Testen serverloser Anwendungen auf AWS

AWS Präskriptive Leitlinien

AWS Präskriptive Leitlinien: Testen serverloser Anwendungen auf AWS

Table of Contents

Einführung 1
-Übersicht 1
Voraussetzungen 2
Definitionen 2
Ziele 2
Erhöhung der Softwarequalität 3
Verkürzung des Zeitaufwands für die Implementierung oder Änderung von Features 5
Testtechniken für Serverless-Anwendungen 7
Mocktests 7
Emulationstest 9
Testen in der Cloud 10
Herausforderungen beim Testen von Serverless-Anwendungen 12
Beispiel: Eine Lambda-Funktion, die einen Amazon S3 S3-Bucket erstellt 12
Beispiel: Eine Lambda-Funktion, die Nachrichten von Amazon SQS verarbeitet 13
Bewährte Methoden zum Testen von Serverless-Anwendungen 14
Priorisieren Sie Tests in der Cloud 14
Bei Bedarf Mocks verwenden 15
Machen Sie sich mit den Kompromissen von Emulationstests vertraut 15
Durchführen von Tests anhand natürlicher Grenzen 16
Identifizieren Sie die Grenzen der Architektur 16
Lambda-Code von Geschäftslogik trennen 17
Behandeln Sie Grenzen wie Verträge 17
Verwenden Sie Testlösungen für asynchrone Workflows 17
Organisieren Sie Cloud-Umgebungen zur Isolierung von Entwicklern 18
Feedback-Schleifen beschleunigen 19
Häufig gestellte Fragen 20
Ich habe eine Lambda-Funktion, die Berechnungen durchführt und ein Ergebnis zurückgibt, ohne andere Dienste aufzurufen. Muss ich das wirklich in der Cloud testen? 20
Wie können Tests in der Cloud beim Einheitentest helfen? Wenn es sich in der Cloud befindet und eine Verbindung zu anderen Ressourcen herstellt, ist das nicht ein Integrationstest? 20
Nächste Schritte und Ressourcen 22
Beispielimplementierungen 22
Weitere Informationen 22
Referenzen 22

Tools 22
Mitwirkende 24
Inhaltserstellung 24
Überprüfend 24
Technisches Schreiben 24
Dokumentverlauf 25
Glossar 26
..... 26
A 27
B 30
C 32
D 35
E 40
F 42
G 44
H 45
I 47
L 49
M 50
O 55
P 58
Q 61
R 61
S 64
T 68
U 70
V 70
W 71
Z 72
..... lxxiii

Serverlose Anwendungen testen auf AWS

Amazon Web Services (???Mitwirkende)

März 2026 (Verlauf der Dokumente)

In diesem Leitfaden werden Methoden für das Testen von Serverless-Anwendungen erörtert, die Herausforderungen beschrieben, denen Sie beim Testen begegnen können, und bewährte Methoden vorgestellt. Diese Testtechniken sollen Ihnen helfen, schneller zu iterieren und Ihren Code sicherer zu veröffentlichen.

Dieses Handbuch richtet sich an Entwickler, die Teststrategien für ihre Serverless-Anwendungen entwickeln möchten. Sie können den Leitfaden als Ausgangspunkt verwenden, um mehr über Teststrategien zu erfahren. Besuchen Sie anschließend das Repository für Serverless-Testbeispiele, um Beispiele von Tests zu sehen, die den in diesem Leitfaden beschriebenen Mustern und bewährten Methoden folgen. In diesem Leitfaden werden Methoden für serverlose Tests beschrieben, die Herausforderungen beschrieben, mit denen Kunden beim Testen serverloser Anwendungen konfrontiert sind, und es werden bewährte Methoden zum Testen serverloser Anwendungen vorgestellt. Diese Techniken sollen Entwicklern helfen, schneller zu iterieren und Releases sicherer zu machen.

-Übersicht

Automatisierte Tests sind wichtige Investitionen, die dazu beitragen, die Anwendungsqualität und die Geschwindigkeit der Entwicklung sicherzustellen. Das Testen beschleunigt auch das Feedback für Entwickler. Als Entwickler möchten Sie in der Lage sein, schnell an Ihrer Anwendung zu iterieren und Feedback zur Qualität Ihres Codes zu erhalten. Viele Entwickler sind es gewohnt, Anwendungen zu schreiben, die sie in einer Umgebung auf ihrem Desktop bereitstellen, entweder direkt auf ihrem Betriebssystem oder in einer containerbasierten Umgebung. Wenn Sie in Desktop- oder Container-basierten Umgebungen arbeiten, schreiben Sie in der Regel Tests für Code, der vollständig auf Ihrem Desktop gehostet wird. In Serverless-Anwendungen können Architekturkomponenten jedoch möglicherweise nicht in einer Desktop-Umgebung bereitgestellt werden, sondern sind eventuell nur in der Cloud vorhanden. Eine cloudbasierte Architektur kann Persistenzschichten, Messaging-Systeme, Sicherheitskonstrukte und andere Komponenten APIs umfassen. Wenn Sie Anwendungscode schreiben, der auf diesen Komponenten basiert, kann es schwierig sein, die beste Methode zum Entwerfen und Ausführen von Tests zu ermitteln.

Dieser Leitfaden hilft Ihnen dabei, sich auf eine Teststrategie einzustellen, die Reibungsverluste und Verwirrung reduziert und die Codequalität verbessert.

Voraussetzungen

In diesem Leitfaden wird davon ausgegangen, dass Sie mit den Grundlagen automatisierter Tests vertraut sind, einschließlich der Art und Weise, wie automatisierte Softwaretests zur Sicherstellung der Softwarequalität eingesetzt werden. Das Handbuch bietet eine allgemeine Einführung in die Teststrategie für serverlose Anwendungen und erfordert keine praktische Erfahrung beim Schreiben von Tests.

Definitionen

In diesem Leitfaden werden die folgenden Begriffe verwendet:

- Einheitentests sind Tests, die isoliert gegen Code für eine einzelne Architekturkomponente ausgeführt werden.
- Integrationstests werden mit zwei oder mehr Architekturkomponenten durchgeführt, typischerweise in einer Cloud-Umgebung.
- End-to-end Tests verifizieren das Verhalten ganzer Anwendungen oder Workflows.
- Emulatoren sind Anwendungen (oft von Drittanbietern bereitgestellt), die darauf ausgelegt sind, einen Cloud-Dienst nachzuahmen, ohne Cloud-Ressourcen bereitzustellen oder aufzurufen.
- Mocks(auch Fälschungen genannt) sind Implementierungen in einer Testanwendung, die eine Abhängigkeit durch eine Simulation dieser Abhängigkeit ersetzen.

Ziele

Die bewährten Methoden in diesem Leitfaden sollen Ihnen helfen, zwei Hauptziele zu erreichen:

- Steigerung der Qualität von Serverless-Anwendungen
  - Testen an Architekturgrenzen
  - Testen an Codegrenzen
- Verkürzung des Zeitaufwands für die Implementierung oder Änderung von Features

Erhöhung der Softwarequalität

Die Qualität einer Anwendung hängt in hohem Maße von der Fähigkeit der Entwickler ab, eine Vielzahl von Szenarien zu testen, um die Funktionalität zu überprüfen. Wenn Sie keine automatisierten Tests implementieren oder, was noch typischer ist, wenn Ihre Tests die erforderlichen Szenarien nicht angemessen abdecken, kann die Qualität Ihrer Anwendung weder bestimmt noch garantiert werden.

In einer serverbasierten Architektur können Teams den Testumfang leicht definieren: Jeder Code, der auf dem Anwendungsserver läuft, muss getestet werden. Andere Komponenten, die den Server aufrufen, oder Abhängigkeiten, die der Server aufruft, werden von dem Team, das für die Anwendung auf dem Server verantwortlich ist, häufig als extern betrachtet und können daher nicht getestet werden.

Serverless-Anwendungen bestehen häufig aus kleineren Arbeitseinheiten, wie AWS Lambda - Funktionen, die in ihrer eigenen Umgebung laufen. Teams werden wahrscheinlich für mehrere dieser kleineren Einheiten innerhalb einer einzigen Anwendung zuständig sein. Einige Anwendungsfunktionalitäten können vollständig an verwaltete Services wie Amazon Simple Storage Service (Amazon S3) oder Amazon Simple Queue Service (Amazon SQS) delegiert werden, ohne dass intern entwickelter Code verwendet wird. Herkömmliche serverbasierte Modelle für Softwaretests schließen verwaltete Services möglicherweise aus, da sie als außerhalb der Anwendung betrachtet werden. Dies kann zu einer unzureichenden Abdeckung führen, sodass kritische Szenarien möglicherweise auf manuelle Sondierungstests oder auf einige wenige Integrationstestfälle beschränkt sind, bei denen das Ergebnis je nach Umgebung unterschiedlich ist. Daher kann die Softwarequalität durch die Einführung von Teststrategien, die das Verhalten von verwalteten Services und Cloud-Konfigurationen umfassen, verbessert werden.

