Zerlegung der Datenbank auf AWS

AWS Präskriptive Leitlinien

AWS Präskriptive Leitlinien: Zerlegung der Datenbank auf AWS

Table of Contents

Datenbankzerlegung auf AWS

Philippe Wanner und Saurabh Sharma, Amazon Web Services

Oktober 2025 (Geschichte der Dokumente)

Die Modernisierung von Datenbanken, insbesondere die Zerlegung monolithischer Datenbanken, ist eine wichtige Aufgabe für Unternehmen, die die Agilität, Skalierbarkeit und Leistung ihrer Datenverwaltungssysteme verbessern möchten. Da Unternehmen wachsen und ihre Datenanforderungen immer komplexer werden, haben herkömmliche monolithische Datenbanken oft Schwierigkeiten, Schritt zu halten. Dies führt zu Leistungsengpässen, Wartungsherausforderungen und Schwierigkeiten bei der Anpassung an sich ändernde Geschäftsanforderungen.

Im Folgenden sind die häufigsten Herausforderungen bei monolithischen Datenbanken aufgeführt:

- Fehlausrichtung von Geschäftsbereichen — Monolithische Datenbanken schaffen es oft nicht, die Technologie auf unterschiedliche Geschäftsbereiche abzustimmen, was das Unternehmenswachstum einschränken kann.
- Einschränkungen bei der Skalierbarkeit — Systeme stoßen häufig an Skalierungsgrenzen, was zu Hindernissen für die Geschäftsexpansion führt.
- Architektonische Steifigkeit — Eng miteinander verbundene Strukturen erschweren die Aktualisierung bestimmter Komponenten, ohne das gesamte System zu beeinträchtigen.
- Leistungseinbußen — Zunehmende Datenlasten und zunehmende Parallelität der Benutzer führen häufig zu einer Verschlechterung der Systemleistung.

Im Folgenden sind die Vorteile der Datenbankzerlegung aufgeführt:

- Verbesserte geschäftliche Flexibilität — Die Dekomposition ermöglicht eine schnelle Anpassung an sich ändernde Geschäftsanforderungen und unterstützt eine unabhängige Skalierung.
- Optimierte Leistung — Decomposition hilft Ihnen bei der Erstellung spezialisierter Datenbanklösungen, die auf bestimmte Anwendungsfälle zugeschnitten sind und jede Datenbank unabhängig skalieren können.
- Verbessertes Kostenmanagement — Die Zerlegung ermöglicht eine effizientere Ressourcennutzung und senkt die Betriebskosten.
- Flexible Lizenzoptionen — Decomposition bietet Möglichkeiten für den Übergang von kostspieligen proprietären Lizenzen zu Open-Source-Alternativen.

- Innovationsförderung — Decomposition erleichtert die Einführung von speziell für bestimmte Workloads entwickelten Datenbanken.

Zielgruppe

Dieser Leitfaden hilft Datenbankarchitekten, Cloud-Lösungsarchitekten, Anwendungsentwicklungsteams und Unternehmensarchitekten. Es soll Ihnen helfen, monolithische Datenbanken in auf Microservices ausgerichtete Datenspeicher zu zerlegen, domänengesteuerte Datenbankarchitekturen zu implementieren, Datenbankmigrationsstrategien zu planen und Datenbankoperationen zu skalieren, um den wachsenden Geschäftsanforderungen gerecht zu werden. Um die Konzepte und Empfehlungen in diesem Handbuch zu verstehen, sollten Sie mit relationalen und NoSQL-Datenbankprinzipien, AWS verwalteten Datenbankdiensten und Mikroservices-Architekturmustern vertraut sein. Dieser Leitfaden soll Organisationen helfen, die sich in der Anfangsphase eines Projekts zur Zerlegung von Datenbanken befinden.

Ziele

Dieser Leitfaden kann Ihrer Organisation helfen, die folgenden Ziele zu erreichen:

- Erfassen Sie die Anforderungen für die Zerlegung Ihrer Zielarchitektur.
- Entwickeln Sie eine systematische Methode zur Risikobewertung und Kommunikation.
- Erstellen Sie einen Zerlegungsplan.
- Definieren Sie Erfolgskennzahlen, wichtige Leistungsindikatoren (KPIs), eine Strategie zur Risikominderung und einen Plan zur Geschäftskontinuität.
- Sorgen Sie für eine bessere Workload-Elastizität, die Ihnen hilft, der Geschäftsnachfrage gerecht zu werden.
- Erfahren Sie, wie Sie spezielle Datenbanken für bestimmte Anwendungsfälle einsetzen können, um Innovationen zu ermöglichen.
- Stärken Sie die Datensicherheit und Governance Ihres Unternehmens.
- Reduzieren Sie die Kosten durch folgende Maßnahmen:
  - Reduzierte Lizenzgebühren
  - Geringere Bindung an einen Anbieter
  - Verbesserter Zugang zu umfassenderer Unterstützung und Innovationen durch die Community
  - Fähigkeit, verschiedene Datenbanktechnologien für verschiedene Komponenten auszuwählen

- Schrittweise Migration, wodurch das Risiko reduziert und die Kosten im Laufe der Zeit verteilt werden
- Verbesserte Ressourcennutzung

Allgemeine Herausforderungen und Verwaltungsaufgaben bei der Datenbankzerlegung

Die Zerlegung von Datenbanken ist ein komplexer Prozess, der eine sorgfältige Planung, Ausführung und Verwaltung erfordert. Wenn Unternehmen versuchen, ihre Dateninfrastruktur zu modernisieren, stoßen sie häufig auf eine Vielzahl von Herausforderungen, die sich auf den Erfolg ihrer Projekte auswirken können. In diesem Abschnitt werden die häufigsten Hürden beschrieben und ein strukturierter Ansatz zur Überwindung dieser Hindernisse vorgestellt.

Übliche Herausforderungen

Ein Projekt zur Zerlegung von Datenbanken steht vor verschiedenen Herausforderungen in technischer, personeller und geschäftlicher Hinsicht. In technischer Hinsicht stellt die Sicherstellung der Datenkonsistenz zwischen verteilten Systemen eine erhebliche Hürde dar. Dies kann während der Übergangsphase auch potenzielle Auswirkungen auf Leistung und Stabilität haben, und Sie müssen sich nahtlos in bestehende Systeme integrieren. Zu den personellen Herausforderungen gehören die mit dem neuen System verbundene Lernkurve, der potenzielle Widerstand der Mitarbeiter gegen Veränderungen und die Verfügbarkeit der erforderlichen Ressourcen.
Aus geschäftlicher Sicht muss sich das Projekt mit den Risiken von Zeitüberschreitungen, Budgetbeschränkungen und möglichen Betriebsunterbrechungen während des Migrationsprozesses auseinandersetzen.

Definition von Rollen und Verantwortlichkeiten

Angesichts dieser komplexen Herausforderungen, die technische, personelle und geschäftliche Dimensionen umfassen, ist die Festlegung klarer Rollen und Verantwortlichkeiten entscheidend für den Projekterfolg. Eine RACI-Matrix (Responsible, Accountable, Consulted and Informed) bietet die notwendige Struktur, um diese Herausforderungen zu bewältigen. Sie definiert ausdrücklich, wer Entscheidungen trifft, wer die Arbeit ausführt, wer Input liefert und wer in jeder Phase der Zerlegung auf dem Laufenden bleiben muss. Diese Klarheit trägt dazu bei, Verzögerungen aufgrund unklarer Entscheidungen zu vermeiden, fördert eine angemessene Einbindung der Interessengruppen und schafft Rechenschaftspflicht für wichtige Ergebnisse. Ohne einen solchen Rahmen könnten Teams mit sich überschneidenden Zuständigkeiten, verpassten Mitteilungen und unklaren Eskalationspfaden zu kämpfen haben — Probleme, die die bestehende technische Komplexität und

die Herausforderungen des Change-Managements verschärfen und gleichzeitig das Risiko von Zeit- und Budgetüberschreitungen erhöhen könnten.

Die folgende RACI-Beispielmatrix ist ein Ausgangspunkt, der Ihnen helfen kann, mögliche Rollen und Verantwortlichkeiten in Ihrem Unternehmen zu klären.

Aufgabe oder Aktivität	Projektleiter	Architekt	Entwickler	Interessensvertreter
Identifizieren Sie Geschäftsergebnisse und Herausforderungen	A/R	R	C	–
Definieren Sie den Umfang und identifizieren Sie die Anforderungen	A	R	C	C/I
Identifizieren Sie die Erfolgskennzahlen des Projekts	A	R	C	I
Erstellen Sie den Kommunikationsplan und führen Sie ihn aus	A/R	C	C	I
Definieren Sie die Zielarchitektur	I	A/R	C	–

Kontrollieren Sie den Datenbank zugriff	I	A/R	R	–
Erstellen Sie den Plan zur Geschäfts kontinuität und führen Sie ihn aus	A/R	C	I	–
Analysieren Sie Kohäsion und Kopplung	I	A/R	R	I
Verschieben Sie die Geschäfts logik (z. B. gespeicherte Prozedure n) von der Datenbank auf die Anwendung sebene	I	A	R	–
Entkoppeln Sie Tabellenb eziehungen, sogenannte Joins	I	A	R	–

Definition des Umfangs und der Anforderungen für die Datenbankzerlegung

Bei der Definition des Umfangs und der Festlegung der Anforderungen für Ihr Projekt zur Datenbankzerlegung müssen Sie von den Anforderungen Ihrer Organisation ausgehen. Dies erfordert einen systematischen Ansatz, der die technische Machbarkeit mit dem geschäftlichen Nutzen in Einklang bringt. Dieser erste Schritt bildet die Grundlage für den gesamten Prozess und hilft Ihnen sicherzustellen, dass die Projektziele mit den Zielen und Fähigkeiten der Organisation übereinstimmen.

In diesem Abschnitt werden folgende Themen behandelt:

- Einrichtung eines zentralen Analyse-Frameworks
- Definition von Systemgrenzen für die Datenbankzerlegung
- Berücksichtigung der Veröffentlichungszyklen
- Bewertung der technischen Einschränkungen bei der Zerlegung von Datenbanken
- Den organisatorischen Kontext verstehen
- Bewertung des Risikos einer Datenbankzerlegung
- Definition von Erfolgskriterien für die Datenbankzerlegung

Einrichtung eines zentralen Analyse-Frameworks

Die Definition des Umfangs beginnt mit einem systematischen Arbeitsablauf, der die Analyse durch vier miteinander verbundene Phasen führt. Dieser umfassende Ansatz stellt sicher, dass die Zerlegung von Datenbanken auf einem gründlichen Verständnis der bestehenden Systeme und betrieblichen Anforderungen basiert. Im Folgenden sind die Phasen des Kernanalyse-Frameworks aufgeführt:

1. Analyse der Akteure — Identifizieren Sie gründlich alle Systeme und Anwendungen, die mit der Datenbank interagieren. Dabei werden sowohl Hersteller, die Schreibvorgänge ausführen, als auch Verbraucher, die Lesevorgänge durchführen, abgebildet und gleichzeitig deren Zugriffsmuster, Frequenzen und Spitzennutzungszeiten dokumentiert. Diese kundenorientierte Ansicht hilft Ihnen, die Auswirkungen von Änderungen zu verstehen und kritische Pfade zu identifizieren, denen bei der Zerlegung besondere Aufmerksamkeit geschenkt werden muss.

2. Aktivitätsanalyse — Machen Sie sich eingehend mit den spezifischen Vorgängen, die die einzelnen Akteure ausführen, vertraut. Sie erstellen detaillierte CRUD-Matrizen (Create, Read, Update und Delete) für jedes System und legen fest, auf welche Tabellen sie wie zugreifen. Diese Analyse hilft Ihnen dabei, natürliche Grenzen für die Zerlegung aufzudecken, und hebt Bereiche hervor, in denen Sie die aktuelle Architektur vereinfachen können.
3. Zuordnung von Abhängigkeiten — Dokumentieren Sie sowohl direkte als auch indirekte Abhängigkeiten zwischen Systemen und erstellen Sie so klare Visualisierungen von Datenflüssen und Beziehungen. Auf diese Weise können potenzielle Schwachstellen und Bereiche identifiziert werden, in denen eine sorgfältige Planung erforderlich ist, um Vertrauen zu gewinnen. Bei der Analyse werden sowohl technische Abhängigkeiten wie gemeinsam genutzte Tabellen und Fremdschlüssel als auch Abhängigkeiten von Geschäftsprozessen wie Workflow-Sequenzen und Berichtsanforderungen berücksichtigt.
4. Konsistenzanforderungen — Untersuchen Sie die Konsistenzanforderungen jedes einzelnen Vorgangs anhand hoher Standards. Ermitteln Sie, bei welchen Vorgängen sofortige Konsistenz erforderlich ist, z. B. bei Finanztransaktionen. Andere Operationen können letztendlich konsistent ablaufen, z. B. Analyseaktualisierungen. Diese Analyse hat direkten Einfluss auf die Wahl der Zerlegungsmuster und die architektonischen Entscheidungen während des gesamten Projekts.

Definition von Systemgrenzen für die Datenbankzerlegung

Systemgrenzen sind logische Perimeter, die definieren, wo ein System endet und wo ein anderes beginnt. Sie umfassen Datenbesitz, Zugriffsmuster und Integrationspunkte. Treffen Sie bei der Definition von Systemgrenzen durchdachte, aber entscheidende Entscheidungen, die eine umfassende Planung mit den praktischen Implementierungsanforderungen in Einklang bringen. Betrachten Sie die Datenbank als logische Einheit, die sich über mehrere physische Datenbanken oder Schemas erstrecken kann. Mit dieser Grenzdefinition werden die folgenden wichtigen Ziele erreicht:

- Identifiziert alle externen Akteure und ihre Interaktionsmuster
- Bildet sowohl eingehende als auch ausgehende Abhängigkeiten umfassend ab
- Dokumentiert technische und betriebliche Einschränkungen
- Definiert klar den Umfang der Zersetzungsbemühungen

Berücksichtigung der Veröffentlichungszyklen

Das Verständnis der Release-Zyklen ist entscheidend für die Planung der Datenbankzerlegung. Überprüfen Sie die Verlängerungszeiten sowohl für das Zielsystem als auch für alle abhängigen Systeme. Identifizieren Sie Möglichkeiten für koordinierte Änderungen. Ziehen Sie jede geplante Außerbetriebnahme verbundener Systeme in Betracht, da dies Ihre Zerlegungsstrategie beeinflussen könnte. Berücksichtigen Sie bestehende Änderungsfenster und Implementierungsbeschränkungen, um Betriebsunterbrechungen zu minimieren. Stellen Sie sicher, dass Ihr Implementierungsplan mit den Veröffentlichungszeitplänen aller verbundenen Systeme übereinstimmt.

Bewertung der technischen Einschränkungen bei der Zerlegung von Datenbanken

Bevor Sie mit der Zerlegung der Datenbank fortfahren, sollten Sie die wichtigsten technischen Einschränkungen abwägen, die Ihren Modernisierungsansatz prägen werden. Untersuchen Sie die Funktionen Ihres aktuellen Technologie-Stacks, einschließlich Datenbankversionen, Frameworks, Leistungsanforderungen und Service Level Agreements. Denken Sie an Sicherheits- und Compliance-Anforderungen, insbesondere für regulierte Branchen. Informieren Sie sich über aktuelle Datenmengen, Wachstumsprognosen und verfügbare Migrationstools, um fundierte Entscheidungen zur Skalierung treffen zu können. Bestätigen Sie abschließend Ihre Zugriffsrechte auf Quellcode und Systemänderungen, da diese die praktikablen Zerlegungsstrategien bestimmen.

