adapter zum Speichern relationaler Daten. Außerdem wird ein weiteres API-Gateway als sekundärer Adapter zum Aufrufen eines externen API-Aufrufs hinzugefügt. Daher verwendet die Architektur mehrere primäre und sekundäre Adapter, die auf unterschiedlichen zugrunde liegenden Rechenschichten in einer Geschäftsdomäne basieren.

Die Domäne ist über Abstraktionen, die als Ports bezeichnet werden, immer lose mit allen primären und sekundären Adaptern verbunden. Die Domäne definiert mithilfe von Ports, was sie von der Außenwelt benötigt. Da es in der Verantwortung des Adapters liegt, den Port zu implementieren, hat der Wechsel von einem Adapter zu einem anderen keine Auswirkungen auf die Domäne. Sie können beispielsweise von Amazon DynamoDB zu Amazon RDS wechseln, indem Sie einen neuen Adapter schreiben, ohne dass sich dies auf die Domain auswirkt.

Fügen Sie weitere Domänen hinzu (verkleinern)

Die sechseckige Architektur passt gut zu den Prinzipien einer Microservices-Architektur. Die bisher gezeigten Architekturbeispiele enthielten eine einzelne Domäne (oder einen begrenzten Kontext). Anwendungen umfassen in der Regel mehrere Domänen, die über primäre und sekundäre Adapter kommunizieren müssen. Jede Domäne stellt einen Microservice dar und ist lose mit anderen Domänen verknüpft.



In dieser Architektur verwendet jede Domain eine andere (n) Gruppe von Rechenumgebung (en). (Jede Domäne kann auch mehrere Rechenumgebungen haben, wie im vorherigen Beispiel.) Jede Domäne definiert ihre erforderlichen Schnittstellen für die Kommunikation mit anderen Domänen über Ports. Ports werden mithilfe von Primär- und Sekundäradaptern implementiert. Auf diese Weise wird die Domäne nicht beeinträchtigt, wenn eine Änderung am Adapter vorgenommen wird. Darüber hinaus sind Domänen voneinander entkoppelt.

In dem im vorherigen Diagramm gezeigten Architekturbeispiel werden Lambda, Amazon EC2, Amazon ECS und AWS Fargate Lambda als Primäradapter verwendet. API Gateway, Elastic Load Balancing und Amazon SQS werden als sekundäre Adapter verwendet. EventBridge

Häufig gestellte Fragen

Warum sollte ich eine sechseckige Architektur verwenden?

Die hexagonale Architektur verlagert den Fokus der Entwickler auf die Domänenlogik, vereinfacht die Testautomatisierung und verbessert die Codequalität und Anpassungsfähigkeit. Diese Verbesserungen führen zu einer schnelleren Markteinführung und einer einfacheren technischen und organisatorischen Skalierung.

Warum sollte ich domänengesteuertes Design verwenden?

Domain-Driven Design (DDD) ermöglicht es Ihnen, Softwarekomponenten und -konstrukte zu erstellen, indem Sie eine gemeinsame Sprache zwischen Geschäftsbeteiligten und Technikern verwenden. DDD hilft Ihnen bei der Verwaltung der Softwarekomplexität und ist eine effektive Strategie für die langfristige Wartung von Softwareprodukten.

Kann ich testgetriebene Entwicklung ohne hexagonale Architektur praktizieren?

Ja. Testgetriebene Entwicklung (TDD) ist nicht auf bestimmte Softwaredesignmuster beschränkt. Die hexagonale Architektur macht es jedoch einfacher, TDD zu praktizieren.

Kann ich mein Produkt ohne hexagonale Architektur und domänengesteuertes Design skalieren?

Ja. Die technische und organisatorische Produktskalierung kann mit den meisten Entwurfsmustern erreicht werden. Die sechseckige Architektur und DDD erleichtern jedoch die Skalierung und sind langfristig für große Projekte effektiver.

Welche Technologien sollte ich verwenden, um eine hexagonale Architektur zu implementieren?

Die hexagonale Architektur ist nicht auf einen bestimmten Technologie-Stack beschränkt. Wir empfehlen Ihnen, sich für eine Technologie zu entscheiden, die die Umkehrung von Abhängigkeiten und Komponententests unterstützt.

Ich entwickle ein Produkt, das am wenigsten praktikabel ist. Macht es Sinn, Zeit damit zu verbringen, über Softwarearchitektur nachzudenken?

Ja. Wir empfehlen Ihnen, Entwurfsmuster zu verwenden, mit denen Sie vertraut sind MVPs. Wir empfehlen Ihnen, die sechseckige Architektur so lange auszuprobieren, bis Ihre Techniker damit vertraut sind. Die Einrichtung einer sechseckigen Architektur für neue Projekte erfordert keinen wesentlich größeren Zeitaufwand als der Start ohne Architektur.

Ich entwickle ein Produkt, das nur minimal realisierbar ist, und habe keine Zeit, Tests zu schreiben.

Wenn Ihr MVP Geschäftslogik enthält, empfehlen wir dringend, dafür automatisierte Tests zu schreiben. Dadurch wird die Feedback-Schleife reduziert und Zeit gespart.

Welche zusätzlichen Entwurfsmuster kann ich bei der hexagonalen Architektur verwenden?

Verwenden Sie das CQRS-Muster, um die Skalierung des Gesamtsystems zu unterstützen. Verwenden Sie das Repository-Muster, um Ihr Domänenmodell zu speichern und wiederherzustellen. Verwenden Sie das Muster der Arbeitseinheit, um die Schritte des Transaktionsprozesses zu verwalten. Verwenden Sie Komposition statt Vererbung, um Domänenaggregate, Entitäten und Wertobjekte zu modellieren. Erstellen Sie keine komplexen Objekthierarchien.