Testen an Architekturgrenzen

Wenn serverlose Anwendungen wachsen, verteilen sie sich naturgemäß auf mehrere Architekturkomponenten. Dabei werden zwar AWS verteilte Funktionen verwendet, das end-to-end Verhalten kann jedoch schwer verständlich sein.

Identifizierung natürlicher Grenzen

Wenn Sie Ihre Architektur nach bewährten Methoden für serverlose Systeme entwerfen (eine Funktion = ein Job, Entkopplung), werden Sie natürliche Grenzen rund um Subsysteme feststellen. Diese Grenzen stellen logische Trennpunkte in Ihrer Anwendung dar.

Grenzen als Testverträge

Diese architektonischen Grenzen eignen sich hervorragend zum Testen von Kanten. Behandeln Sie jede Grenze als Vertrag und überprüfen Sie, ob sie sich gemäß der definierten Spezifikation verhält. Stellen Sie sich diese Grenzen als Nähte in Ihrer Anwendung vor, in die Sie eine Testvalidierung einfügen können.

Wichtigste Vorteile

Im Folgenden sind die wichtigsten Vorteile von Tests an Architekturgrenzen aufgeführt:

- Fokussierter Testumfang — Testen Sie Subsysteme unabhängig voneinander, ohne die gesamte Anwendung verstehen zu müssen.
- Vertragsvalidierung — Stellen Sie sicher, dass jede Grenze ihr erwartetes Verhalten beibehält, während sich das System weiterentwickelt
- Instrumentierung mit doppeltem Verwendungszweck — Dieselben Grenzen bieten hervorragende Beobachtungsmöglichkeiten in der Produktion
- Test Harness — Ermöglicht das Testen asynchroner serverloser Systeme. Es hilft Ihnen beim Testen ereignisgesteuerter Architekturen, indem es Ereignisse erfasst und validiert, während sie Ihr Subsystem durchlaufen.

Testen an Codegrenzen

Definieren Sie klare Codegrenzen, indem Sie Infrastrukturcode wie Lambda-Code von Ihrer Kerngeschäftslogik trennen. Durch diese Trennung entstehen unterschiedliche Testbereiche, die Ihre Teststrategie vereinfachen.

Das Grenzmuster

Richten Sie in Ihren Lambda-Funktionen zwei klare Codegrenzen ein:

- Äußere Grenze (Lambda-Handler) — Eine schlanke Adapterschicht, die spezifische Probleme behandelt AWS Lambda
- Innere Grenze (Geschäftslogik) — Reine Geschäftslogikmethoden unabhängig von der Lambda-Laufzeit

Handler als Adapter (äußerer Geltungsbereich)

Ihr Lambda-Funktionshandler sollte eine dünne Schicht sein, die:

- Extrahiert Daten aus den eingehenden `event` Objekten `context`
- Validiert die extrahierten Daten
- Übergibt nur relevante Details an Methoden der Geschäftslogik
- Gibt Ergebnisse im erwarteten Format für Lambda zurück

Geschäftslogik (innerer Geltungsbereich)

Ihre Kerngeschäftslogik sollte:

- Funktioniert unabhängig von Lambda-spezifischen Details
- Akzeptieren Sie einfache, validierte Eingaben
- Gibt vorhersehbare Ergebnisse zurück
- Erfordern minimale Abhängigkeiten für die Initialisierung

Vorteile des Testens nach Umfang

- Tests innerer Grenzen — Umfassende Komponententests rund um Geschäftslogik ohne Lambda-Komplexität oder Umgebungskonfiguration
- Tests an äußeren Grenzen — Gezielte Integrationstests zur Validierung der Ereignisbehandlung und Datenextraktion auf der Adapterebene
- Minimaler Testaufwand — Für die meisten Ihrer Tests sind keine komplexen Umgebungen oder umfangreichen Abhängigkeiten erforderlich

Dieser grenzenbasierte Ansatz ermöglicht es Ihnen, den größten Teil Ihres Codes als reine Funktionen zu testen und gleichzeitig die Lambda-Tests minimal und zielgerichtet zu halten.

Verkürzung des Zeitaufwands für die Implementierung oder Änderung von Features

Sie können die Auswirkungen von Softwarefehlern und Konfigurationsproblemen auf Kosten und Zeitpläne minimieren, indem Sie diese Probleme in einem iterativen Entwicklungszyklus beheben. Wenn ein Entwickler diese Probleme nicht erkennt, müssen mehr Mitarbeiter zusätzliche Anstrengungen unternehmen, um die Probleme zu identifizieren.

Eine Serverless-Architektur kann verwaltete Services umfassen, die wichtige Anwendungsfunktionen über API-Aufrufe bereitstellen. Aus diesem Grund sollte Ihr Entwicklungszyklus Tests beinhalten,

die sowohl den glücklichen Weg (bei dem sich Interaktionen mit diesen Diensten erwartungsgemäß verhalten) als auch den traurigen Pfad (bei dem Anrufe fehlschlagen, unerwartete Antworten zurückgeben oder sich in den Umgebungen unterschiedlich verhalten) validieren. Ohne diese Tests können Probleme auftreten, die auf Unterschiede zwischen Ihrer lokalen Umgebung und der bereitgestellten Umgebung zurückzuführen sind. In diesem Fall müssen Sie zusätzliche Zeit damit verbringen, einen Fix zu reproduzieren und zu verifizieren, da jetzt bei jeder Iteration Änderungen anhand einer Umgebung validiert werden müssen, die sich von Ihrer bevorzugten Konfiguration unterscheidet.

Eine geeignete Serverless-Teststrategie verbessert Ihre Iterationszeit, indem sie genaue Ergebnisse für Tests liefert, die Aufrufe anderer Services beinhalten.

Testtechniken für Serverless-Anwendungen

Es gibt drei Hauptansätze für die Ausführung von Tests mit serverlosem Anwendungscode: Mock-Tests, Emulationstests und Tests in der Cloud. Im Allgemeinen finden Sie im Rahmen eines einzelnen Projekts eine Mischung dieser Ansätze. Sie können beispielsweise Testtests verwenden, wenn Sie Ihren Code lokal entwickeln, Emulationstests, um das Dienstverhalten vor der Bereitstellung genauer nachzubilden, und Cloud-Tests als Teil eines nächtlichen CI/CD-Prozesses (Continuous Integration and Continuous Delivery) verwenden.

Mocktests

Mocktests sind eine Strategie, bei der Sie Ersatzobjekte in Ihrem Code erstellen, die das Verhalten von Cloud-Services simulieren. Sie können beispielsweise einen Test schreiben, der ein Modell des Amazon S3 S3-Service verwendet, und Sie können dieses Modell so konfigurieren, dass bei jedem Aufruf der `PutObject` Methode ein bestimmtes Antwortobjekt zurückgegeben wird. Wenn der Test ausgeführt wird, gibt der Mock die Antwort zurück, ohne Amazon S3 oder andere Service-Endpunkte aufzurufen.

Diese Ersatzobjekte werden häufig von einem Mock-Framework generiert, um den Implementierungsaufwand für einen bestimmten Test zu reduzieren. Einige Schein-Frameworks sind generisch und andere wurden speziell dafür entwickelt AWS SDKs.

Mocks fallen zusammen mit Stummeln in die breitere Kategorie von Fälschungen. Mocks unterscheiden sich von Emulatoren (auf die weiter unten in diesem Abschnitt eingegangen wird) dadurch, dass Mocks in der Regel von einem Entwickler als Teil des Testcodes erstellt oder konfiguriert werden, wohingegen Emulatoren eigenständige Anwendungen sind, die APIs (wie REST APIs) auf die gleiche Weise verfügbar gemacht werden wie die Systeme, die sie emulieren.