Den organisatorischen Kontext verstehen

Eine erfolgreiche Datenbankzerlegung setzt voraus, dass Sie die breitere organisatorische Landschaft verstehen, in der das System arbeitet. Ordnen Sie abteilungsübergreifende Abhängigkeiten zu und richten Sie klare Kommunikationskanäle zwischen den Teams ein. Beurteilen Sie die technischen Fähigkeiten Ihres Teams und ermitteln Sie alle Schulungsbedürfnisse oder Qualifikationslücken, die Sie beheben müssen. Berücksichtigen Sie die Auswirkungen des Change-Managements, einschließlich der Bewältigung von Übergängen und der Aufrechterhaltung der Geschäftskontinuität. Bewerten Sie die verfügbaren Ressourcen und etwaige Einschränkungen, z. B. Budget- oder Personalbeschränkungen. Schließlich sollten Sie Ihre Zerlegungsstrategie an den Erwartungen und Prioritäten der Beteiligten ausrichten, um eine kontinuierliche Unterstützung während des gesamten Projekts zu gewährleisten.

Bewertung des Risikos einer Datenbankzerlegung

Eine umfassende Risikobewertung ist für den Erfolg der Datenbankzerlegung unerlässlich. Bewerten Sie sorgfältig Risiken wie Datenintegrität während der Migration, mögliche Beeinträchtigungen der Systemleistung, mögliche Integrationsfehler und Sicherheitslücken. Diese technischen Herausforderungen müssen gegen Geschäftsrisiken wie mögliche Betriebsunterbrechungen, Ressourcenbeschränkungen, Zeitverzögerungen und Budgetbeschränkungen abgewogen werden. Entwickeln Sie für jedes identifizierte Risiko spezifische Minderungsstrategien und Notfallpläne, um die Dynamik des Projekts aufrechtzuerhalten und gleichzeitig den Geschäftsbetrieb zu schützen.

Erstellen Sie eine Risikomatrix, die sowohl die Auswirkungen als auch die Wahrscheinlichkeit potenzieller Probleme bewertet. Arbeiten Sie mit technischen Teams und Geschäftsbeteiligten zusammen, um Risiken zu identifizieren, klare Schwellenwerte für Interventionen festzulegen und spezifische Strategien zur Risikominderung zu entwickeln. Wenn Sie beispielsweise das Risiko eines Datenverlusts mit hoher Auswirkung und geringer Wahrscheinlichkeit einstufen, sind robuste Backup-Strategien erforderlich. Geringfügige Leistungseinbußen können mittlere Auswirkungen und hohe Wahrscheinlichkeit haben und erfordern eine proaktive Überwachung.

Richten Sie regelmäßige Risikoüberprüfungszyklen ein, um Prioritäten neu zu bewerten und die Pläne zur Risikominderung an die Weiterentwicklung des Projekts anzupassen. Dieser systematische Ansatz stellt sicher, dass sich die Ressourcen auf die kritischsten Risiken konzentrieren und gleichzeitig klare Eskalationspfade für neu auftretende Probleme beibehalten werden.

Definition von Erfolgskriterien für die Datenbankzerlegung

Erfolgskriterien für die Zerlegung von Datenbanken müssen klar definiert und in mehreren Dimensionen messbar sein. Aus geschäftlicher Sicht sollten Sie spezifische Ziele für Kostensenkung time-to-market, Verbesserung der Systemverfügbarkeit und Kundenzufriedenheit festlegen.
Der technische Erfolg sollte an quantifizierbaren Verbesserungen der Systemleistung, der Bereitstellungseffizienz, der Datenkonsistenz und der allgemeinen Zuverlässigkeit gemessen werden. Definieren Sie für den Migrationsprozess strenge Anforderungen in Bezug auf die Vermeidung von Datenverlusten, akzeptable Grenzwerte für Betriebsunterbrechungen, die Einhaltung von Budgets und die Einhaltung von Zeitplänen.

Dokumentieren Sie diese Kriterien gründlich, indem Sie Basis- und Zielkennzahlen, klare Messmethoden und regelmäßige Überprüfungspläne beibehalten. Weisen Sie jeder Erfolgsmetrik eindeutige Eigentümer zu und ordnen Sie Abhängigkeiten zwischen verschiedenen Kennzahlen zu. Dieser umfassende Ansatz zur Erfolgsmessung bringt technische Errungenschaften mit

Geschäftsergebnissen in Einklang und gewährleistet gleichzeitig die Rechenschaftspflicht während der gesamten Zerlegung.

Steuerung des Datenbankzugriffs während der Zerlegung

Viele Unternehmen stehen vor einem gemeinsamen Szenario: einer zentralen Datenbank, die über viele Jahre organisch gewachsen ist und auf die mehrere Dienste und Teams direkt zugreifen. Dies führt zu mehreren kritischen Problemen:

- Unkontrolliertes Wachstum — Da Teams kontinuierlich neue Funktionen hinzufügen und Schemas ändern, wird die Datenbank immer komplexer und schwieriger zu verwalten.
- Bedenken hinsichtlich der Leistung — Selbst bei Hardwareverbesserungen droht die wachsende Auslastung irgendwann, die Kapazitäten der Datenbank zu übersteigen. Aufgrund der Komplexität des Schemas oder mangelnder Fähigkeiten ist es unmöglich, Abfragen zu optimieren. Die Systemleistung kann nicht vorhergesagt oder erklärt werden.
- Dekompositionslähmung — Es wird fast unmöglich, die Datenbank aufzuteilen oder umzugestalten, während sie von mehreren Teams aktiv geändert wird.

> **Note**
>
> Monolithische Datenbanksysteme verwenden häufig dieselben Anmeldeinformationen für Anwendungen, Dienste oder für die Verwaltung wieder. Dies führt zu einer schlechten Rückverfolgbarkeit der Datenbank. Durch die Einrichtung spezieller Rollen und die Anwendung des Prinzips der geringsten Rechte können Sie die Sicherheit und Verfügbarkeit erhöhen.

Bei einer monolithischen Datenbank, die unhandlich geworden ist, wird eines der effektivsten Muster zur Zugriffskontrolle als Datenbank-Wrapper-Service bezeichnet. Er stellt einen strategischen ersten Schritt bei der Verwaltung komplexer Datenbanksysteme dar. Es ermöglicht einen kontrollierten Datenbankzugriff und ermöglicht eine schrittweise Modernisierung bei gleichzeitiger Risikominderung. Dieser Ansatz schafft die Grundlage für schrittweise Verbesserungen, indem er einen klaren Einblick in die Datennutzungsmuster und Abhängigkeiten bietet. Es handelt sich um eine Übergangsarchitektur, die als Schritt zur vollständigen Zerlegung der Datenbank dient. Der Wrapper-Service bietet die Stabilität und Kontrolle, die für einen erfolgreichen Übergang erforderlich sind.

In diesem Abschnitt werden folgende Themen behandelt:

- Steuerung des Zugriffs mit dem Datenbank-Wrapper-Servicemuster

- Steuerung des Zugriffs mit dem CQRS-Muster

Steuerung des Zugriffs mit dem Datenbank-Wrapper-Servicemuster

Ein Wrapper-Service ist eine Service-Ebene, die als Fassade für die Datenbank fungiert. Dieser Ansatz ist besonders nützlich, wenn Sie bestehende Funktionen beibehalten und sich gleichzeitig auf eine future Zerlegung vorbereiten müssen. Dieses Muster folgt einem einfachen Prinzip: Wenn etwas zu chaotisch ist, fangen Sie damit an, das Chaos einzudämmen. Der Wrapper-Service wird zur einzig autorisierten Methode für den Zugriff auf die Datenbank. Er bietet eine kontrollierte Schnittstelle und verbirgt gleichzeitig die zugrundeliegende Komplexität.

Verwenden Sie dieses Muster, wenn eine sofortige Datenbankzerlegung aufgrund komplexer Schemas nicht möglich ist oder wenn mehrere Dienste kontinuierlichen Datenzugriff erfordern. Es ist besonders in Übergangsphasen nützlich, da es Zeit für ein sorgfältiges Refactoring bietet und gleichzeitig die Systemstabilität gewährleistet. Das Muster eignet sich gut für die Konsolidierung des Dateneigentums auf bestimmte Teams oder wenn neue Anwendungen aggregierte Ansichten über mehrere Tabellen hinweg benötigen.

Wenden Sie dieses Muster beispielsweise an, wenn:

- Die Komplexität des Schemas verhindert eine sofortige Trennung
- Mehrere Teams benötigen fortlaufenden Datenzugriff
- Eine schrittweise Modernisierung wird bevorzugt
- Die Umstrukturierung von Teams erfordert eine klare Datenverantwortung
- Neue Anwendungen benötigen konsolidierte Datenansichten

Vorteile und Einschränkungen des Database Wrapper-Service Patterns

Im Folgenden sind die Vorteile des Datenbank-Wrapper-Musters aufgeführt:

- Kontrolliertes Wachstum — Der Wrapper-Service verhindert weitere unkontrollierte Ergänzungen zum Datenbankschema.
- Klare Grenzen — Der Implementierungsprozess hilft Ihnen dabei, klare Eigentums- und Verantwortungsgrenzen festzulegen.
- Freiheit beim Refactoring — Mit einem Wrapper-Service können Sie interne Änderungen vornehmen, ohne die Verbraucher zu beeinträchtigen.

- Verbesserte Beobachtbarkeit — Ein Wrapper-Service ist eine zentrale Anlaufstelle für Überwachung und Protokollierung.
- Vereinfachtes Testen — Ein Wrapper-Service erleichtert es nutzenden Diensten, vereinfachte Testversionen zu erstellen.

Im Folgenden sind die Einschränkungen des Datenbank-Wrapper-Musters aufgeführt.

- Technologiekopplung — Ein Wrapper-Service funktioniert am besten, wenn er denselben Technologie-Stack verwendet wie die Dienste, die ihn nutzen.
- Anfänglicher Overhead — Der Wrapper-Service erfordert zusätzliche Infrastruktur, was sich auf die Leistung auswirken kann.
- Migrationsaufwand — Um den Wrapper-Service zu implementieren, müssen Sie sich teamübergreifend abstimmen, um den direkten Zugriff zu vermeiden.
- Leistung — Wenn der Wrapping-Service stark frequentiert, stark genutzt wird oder häufig darauf zugegriffen wird, kann es bei der Nutzung von Diensten zu Leistungseinbußen kommen. Zusätzlich zur Datenbank muss der Wrapper-Service Seitennummerierung, Cursor und Datenbankverbindungen verarbeiten. Abhängig von Ihrem Anwendungsfall ist er möglicherweise nicht gut skalierbar und eignet sich möglicherweise nicht für ETL-Workloads (Extrahieren, Transformieren und Laden).

Implementierung des Datenbank-Wrapper-Servicemusters

Es gibt zwei Phasen, um das Datenbank-Wrapper-Servicemuster zu implementieren. Zunächst erstellen Sie den Datenbank-Wrapper-Dienst. Anschließend leiten Sie den gesamten Zugriff über ihn und dokumentieren die Zugriffsmuster.

Phase 1: Erstellen des Datenbank-Wrapper-Dienstes

Erstellen Sie eine einfache Serviceebene, die als Gatekeeper für Ihre Datenbank fungiert. Anfänglich sollte sie alle vorhandenen Funktionen widerspiegeln. Dieser Wrapper-Service wird zum obligatorischen Zugriffspunkt für alle Datenbankoperationen, wodurch direkte Datenbankabhängigkeiten in Abhängigkeiten auf Dienstebene umgewandelt werden. Implementieren Sie detaillierte Protokollierung und Überwachung auf dieser Ebene, um Nutzungsmuster, Leistungskennzahlen und Zugriffshäufigkeiten zu verfolgen. Behalten Sie Ihre vorhandenen gespeicherten Prozeduren bei, stellen Sie jedoch sicher, dass auf sie nur über diese neue Serviceschnittstelle zugegriffen wird.

Phase 2: Implementierung der Zugriffskontrolle

Leiten Sie den gesamten Datenbankzugriff systematisch über den Wrapper-Service um und widerrufen Sie dann direkte Datenbankberechtigungen von externen Systemen, die direkt auf die Datenbank zugreifen. Dokumentieren Sie jedes Zugriffsmuster und jede Abhängigkeit, während Dienste migriert werden. Dieser kontrollierte Zugriff ermöglicht ein internes Refactoring von Datenbankkomponenten, ohne externe Nutzer zu stören. Beginnen Sie beispielsweise mit risikoarmen, schreibgeschützten Vorgängen statt mit komplexen Transaktionsworkflows.

Phase 3: Überwachen Sie die Datenbankleistung

Verwenden Sie den Wrapper-Service als zentralen Überwachungspunkt für die Datenbankleistung. Verfolgen Sie wichtige Kennzahlen wie Antwortzeiten für Abfragen, Nutzungsmuster, Fehlerquoten und Ressourcennutzung. Richten Sie Warnmeldungen für Leistungsschwellenwerte und ungewöhnliche Muster ein. Überwachen Sie beispielsweise langsam laufende Abfragen, die Auslastung des Verbindungspools und den Transaktionsdurchsatz, um potenzielle Probleme proaktiv zu identifizieren.

Verwenden Sie diese konsolidierte Ansicht, um die Datenbankleistung durch Abfrageoptimierung, Anpassungen der Ressourcenzuweisung und Analyse von Nutzungsmustern zu optimieren. Der zentralisierte Charakter des Wrapper-Service erleichtert die Implementierung von Verbesserungen und die Überprüfung ihrer Auswirkungen auf alle Verbraucher bei gleichzeitiger Beibehaltung konsistenter Leistungsstandards.

Bewährte Methoden für die Implementierung eines Datenbank-Wrapper-Service

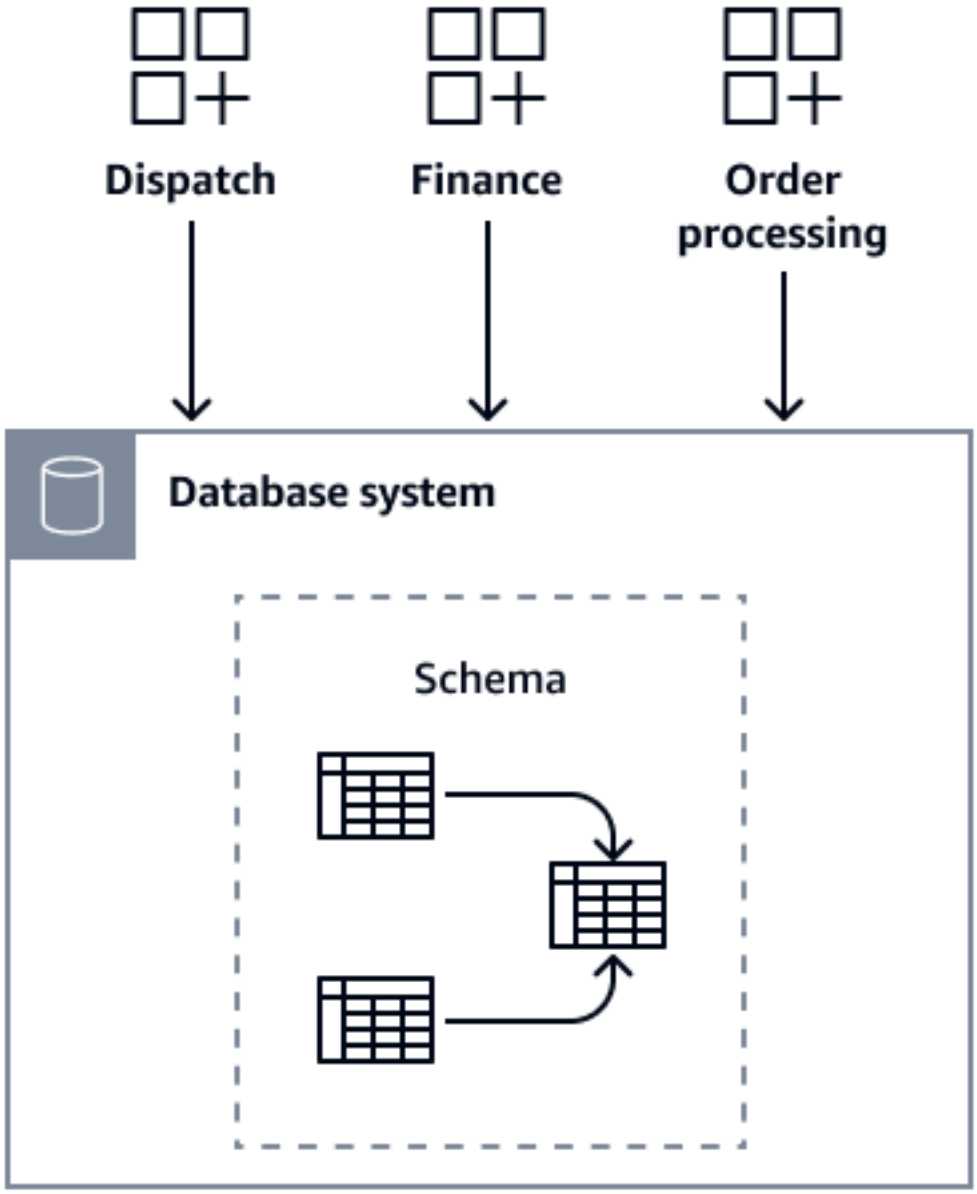
Die folgenden bewährten Methoden können Ihnen bei der Implementierung eines Datenbank-Wrapper-Dienstes helfen:

- Fangen Sie klein an — Beginnen Sie mit einem minimalen Wrapper, der lediglich vorhandene Funktionen als Proxy bereitstellt
- Stabilität aufrechterhalten — Sorgen Sie für eine stabile Serviceschnittstelle und nehmen Sie gleichzeitig interne Verbesserungen vor
- Nutzung überwachen — Implementieren Sie eine umfassende Überwachung, um die Zugriffsmuster zu verstehen
- Klare Verantwortung — Weisen Sie ein engagiertes Team zu, das sowohl den Wrapper als auch das zugrundeliegende Schema verwaltet

- Förderung der lokalen Speicherung — Motivieren Sie Teams, ihre Daten in ihren eigenen Datenbanken zu speichern

Ein szenariobasiertes Beispiel

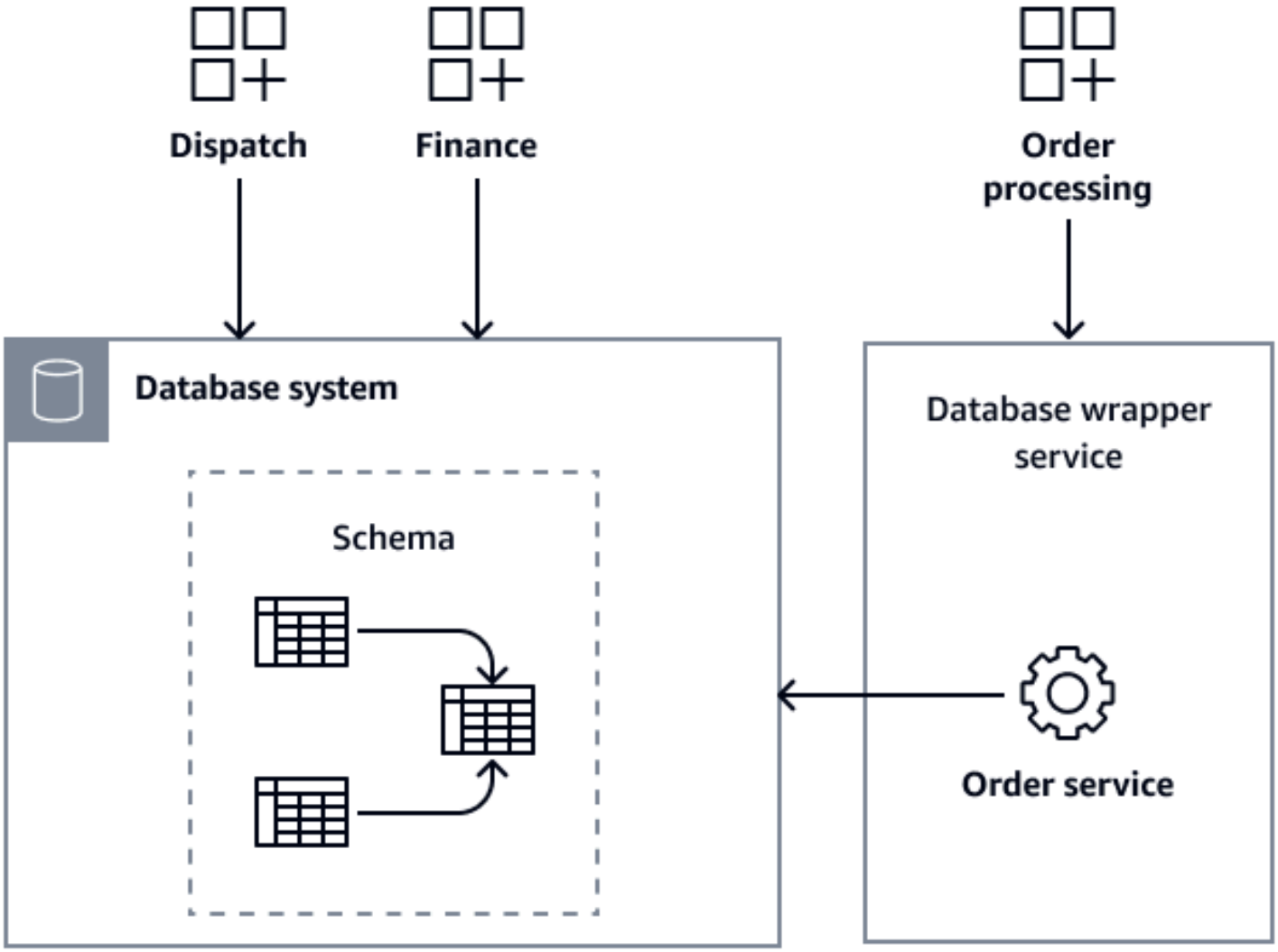
In diesem Abschnitt wird ein Beispiel dafür beschrieben, wie ein fiktives Unternehmen namens AnyCompany Books das Datenbank-Wrapper-Muster verwenden könnte, um den Zugriff auf sein monolithisches Datenbanksystem zu kontrollieren. Bei AnyCompany Books gibt es drei wichtige Dienste: Versand, Finanzen und Auftragsabwicklung. Diese Dienste teilen sich den Zugriff auf eine zentrale Datenbank. Jeder Dienst wird von einem anderen Team verwaltet. Im Laufe der Zeit ändern sie das Datenbankschema unabhängig voneinander, um ihren spezifischen Anforderungen gerecht zu werden. Dies hat zu einem verworrenen Netz von Abhängigkeiten und einer immer komplexeren Datenbankstruktur geführt.



Der Anwendungs- oder Unternehmensarchitekt des Unternehmens ist sich der Notwendigkeit bewusst, diese monolithische Datenbank zu zerlegen. Ihr Ziel ist es, jedem Dienst eine eigene Datenbank zur Verfügung zu stellen, um die Wartbarkeit zu verbessern und teamübergreifende Abhängigkeiten zu reduzieren. Sie stehen jedoch vor einer großen Herausforderung: Es ist fast unmöglich, die Datenbank zu zerlegen, solange alle drei Teams sie weiterhin aktiv für ihre laufenden Projekte modifizieren. Aufgrund der ständigen Schemaänderungen und der mangelnden Koordination zwischen den Teams ist es äußerst riskant, eine signifikante Umstrukturierung zu versuchen.

Der Architekt verwendet das Database-Wrapper-Servicemuster, um mit der Steuerung des Zugriffs auf die monolithische Datenbank zu beginnen. Zunächst richten sie den Datenbank-Wrapper-

Service für ein bestimmtes Modul ein, den sogenannten Order-Service. Dann leiten sie den Order Processing Service weiter, um auf den Wrapper-Service zuzugreifen, anstatt direkt auf die Datenbank zuzugreifen. Das folgende Bild zeigt die modifizierte Infrastruktur.

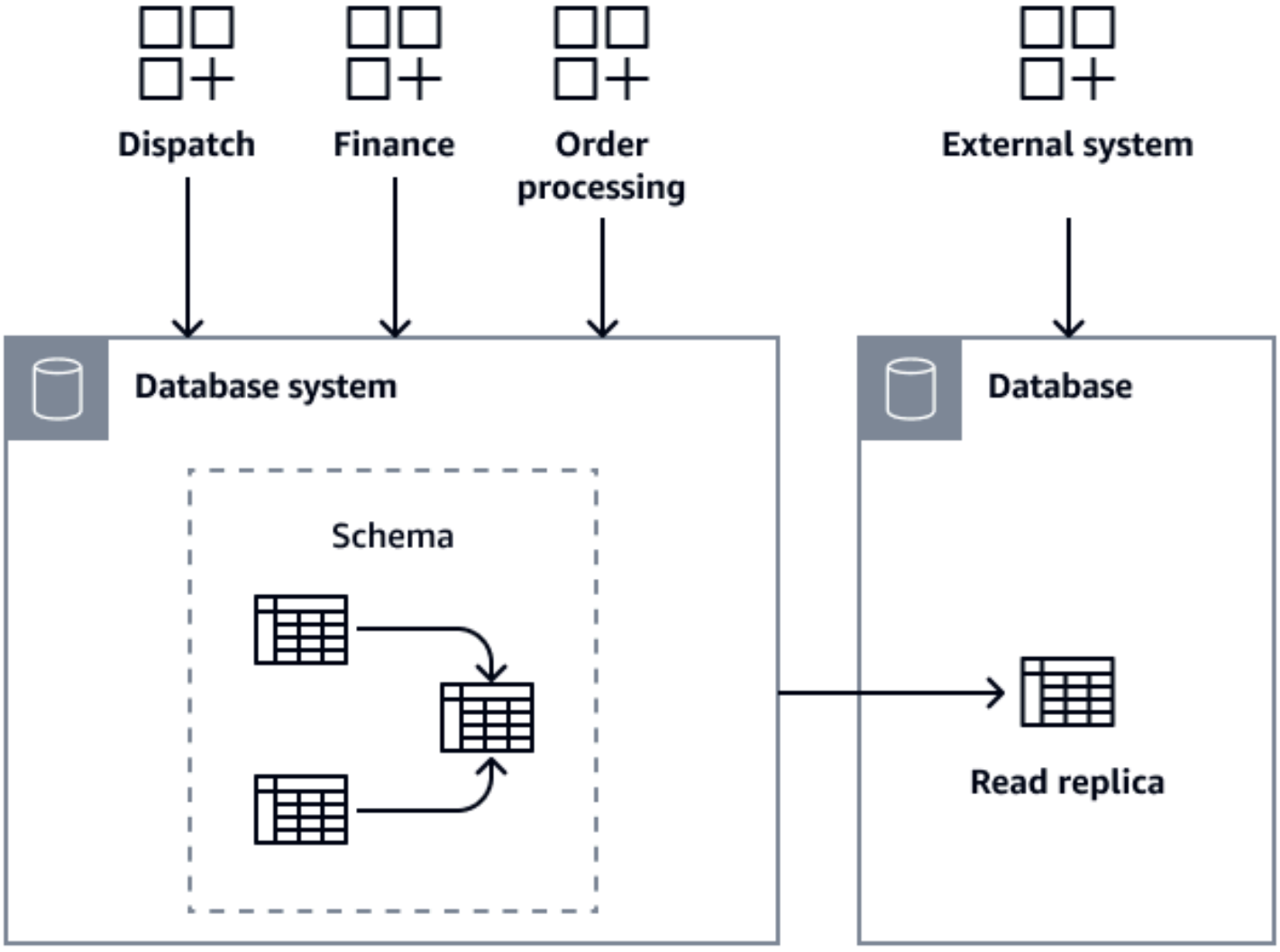


Steuerung des Zugriffs mit dem CQRS-Muster

Ein weiteres Muster, mit dem Sie externe Systeme isolieren können, die eine Verbindung zu dieser zentralen Datenbank herstellen, ist Command Query Responsibility Segregation (CQRS). Wenn einige der externen Systeme hauptsächlich für Lesevorgänge, z. B. für Analysen, Berichte oder andere leseintensive Operationen, eine Verbindung zu Ihrer zentralen Datenbank herstellen, können Sie separate leseoptimierte Datenspeicher erstellen.

Dieses Muster schützt diese externen Systeme effektiv vor den Auswirkungen von Datenbankzerlegung und Schemaänderungen. Durch die Bereitstellung spezieller Lesereplikate

oder speziell für bestimmte Abfragemuster erstellter Datenspeicher können Teams ihren Betrieb fortsetzen, ohne von Änderungen in der primären Datenbankstruktur beeinträchtigt zu werden. Während Sie beispielsweise Ihre monolithische Datenbank zerlegen, können Berichtssysteme weiterhin mit ihren vorhandenen Datenansichten arbeiten, und analytische Workloads können ihre aktuellen Abfragemuster über spezielle Analysespeicher beibehalten. Dieser Ansatz bietet technische Isolierung und ermöglicht organisatorische Autonomie, da verschiedene Teams ihre Systeme unabhängig voneinander weiterentwickeln können, ohne dass eine enge Kopplung mit der Transformation der Primärdatenbank erforderlich ist.



Weitere Informationen zu diesem Muster und ein Beispiel für seine Verwendung zum Entkoppeln von Tabellenbeziehungen finden Sie weiter unten in diesem HandbuchCQRS-Muster.

Analyse von Kohäsion und Kopplung für die Datenbankzerlegung

Dieser Abschnitt hilft Ihnen bei der Analyse von Kopplungs- und Kohäsionsmustern in Ihrer monolithischen Datenbank, um deren Zerlegung zu steuern. Zu verstehen, wie Datenbankkomponenten interagieren und voneinander abhängen, ist entscheidend für die Identifizierung natürlicher Bruchstellen, die Bewertung der Komplexität und die Planung eines schrittweisen Migrationsansatzes. Diese Analyse deckt verborgene Abhängigkeiten auf, hebt Bereiche hervor, die für eine sofortige Trennung geeignet sind, und hilft Ihnen, die Zerlegungsbemühungen zu priorisieren und gleichzeitig die Transformationsrisiken zu minimieren. Indem Sie sowohl die Kopplung als auch den Zusammenhalt untersuchen, können Sie fundierte Entscheidungen über die Reihenfolge der Trennung der Komponenten treffen, um die Systemstabilität während des gesamten Transformationsprozesses aufrechtzuerhalten.