Nächste Schritte

- Machen Sie sich anhand der im Abschnitt gesammelten Links weiter mit domänenorientierten Designkonzepten vertraut. Ressourcen
- Wenn Sie ein neues Projekt implementieren, verwenden Sie die in diesem Handbuch enthaltene Projektstrukturvorlage und implementieren Sie einige Funktionen.
- Wenn Sie gerade dabei sind, ein vorhandenes Projekt zu implementieren, identifizieren Sie Code, der zwischen schreibgeschützten und schreibgeschützten Vorgängen aufgeteilt werden kann. Abstrahieren Sie schreibgeschützten Code in Abfragediensten und platzieren Sie schreibgeschützten Code in Befehlsbehandlungsroutinen.
- Wenn Ihre grundlegende Projektstruktur eingerichtet ist, schreiben Sie Komponententests, richten Sie Continuous Integration (CI) mit Testautomatisierung ein und folgen Sie den Methoden der testgesteuerten Entwicklung (TDD).

Ressourcen

Referenzen

- Strukturieren Sie ein Python-Projekt in hexagonaler Architektur mithilfe von AWS Lambda (AWS Prescriptive Guidance Pattern)
- Agile Teams (Website für skaliertes Agile Framework)
- Architekturmuster mit Python, von Harry Percival und Bob Gregory (O'Reilly Media, 31. März 2020), insbesondere die folgenden Kapitel:
  - Befehle und Befehlshandler
  - Trennung von Verantwortlichkeiten zwischen Befehl und Abfrage (CQRS)
  - Repository-Muster
- Event Storming: Der intelligenteste Ansatz für Zusammenarbeit über Silogrenzen hinweg, von Alberto Brandolini (Event Storming-Website)
- Das Gesetz von Conway entmystifizieren, von Sam Newman (Thoughtworks-Website, 30. Juni 2014)
- Entwicklung einer evolutionären Architektur mit AWS Lambda, von James Beswick (Compute Blog, 8. Juli 2021)AWS
- Domänensprache: Bewältigung der Komplexität im Herzen von Software (Domain Language-Website)
- Facade, aus Dive Into Design Patterns von Alexander Shvets (E-Book, 5. Dezember 2018)
- GivenWhenThen, von Martin Fowler (21. August 2013)
- Implementierung eines domänengesteuerten Designs, von Vaughn Vernon (Addison-Wesley Professional, Februar 2013)
- Inverses Conway-Manöver (Thoughtworks-Website, 8. Juli 2014)
- Muster des Monats: Red Green Refactor (DZone Website, 2. Juni 2017)
- Erläuterung der SOLID-Entwurfsprinzipien: Prinzip der Abhängigkeitsinversion anhand von Codebeispielen, von Thorben Janssen (Stackify-Website, 7. Mai 2018)
- SOLID-Prinzipien: Erklärung und Beispiele von Simon Hoiberg (ITNEXT-Website, 1. Januar 2019)
- Die Kunst der agilen Entwicklung: Testgetriebene Entwicklung, von James Shore und Shane Warden (O'Reilly Media, 25. März 2010)

- Die SOLID-Prinzipien der objektorientierten Programmierung in einfachem Englisch erklärt, von Yigit Kemal Erinc (Beiträge zur objektorientierten Programmierung, 20. August 2020) freeCodeCamp
- Was ist eine ereignisgesteuerte Architektur? (Webseite)AWS

AWS Dienstleistungen

- Amazon API Gateway
- Amazon Aurora
- Amazon-DynamoDB
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- Elastic Load Balancing
- Amazon EventBridge
- AWS Fargate
- AWS Lambda
- Amazon Relational Database Service (Amazon RDS)
- Amazon-Simple-Notification-Service (Amazon-SNS)
- Amazon-Simple-Queue-Service (Amazon SQS)

Andere Tools

- Moto
- LocalStack

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
Erste Veröffentlichung	—	15. Juni 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

Weitere Informationen finden Sie unter 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 Transaktionen von verbindenden Anwendungen verarbeitet, 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.

autoritative 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 Cloud-Einführung (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 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 in der Regel 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 für die AWS 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 oderBitbucket 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 is commonly described as a pipeline. CI/CDkann 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 unterKontinuierliche 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 Variation 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.

betroffene 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.

Bereitstellung

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, wie z. B. unbeabsichtigte Fehlkonfigurationen oder ein Malware-Angriff.

Disaster Recovery (DR)

Die Strategie und der Prozess, die Sie verwenden, um Ausfallzeiten und Datenverluste aufgrund einer Katastrophe zu 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 Kerngeschäftszielen 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 Sicherheitsthemen 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, 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.

Holdout-Daten

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 Translationsdaten. 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 sich Infrastruktur als Code an.

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) IIo

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-Servicemanagement)

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

Weitere Informationen finden Sie in der IT-Informationsbibliothek.

ITSM

Siehe IT-Servicemanagement.

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 Infrastrukturumgebung 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

Weitere Informationen finden Sie unter Label-basierte 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 Nachverfolgung, 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 in sind. AWS Organizations Ein Konto kann jeweils nur einer Organisation angehören.

DURCHEINANDER

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.

OAI

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. Weitere Informationen finden Sie unter Datenpersistenz in Microservices aktivieren.

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. Bei dieser Entität handelt es sich in der Regel um einen 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 mit der Steuerung konform ist, 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).

LAPPEN

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.

Refaktorierung

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 der Wiederherstellungszeit.

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 Management Console 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 Benutzeranmeldedaten, 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 EC2 Amazon-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-Endpunkt

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 unbekannte 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. Weitere Informationen finden Sie im Leitfaden Quantifizieren der Unsicherheit in Deep-Learning-Systemen.

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.

Zero-Shot-Aufforderung

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.