Die Verwendung von Mocks bietet folgende Vorteile:

- Mocks kann Dienste von Drittanbietern simulieren, auf die Ihre Anwendung keinen Einfluss hat, z. B. SaaS-Anbieter (Software as a Service), ohne direkten Zugriff auf diese Dienste zu benötigen. APIs
- Mocks sind auch für das Testen von Fehlerbedingungen nützlich, insbesondere wenn solche Bedingungen (wie z. B. Serviceausfälle) schwer zu simulieren sind.
- Wie Emulatoren können Mock-Frameworks nach ihrer Konfiguration eine schnelle lokale Entwicklung ermöglichen.

- Mocks können Ersatzverhalten für praktisch jede Art von Objekt bereitstellen, so dass Mocking-Strategien eine breitere Palette von Diensten abdecken können als Emulatoren.
- Wenn neue Funktionen oder Verhaltensweisen verfügbar werden, können Scheintests schneller reagieren, indem sie ein generisches Mock-Framework verwenden (oder darauf zurückgreifen), das die neuen Funktionen simulieren kann, sobald das aktualisierte AWS SDK verfügbar ist.

Mock-Tests haben die folgenden Nachteile:

- Mocks erfordern in der Regel einen nicht unerheblichen Einrichtungs- und Konfigurationsaufwand, insbesondere dann, wenn versucht wird, Rückgabewerte von verschiedenen Services zu ermitteln, um die Antworten korrekt nachzuahmen.
- Da Mocks von den Entwicklern geschrieben oder konfiguriert werden, die die Tests schreiben, ist dies eine zusätzliche Verantwortung für Entwickler.
- Möglicherweise benötigen Sie Zugriff auf die Cloud, um die Werte APIs und Rückgabewerte von Diensten zu verstehen.
- Mocks können auch schwierig zu verwalten sein, da sie immer dann aktualisiert werden müssen, wenn sich die simulierten Cloud-API-Signaturen ändern, Rückgabewertschemas weiterentwickelt werden oder die getestete Anwendungslogik erweitert wird, um neue Aufrufe zu tätigen. APIs Diese Änderungen bedeuten zusätzlichen Workload für Entwickler bei der Testentwicklung.
- Scheintests können lokal erfolgreich sein, schlagen aber in der Cloud fehl, weil sie das Verhalten realer Dienste simulieren — und nicht replizieren —, sodass umgebungsspezifische Probleme unentdeckt bleiben.
- Schein-Frameworks können ebenso wie Emulatoren keine AWS Identity and Access Management (IAM-) Richtlinienverstöße, Servicekontingentbeschränkungen oder Beschränkungen der Nutzlastgröße erkennen und lösen auch keine Zusatzdienste wie Amazon oder Amazon aus CloudWatch AWS CloudTrail, GuardDuty die das Anwendungsverhalten in einer bereitgestellten Umgebung beeinflussen können.
- Obwohl Mocks besser simulieren können, was eine Anwendung tun wird, wenn die Autorisierung fehlschlägt oder ein Kontingent überschritten wird, kann anhand von Scheintests nicht festgestellt werden, welches Ergebnis tatsächlich eintritt, wenn der Code in der Produktionsumgebung bereitgestellt wird.

Emulationstest

Emulationstests werden durch lokal ausgeführte Anwendungen, sogenannte Emulatoren, ermöglicht, die sich ähneln. AWS-Services Emulatoren haben ähnliche APIs Eigenschaften wie ihre Cloud-Gegenstücke und liefern ähnliche Rückgabewerte. Sie können auch Zustandsänderungen simulieren, die durch API-Aufrufe initiiert werden. Beispielsweise könnte ein Amazon S3 S3-Emulator ein Objekt auf der lokalen Festplatte speichern, wenn die `PutObject` Methode aufgerufen wird. Wenn `GetObject` es mit demselben Schlüssel aufgerufen wird, liest der Emulator dasselbe Objekt von der Festplatte und gibt es zurück.

Zu den Vorteilen von Emulationstests gehören folgende:

- Sie bieten Entwicklern, die es gewohnt sind, Code in einer lokalen Umgebung zu entwickeln, eine vertraute Umgebung. Wenn Sie beispielsweise mit der Entwicklung einer n-Tier-Anwendung vertraut sind, haben Sie möglicherweise eine Datenbank-Engine und einen Webserver, ähnlich denen, die in der Produktion laufen, auf Ihrem lokalen Computer ausgeführt, um schnelle, lokale, isolierte Testfunktionen bereitzustellen.
- Da keine Änderungen an der Cloud-Infrastruktur (wie Cloud-Konten für Entwickler) erforderlich sind, ist die Implementierung mit bestehenden Testmustern einfach. Emulationstests bieten die Vorteile schneller, lokaler Entwicklungsiterationen.

Emulatoren haben jedoch auch mehrere Nachteile:

- Sie sind oft schwer einzurichten und zu replizieren, insbesondere wenn sie als Teil von CI/CD Pipelines verwendet werden. Dies könnte den Workload und den Wartungsaufwand für IT-Mitarbeiter oder Entwickler, die ihre eigenen Softwareinstallationen verwalten, erhöhen.
- Emulierte Funktionen hinken APIs in der Regel den Änderungen der Dienste hinterher und können die Einführung neuer Funktionen behindern, bis Unterstützung hinzugefügt wird.
- Emulatoren benötigen möglicherweise Support, Updates, Bugfixes oder Verbesserungen der Featureparität, für die der Autor des Emulators verantwortlich ist (bei dem es sich häufig um einen Drittanbieter handelt).
- Tests, die auf Emulatoren basieren, können lokal zu erfolgreichen Ergebnissen führen, in der Cloud können sie jedoch aufgrund von Interaktionen mit anderen Aspekten AWS wie IAM-Richtlinien und -Kontingenten fehlschlagen, die im Allgemeinen nicht emuliert werden.

- In einigen AWS-Services Fällen sind keine Emulatoren verfügbar. Wenn Sie sich also nur auf Emulation verlassen, haben Sie möglicherweise keine zufriedenstellende Testoption für Teile Ihrer Anwendung.

Testen in der Cloud

Beim Testen in der Cloud werden Tests mit Code ausgeführt, der in einer Cloud-Umgebung statt in einer Desktop-Umgebung bereitgestellt wird. Der Wert von Tests in der Cloud steigt mit der cloudnativen Anwendungsentwicklung. Beispiel:

- Sie können Tests in der Cloud mit den neuesten Funktionen APIs und Dienstfunktionen durchführen und so sicherstellen, dass Ihre Tests die neuesten Rückgabewerte und Verhaltensweisen widerspiegeln, während AWS ständig neue Dienste und Funktionen eingeführt werden.
- Ihre Tests können sich auf IAM-Richtlinien, Service Quotas, Konfigurationen und alle Services beziehen.
- Ihre Cloud-Testumgebung entspricht am ehesten Ihrer Produktions-Laufzeitumgebung, so dass Tests, die Sie in der Cloud ausführen, wahrscheinlich die konsistentesten Ergebnisse erzielen, während sie durch die Umgebungen laufen.

Zu den Nachteilen von Tests in der Cloud gehören:

- Bereitstellungen in Cloud-Umgebungen benötigen in der Regel mehr Zeit als Bereitstellungen in Desktop-Umgebungen. Tools wie AWS Serverless Application Model (AWS SAM) Accelerate, AWS Cloud Development Kit (AWS CDK) watch mode, und SST tragen dazu bei, die Latenz zu verringern, die mit Iterationen der Cloud-Bereitstellung verbunden ist.
- Beim Testen in der Cloud können im Gegensatz zum lokalen Testen, bei dem Ihre lokale Entwicklungsumgebung genutzt wird, zusätzliche Servicekosten anfallen.
- Tests in der Cloud sind möglicherweise weniger durchführbar, wenn Sie keinen Hochgeschwindigkeits-Internetzugang haben.
- In regulierten Branchen können Unternehmenssicherheitsrichtlinien den Zugriff von Entwicklern auf Cloud-Umgebungen einschränken, was es schwierig oder unmöglich macht, Cloud-Tests als Teil eines lokalen Entwicklungsworkflows durchzuführen.
- Umgebungsgrenzen werden häufig auf Stack-Ebene in gemeinsam genutzten Konten für Entwicklerumgebungen gezogen, manchmal mithilfe von Strategien vom Typ Namespace, wie

z. B. der Verwendung von Präfixen zur Identifizierung von Eigentümern. In Vorproduktions- und Produktionsumgebungen werden die Grenzen in der Regel auf Kontoebene gezogen, um Workloads von „Noisy-Neighbor“-Problemen zu isolieren und Sicherheitskontrollen mit geringsten Berechtigungen zum Schutz sensibler Daten zu implementieren. Die Anforderung, isolierte Umgebungen zu erstellen, kann die DevOps Teams zusätzlich belasten, insbesondere wenn sie in einem Unternehmen tätig sind, das strenge Kontrollen in Bezug auf Konten und Infrastruktur hat.