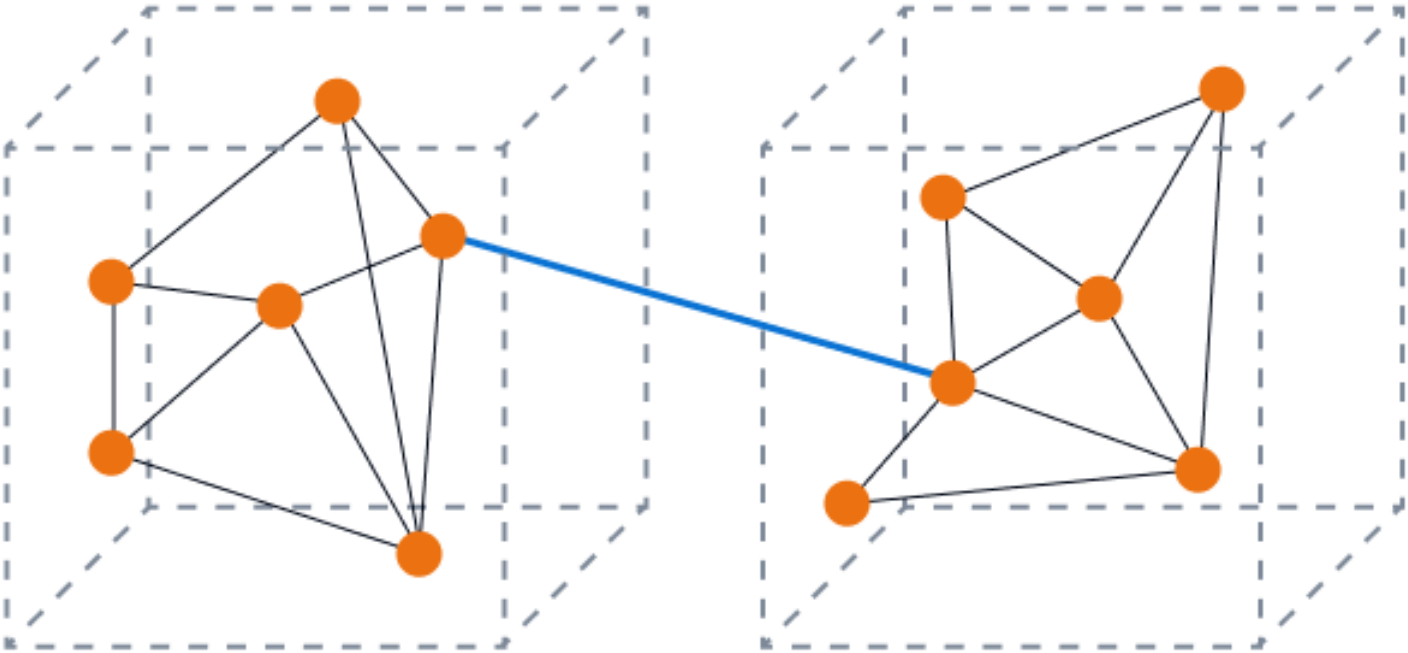
In diesem Abschnitt werden folgende Themen behandelt:

- Über Kohäsion und Kopplung
- Übliche Kopplungsmuster in monolithischen Datenbanken
- Gemeinsame Kohäsionsmuster in monolithischen Datenbanken
- Implementierung niedriger Kopplung und hoher Kohäsion

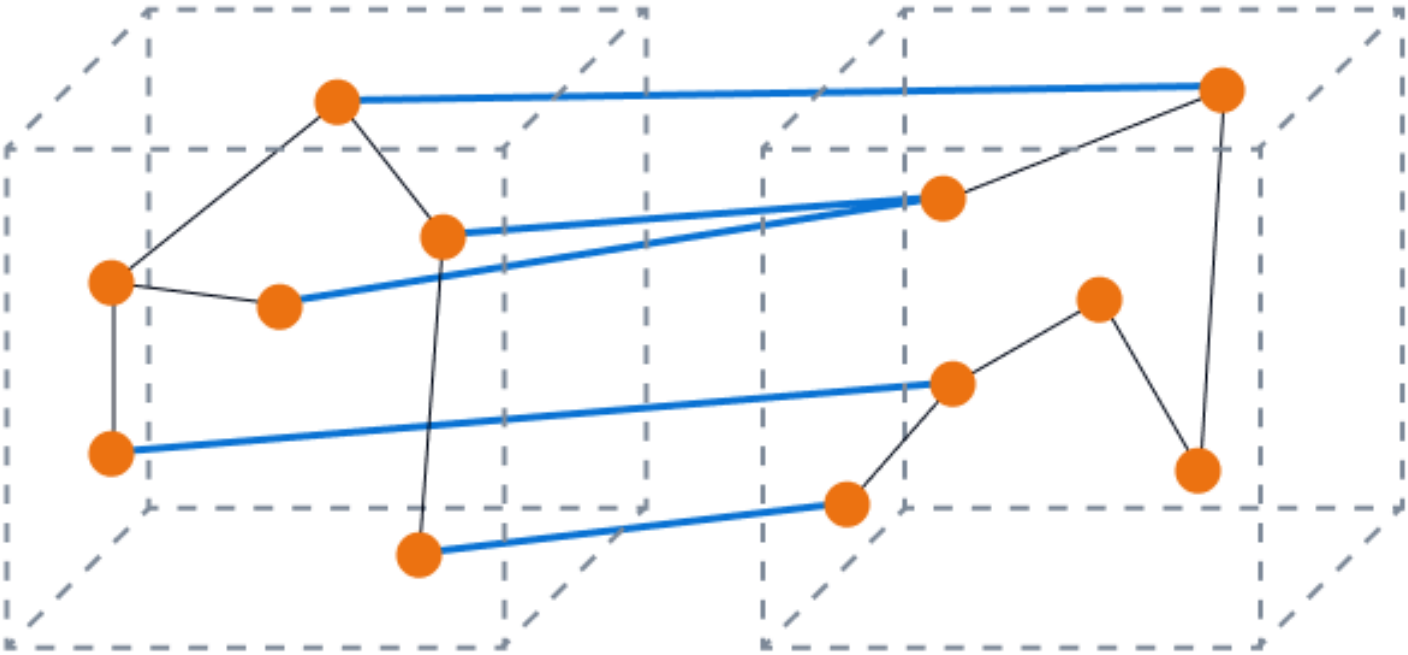
Über Kohäsion und Kopplung

Die Kopplung misst den Grad der Interdependenz zwischen Datenbankkomponenten. In einem gut konzipierten System möchten Sie eine lose Kopplung erreichen, bei der Änderungen an einer Komponente nur minimale Auswirkungen auf andere haben. Kohäsion misst, wie gut die Elemente innerhalb einer Datenbankkomponente zusammenarbeiten, um einem einzigen, genau definierten Zweck zu dienen. Ein hoher Zusammenhalt bedeutet, dass die Elemente einer Komponente eng miteinander verwandt sind und sich auf eine bestimmte Funktion konzentrieren. Bei der Zerlegung einer monolithischen Datenbank müssen Sie sowohl den Zusammenhalt innerhalb einzelner Komponenten als auch die Kopplung zwischen ihnen analysieren. Diese Analyse hilft Ihnen dabei, fundierte Entscheidungen darüber zu treffen, wie Sie die Datenbank aufschlüsseln und gleichzeitig die Systemintegrität und Leistung aufrechterhalten können.

Die folgende Abbildung zeigt eine lose Kopplung mit hoher Kohäsion. Die Komponenten in der Datenbank arbeiten zusammen, um eine bestimmte Funktion auszuführen, und Sie minimieren die Auswirkungen von Änderungen auf eine einzelne Komponente. Dies ist der ideale Zustand.



Das folgende Bild zeigt eine hohe Kopplung bei geringer Kohäsion. Die Datenbankkomponenten sind getrennt, und es ist sehr wahrscheinlich, dass sich Änderungen auf andere Komponenten auswirken.



Übliche Kopplungsmuster in monolithischen Datenbanken

Es gibt mehrere Kopplungsmuster, die häufig vorkommen, wenn eine monolithische Datenbank in Microservice-spezifische Datenbanken zerlegt wird. Das Verständnis dieser Muster ist entscheidend für erfolgreiche Initiativen zur Datenbankmodernisierung. In diesem Abschnitt werden die einzelnen Muster, ihre Herausforderungen und bewährte Methoden zur Reduzierung der Kopplung beschrieben.

Kopplungsmuster bei der Implementierung

Definition: Komponenten sind auf Code- und Schemaebene eng miteinander verbunden. Wenn Sie beispielsweise die Struktur einer `customer` Tabelle ändern`order`, wirkt sich dies auf`inventory`, und `billing` Dienste aus.

Auswirkungen der Modernisierung: Jeder Microservice benötigt ein eigenes Datenbankschema und eine eigene Datenzugriffsebene.

Herausforderungen:

- Änderungen an gemeinsam genutzten Tabellen wirken sich auf mehrere Dienste aus
- Hohes Risiko unbeabsichtigter Nebenwirkungen
- Höhere Komplexität der Tests
- Es ist schwierig, einzelne Komponenten zu modifizieren

Bewährte Verfahren zur Reduzierung der Kopplung:

- Definieren Sie klare Schnittstellen zwischen Komponenten
- Verwenden Sie Abstraktionsebenen, um Implementierungsdetails zu verbergen
- Implementieren Sie domänenspezifische Schemas

Temporales Kopplungsmuster

Definition: Operationen müssen in einer bestimmten Reihenfolge ausgeführt werden. Beispielsweise kann die Auftragsabwicklung erst fortgesetzt werden, wenn die Inventaraktualisierungen abgeschlossen sind.

Auswirkungen der Modernisierung: Jeder Microservice benötigt eine autonome Datensteuerung.

Herausforderungen:

- Aufhebung synchroner Abhängigkeiten zwischen Diensten
- Leistungsengpässe
- Schwer zu optimieren
- Eingeschränkte Parallelverarbeitung

Bewährte Methoden zur Reduzierung der Kopplung:

- Implementieren Sie nach Möglichkeit asynchrone Verarbeitung
- Verwenden Sie ereignisgesteuerte Architekturen
- Entwerfen Sie gegebenenfalls auf Konsistenz

Muster der Einsatzkopplung

Definition: Systemkomponenten müssen als eine Einheit bereitgestellt werden. Beispielsweise erfordert eine geringfügige Änderung der Zahlungsverarbeitungslogik eine erneute Bereitstellung der gesamten Datenbank.

Auswirkungen der Modernisierung: Unabhängige Datenbankbereitstellungen pro Service

Herausforderungen:

- Bereitstellungen mit hohem Risiko
- Begrenzte Einsatzhäufigkeit
- Komplexe Rollback-Verfahren

Bewährte Verfahren zur Reduzierung der Kopplung:

- Unterteilen Sie sich in unabhängig voneinander einsetzbare Komponenten
- Implementieren Sie Datenbank-Sharding-Strategien
- Verwenden Sie blaugrüne Bereitstellungsmuster

Muster für die Domänenkopplung

Definition: Geschäftsdomänen teilen sich Datenbankstrukturen und Logik. Beispielsweise teilen sich die `inventory` Domänen `customerorder`, und Tabellen und gespeicherte Prozeduren gemeinsam.

Auswirkungen der Modernisierung: Domänenspezifische Datenisolierung

Herausforderungen:

- Komplexe Domänengrenzen
- Es ist schwierig, einzelne Domänen zu skalieren
- Verworrene Geschäftsregeln

Bewährte Verfahren zur Reduzierung der Kopplung:

- Identifizieren Sie klare Domänengrenzen
- Trennen Sie die Daten nach Domänenkontext
- Implementieren Sie domänenspezifische Dienste

Gemeinsame Kohäsionsmuster in monolithischen Datenbanken

Es gibt mehrere Kohäsionsmuster, die häufig bei der Bewertung von Datenbankkomponenten im Hinblick auf ihre Zerlegung gefunden werden. Das Verständnis dieser Muster ist entscheidend für die Identifizierung gut strukturierter Datenbankkomponenten. In diesem Abschnitt werden jedes Muster, seine Merkmale und bewährte Verfahren zur Stärkung des Zusammenhalts beschrieben.

Funktionelles Kohäsionsmuster

Definition: Alle Elemente unterstützen direkt die Ausführung einer einzigen, genau definierten Funktion und tragen dazu bei. Beispielsweise verarbeiten alle gespeicherten Prozeduren und Tabellen in einem Zahlungsverarbeitungsmodul nur zahlungsbezogene Vorgänge.

Auswirkungen der Modernisierung: Ideales Muster für das Design von Microservice-Datenbanken

Herausforderungen:

- Identifizierung klarer funktionaler Grenzen

- Trennung von Komponenten mit gemischten Verwendungszwecken
- Beibehaltung der alleinigen Verantwortung

Bewährte Verfahren zur Stärkung des Zusammenhalts:

- Gruppieren Sie verwandte Funktionen
- Entfernen Sie Funktionen, die nicht miteinander verwandt sind
- Definieren Sie klare Komponentengrenzen

Sequentielles Kohäsionsmuster

Definition: Die Ausgabe eines Elements wird zur Eingabe für ein anderes. Beispielsweise fließen die Validierungsergebnisse für eine Bestellung in die Auftragsabwicklung ein.

Auswirkungen der Modernisierung: Erfordert eine sorgfältige Workflow-Analyse und Datenflussabbildung

Herausforderungen:

- Verwaltung von Abhängigkeiten zwischen den Schritten
- Umgang mit Ausfallszenarien
- Aufrechterhaltung der Prozessreihenfolge

Bewährte Verfahren zur Stärkung des Zusammenhalts:

- Dokumentieren Sie klare Datenflüsse
- Implementieren Sie die richtige Fehlerbehandlung
- Entwerfen Sie klare Schnittstellen zwischen den Schritten

Muster des kommunikativen Zusammenhalts

Definition: Elemente arbeiten mit denselben Daten. Beispielsweise arbeiten alle Funktionen zur Verwaltung von Kundenprofilen mit Kundendaten.

Auswirkungen der Modernisierung: Hilft bei der Identifizierung von Datengrenzen für die Trennung von Diensten, um die Kopplung zwischen Modulen zu verringern

Herausforderungen:

- Bestimmung des Dateneigentums
- Verwaltung des gemeinsamen Datenzugriffs
- Wahrung der Datenkonsistenz

Bewährte Verfahren zur Stärkung des Zusammenhalts:

- Definieren Sie klare Dateneigentumsrechte
- Implementieren Sie die richtigen Datenzugriffsmuster
- Entwerfen Sie eine effektive Datenpartitionierung

Prozedurales Kohäsionsmuster

Definition: Elemente werden gruppiert, weil sie in einer bestimmten Reihenfolge ausgeführt werden müssen, sie aber möglicherweise nicht funktionell zusammenhängen. Bei der Auftragsverarbeitung wird beispielsweise eine gespeicherte Prozedur, die sowohl die Auftragsvalidierung als auch die Benutzerbenachrichtigung verarbeitet, einfach deshalb gruppiert, weil sie nacheinander ablaufen, obwohl sie unterschiedlichen Zwecken dienen und von separaten Diensten verarbeitet werden könnten.

Auswirkungen der Modernisierung: Erfordert eine sorgfältige Trennung der Verfahren unter Beibehaltung des Prozessablaufs

Herausforderungen:

- Aufrechterhaltung des korrekten Prozessablaufs nach der Zersetzung
- Identifizierung echter funktionaler Grenzen im Vergleich zu prozeduralen Abhängigkeiten

Bewährte Verfahren zur Stärkung des Zusammenhalts:

- Separate Verfahren, die auf ihrem funktionalen Zweck und nicht auf der Reihenfolge ihrer Ausführung basieren
- Verwenden Sie Orchestrierungsmuster, um den Prozessablauf zu verwalten
- Implementieren Sie Workflow-Management-Systeme für komplexe Abläufe

- Entwerfen Sie ereignisgesteuerte Architekturen, um Prozessschritte unabhängig voneinander abzuwickeln

Temporales Kohäsionsmuster

Definition: Die Elemente sind durch zeitliche Anforderungen miteinander verknüpft. Wenn beispielsweise eine Bestellung aufgegeben wird, müssen mehrere Vorgänge gleichzeitig ausgeführt werden: Inventarprüfung, Zahlungsabwicklung, Auftragsbestätigung und Versandbenachrichtigung müssen alle innerhalb eines bestimmten Zeitfensters erfolgen, um einen konsistenten Bestellstatus aufrechtzuerhalten.

Auswirkungen der Modernisierung: In verteilten Systemen ist möglicherweise eine besondere Behandlung erforderlich

Herausforderungen:

- Koordination der zeitlichen Abhängigkeiten zwischen verteilten Diensten
- Verwaltung verteilter Transaktionen
- Bestätigung des Abschlusses des Prozesses für mehrere Komponenten

Bewährte Verfahren zur Stärkung des Zusammenhalts:

- Implementieren Sie geeignete Planungsmechanismen und Timeouts
- Verwenden Sie ereignisgesteuerte Architekturen mit klarer Sequenzbehandlung
- Achten Sie bei der Planung auf Konsistenz mit den Vergütungsmustern
- Implementieren Sie Saga-Muster für verteilte Transaktionen

Logisches oder zufälliges Kohäsionsmuster

Definition: Elemente werden logisch so kategorisiert, dass sie dieselben Dinge bewirken, auch wenn sie schwache oder keine bedeutungsvollen Beziehungen haben. Ein Beispiel ist das Speichern von Kundenauftragsdaten, Lagerbestandszahlen und Marketing-E-Mail-Vorlagen in demselben Datenbankschema, da sie sich alle auf Vertriebsabläufe beziehen, obwohl unterschiedliche Zugriffsmuster, Lebenszyklusmanagement und Skalierungsanforderungen gelten. Ein anderes Beispiel ist die Kombination von Auftragszahlungsabwicklung und Produktkatalogverwaltung

innerhalb derselben Datenbankkomponente, da beide Teil des E-Commerce-Systems sind, obwohl sie unterschiedliche Geschäftsfunktionen mit unterschiedlichen betrieblichen Anforderungen erfüllen.

Auswirkungen der Modernisierung: Sollte überarbeitet oder neu organisiert werden

Herausforderungen:

- Identifizierung besserer Organisationsmuster
- Überflüssige Abhängigkeiten aufbrechen
- Restrukturierung von Komponenten, die willkürlich gruppiert wurden

Bewährte Verfahren zur Stärkung des Zusammenhalts:

- Reorganisieren Sie sich auf der Grundlage echter funktionaler Grenzen und Geschäftsbereiche
- Entfernen Sie willkürliche Gruppierungen, die auf oberflächlichen Beziehungen basieren
- Implementieren Sie die richtige Trennung der Elemente auf der Grundlage der Geschäftskapazitäten
- Passen Sie die Datenbankkomponenten an ihre spezifischen betrieblichen Anforderungen an

Implementierung niedriger Kopplung und hoher Kohäsion

Best Practices

Die folgenden bewährten Verfahren können Ihnen dabei helfen, eine geringe Kopplung zu erreichen:

- Minimiert die Abhängigkeiten zwischen Datenbankkomponenten
- Verwenden Sie klar definierte Schnittstellen für die Interaktion mit Komponenten
- Vermeiden Sie gemeinsame Status- und globale Datenstrukturen

Die folgenden bewährten Methoden können Ihnen dabei helfen, eine hohe Kohäsion zu erreichen:

- Gruppieren Sie zusammengehörige Daten und Vorgänge
- Stellen Sie sicher, dass für jede Komponente eine einzige, klare Verantwortung besteht
- Halten Sie klare Grenzen zwischen verschiedenen Geschäftsbereichen aufrecht

Phase 1: Ordnen Sie Datenabhängigkeiten zu

Ordnen Sie Datenbeziehungen zu und identifizieren Sie natürliche Grenzen. Sie können Tools verwenden, um beispielsweise die Datenbank zu visualisieren SchemaSpy, indem Sie die Tabellen in einem Entity-Relationship-Diagramm (ER) anzeigen. Dies ermöglicht eine statische Analyse der Datenbank und zeigt einige klare Grenzen und Abhängigkeiten innerhalb der Datenbank auf.

Sie können Ihre Datenbankschemas auch in eine Graphdatenbank oder in ein Jupiter Notizbuch exportieren. Anschließend können Sie Algorithmen für Clustering oder miteinander verbundene Komponenten anwenden, um natürliche Grenzen und Abhängigkeiten zu identifizieren. Andere AWS Partner Tools, wie z. B. CAST Imaging, können Ihnen helfen, Ihre Datenbankabhängigkeiten zu verstehen.

Phase 2: Analysieren Sie Transaktionsgrenzen und Zugriffsmuster

Analysieren Sie Transaktionsmuster, um die Eigenschaften Atomizität, Konsistenz, Isolation und Dauerhaftigkeit (ACID) aufrechtzuerhalten und zu verstehen, wie auf Daten zugegriffen und wie sie verändert werden. Sie können Datenbankanalyse- und Diagnosetools wie Oracle Automatic Workload Repository (AWR)oder verwenden. PostgreSQL pg_stat_statements Diese Analyse hilft Ihnen zu verstehen, wer auf die Datenbank zugreift und welche Transaktionsgrenzen bestehen. Sie kann Ihnen auch dabei helfen, den Zusammenhalt und die Kopplung zwischen Tabellen zur Laufzeit zu verstehen. Sie können auch Tools zur Überwachung und Profilerstellung verwenden, mit denen Code- und Datenbankausführungsprofile verknüpft werden können, z. B. Dynatrace AppEngine

KI-Tools wie können Ihnen helfen vFunction, Domänengrenzen zu identifizieren, indem sie die Funktions- und Domänengrenzen der Anwendung analysieren. Obwohl vFunction in erster Linie die Anwendungsebene analysiert wird, können die gewonnenen Erkenntnisse als Grundlage für die Zerlegung sowohl der Anwendung als auch der Datenbank dienen und so die Ausrichtung auf Geschäftsdomänen unterstützen.

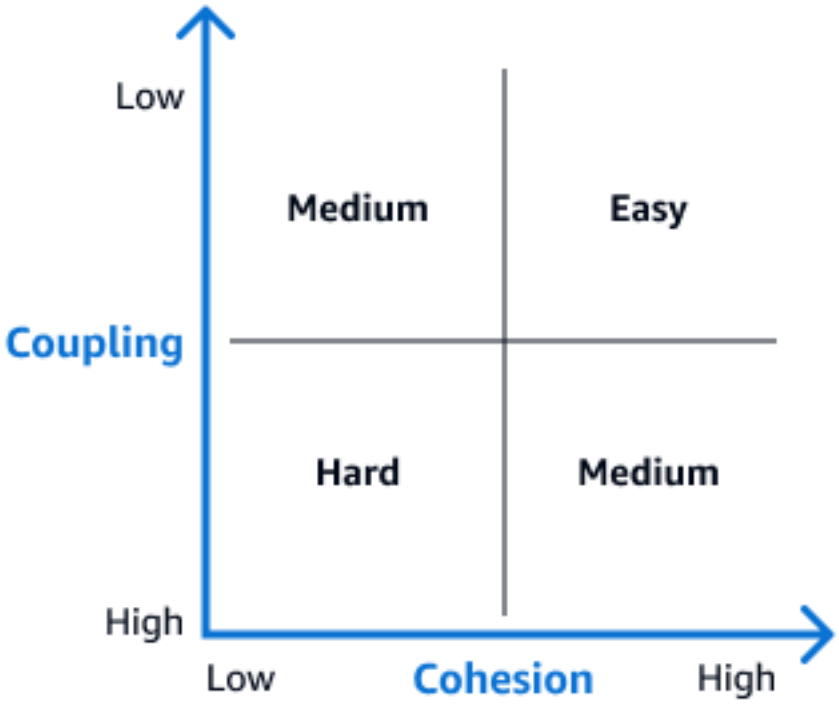
Phase 3: Identifizieren Sie eigenständige Tabellen

Suchen Sie nach Tabellen, die zwei Hauptmerkmale aufweisen:

- Hohe Kohäsion — Die Inhalte der Tabelle stehen in engem Zusammenhang
- Geringe Kopplung — Sie haben nur minimale Abhängigkeiten von anderen Tabellen.

Anhand der folgenden Kopplungs- und Kohäsionsmatrix können Sie erkennen, wie schwierig es ist, die einzelnen Tabellen zu entkoppeln. Tabellen, die im oberen rechten Quadranten

dieser Matrix erscheinen, eignen sich ideal für anfängliche Entkopplungsversuche, da sie sich am leichtesten trennen lassen. In einem ER-Diagramm weisen diese Tabellen nur wenige Fremdschlüsselbeziehungen oder andere Abhängigkeiten auf. Nachdem Sie diese Tabellen entkoppelt haben, gehen Sie zu Tabellen mit komplexeren Beziehungen über.



> **Note**
>
> Die Datenbankstruktur spiegelt häufig die Anwendungsarchitektur wider. Tabellen, die auf Datenbankebene einfacher zu entkoppeln sind, entsprechen in der Regel Komponenten, die sich auf Anwendungsebene leichter in Microservices umwandeln lassen.

Migration der Geschäftslogik von der Datenbank zur Anwendungsebene

Die Migration der Geschäftslogik von in der Datenbank gespeicherten Prozeduren, Triggern und Funktionen zu Diensten auf Anwendungsebene ist ein entscheidender Schritt bei der Zerlegung monolithischer Datenbanken. Diese Transformation verbessert die Autonomie der Dienste, vereinfacht die Wartung und verbessert die Skalierbarkeit. Dieser Abschnitt enthält Anleitungen zur Analyse der Datenbanklogik, zur Planung der Migrationsstrategie und zur anschließenden Implementierung der Transformation unter Wahrung der Geschäftskontinuität. Außerdem wird die Erstellung eines effektiven Rollback-Plans erörtert.

In diesem Abschnitt werden folgende Themen behandelt:

Phase 1: Analyse der Geschäftslogik

Bei der Modernisierung monolithischer Datenbanken müssen Sie zunächst eine umfassende Analyse Ihrer vorhandenen Datenbanklogik durchführen. Diese Phase konzentriert sich auf drei Hauptkategorien:

- Gespeicherte Prozeduren enthalten häufig kritische Geschäftsvorgänge, darunter Datenmanipulationslogik, Geschäftsregeln, Validierungsprüfungen und Berechnungen. Als Kernkomponenten der Geschäftslogik der Anwendung müssen sie sorgfältig zerlegt werden. Beispielsweise können die gespeicherten Prozeduren einer Finanzorganisation Zinsberechnungen, Kontenabstimmungen und Konformitätsprüfungen übernehmen.
- Trigger sind wichtige Datenbankkomponenten, die Prüfpfade, Datenvalidierung, Berechnungen und tabellenübergreifende Konsistenz verwalten. Beispielsweise könnte ein Einzelhandelsunternehmen Trigger verwenden, um Inventaraktualisierungen in seinem gesamten Auftragsverarbeitungssystem zu verwalten, was die Komplexität automatisierter Datenbankvorgänge verdeutlicht.
- Funktionen in Datenbanken verwalten hauptsächlich Datentransformationen, Berechnungen und Suchvorgänge. Sie sind häufig in mehrere Verfahren und Anwendungen eingebettet.

Eine Organisation im Gesundheitswesen könnte beispielsweise Funktionen verwenden, um Patientendaten zu normalisieren oder medizinische Codes nachzuschlagen.

Jede Kategorie steht für unterschiedliche Aspekte der Geschäftslogik, die in die Datenbankschicht eingebettet ist. Sie müssen alle sorgfältig evaluieren und planen, um sie auf die Anwendungsebene zu migrieren.

Während dieser Analysephase stehen Kunden in der Regel vor drei großen Herausforderungen. Erstens entstehen komplexe Abhängigkeiten durch verschachtelte Prozeduraufrufe, schemaübergreifende Verweise und implizite Datenabhängigkeiten. Zweitens wird das Transaktionsmanagement von entscheidender Bedeutung, insbesondere wenn es um mehrstufige Transaktionen geht und die Datenkonsistenz über verteilte Systeme hinweg aufrechterhalten wird. Drittens müssen Leistungsaspekte sorgfältig abgewogen werden, insbesondere bei Stapelverarbeitungsvorgängen, Massendatenaktualisierungen und Echtzeitberechnungen, die derzeit von der Nähe zu den Daten profitieren.

Um diesen Herausforderungen effektiv zu begegnen, können Sie AWS Schema Conversion Tool (AWS SCT) für die Erstanalyse und anschließend detaillierte Tools zur Abhängigkeitsabbildung verwenden. Dieser Ansatz hilft Ihnen, den vollen Umfang Ihrer Datenbanklogik zu verstehen und eine umfassende Migrationsstrategie zu entwickeln, die die Geschäftskontinuität während der Zerlegung gewährleistet.

Wenn Sie diese Komponenten und Herausforderungen gründlich verstehen, können Sie Ihre Modernisierung besser planen und fundierte Entscheidungen darüber treffen, welche Elemente Sie bei der Migration zu einer Microservices-basierten Architektur priorisieren sollten.

Erstellen Sie bei der Analyse von Datenbankcodekomponenten eine umfassende Dokumentation für jede gespeicherte Prozedur, jeden Trigger und jede Funktion. Beschreiben Sie zunächst klar ihren Zweck und ihre Kernfunktionen, einschließlich der implementierten Geschäftsregeln. Erläutern Sie alle Eingabe- und Ausgabeparameter und notieren Sie sich ihre Datentypen und gültigen Bereiche. Ordnen Sie Abhängigkeiten von anderen Datenbankobjekten, externen Systemen und nachgelagerten Prozessen zu. Definieren Sie klar die Transaktionsgrenzen und Isolationsanforderungen, um die Datenintegrität aufrechtzuerhalten. Dokumentieren Sie alle Leistungserwartungen, einschließlich der Anforderungen an die Reaktionszeit und der Muster der Ressourcennutzung. Analysieren Sie abschließend die Nutzungsmuster, um Spitzenlasten, Ausführungshäufigkeit und kritische Geschäftsperioden zu verstehen.

Phase 2: Klassifizierung der Geschäftslogik

Eine effektive Datenbankzerlegung erfordert eine systematische Kategorisierung der Datenbanklogik nach Schlüsseldimensionen: Komplexität, geschäftliche Auswirkungen, Abhängigkeiten, Nutzungsmuster und Migrationsschwierigkeiten. Diese Klassifizierung hilft Ihnen dabei, Komponenten mit hohem Risiko zu identifizieren, Testanforderungen zu ermitteln und Migrationsprioritäten festzulegen. Beispielsweise erfordern komplexe gespeicherte Prozeduren mit großen geschäftlichen Auswirkungen und häufiger Nutzung eine sorgfältige Planung und umfangreiche Tests. Einfache, selten verwendete Funktionen mit minimalen Abhängigkeiten könnten sich jedoch für frühe Migrationsphasen eignen.

Dieser strukturierte Ansatz schafft einen ausgewogenen Migrationsplan, der Geschäftsunterbrechungen minimiert und gleichzeitig die Systemstabilität gewährleistet. Wenn Sie diese Zusammenhänge verstehen, können Sie die Reihenfolge Ihrer Zerlegungsmaßnahmen verbessern und Ressourcen angemessen zuweisen.

Phase 3: Migration der Geschäftslogik

Nachdem Sie Ihre Geschäftslogik analysiert und klassifiziert haben, ist es an der Zeit, sie zu migrieren. Bei der Migration von Geschäftslogik aus einer monolithischen Datenbank gibt es zwei Ansätze: Verschieben Sie die Datenbanklogik auf die Anwendungsebene oder verschieben Sie die Geschäftslogik in eine andere Datenbank, die Teil des Microservices ist.

Wenn Sie die Geschäftslogik zur Anwendung migrieren, speichern die Datenbanktabellen nur die Daten, und die Datenbank enthält keine Geschäftslogik. Dies ist der empfohlene Ansatz. Sie können Ispirer oder generative KI-Tools wie Amazon Q Developer oder verwenden Kiro, um die Datenbank-Geschäftslogik für die Anwendungsebene zu konvertieren, z. B. die Konvertierung in Java. Weitere Informationen finden Sie unter Migrieren von Geschäftslogik von der Datenbank zur Anwendung für schnellere Innovation und Flexibilität (AWS Blogbeitrag).