Herausforderungen beim Testen von Serverless-Anwendungen

Wenn Sie Emulatoren und simulierte Aufrufe verwenden, um Ihre serverlose Anwendung auf Ihrem lokalen Desktop zu testen, kann es zu Testinkonsistenzen kommen, wenn Ihr Code in Ihrer Pipeline von Umgebung zu Umgebung fortschreitet. CI/CD Die Komponententests, die Sie auf Ihrem Desktop erstellen, um die Geschäftslogik Ihrer Anwendung zu überprüfen, beinhalten möglicherweise wichtige Aspekte von Cloud-Diensten nicht oder stellen diese nicht korrekt dar. Vollständige Tests können nicht isoliert lokal durchgeführt werden. Sie erfordern die Überprüfung der Berechtigungen und Konfigurationen mehrerer Services.

In den folgenden Abschnitten werden die Herausforderungen beschrieben, auf die Sie bei der Implementierung einer Cloud-Teststrategie stoßen könnten. In den folgenden Abschnitten werden die Herausforderungen beschrieben, mit denen Kunden bei der Implementierung einer Cloud-Teststrategie konfrontiert sind, sowie unsere Empfehlungen zu bewährten Verfahren für eine effektive Testabdeckung.

Beispiel: Eine Lambda-Funktion, die einen Amazon S3 S3-Bucket erstellt

Wenn die Logik einer Lambda-Funktion von der Erstellung eines Amazon S3-Buckets abhängt, sollte ein vollständiger Test bestätigen, dass Amazon S3 erfolgreich aufgerufen und der Bucket erfolgreich erstellt wurde. In einem Test-Setup können Sie eine Erfolgsreaktion simulieren und möglicherweise einen Testfall hinzufügen, um eine Fehlerreaktion zu behandeln. In einem Emulationstestszenario könnte die `CreateBucket` API zwar aufgerufen werden, aber die Identität, die den Aufruf tätigt, stammt nicht von der Übernahme einer Rolle durch den Lambda-Service, sondern es wird stattdessen eine Platzhalterauthentifizierung verwendet — dies ist oft Ihre tolerantere Rollen- oder Benutzeridentität.

Die zuvor besprochenen Mock- und Emulations-Setups werden höchstwahrscheinlich testen, was die Lambda-Funktion tun wird, wenn sie erfolgreich (oder erfolglos) Amazon S3 aufruft. Diese Tests können jedoch nicht erfassen, ob die Lambda-Funktion den Bucket aufgrund der Konfiguration der Funktion erfolgreich erstellen kann. Diese Konfiguration wird wahrscheinlich durch Infrastructure as Code (IaC) für Produkte und Dienste wie AWS CloudFormation AWS SAM, oder HashiCorp Terraform dargestellt. Ein mögliches Problem besteht darin, dass der Rolle, die der Funktion

zugewiesen ist, keine Richtlinie angehängt ist, die die `s3:CreateBucket` Aktion zulässt. Daher schlägt die Funktion immer fehl, wenn sie in einer Cloud-Umgebung bereitgestellt wird.

Beispiel: Eine Lambda-Funktion, die Nachrichten von Amazon SQS verarbeitet

Wenn eine Amazon Simple Queue Service (Amazon SQS) -Warteschlange die Quelle einer Lambda-Funktion ist, sollte ein vollständiger Test sicherstellen, dass die Lambda-Funktion erfolgreich aufgerufen wurde, wenn eine Nachricht in eine Warteschlange gestellt wird. Emulationstests und Scheintests sind im Allgemeinen so eingerichtet, dass der Lambda-Funktionscode direkt ausgeführt wird und die Amazon SQS SQS-Integration simuliert wird, indem eine JSON-Ereignisnutzlast (oder ein deserialisiertes Objekt) als Eingabe für den Funktionshandler übergeben wird.

Lokale Tests, die die Amazon SQS-Integration simulieren, testen, was die Lambda-Funktion tut, wenn sie von Amazon SQS mit einer bestimmten Nutzlast aufgerufen wird, aber sie stellen nicht sicher, dass Amazon SQS die Lambda-Funktion erfolgreich aufruft, wenn sie in einer Cloud-Umgebung bereitgestellt wird.

Einige Beispiele für Konfigurationsprobleme, die bei Amazon SQS und Lambda auftreten können, sind die folgenden:

- Das Amazon-SQS-Sichtbarkeits-Timeout ist zu niedrig, was zu mehreren Aufrufen führt, obwohl nur einer vorgesehen war.
- Die Ausführungsrolle der Lambda-Funktion erlaubt nicht das Lesen von Nachrichten aus der Warteschlange (durch `sqs:ReceiveMessagesqs:DeleteMessage`, oder`sqs:GetQueueAttributes`).
- Das Beispielereignis, das an die Lambda-Funktion übergeben wird, überschreitet das Amazon-SQS-Nachrichtengrößenkontingent. Daher ist der Test ungültig, da Amazon SQS niemals in der Lage wäre, eine Nachricht dieser Größe zu senden.

Wie diese Beispiele zeigen, liefern Tests, die sich mit der Geschäftslogik, aber nicht mit den Konfigurationen zwischen Cloud-Services befassen, wahrscheinlich zu unzuverlässigen Ergebnissen.

Bewährte Methoden zum Testen von Serverless-Anwendungen

In den folgenden Abschnitten werden bewährte Methoden für die Erzielung einer effektiven Abdeckung beim Testen von Serverless-Anwendungen beschrieben.

Priorisieren Sie Tests in der Cloud

Für gut konzipierte Anwendungen können Sie eine Vielzahl von Testtechniken einsetzen, um eine Reihe von Anforderungen und Bedingungen zu erfüllen. Auf der Grundlage der aktuellen Tools empfehlen wir Ihnen jedoch, sich so weit wie möglich auf das Testen in der Cloud zu konzentrieren. Obwohl Tests in der Cloud zu Latenz für Entwickler führen, die Kosten erhöhen und manchmal Investitionen in zusätzliche DevOps Kontrollen erfordern können, bietet diese Technik die zuverlässigste, genaueste und vollständigste Testabdeckung.

Sie sollten Zugriff auf isolierte Umgebungen haben, in denen Sie Tests durchführen können. Idealerweise sollte jeder Entwickler über ein eigenes Tool verfügen, AWS-Konto um Probleme mit der Benennung von Ressourcen zu vermeiden, die auftreten können, wenn mehrere Entwickler, die am selben Code arbeiten, versuchen, API-Aufrufe für Ressourcen mit identischen Namen bereitzustellen oder aufzurufen. Diese Umgebungen sollten mit den entsprechenden Warnungen und Kontrollen konfiguriert werden, um unnötige Ausgaben zu vermeiden. Sie können beispielsweise die Art, Stufe oder Größe der Ressourcen einschränken, die erstellt werden können, und E-Mail-Benachrichtigungen einrichten, wenn die geschätzten Kosten einen bestimmten Schwellenwert überschreiten.

Wenn Sie eine einzelne Ressource AWS-Konto mit anderen Entwicklern teilen müssen, sollten automatisierte Testprozesse Ressourcen so benennen, dass sie für jeden Entwickler eindeutig sind. Aktualisierungsskripts oder TOML-Konfigurationsdateien, die die AWS SAM CLI sam deploy - oder sam sync-Befehle auslösen, können beispielsweise automatisch einen Stacknamen angeben, der den Benutzernamen des lokalen Entwicklers enthält.

Das Testen in der Cloud ist für alle Testphasen nützlich, einschließlich Unit-Tests, Integrationstests und end-to-end Tests.

Bei Bedarf Mocks verwenden

Mock-Frameworks sind ein wertvolles Tool zum Schreiben schneller Einheitentests. Sie sind besonders wertvoll, wenn Tests komplexe interne Geschäftslogiken wie mathematische oder finanzielle Berechnungen oder Simulationen abdecken müssen. Suchen Sie nach Einheitentests, die eine große Anzahl von Testfällen oder Eingabevariationen enthalten, bei denen die Eingaben das Muster oder den Inhalt von Aufrufen an andere Cloud-Services nicht ändern. Die Erstellung von Mocktests für diese Szenarien kann die Iterationszeiten der Entwickler verbessern.