Wenn Sie die Geschäftslogik in eine andere Datenbank migrieren, können Sie AWS Schema Conversion Tool (AWS SCT) verwenden, um vorhandene Datenbankschemas und Codeobjekte in Ihre Zieldatenbank zu konvertieren. Es unterstützt speziell entwickelte AWS Datenbankdienste wie Amazon DynamoDB, AmazonAurora und Amazon Redshift. Durch die Bereitstellung eines umfassenden Bewertungsberichts und automatisierter Konvertierungsfunktionen wird der AWS SCT Übergangsprozess optimiert, sodass Sie sich auf die Optimierung Ihrer neuen Datenbankstruktur konzentrieren können, um Leistung und Skalierbarkeit zu verbessern. AWS SCT Kann im Verlauf

Ihres Modernisierungsprojekts schrittweise Konvertierungen durchführen, um einen schrittweisen Ansatz zu unterstützen, sodass Sie jeden Schritt Ihrer Datenbanktransformation validieren und optimieren können.

Rollback-Strategie für Geschäftslogik

Zwei wichtige Aspekte jeder Zerlegungsstrategie sind die Wahrung der Abwärtskompatibilität und die Implementierung umfassender Rollback-Verfahren. Diese Elemente tragen zusammen dazu bei, den Betrieb während der Übergangsphase zu schützen. In diesem Abschnitt wird beschrieben, wie die Kompatibilität während des Zerlegungsprozesses gewährleistet und effektive Rollback-Funktionen für Notfälle eingerichtet werden können, die vor potenziellen Problemen schützen.

Wahrung der Abwärtskompatibilität

Bei der Zerlegung der Datenbank ist die Aufrechterhaltung der Abwärtskompatibilität für reibungslose Übergänge unerlässlich. Behalten Sie die bestehenden Datenbankprozeduren vorübergehend bei und implementieren Sie gleichzeitig schrittweise neue Funktionen. Verwenden Sie die Versionskontrolle, um alle Änderungen nachzuverfolgen und mehrere Datenbankversionen gleichzeitig zu verwalten. Planen Sie einen längeren Koexistenzzeitraum ein, in dem sowohl das Quell- als auch das Zielsystem zuverlässig funktionieren müssen. Auf diese Weise haben Sie Zeit, das neue System zu testen und zu validieren, bevor ältere Komponenten ausgemustert werden. Dieser Ansatz minimiert Betriebsunterbrechungen und bietet ein Sicherheitsnetz für Rollbacks, falls erforderlich.

Rollback-Plan für Notfälle

Eine umfassende Rollback-Strategie ist für eine sichere Datenbankzerlegung unerlässlich. Implementieren Sie Feature-Flags in Ihrem Code, um zu kontrollieren, welche Version der Geschäftslogik aktiv ist. Auf diese Weise können Sie sofort zwischen der neuen und der ursprünglichen Implementierung wechseln, ohne Änderungen an der Bereitstellung vornehmen zu müssen. Dieser Ansatz bietet eine detaillierte Kontrolle über den Übergang und hilft Ihnen, bei Problemen schnell ein Rollback durchzuführen. Behalten Sie die ursprüngliche Logik als verifiziertes Backup bei und führen Sie detaillierte Rollback-Verfahren ein, in denen Auslöser, Zuständigkeiten und Wiederherstellungsschritte festgelegt sind.

Testen Sie diese Rollback-Szenarien regelmäßig unter verschiedenen Bedingungen, um ihre Wirksamkeit zu überprüfen und sicherzustellen, dass die Teams mit den Notfallmaßnahmen vertraut sind. Feature-Flags ermöglichen auch eine schrittweise Einführung, indem neue Funktionen selektiv

für bestimmte Benutzergruppen oder Transaktionen aktiviert werden. Dies bietet eine zusätzliche Ebene der Risikominderung während der Umstellung.

Entkopplung von Tabellenbeziehungen während der Datenbankzerlegung

Dieser Abschnitt enthält Anleitungen zum Aufschlüsseln komplexer Tabellenbeziehungen und JOIN-Operationen bei der Zerlegung monolithischer Datenbanken. Eine Tabellenverknüpfung kombiniert Zeilen aus zwei oder mehr Tabellen auf der Grundlage einer zugehörigen Spalte zwischen ihnen. Das Ziel der Trennung dieser Beziehungen besteht darin, die hohe Kopplung zwischen Tabellen zu reduzieren und gleichzeitig die Datenintegrität zwischen Microservices aufrechtzuerhalten.

In diesem Abschnitt werden folgende Themen behandelt:

- Strategie zur Denormalisierung
- Reference-by-key Strategie
- CQRS-Muster
- Ereignisbasierte Datensynchronisierung
- Implementierung von Alternativen zu Tabellenverknüpfungen
- Szenariobasiertes Beispiel

Strategie zur Denormalisierung

Die Denormalisierung ist eine Strategie für den Datenbankentwurf, bei der bewusst Redundanz eingeführt wird, indem Daten tabellenübergreifend kombiniert oder dupliziert werden. Bei der Aufteilung einer großen Datenbank in kleine Datenbanken kann es sinnvoll sein, einige Daten dienstübergreifend zu duplizieren. Wenn beispielsweise grundlegende Kundendaten wie Name und E-Mail-Adressen sowohl in einem Marketingservice als auch in einem Bestellservice gespeichert werden, entfällt die Notwendigkeit, ständig serviceübergreifende Abfragen durchzuführen. Der Marketingservice benötigt möglicherweise Kundenpräferenzen und Kontaktinformationen für die Kampagnenausrichtung, während der Bestellservice dieselben Daten für die Auftragsabwicklung und Benachrichtigungen benötigt. Dies führt zwar zu einer gewissen Datenredundanz, kann aber die Serviceleistung und Unabhängigkeit erheblich verbessern, sodass das Marketingteam seine Kampagnen durchführen kann, ohne auf Kundenanfragen in Echtzeit angewiesen zu sein.

Konzentrieren Sie sich bei der Implementierung der Denormalisierung auf Felder, auf die häufig zugegriffen wird und die Sie anhand einer sorgfältigen Analyse der Datenzugriffsmuster identifizieren. Sie können Tools wie Oracle AWR Berichte oder verwenden, um zu verstehenpg_stat_statements,

welche Daten häufig zusammen abgerufen werden. Fachexperten können auch wertvolle Einblicke in natürliche Datengruppierungen geben. Denken Sie daran, dass Denormalisierung kein all-or-nothing Ansatz ist, sondern nur doppelte Daten, die nachweislich die Systemleistung verbessern oder komplexe Abhängigkeiten reduzieren.

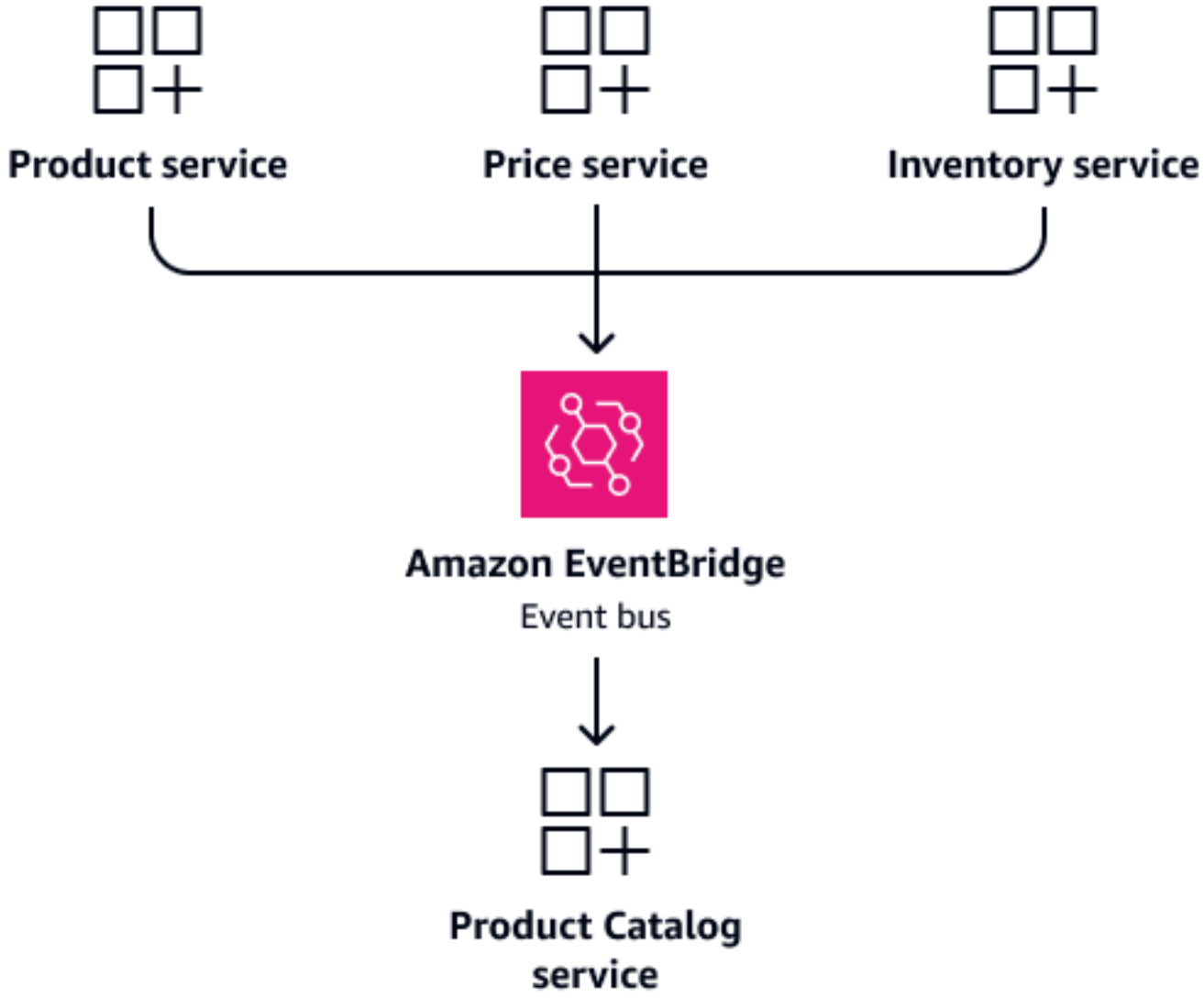
Reference-by-key Strategie

Eine reference-by-key Strategie ist ein Datenbankentwurfsmuster, bei dem Beziehungen zwischen Entitäten durch eindeutige Schlüssel aufrechterhalten werden, anstatt die eigentlichen zugehörigen Daten zu speichern. Anstatt herkömmlicher Fremdschlüsselbeziehungen speichern moderne Microservices oft nur die eindeutigen Identifikatoren verwandter Daten. Anstatt beispielsweise alle Kundendaten in der Bestelltabelle zu speichern, speichert der Bestellservice nur die Kundennummer und ruft bei Bedarf zusätzliche Kundeninformationen über einen API-Aufruf ab. Dieser Ansatz gewährleistet die Unabhängigkeit des Dienstes und gewährleistet gleichzeitig den Zugriff auf zugehörige Daten.

CQRS-Muster

Das CQRS-Muster (Command Query Responsibility Segregation) trennt die Lese- und Schreibvorgänge eines Datenspeichers. Dieses Muster ist besonders nützlich in komplexen Systemen mit Hochleistungsanforderungen, insbesondere solchen mit asymmetrischen read/write Lasten. Wenn Ihre Anwendung häufig Daten aus mehreren Quellen kombiniert benötigt, können Sie anstelle komplexer Verknüpfungen ein spezielles CQRS-Modell erstellen. Anstatt beispielsweise `Inventory` Tabellen bei jeder Anfrage zu `Pricing` verknüpfen`Product`, sollten Sie eine konsolidierte `Product Catalog` Tabelle verwalten, die die erforderlichen Daten enthält. Die Vorteile dieses Ansatzes können die Kosten der zusätzlichen Tabelle überwiegen.

Stellen Sie sich ein Szenario vor `ProductPrice`, in dem, und `Inventory` Dienste häufig Produktinformationen benötigen. Anstatt diese Dienste so zu konfigurieren, dass sie direkt auf gemeinsam genutzte Tabellen zugreifen, sollten Sie einen speziellen `Product Catalog` Dienst erstellen. Dieser Dienst unterhält eine eigene Datenbank, die die konsolidierten Produktinformationen enthält. Er dient als zentrale Informationsquelle für produktbezogene Anfragen. Wenn sich Produktdetails, Preise oder Lagerbestände ändern, können die jeweiligen Dienste Ereignisse veröffentlichen, um den `Product Catalog` Service zu aktualisieren. Dies sorgt für Datenkonsistenz und gewährleistet gleichzeitig die Unabhängigkeit der Dienste. Die folgende Abbildung zeigt diese Konfiguration, bei der Amazon als Event-Bus EventBridge dient.



Wie im nächsten Abschnitt beschriebenEreignisbasierte Datensynchronisierung, sollten Sie das CQRS-Modell anhand von Ereignissen auf dem neuesten Stand halten. Wenn sich Produktdetails, Preise oder Lagerbestände ändern, veröffentlichen die jeweiligen Dienste Ereignisse. Der `Product Catalog` Service abonniert diese Ereignisse und aktualisiert seine konsolidierte Ansicht. Dies ermöglicht schnelle Lesevorgänge ohne komplexe Verknüpfungen und gewährleistet die Unabhängigkeit des Dienstes.

Ereignisbasierte Datensynchronisierung

Die ereignisbasierte Datensynchronisierung ist ein Muster, bei dem Änderungen an Daten erfasst und als Ereignisse weitergegeben werden, sodass verschiedene Systeme oder Komponenten synchronisierte Datenzustände beibehalten können. Wenn sich Daten ändern, veröffentlichen Sie

ein Ereignis, um abonnierte Dienste zu benachrichtigen, anstatt alle zugehörigen Datenbanken sofort zu aktualisieren. Wenn ein Kunde beispielsweise seine Lieferadresse im Service ändert, führt ein `CustomerUpdated` Ereignis dazu, dass der `Customer` Service und der `Order` Service entsprechend dem Zeitplan der einzelnen `Delivery` Services aktualisiert werden. Bei diesem Ansatz werden starre Tabellenverknüpfungen durch flexible, skalierbare, ereignisgesteuerte Updates ersetzt. Einige Dienste verfügen möglicherweise kurzzeitig über veraltete Daten, der Nachteil besteht jedoch in einer verbesserten Systemskalierbarkeit und Dienstunabhängigkeit.

Implementierung von Alternativen zu Tabellenverknüpfungen

Beginnen Sie Ihre Datenbankzerlegung mit Lesevorgängen, da diese in der Regel einfacher zu migrieren und zu validieren sind. Sobald die Lesepfade stabil sind, können Sie die komplexeren Schreibvorgänge in Angriff nehmen. Bei kritischen Hochleistungsanforderungen sollten Sie die Implementierung des CQRS-Musters in Betracht ziehen. Verwenden Sie eine separate, optimierte Datenbank für Lesevorgänge und verwalten Sie eine weitere Datenbank für Schreibvorgänge.

Bauen Sie belastbare Systeme auf, indem Sie Wiederholungslogik für dienstübergreifende Aufrufe hinzufügen und entsprechende Caching-Ebenen implementieren. Überwachen Sie die Serviceinteraktionen genau und richten Sie Warnmeldungen für Probleme mit der Datenkonsistenz ein. Das Endziel ist nicht überall perfekte Konsistenz — es geht darum, unabhängige Dienste zu schaffen, die eine gute Leistung erbringen und gleichzeitig eine akzeptable Datengenauigkeit für Ihre Geschäftsanforderungen gewährleisten.

Der entkoppelte Charakter von Microservices führt zu folgenden neuen Komplexitäten im Datenmanagement:

- Daten werden verteilt. Daten befinden sich jetzt in separaten Datenbanken, die von unabhängigen Diensten verwaltet werden.
- Die Echtzeitsynchronisation zwischen Diensten ist oft nicht praktikabel und erfordert letztendlich ein Konsistenzmodell.
- Operationen, die zuvor innerhalb einer einzigen Datenbanktransaktion stattfanden, umfassen jetzt mehrere Dienste.