Code, der durch Komponententests mit Mocks-Tests abgedeckt wird, muss auch durch Tests in der Cloud abgedeckt werden. Dies ist notwendig, da die Mocks immer noch auf dem Laptop oder der Build-Maschine eines Entwicklers laufen und die Umgebung möglicherweise anders konfiguriert ist als in der Cloud. Ihr Code könnte beispielsweise AWS Lambda Funktionen enthalten, die mehr Speicher verbrauchen oder mehr Zeit in Anspruch nehmen, als Lambda für die Zuweisung konfiguriert ist, wenn es mit bestimmten Eingabeparametern ausgeführt wird. Oder Ihr Code könnte Umgebungsvariablen enthalten, die nicht auf die gleiche Weise (oder überhaupt nicht) konfiguriert sind, und die Unterschiede können dazu führen, dass sich der Code anders verhält oder fehlschlägt.

Verwenden Sie keine Modelle von Cloud-Diensten, um die korrekte Implementierung dieser Dienstintegrationen zu überprüfen. Es mag zwar akzeptabel sein, sich über einen Cloud-Dienst lustig zu machen, wenn Sie andere Funktionen testen, aber Sie sollten Cloud-Dienstaufrufe in der Cloud testen, um die korrekte Konfiguration und funktionale Implementierung zu überprüfen.

Mocks können bei Komponententests einen Mehrwert bieten, vor allem, wenn Sie häufig eine große Anzahl von Fällen testen. Dieser Vorteil ist bei Integrationstests geringer, da der Aufwand für die Implementierung der erforderlichen Mocks mit der Anzahl der Verbindungspunkte zunimmt. End-to-endBeim Testen sollten keine Mocks verwendet werden, da sich diese Tests im Allgemeinen mit Zuständen und komplexer Logik befassen, die mit Schein-Frameworks nicht einfach simuliert werden können.

Machen Sie sich mit den Kompromissen von Emulationstests vertraut

Emulatoren können für bestimmte Anwendungsfälle eine praktische Wahl sein. Beispielsweise könnte ein Entwicklungsteam mit eingeschränktem, inkonsistentem oder langsamem Internetzugang feststellen, dass Emulationstests die zuverlässigste Methode sind, Code zu iterieren, bevor er in eine Cloud-Umgebung wechselt.

In den meisten anderen Fällen sollten Sie Emulatoren selektiv verwenden. Wenn Sie sich stark auf einen Emulator verlassen, kann es schwierig werden, neue AWS Servicefunktionen in Ihre Tests zu integrieren, bis der Emulationsanbieter ein Update veröffentlicht, das die gleiche Funktionalität gewährleistet. Emulatoren erfordern außerdem im Voraus laufende Investitionen in die Einrichtung und Konfiguration aller Entwicklungssysteme und Build-Maschinen. Darüber hinaus sind für viele Cloud-Dienste keine Emulatoren verfügbar. Wenn Sie eine Strategie wählen, bei der die Emulation an erster Stelle steht, kann dies entweder die Nutzung dieser Dienste ausschließen oder Code und Konfigurationen erzeugen, die nicht ausreichend anhand des tatsächlichen Dienstverhaltens getestet wurden.

Wenn Sie Emulationstests verwenden, sollten Sie diese so weit wie möglich durch Cloud-Tests ergänzen, um zu überprüfen, ob die richtigen Cloud-Konfigurationen vorhanden sind, und um Interaktionen mit Diensten zu testen, die nur in einer emulierten Umgebung simuliert oder simuliert werden können.

Emulationstests können schnelles Feedback für Komponententests liefern und je nach den Funktionen und der Verhaltensparität der Emulationssoftware auch einige Integrationen und Tests unterstützen. end-to-end

Durchführen von Tests anhand natürlicher Grenzen

Da serverlose Anwendungen immer mehr architektonische Komponenten umfassen, entstehen natürliche Grenzen rund um Subsysteme — vor allem, wenn bewährte Verfahren wie Funktionen für einzelne Zwecke und ereignisgesteuerte Entkopplung befolgt werden. Diese Grenzen dienen als effektive Testgrenzen, an denen Sie Verträge zwischen Komponenten validieren können.

Identifizieren Sie die Grenzen der Architektur

Achten Sie in Ihrem Anwendungsdesign auf natürliche Nähte:

- Zwischen Diensten, z. B. einer EventBridge Amazon-Regel, die einen Verlag mit einem Verbraucher verbindet
- An API-Edges, wie z. B. Amazon API Gateway Gateway-Endpunkten, die Lambda-Funktionen unterstützen
- Rund um Workflows, wie z. B. die AWS Step Functions Orchestrierung mehrerer Dienste
- Auf Speicherebenen wie Amazon DynamoDB DynamoDB-Streams, die die Downstream-Verarbeitung auslösen

Lambda-Code von Geschäftslogik trennen

Vereinfachen Sie Ihre Tests, indem Sie Lambda-Code von der Kerngeschäftslogik isolieren. Ihr Lambda-Handler sollte als dünner Adapter zwischen der AWS Laufzeit und Ihrer Anwendungslogik fungieren. Es sollte Ereignisdaten extrahieren und validieren und dann an eine testbare Funktion delegieren, die keine Lambda-Abhängigkeiten hat. Dadurch ist Ihre Geschäftslogik portabel, leichter nachvollziehbar und einfach zu testen, ohne Lambda-Objekte zu verspotten oder komplexe Umgebungen einzurichten.

Behandeln Sie Grenzen wie Verträge

Testen Sie an der Grenze, nicht durch sie hindurch. Prüfen Sie, was die Grenze überschreitet, ohne dass das gesamte nachgeschaltete System erforderlich ist. Dieselben Grenzen dienen auch als Haken für die Beobachtbarkeit in der Produktion. Die architektonischen Nähte, an denen Sie testen, können mithilfe von Amazon CloudWatch Logs, AWS X-Ray Traces und EventBridge Events für die Überwachung instrumentiert werden.

Verwenden Sie Testlösungen für asynchrone Workflows

Serverlose Anwendungen basieren häufig auf asynchronen Mustern, bei denen Ereignisse die Verarbeitung auslösen, Nachrichten durch Warteschlangen fließen und Workflows sich über mehrere Dienste erstrecken, ohne dass sofort reagiert wird. Sie können nicht einfach eine Funktion aufrufen und einen Rückgabewert überprüfen. Das Ergebnis kann später in einer Datenbank, einem Protokollstream oder einem anderen Dienst erscheinen.

Ein Test-Harness ist eine Testinfrastruktur, die Sie zusammen mit Ihrer Anwendung bereitstellen, um dieses asynchrone Verhalten zu beobachten und zu validieren. Zu den Testkabelbäumen gehören in der Regel:

- Ereignis-Listener, die dieselben Ereignisse abonnieren, die Ihre Anwendung erzeugt
- Speichermechanismen (wie DynamoDB-Tabellen oder Amazon S3 S3-Buckets), mit denen Testergebnisse erfasst werden können
- Abfragelogik in Ihrem Testcode, die darauf wartet, dass erwartete Ergebnisse angezeigt werden

Ihr Testcode löst ein Ereignis aus, wartet, bis der Workflow abgeschlossen ist, und fragt dann den Testcode ab, um zu überprüfen, ob das erwartete Ergebnis eingetreten ist.

Im Folgenden finden Sie bewährte Methoden:

- Definieren Sie „Clear“ SLAs für asynchrone Operationen — Legen Sie fest, wie lange Workflows dauern sollen, und verwenden Sie diese als Timeouts für Abfragen in Ihren Tests
- Verwenden Sie eindeutige Identifikatoren für die Testisolierung — Generieren Sie eindeutige Dateinamen, Nachrichten- oder Korrelationstoken pro IDs Testlauf, um Interferenzen zwischen Tests zu vermeiden
- Stellen Sie die Testinfrastruktur zusammen mit Ihrer Anwendung bereit — Nehmen Sie Test-Harness-Ressourcen in Ihre infrastructure-as-code Vorlagen auf, damit sie bei der Weiterentwicklung Ihrer Anwendung synchron bleiben
- Bereinigen Sie die Testdaten nach Testläufen — Dadurch wird verhindert, dass sich Testartefakte in Ihrer Cloud-Umgebung ansammeln

Testkabelbäume sind am wertvollsten für Integrationstests, bei denen Workflows über mehrere Services hinweg validiert werden, end-to-end Tests, die komplette Benutzererfahrung verifizieren, und ereignisgesteuerte Architekturen, bei denen Dienste über Amazon SNS EventBridge, Amazon SQS oder Amazon Kinesis kommunizieren.

Organisieren Sie Cloud-Umgebungen zur Isolierung von Entwicklern

Das Testen in der Cloud erfordert Umgebungen, die voneinander isoliert sind. Wenn sich Entwickler ein einzelnes Konto teilen AWS-Konto, z. B. ein Teamentwicklungskonto, sollten Sie erwägen, für jeden Entwickler oder jeden Feature-Branch einen separaten Anwendungsstapel zu erstellen. Dadurch werden Ressourcen isoliert, Namenskonflikte vermieden und Quotenkonflikte oder Probleme mit störenden Nachbarn beim Testen vermieden.

Verwenden Sie AWS Systems Manager Parameter Store oder ähnliche Tools, um stapelspezifische Konfigurationen wie API-Endpunkte und Warteschlangennamen zu verwalten. Aus Kostengründen sollten Sie teure Ressourcen wie Amazon Relational Database Service (Amazon RDS) -Cluster auf mehrere Entwickler-Stacks verteilen und gleichzeitig leichtgewichtige serverlose Ressourcen (wie Lambda-Funktionen, API Gateway Gateway-Stufen und DynamoDB-Tabellen) pro Stack isoliert halten.

In regulierten Branchen können Sicherheitsrichtlinien für Unternehmen den Zugriff von Entwicklern auf Cloud-Umgebungen einschränken, was es schwierig macht, Cloud-Tests als Teil eines lokalen

Entwicklungsworkflows durchzuführen. In diesen Fällen können Emulationstests die Lücke zwischen lokalen Scheintests und vollständiger Cloud-Validierung schließen. Sie sollten jedoch durch Cloud-Tests ergänzt werden, sofern der Zugriff dies zulässt.

Feedback-Schleifen beschleunigen

Wenn Sie in der Cloud testen, verwenden Sie Tools und Techniken, um die Feedback-Schleifen zur Entwicklung zu beschleunigen. Verwenden Sie beispielsweise AWS SAM den Beschleunigungs - und AWS CDK Überwachungsmodus, um die Zeit zu verkürzen, die benötigt wird, um Codeänderungen in eine Cloud-Umgebung zu übertragen. In den Beispielen im Repository GitHub Serverless Test Samples werden einige dieser Techniken untersucht.

Wir empfehlen Ihnen außerdem, Cloud-Ressourcen so früh wie möglich während der Entwicklung auf Ihrem lokalen Computer zu erstellen und zu testen — nicht erst, nachdem Sie sich bei der Quellcodeverwaltung angemeldet haben. Diese Praxis ermöglicht ein schnelleres Erkunden und Experimentieren bei der Entwicklung von Lösungen. Die Möglichkeit, die Bereitstellung von einem Entwicklungscomputer aus zu automatisieren, hilft Ihnen zudem, Probleme mit der Cloud-Konfiguration schneller zu erkennen und den Aufwand für die Aktualisierung und Genehmigung von Änderungen in der Versionsverwaltung zu reduzieren.

Häufig gestellte Fragen

Ich habe eine Lambda-Funktion, die Berechnungen durchführt und ein Ergebnis zurückgibt, ohne andere Dienste aufzurufen. Muss ich das wirklich in der Cloud testen?

Ja. AWS Lambda Funktionen haben Konfigurationsparameter, die das Ergebnis des Tests ändern könnten. Der gesamte Lambda-Funktionscode ist von Timeout- und Speichereinstellungen abhängig, was dazu führen kann, dass die Funktion fehlschlägt, wenn sie nicht richtig eingestellt sind. Lambda-Richtlinien ermöglichen auch die Standardausgabeprotokollierung an Amazon CloudWatch. Auch wenn Ihr Code nicht CloudWatch direkt aufruft, ist eine Genehmigung erforderlich, um die Protokollierung zu aktivieren, und diese Erlaubnis kann nicht korrekt verspottet oder emuliert werden.

Wie können Tests in der Cloud beim Einheitentest helfen? Wenn es sich in der Cloud befindet und eine Verbindung zu anderen Ressourcen herstellt, ist das nicht ein Integrationstest?

Wir definieren Unit-Tests als Tests, die isoliert an Architekturkomponenten arbeiten. Diese Definition schließt die Verwendung von Serviceanrufen oder anderer Netzwerkkommunikation nicht unbedingt aus.

Viele Serverless-Anwendungen verfügen über Architekturkomponenten, die isoliert getestet werden können, sogar in der Cloud. Ein einfaches Beispiel ist eine Lambda-Funktion, die Eingaben entgegennimmt, interpretiert und eine Nachricht an eine Amazon Simple Queue Service (Amazon SQS) -Warteschlange sendet. Ein Komponententest einer solchen Funktion würde wahrscheinlich testen, ob Eingabewerte dazu führen, dass bestimmte Werte in der Nachricht in der Warteschlange vorhanden sind. Stellen Sie sich einen Test vor, der nach dem Muster arrange, act, assert geschrieben wurde:

- Anordnen — Weist Ressourcen zu (eine Warteschlange für den Empfang von Nachrichten und die zu testende Funktion).
- Aktion — Ruft die zu testende Funktion auf.
- Assert — Ruft die von der Funktion gesendete Nachricht ab und validiert die Ausgabe.

Ein Mock-Testansatz würde das Mocking der Warteschlange mit einem prozessinternen Mock-Objekt und das Erstellen einer prozessinternen Instance der Klasse oder des Moduls, das den Lambda-Funktionscode enthält, beinhalten. Während der Assert-Phase würde die Nachricht in der Warteschlange aus dem simulierten Objekt abgerufen werden.

Bei einem cloud-basierten Ansatz würde der Test eine Amazon-SQS-Warteschlange für die Testzwecke erstellen und die Lambda-Funktion mit Umgebungsvariablen bereitstellen, die so konfiguriert sind, dass sie die isolierte Amazon-SQS-Warteschlange als Ausgabeziel verwenden. Nach dem Ausführen der Lambda-Funktion würde der Test die Nachricht aus der Amazon-SQS-Warteschlange abrufen.

Der cloudbasierte Test würde denselben Code ausführen, dasselbe Verhalten bestätigen und die funktionale Korrektheit der Anwendung überprüfen. Es hätte jedoch den zusätzlichen Vorteil, dass die folgenden Einstellungen der Lambda-Funktion validiert werden könnten: die AWS Identity and Access Management (IAM-) Rolle, die IAM-Richtlinien und die Timeout- und Speichereinstellungen der Funktion.

Nächste Schritte und Ressourcen

Verwenden Sie die folgenden Ressourcen für weiterführende Informationen und weitere konkrete Beispiele.

Beispielimplementierungen

Das Serverless Test Samples Repository on GitHub enthält konkrete Beispiele für Tests, die den in diesem Handbuch beschriebenen Mustern und bewährten Methoden folgen. Das Repository enthält Beispielcode und Anleitungen zu den Mock-, Emulations- und Cloud-Testprozessen, die in den vorherigen Abschnitten beschrieben wurden. Verwenden Sie dieses Repository, um sich über die neuesten Anleitungen zu serverlosen Tests von auf dem Laufenden zu halten. AWS

Weitere Informationen

Besuchen Sie Serverless Land, um auf die neuesten Blogs, Videos und Schulungen für AWS serverlose Technologien zuzugreifen.

Referenzen

- Beschleunigen Sie die serverlose Entwicklung mit AWS SAM Accelerate (Blogbeitrag)AWS
- Erhöhung der Entwicklungsgeschwindigkeit mit CDK Watch (Blogbeitrag)AWS
- Serviceintegrationen mit AWS Step Functions Local verspotten (Blogbeitrag)AWS
- Erste Schritte beim Testen serverloser Anwendungen (Blogbeitrag)AWS
- Beschleunigen Sie serverlose Tests mit der LocalStack Integration in VS Code IDE (AWS Blogbeitrag)
- Lokales Debuggen von Funktionen mit AWS SAM(Dokumentation)AWS

Tools

- AWS SAM — Testen und Debuggen serverloser Anwendungen
- AWS SAM — Integration mit automatisierten Tests
- AWS Lambda — Testen von Lambda-Funktionen in der Konsole

- LocalStack Cloud-Emulator — Verbessern Sie das lokale Testerlebnis für serverlose Anwendungen mit LocalStack

Mitwirkende

Inhaltserstellung

- Dan Fox, AWS
- Leslie Raj, AWS
- Rohan Mehta, AWS
- Rob Hill, AWS

Überprüfend

- Brian Krygsman, AWS

Technisches Schreiben

- Lilly, AbouHarb AWS

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
Aktualisierungen	Wir haben Leitlinien für Tests an Architektur- und Codegrenzen, Testlösungen für asynchrone Workflows und zur Isolierung von Entwickle rumgebungen hinzugefügt. Wir haben auch die Empfehlungen für Emulationstests aktualisi ert.	18. März 2026
Erste Veröffentlichung	—	9. Dezember 2022

AWS Glossar zu präskriptiven Leitlinien

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

Zahlen

7 Rs

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

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

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

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

A

ABAC

Siehe attributbasierte Zugriffskontrolle.

abstrahierte Dienste

Siehe Managed Services.