Gehen Sie wie folgt vor, um diesen Herausforderungen zu begegnen:

- Implementieren Sie eine ereignisgesteuerte Architektur — Verwenden Sie Nachrichtenwarteschlangen und die Veröffentlichung von Ereignissen, um Datenänderungen

dienstübergreifend zu verbreiten. Weitere Informationen finden Sie unter Aufbau ereignisgesteuerter Architekturen auf serverlosem Land.

- Verwenden Sie das Saga-Orchestrierungsmuster — Dieses Muster hilft Ihnen, verteilte Transaktionen zu verwalten und die Datenintegrität zwischen den Diensten aufrechtzuerhalten. Weitere Informationen finden Sie in Blogs unter Erstellen einer serverlosen verteilten Anwendung mithilfe eines Saga-Orchestrierungsmusters. AWS
- Design for Failure — Integrieren Sie Wiederholungsmechanismen, Schutzschalter und Ausgleichstransaktionen, um Netzwerkprobleme oder Dienstausfälle zu beheben.
- Verwenden Sie Versionsstempel — Verfolgen Sie Datenversionen, um Konflikte zu lösen und sicherzustellen, dass die neuesten Updates angewendet werden.
- Regelmäßiger Abgleich — Implementieren Sie regelmäßige Datensynchronisierungsprozesse, catch etwaige Inkonsistenzen aufzudecken und zu korrigieren.

Szenariobasiertes Beispiel

Das folgende Schemabeispiel enthält zwei Tabellen, eine `Customer` Tabelle und eine Tabelle: `Order`

```
-- Customer table
CREATE TABLE customer (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    email VARCHAR(255),
    phone VARCHAR(20),
    address TEXT,
    created_at TIMESTAMP
);

-- Order table
CREATE TABLE order (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date TIMESTAMP,
    total_amount DECIMAL(10,2),
    status VARCHAR(50),
    FOREIGN KEY (customer_id) REFERENCES customers(id)
);
```

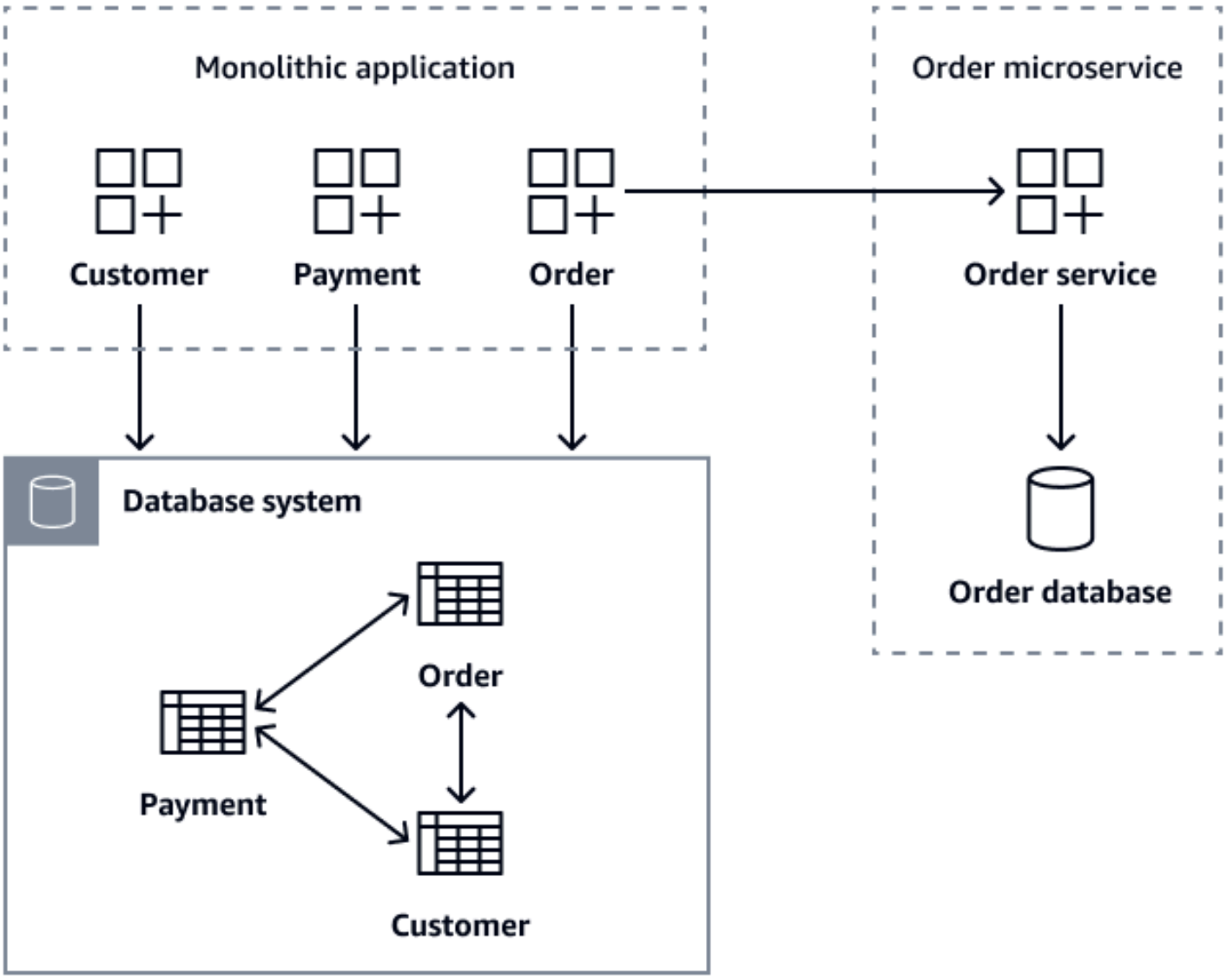
Das Folgende ist ein Beispiel dafür, wie Sie einen denormalisierten Ansatz verwenden könnten:

```
CREATE TABLE order (
    order_id INT PRIMARY KEY,
    customer_id INT,                   -- Reference only
    customer_first_name VARCHAR(100), -- Denormalized
    customer_last_name VARCHAR(100),  -- Denormalized
    customer_email VARCHAR(255),      -- Denormalized
    order_date TIMESTAMP,
    total_amount DECIMAL(10,2),
    status VARCHAR(50)
);
```

Die neue `Order` Tabelle enthält denormalisierte Kundennamen und E-Mail-Adressen. Es `customer_id` wird auf die verwiesen, und es gibt keine Fremdschlüsseleinschränkung für die Tabelle. `Customer` Im Folgenden sind die Vorteile dieses denormalisierten Ansatzes aufgeführt:

- Der `Order` Service kann die Bestellhistorie mit Kundendetails anzeigen und benötigt keine API-Aufrufe an den Microservice. `Customer`
- Wenn der `Customer` Dienst nicht verfügbar ist, ist der `Order` Dienst weiterhin voll funktionsfähig.
- Anfragen für die Auftragsabwicklung und Berichterstattung laufen schneller.

Das folgende Diagramm zeigt eine monolithische Anwendung, die Bestelldaten mithilfe von`getOrder(customer_id)`, `getOrder(order_id)``getCustomerOders(customer_id)`, und `createOrder(Order order)` API-Aufrufen an den `Order` Microservice abruft.



Während der Microservices-Migration können Sie die `Order` Tabelle in der monolithischen Datenbank als vorübergehende Sicherheitsmaßnahme beibehalten und so sicherstellen, dass die Legacy-Anwendung weiterhin funktionsfähig bleibt. Es ist jedoch entscheidend, dass alle neuen auftragsbezogenen Operationen über die `Order` Microservice-API geleitet werden, die ihre eigene Datenbank verwaltet und gleichzeitig als Backup in die alte Datenbank schreibt. Dieses Dual-Write-Muster bietet ein Sicherheitsnetz. Es ermöglicht eine schrittweise Migration unter Beibehaltung der Systemstabilität. Nachdem alle Kunden erfolgreich auf den neuen Microservice migriert haben, können Sie die `Order` Legacy-Tabelle in der monolithischen Datenbank als veraltet markieren. Nach der Zerlegung der monolithischen Anwendung und ihrer Datenbank in separate `Customer` und `Order` Microservices wird die Aufrechterhaltung der Datenkonsistenz zur größten Herausforderung.

Bewährte Methoden für die Datenbankzerlegung

Bei der Zerlegung einer monolithischen Datenbank müssen Unternehmen klare Rahmenbedingungen für die Nachverfolgung von Fortschritten, die Aufrechterhaltung des Systemwissens und die Bewältigung neuer Herausforderungen einrichten. In diesem Abschnitt finden Sie bewährte Verfahren zur Erfolgsmessung bei der Zerlegung, zur Pflege wichtiger Unterlagen, zur Implementierung kontinuierlicher Verbesserungsprozesse und zur Bewältigung häufiger Herausforderungen. Wenn Sie diese Richtlinien verstehen und befolgen, können Sie sicherstellen, dass die Bemühungen zur Datenbankzerlegung ihren beabsichtigten Nutzen bringen und gleichzeitig Betriebsunterbrechungen und technische Schulden minimieren.

In diesem Abschnitt werden folgende Themen behandelt:

- Erfolgsmessung
- Anforderungen an die Dokumentation
- Strategie zur kontinuierlichen Verbesserung
- Überwindung häufiger Herausforderungen bei der Zerlegung von Datenbanken

Erfolgsmessung

Verfolgen Sie den Erfolg der Zerlegung anhand einer Mischung aus technischen, betrieblichen und geschäftlichen Kennzahlen. Überwachen Sie technisch gesehen die Antwortzeiten von Anfragen, die Verbesserung der Systemverfügbarkeit und die Erhöhung der Bereitstellungshäufigkeit. Messen Sie betrieblich die Reduzierung von Zwischenfällen, die Geschwindigkeit der Problemlösung und die Verbesserung der Ressourcennutzung. Bei der Entwicklung sollten Sie die Geschwindigkeit der Funktionsimplementierung, die Beschleunigung des Release-Zyklus und die Verringerung teamübergreifender Abhängigkeiten verfolgen. Die Auswirkungen auf das Geschäft sollten zu niedrigeren Betriebskosten, einer schnelleren time-to-market und besseren Kundenzufriedenheit führen. Diese Kennzahlen werden häufig während der Umfangsphase definiert. Weitere Informationen finden Sie in diesem Handbuch unter Definition des Umfangs und der Anforderungen für die Datenbankzerlegung.

Anforderungen an die Dokumentation

Pflegen Sie die Dokumentation zur up-to-date Systemarchitektur mit klaren Dienstgrenzen, Datenflüssen und Schnittstellenspezifikationen. Verwenden Sie Aufzeichnungen über

Architekturentscheidungen (ADRs), um wichtige technische Entscheidungen zu erfassen, einschließlich ihres Kontextes, ihrer Konsequenzen und der in Betracht gezogenen Alternativen. Dokumentieren Sie beispielsweise, warum bestimmte Dienste zuerst getrennt wurden oder wie bestimmte Kompromisse bei der Datenkonsistenz eingegangen wurden.

Planen Sie monatliche Architekturprüfungen ein, um den Systemzustand anhand wichtiger Kennzahlen zu bewerten: Leistungstrends, Einhaltung von Sicherheitsbestimmungen und dienstübergreifende Abhängigkeiten. Fügen Sie Feedback von Entwicklungsteams zu Integrationsherausforderungen und betrieblichen Problemen hinzu. Dieser regelmäßige Überprüfungszyklus hilft Ihnen dabei, neu auftretende Probleme frühzeitig zu erkennen, und bestätigt, dass die Bemühungen zur Zerlegung weiterhin mit den Geschäftszielen im Einklang stehen.

Strategie zur kontinuierlichen Verbesserung

Behandeln Sie die Datenbankzerlegung als iterativen Prozess und nicht als einmaliges Projekt. Überwachen Sie Systemleistungskennzahlen und Serviceinteraktionen, um Optimierungsmöglichkeiten zu identifizieren. Priorisieren Sie in jedem Quartal die Behebung technischer Schulden auf der Grundlage der betrieblichen Auswirkungen und der Wartungskosten. Automatisieren Sie beispielsweise häufig ausgeführte Datenbankoperationen, verbessern Sie die Überwachungsabdeckung und verfeinern Sie die Bereitstellungsverfahren auf der Grundlage erlernter Muster.

Überwindung häufiger Herausforderungen bei der Zerlegung von Datenbanken

Die Leistungsoptimierung erfordert einen vielschichtigen Ansatz. Implementieren Sie strategisches Caching an Dienstgrenzen, optimieren Sie Abfragemuster auf der Grundlage der tatsächlichen Nutzung und überwachen Sie kontinuierlich wichtige Kennzahlen. Gehen Sie proaktiv auf Leistungsengpässe ein, indem Sie Trends analysieren und klare Schwellenwerte für Interventionen festlegen.

Probleme mit der Datenkonsistenz erfordern sorgfältige Architekturentscheidungen. Implementieren Sie ereignisgesteuerte Muster für dienstübergreifende Updates und verwenden Sie Saga-Orchestrierungsmuster für komplexe Transaktionen. Definieren Sie klare Servicegrenzen und akzeptieren Sie eine eventuelle Konsistenz, sofern die Geschäftsanforderungen dies zulassen. Dieses Gleichgewicht zwischen Konsistenz und Serviceautonomie ist entscheidend für eine erfolgreiche Zerlegung.

Operative Exzellenz erfordert die Automatisierung von Routineaufgaben und standardisierte Verfahren für alle Dienste. Sorgen Sie für eine umfassende Überwachung mit klaren Alarmschwellen und investieren Sie in regelmäßige Teamschulungen zur Einführung neuer Muster und Tools. Dieser systematische Betriebsansatz fördert eine zuverlässige Servicebereitstellung bei gleichzeitiger Bewältigung der Komplexität.

Häufig gestellte Fragen zur Datenbankzerlegung

In diesem umfassenden FAQ-Bereich werden die häufigsten Fragen und Herausforderungen behandelt, mit denen Unternehmen bei der Durchführung von Projekten zur Datenbankzerlegung konfrontiert sind. Von der Definition des anfänglichen Umfangs und der Anforderungen bis hin zur Migration von gespeicherten Prozeduren bieten diese Fragen praktische Einblicke und strategische Ansätze, um Teams dabei zu unterstützen, ihre Datenbank erfolgreich zu modernisieren. Ganz gleich, ob Sie sich in der Planungsphase befinden oder Ihre Zerlegungsstrategie bereits umsetzen, diese Antworten können Ihnen helfen, häufig auftretende Fallstricke zu vermeiden und bewährte Verfahren zu implementieren, um optimale Ergebnisse zu erzielen.

In diesem Abschnitt werden folgende Themen behandelt:

- FAQs über die Definition von Umfang und Anforderungen
- FAQs über die Steuerung des Datenbankzugriffs
- FAQs über die Analyse von Kohäsion und Kopplung
- FAQs über die Migration der Geschäftslogik auf die Anwendungsebene

FAQs über die Definition von Umfang und Anforderungen

Im Definition des Umfangs und der Anforderungen für die Datenbankzerlegung Abschnitt dieses Handbuchs wird erläutert, wie Interaktionen analysiert, Abhängigkeiten zugeordnet und Erfolgskriterien festgelegt werden. In diesem FAQ-Bereich werden wichtige Fragen zur Festlegung und Verwaltung von Projektgrenzen behandelt. Ganz gleich, ob Sie es mit unklaren technischen Einschränkungen, widersprüchlichen Abteilungsanforderungen oder sich ändernden Geschäftsanforderungen zu tun haben, diese FAQs bieten praktische Hinweise zur Aufrechterhaltung eines ausgewogenen Ansatzes.