ACID

Siehe Atomarität, Konsistenz, Isolierung und Haltbarkeit.

Aktiv-Aktiv-Migration

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

Aktiv-Passiv-Migration

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

Aggregatfunktion

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

AI

Siehe künstliche Intelligenz.

AIOps

Siehe Operationen im Bereich künstliche Intelligenz.

Anonymisierung

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

Anti-Muster

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

Anwendungssteuerung

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

Anwendungsportfolio

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

künstliche Intelligenz (KI)

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

Operationen mit künstlicher Intelligenz (AIOps)

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

Asymmetrische Verschlüsselung

Ein Verschlüsselungsalgorithmus, der ein Schlüsselpaar, einen öffentlichen Schlüssel für die Verschlüsselung und einen privaten Schlüssel für die Entschlüsselung verwendet. Sie können den

öffentlichen Schlüssel teilen, da er nicht für die Entschlüsselung verwendet wird. Der Zugriff auf den privaten Schlüssel sollte jedoch stark eingeschränkt sein.

Atomizität, Konsistenz, Isolierung, Haltbarkeit (ACID)

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

Attributbasierte Zugriffskontrolle (ABAC)

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

maßgebliche Datenquelle

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

Availability Zone

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

AWS Framework für die Einführung der Cloud (AWS CAF)

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

AWS Workload-Qualifizierungsrahmen (AWS WQF)

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

B

schlechter Bot

Ein Bot, der Einzelpersonen oder Organisationen stören oder ihnen Schaden zufügen soll.

BCP

Siehe Planung der Geschäftskontinuität.

Verhaltensdiagramm

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

Big-Endian-System

Ein System, welches das höchstwertige Byte zuerst speichert. Siehe auch Endianness.

Binäre Klassifikation

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

Bloom-Filter

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

Blau/Grün-Bereitstellung

Eine Bereitstellungsstrategie, bei der Sie zwei separate, aber identische Umgebungen erstellen. Sie führen die aktuelle Anwendungsversion in einer Umgebung (blau) und die neue

Anwendungsversion in der anderen Umgebung (grün) aus. Mit dieser Strategie können Sie schnell und mit minimalen Auswirkungen ein Rollback durchführen.

Bot

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

Botnetz

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

branch

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

Zugang durch Glasbruch

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

Brownfield-Strategie

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

Puffer-Cache

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

Geschäftsfähigkeit

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

Planung der Geschäftskontinuität (BCP)

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

C

CAF

Weitere Informationen finden Sie unter Framework AWS für die Cloud-Einführung.

Bereitstellung auf Kanaren

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

CCoE

Weitere Informationen finden Sie im Cloud Center of Excellence.

CDC

Siehe Erfassung von Änderungsdaten.

Erfassung von Datenänderungen (CDC)

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

Chaos-Technik

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

CI/CD

Siehe Continuous Integration und Continuous Delivery.

Klassifizierung

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

clientseitige Verschlüsselung

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

Cloud-Exzellenzzentrum (CCoE)

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

Cloud Computing

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

Cloud-Betriebsmodell

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

Phasen der Einführung der Cloud

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

- Projekt – Durchführung einiger Cloud-bezogener Projekte zu Machbarkeitsnachweisen und zu Lernzwecken
- Fundament — Tätigen Sie grundlegende Investitionen, um Ihre Cloud-Einführung zu skalieren (z. B. Einrichtung einer landing zone, Definition eines CCo E, Einrichtung eines Betriebsmodells)

- Migration – Migrieren einzelner Anwendungen
- Neuentwicklung – Optimierung von Produkten und Services und Innovation in der Cloud

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

CMDB

Siehe Datenbank für das Konfigurationsmanagement.

Code-Repository

Ein Ort, an dem Quellcode und andere Komponenten wie Dokumentation, Beispiele und Skripts gespeichert und im Rahmen von Versionskontrollprozessen aktualisiert werden. Zu den gängigen Cloud-Repositorys gehören GitHub 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 wird allgemein als Pipeline beschrieben. CI/CD kann Ihnen helfen, Prozesse zu automatisieren, die Produktivität zu steigern, die Codequalität zu verbessern und schneller zu liefern. Weitere Informationen finden Sie unter Vorteile der kontinuierlichen Auslieferung. CD kann auch für kontinuierliche Bereitstellung stehen. Weitere Informationen finden Sie 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 Abweichung zwischen den Produktionsdaten und den Daten, die zum Trainieren eines ML-Modells verwendet wurden, oder eine signifikante Änderung der Eingabedaten im Laufe der Zeit. Datendrift kann die Gesamtqualität, Genauigkeit und Fairness von ML-Modellvorhersagen beeinträchtigen.

Daten während der Übertragung

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

Datennetz

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

Datenminimierung

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

Datenperimeter

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

Vorverarbeitung der Daten

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

Herkunft der Daten

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

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.

Einsatz

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

Entwicklungsumgebung

Siehe Umgebung.

Detektivische Kontrolle

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

Abbildung des Wertstroms in der Entwicklung (DVSM)

Ein Prozess zur Identifizierung und Priorisierung von Einschränkungen, die sich negativ auf Geschwindigkeit und Qualität im Lebenszyklus der Softwareentwicklung auswirken. DVSM erweitert den Prozess der Wertstromanalyse, der ursprünglich für Lean-Manufacturing-Praktiken konzipiert wurde. Es konzentriert sich auf die Schritte und Teams, die erforderlich sind, um durch den Softwareentwicklungsprozess Mehrwert zu schaffen und zu steigern.

digitaler Zwilling

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

Maßtabelle

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

Katastrophe

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

Notfallwiederherstellung (DR)

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

DML

Siehe Sprache zur Datenbankmanipulation.

Domainorientiertes Design

Ein Ansatz zur Entwicklung eines komplexen Softwaresystems, bei dem seine Komponenten mit sich entwickelnden Domains oder 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 Sicherheitsepen AWS von CAF gehören beispielsweise Identitäts- und Zugriffsmanagement, Detektivkontrollen, Infrastruktursicherheit, Datenschutz und

Reaktion auf Vorfälle. Weitere Informationen zu Epics in der AWS -Migrationsstrategie finden Sie im Leitfaden zur Programm-Implementierung.

ERP

Siehe Enterprise Resource Planning.

Explorative Datenanalyse (EDA)

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

F

Faktentabelle

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

schnell scheitern

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

Grenze zur Fehlerisolierung

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

Feature-Zweig

Siehe Zweig.

Features

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

Bedeutung der Feature

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

Featuretransformation

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

Eingabeaufforderung mit wenigen Klicks

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

FGAC

Siehe detaillierte Zugriffskontrolle.

Feinkörnige Zugriffskontrolle (FGAC)

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

Flash-Cut-Migration

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

FM

Siehe Fundamentmodell.

Fundamentmodell (FM)

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

G

Generative KI

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

Geoblocking

Siehe geografische Einschränkungen.

Geografische Einschränkungen (Geoblocking)

Bei Amazon eine Option CloudFront, um zu verhindern, dass Benutzer in bestimmten Ländern auf Inhaltsverteilungen zugreifen. Sie können eine Zulassungsliste oder eine Sperrliste verwenden, um zugelassene und gesperrte Länder anzugeben. Weitere Informationen finden Sie in der Dokumentation unter Beschränkung der geografischen Verteilung Ihrer Inhalte. CloudFront

Gitflow-Workflow

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

goldenes Bild

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

Greenfield-Strategie

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

Integritätsschutz

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

H

HEKTAR

Siehe Hochverfügbarkeit.

Heterogene Datenbankmigration

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

hohe Verfügbarkeit (HA)

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

historische Modernisierung

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

Daten zurückhalten

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

Homogene Datenbankmigration

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

heiße Daten

Daten, auf die häufig zugegriffen wird, z. B. Echtzeitdaten oder aktuelle 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 Infrastruktur als Code.

Identitätsbasierte Richtlinie

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

Leerlaufanwendung

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

IIoT

Siehe Industrielles Internet der Dinge.

unveränderliche Infrastruktur

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

Eingehende (ingress) VPC

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

Inkrementelle Migration

Eine Cutover-Strategie, bei der Sie Ihre Anwendung in kleinen Teilen migrieren, anstatt eine einziges vollständiges Cutover durchzuführen. Beispielsweise könnten Sie zunächst nur einige Microservices oder Benutzer auf das neue System umstellen. Nachdem Sie sich vergewissert haben, dass alles ordnungsgemäß funktioniert, können Sie weitere Microservices oder Benutzer

schrittweise verschieben, bis Sie Ihr Legacy-System außer Betrieb nehmen können. Diese Strategie reduziert die mit großen Migrationen verbundenen Risiken.

Industrie 4.0

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

Infrastruktur

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

Infrastructure as Code (IaC)

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

industrielles Internet der Dinge (T) 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

Siehe 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

Siehe Labelbasierte Zugriffskontrolle.

Geringste Berechtigung

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

Lift and Shift

Siehe 7 Rs.

Little-Endian-System

Ein System, welches das niedrigwertigste Byte zuerst speichert. Siehe auch Endianness.

LLM

Siehe großes Sprachmodell.

Niedrigere Umgebungen

Siehe Umgebung.

M

Machine Learning (ML)

Eine Art künstlicher Intelligenz, die Algorithmen und Techniken zur Mustererkennung und zum Lernen verwendet. ML analysiert aufgezeichnete Daten, wie z. B. Daten aus dem Internet der

Dinge (IoT), und lernt daraus, um ein statistisches Modell auf der Grundlage von Mustern zu erstellen. Weitere Informationen finden Sie unter Machine Learning.

Hauptzweig

Siehe Filiale.

Malware

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

verwaltete Dienste

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

Manufacturing Execution System (MES)

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

MAP

Siehe Migration Acceleration Program.

Mechanismus

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

Mitgliedskonto

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

MES

Siehe Manufacturing Execution System.

Message Queuing-Telemetrietransport (MQTT)

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

Microservice

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

Microservices-Architekturen

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

Migration Acceleration Program (MAP)

Ein AWS Programm, das Beratung, Unterstützung, Schulungen und Services bietet, um Unternehmen dabei zu unterstützen, eine solide betriebliche Grundlage für die Umstellung auf die Cloud zu schaffen und die anfänglichen Kosten von Migrationen auszugleichen. MAP umfasst eine Migrationsmethode für die methodische Durchführung von Legacy-Migrationen sowie eine Reihe von Tools zur Automatisierung und Beschleunigung gängiger Migrationsszenarien.

Migration in großem Maßstab

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

Migrationsfabrik

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

Migrationsmetadaten

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

Migrationsmuster

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

Migration Portfolio Assessment (MPA)

Ein Online-Tool, das Informationen zur Validierung des Geschäftsszenarios für die Migration auf das bereitstellt. AWS Cloud MPA bietet eine detaillierte Portfoliobewertung (richtige Servergröße, Preisgestaltung, Gesamtbetriebskostenanalyse, Migrationskostenanalyse) sowie Migrationsplanung (Anwendungsdatenanalyse und Datenerfassung, Anwendungsgruppierung, Migrationspriorisierung und Wellenplanung). Das MPA-Tool (Anmeldung erforderlich) steht allen AWS Beratern und APN-Partnerberatern kostenlos zur Verfügung.

Migration Readiness Assessment (MRA)

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

Migrationsstrategie

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

ML

Siehe maschinelles Lernen.

Modernisierung

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

Bewertung der Modernisierungsfähigkeit

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

Monolithische Anwendungen (Monolithen)

Anwendungen, die als ein einziger Service mit eng gekoppelten Prozessen ausgeführt werden. Monolithische Anwendungen haben verschiedene Nachteile. Wenn ein Anwendungs-Feature stark nachgefragt wird, muss die gesamte Architektur skaliert werden. Das Hinzufügen oder Verbessern der Feature einer monolithischen Anwendung wird ebenfalls komplexer, wenn die Codebasis wächst. Um diese Probleme zu beheben, können Sie eine Microservices-Architektur verwenden. Weitere Informationen finden Sie unter Zerlegen von Monolithen in Microservices.

MPA

Siehe Bewertung des Migrationsportfolios.

MQTT

Siehe Message Queuing-Telemetrietransport.

Mehrklassen-Klassifizierung

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

veränderbare Infrastruktur

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

O

OAC

Siehe Origin Access Control.

EICHE

Siehe Zugriffsidentität von Origin.

COM

Siehe organisatorisches Change-Management.

Offline-Migration

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

OI

Siehe Betriebsintegration.

OLA

Siehe Vereinbarung auf operativer Ebene.

Online-Migration

Eine Migrationsmethode, bei der der Quell-Workload auf das Zielsystem kopiert wird, ohne offline genommen zu werden. Anwendungen, die mit dem Workload verbunden sind, können während

der Migration weiterhin funktionieren. Diese Methode beinhaltet keine bis minimale Ausfallzeit und wird in der Regel für kritische Produktionsworkloads verwendet.

OPC-UA

Siehe Open Process Communications — Unified Architecture.

Offene Prozesskommunikation — Einheitliche Architektur (OPC-UA)

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

Vereinbarung auf Betriebsebene (OLA)

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

Überprüfung der Betriebsbereitschaft (ORR)

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

Betriebstechnologie (OT)

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

Betriebsintegration (OI)

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

Organisationspfad

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

Organisatorisches Veränderungsmanagement (OCM)

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

Ursprungszugriffskontrolle (OAC)

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

Ursprungszugriffsidentität (OAI)

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

ORR

Weitere Informationen finden Sie unter Überprüfung der Betriebsbereitschaft.

NICHT

Siehe Betriebstechnologie.

Ausgehende (egress) VPC

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

P

Berechtigungsgrenze

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

persönlich identifizierbare Informationen (PII)

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

Personenbezogene Daten

Siehe persönlich identifizierbare Informationen.

Playbook

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

PLC

Siehe programmierbare Logiksteuerung.

PLM

Siehe Produktlebenszyklusmanagement.

policy

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

Polyglotte Beharrlichkeit

Unabhängige Auswahl der Datenspeichertechnologie eines Microservices auf der Grundlage von Datenzugriffsmustern und anderen Anforderungen. Wenn Ihre Microservices über dieselbe Datenspeichertechnologie verfügen, kann dies zu Implementierungsproblemen oder zu

Leistungseinbußen führen. Microservices lassen sich leichter implementieren und erzielen eine bessere Leistung und Skalierbarkeit, wenn sie den Datenspeicher verwenden, der ihren Anforderungen am besten entspricht.

Portfoliobewertung

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

predicate

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

Prädikat Pushdown

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

Präventive Kontrolle

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

Prinzipal

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

Datenschutz von Natur aus

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

Privat gehostete Zonen

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

proaktive Steuerung

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

Produktlebenszyklusmanagement (PLM)

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

Produktionsumgebung

Siehe Umgebung.

Speicherprogrammierbare Steuerung (SPS)

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

schnelle Verkettung

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

Pseudonymisierung

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

publish/subscribe (pub/sub)

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

Q

Abfrageplan

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

Abfrageplanregression

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

R

RACI-Matrix

Siehe verantwortlich, rechenschaftspflichtig, konsultiert, informiert (RACI).

RAG

Siehe Erweiterte Generierung beim Abrufen.

Ransomware

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

RASCI-Matrix

Siehe verantwortlich, rechenschaftspflichtig, konsultiert, informiert (RACI).

RCAC

Siehe Zugriffskontrolle für Zeilen und Spalten.

Read Replica

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

neu strukturieren

Siehe 7 Rs.

Recovery Point Objective (RPO)

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

Wiederherstellungszeitziel (RTO)

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

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 für die Erholungszeit.

Runbook

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

S

SAML 2.0

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

SCADA

Siehe Aufsichtskontrolle und Datenerfassung.

SCP

Siehe Richtlinie zur Dienstkontrolle.

Secret

Interne AWS Secrets Manager, vertrauliche oder eingeschränkte Informationen, wie z. B. ein Passwort oder 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 Amazon EC2 EC2-Instance oder das Rotieren von Anmeldeinformationen.

Serverseitige Verschlüsselung

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

Service-Kontrollrichtlinie (SCP)

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

Service-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.

Eingabeaufforderung ohne Zwischenfälle

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

Zombie-Anwendung

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

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