Dieser Abschnitt enthält die folgenden Fragen:

- Wie detailliert sollte die ursprüngliche Definition des Geltungsbereichs sein?
- Was ist, wenn ich nach dem Start des Projekts weitere Abhängigkeiten feststelle?
- Wie gehe ich mit Stakeholdern aus verschiedenen Abteilungen um, die widersprüchliche Anforderungen haben?
- Wie lassen sich technische Einschränkungen am besten beurteilen, wenn die Dokumentation schlecht oder veraltet ist?

- Wie bringe ich unmittelbare Geschäftsanforderungen mit langfristigen technischen Zielen in Einklang?
- Wie stelle ich sicher, dass ich wichtige Anforderungen von unauffälligen Stakeholdern nicht übersehe?
- Gelten diese Empfehlungen für monolithische Mainframe-Datenbanken?

Wie detailliert sollte die ursprüngliche Definition des Geltungsbereichs sein?

Gehen Sie von den Bedürfnissen Ihrer Kunden aus und definieren Sie den Projektumfang ausreichend detailliert, um Systemgrenzen und kritische Abhängigkeiten zu identifizieren und gleichzeitig die Flexibilität bei der Erfassung zu wahren. Ordnen Sie wichtige Elemente wie Systemschnittstellen, wichtige Stakeholder und wichtige technische Einschränkungen zu. Fangen Sie klein an, indem Sie einen begrenzten Teil des Systems mit geringem Risiko auswählen, der einen messbaren Nutzen bietet. Dieser Ansatz hilft Teams dabei, Strategien zu erlernen und anzupassen, bevor sie komplexere Komponenten in Angriff nehmen.

Dokumentieren Sie wichtige Geschäftsanforderungen, die den Aufwand für die Zerlegung vorantreiben, vermeiden Sie jedoch, Details zu spezifizieren, die sich während der Implementierung ändern könnten. Dieser ausgewogene Ansatz stellt sicher, dass die Teams mit Klarheit vorankommen können und gleichzeitig anpassungsfähig bleiben an neue Erkenntnisse und Herausforderungen, die sich im Zuge der Modernisierung ergeben.

Was ist, wenn ich nach dem Start des Projekts weitere Abhängigkeiten feststelle?

Rechnen Sie damit, im Laufe des Projekts weitere Abhängigkeiten aufzudecken. Führen Sie ein aktuelles Abhängigkeitsprotokoll und führen Sie regelmäßige Umfangsprüfungen durch, um die Auswirkungen auf Zeitpläne und Ressourcen zu bewerten. Implementieren Sie einen klaren Change-Management-Prozess und planen Sie Pufferzeit in die Projektpläne ein, um auf unerwartete Entdeckungen reagieren zu können. Das Ziel besteht nicht darin, Änderungen zu verhindern, sondern sie effektiv zu verwalten. Dies hilft den Teams, sich schnell anzupassen und gleichzeitig die Dynamik des Projekts aufrechtzuerhalten.

Wie gehe ich mit Stakeholdern aus verschiedenen Abteilungen um, die widersprüchliche Anforderungen haben?

Bewältigen Sie widersprüchliche Abteilungsanforderungen durch eine klare Priorisierung, die auf dem Geschäftswert und den Auswirkungen auf das System basiert. Sichern Sie sich die Unterstützung von Führungskräften, um wichtige Entscheidungen zu treffen und Konflikte schnell zu lösen. Vereinbaren Sie regelmäßige Treffen zur Abstimmung der Interessengruppen, um Kompromisse zu erörtern und Transparenz zu wahren. Dokumentieren Sie alle Entscheidungen und ihre Begründung, um eine klare Kommunikation zu fördern und die Dynamik des Projekts aufrechtzuerhalten. Konzentrieren Sie die Diskussionen auf quantifizierbare Geschäftsvorteile und nicht auf die Präferenzen der Abteilungen.

Wie lassen sich technische Einschränkungen am besten beurteilen, wenn die Dokumentation schlecht oder veraltet ist?

Kombinieren Sie bei schlechter Dokumentation traditionelle Analysen mit modernen KI-Tools. Verwenden Sie umfangreiche Sprachmodelle (LLMs), um Code-Repositorys, Protokolle und bestehende Dokumentation zu analysieren, um Muster und potenzielle Einschränkungen zu identifizieren. Befragen Sie erfahrene Entwickler und Datenbankarchitekten, um KI-Ergebnisse zu validieren und undokumentierte Einschränkungen aufzudecken. Setzen Sie Überwachungstools mit erweiterten KI-Fähigkeiten ein, um das Systemverhalten zu beobachten und potenzielle Probleme vorherzusagen.

Erstellen Sie kleine technische Experimente, die Ihre Annahmen bestätigen. Sie können KI-gestützte Testtools verwenden, um den Prozess zu beschleunigen. Dokumentieren Sie die Ergebnisse in einer Wissensdatenbank, die durch KI-gestützte Updates kontinuierlich erweitert werden kann. Erwägen Sie, Fachexperten für komplexe Bereiche hinzuzuziehen und nutzen Sie KI-Tools zur Paarprogrammierung, um deren Analyse- und Dokumentationsbemühungen zu beschleunigen.

Wie bringe ich unmittelbare Geschäftsanforderungen mit langfristigen technischen Zielen in Einklang?

Erstellen Sie eine schrittweise Projekt-Roadmap, die unmittelbare Geschäftsanforderungen mit langfristigen technischen Zielen in Einklang bringt. Identifizieren Sie frühzeitig schnelle Erfolge, die einen spürbaren Mehrwert bieten, sodass Sie das Vertrauen Ihrer Stakeholder stärken können. Unterteilen Sie die Aufschlüsselung in klare Meilensteine. Jeder von ihnen sollte messbare Geschäftsvorteile bieten und gleichzeitig architektonische Ziele erreichen. Sorgen Sie durch

regelmäßige Überprüfungen und Anpassungen der Roadmap für Flexibilität, um dringenden Geschäftsanforderungen gerecht zu werden.

Wie stelle ich sicher, dass ich wichtige Anforderungen von unauffälligen Stakeholdern nicht übersehe?

Ordnen Sie alle potenziellen Interessengruppen im gesamten Unternehmen zu, einschließlich nachgeschalteter Systembesitzer und indirekter Nutzer. Schaffen Sie durch strukturierte Interviews, Workshops und regelmäßige Überprüfungssitzungen mehrere Feedback-Kanäle. Entwickeln proof-of-concepts und erstellen Sie Prototypen, um Anforderungen greifbar zu machen und sinnvolle Diskussionen anzuregen. Ein einfaches Dashboard, das Systemabhängigkeiten aufzeigt, deckt beispielsweise häufig verborgene Stakeholder und Anforderungen auf, die zunächst nicht ersichtlich waren.

Führen Sie regelmäßige Validierungssitzungen mit sowohl lautstarken als auch stillen Stakeholdern durch und stellen Sie sicher, dass alle Perspektiven berücksichtigt werden. Kritische Erkenntnisse kommen oft von denjenigen, die dem Tagesgeschäft am nächsten stehen, und nicht von den lautesten Stimmen in den Planungssitzungen.

Gelten diese Empfehlungen für monolithische Mainframe-Datenbanken?

Die in diesem Leitfaden beschriebene Methodik gilt auch für die Zerlegung monolithischer Mainframe-Datenbanken. Die wichtigsten Herausforderungen bei diesen Datenbanken sind die Verwaltung der Anforderungen der verschiedenen Interessengruppen. Die Technologieempfehlungen in diesem Leitfaden könnten für monolithische Mainframe-Datenbanken gelten. Wenn der Mainframe über eine relationale Datenbank verfügt, z. B. eine OLTP-Datenbank (Online Transaction Processing), gelten viele der Empfehlungen. Für OLAP-Datenbanken (Online Analytical Processing), wie sie beispielsweise zur Generierung von Geschäftsberichten verwendet werden, gelten nur einige der Empfehlungen.

FAQs über die Steuerung des Datenbankzugriffs

Die Steuerung des Datenbankzugriffs mithilfe des Database Wrapper-Service Patterns wird im Steuerung des Datenbankzugriffs während der Zerlegung Abschnitt dieses Handbuchs behandelt. In diesem FAQ-Bereich werden häufig gestellte Bedenken und Fragen zur Einführung eines Datenbank-Wrapper-Service behandelt, einschließlich seiner potenziellen Auswirkungen auf die Leistung, den Umgang mit vorhandenen gespeicherten Prozeduren, die Verwaltung komplexer Transaktionen und die Überwachung von Schemaänderungen.

Dieser Abschnitt enthält die folgenden Fragen:

Wird der Wrapper-Service nicht zu einem neuen Engpass?

Der Datenbank-Wrapper-Service fügt zwar einen zusätzlichen Netzwerk-Hop hinzu, die Auswirkungen sind jedoch normalerweise minimal. Sie können den Service horizontal skalieren, und die Vorteile des kontrollierten Zugriffs überwiegen in der Regel die geringen Leistungskosten. Betrachten Sie es als einen vorübergehenden Kompromiss zwischen Leistung und Wartbarkeit.

Was passiert mit vorhandenen gespeicherten Prozeduren?

Anfänglich kann der Datenbank-Wrapper-Dienst gespeicherte Prozeduren als Dienstmethoden verfügbar machen. Im Laufe der Zeit können Sie die Logik schrittweise in die Anwendungsebene verlagern, wodurch Tests und Versionskontrolle verbessert werden. Migrieren Sie die Geschäftslogik schrittweise, um das Risiko zu minimieren.

Wie verwalte ich Schemaänderungen während der Umstellung?

Zentralisieren Sie die Kontrolle über Schemaänderungen durch das Wrapper-Serviceteam. Dieses Team ist dafür verantwortlich, für einen umfassenden Überblick über alle Verbraucher zu sorgen. Dieses Team überprüft die vorgeschlagenen Änderungen auf ihre systemweite Wirkung, stimmt sich mit den betroffenen Teams ab und implementiert die Änderungen mithilfe eines kontrollierten Implementierungsprozesses. Wenn beispielsweise neue Felder hinzugefügt werden, sollte dieses Team die Abwärtskompatibilität wahren, indem es Standardwerte implementiert oder zunächst Nullwerte zulässt.

Richten Sie einen klaren Prozess für das Änderungsmanagement ein, der Folgenabschätzungen, Testanforderungen und Rollback-Verfahren umfasst. Verwenden Sie Tools zur Datenbank-Versionierung und sorgen Sie für eine klare Dokumentation aller Änderungen. Dieser zentralisierte Ansatz verhindert, dass durch Schemaänderungen abhängige Dienste unterbrochen werden, und gewährleistet die Systemstabilität.

FAQs über die Analyse von Kohäsion und Kopplung

Das Verständnis und die effektive Analyse von Datenbankkopplung und Kohäsion sind für eine erfolgreiche Datenbankzerlegung von grundlegender Bedeutung. Kopplung und Kohäsion werden im Analyse von Kohäsion und Kopplung für die Datenbankzerlegung Abschnitt dieses Leitfadens erörtert. In diesem Abschnitt mit häufig gestellten Fragen werden wichtige Fragen zur Identifizierung geeigneter Granularitätsstufen, zur Auswahl der richtigen Analysetools, zur Dokumentation der Ergebnisse und zur Priorisierung von Kopplungsproblemen behandelt.

Dieser Abschnitt enthält die folgenden Fragen:

- Wie identifiziere ich bei der Kopplungsanalyse das richtige Maß an Granularität?
- Welche Tools kann ich verwenden, um Datenbankkopplung und Kohäsion zu analysieren?
- Wie lassen sich die Ergebnisse der Kopplung und Kohäsion am besten dokumentieren?
- Wie priorisiere ich, welche Kopplungsprobleme zuerst angegangen werden sollen?
- Wie gehe ich mit Transaktionen um, die sich über mehrere Operationen erstrecken?

Wie identifiziere ich bei der Kopplungsanalyse das richtige Maß an Granularität?

Beginnen Sie mit einer umfassenden Analyse der Datenbankbeziehungen und gehen Sie dann systematisch nach, um natürliche Trennungspunkte zu identifizieren. Verwenden Sie Datenbankanalysetools, um Beziehungen auf Tabellenebene, Schemaabhängigkeiten und Transaktionsgrenzen abzubilden. Untersuchen Sie beispielsweise Verbindungsmuster in SQL-Abfragen, um die Abhängigkeiten beim Datenzugriff zu verstehen. Sie können auch Transaktionsprotokolle analysieren, um die Grenzen von Geschäftsprozessen zu ermitteln.

Konzentrieren Sie sich auf Bereiche, in denen die Kopplung naturgemäß minimal ist. Diese orientieren sich häufig an den Grenzen der Geschäftsbereiche und stellen optimale Zerlegungspunkte dar. Bei der Festlegung geeigneter Dienstgrenzen sollten Sie sowohl die technische Kopplung (z. B. gemeinsam genutzte Tabellen und Fremdschlüssel) als auch die geschäftliche Kopplung (z. B. Prozessabläufe und Berichtsanforderungen) berücksichtigen.

Welche Tools kann ich verwenden, um Datenbankkopplung und Kohäsion zu analysieren?

Sie können eine Kombination aus automatisierten Tools und manueller Analyse verwenden, um die Datenbankkopplung und den Zusammenhalt zu bewerten. Die folgenden Tools können Ihnen bei dieser Bewertung helfen:

- Tools zur Schemavisualisierung — Sie können Tools wie SchemaSpyoder pgAdminzum Generieren von ER-Diagrammen verwenden. Diese Diagramme zeigen Tabellenbeziehungen und mögliche Kopplungspunkte.
- Tools zur Abfrageanalyse — Sie können pg_stat_statementsoder verwenden SQL Server Query Store, um häufig verknüpfte Tabellen und Zugriffsmuster zu identifizieren.
- Tools zur Datenbankprofilerstellung — Tools wie Oracle SQL Developeroder MySQL Workbenchbieten Einblicke in die Abfrageleistung und Datenabhängigkeiten.
- Tools zur Zuordnung von Abhängigkeiten — Mithilfe von AWS Schema Conversion Tool (AWS SCT) können Sie Schemabeziehungen visualisieren und eng miteinander verknüpfte Komponenten identifizieren. vFunctionkann Ihnen helfen, Domänengrenzen zu identifizieren, indem es die Funktions- und Domänengrenzen der Anwendung analysiert.
- Tools zur Transaktionsüberwachung — Sie können datenbankspezifische Tools wie Oracle Enterprise Manageroder verwenden, um SQL Server Extended EventsTransaktionsgrenzen zu analysieren.
- Migrationstools für Geschäftslogik — Sie können generative KI-Tools wie Amazon Q Developer Ispireroder verwenden Kiro, um die Datenbank-Geschäftslogik für die Anwendungsebene zu konvertieren, z. B. die Konvertierung in Java.

Kombinieren Sie diese automatisierten Analysen mit einer manuellen Überprüfung von Geschäftsprozessen und Fachwissen, um die Systemkopplung vollständig zu verstehen. Dieser facettenreiche Ansatz stellt sicher, dass sowohl technische als auch geschäftliche Gesichtspunkte in Ihrer Zerlegungsstrategie berücksichtigt werden.

Wie lassen sich die Ergebnisse der Kopplung und Kohäsion am besten dokumentieren?

Erstellen Sie eine umfassende Dokumentation, die Datenbankbeziehungen und Nutzungsmuster visualisiert. Die folgenden Arten von Ressourcen können Sie verwenden, um Ihre Ergebnisse aufzuzeichnen:

- Abhängigkeitsmatrizen — Ordnen Sie Tabellenabhängigkeiten zu und heben Sie Bereiche mit hoher Kopplung hervor.
- Beziehungsdiagramme — Verwenden Sie ER-Diagramme, um Schemaverbindungen und Fremdschlüsselbeziehungen darzustellen.
- Heatmaps zur Tabellennutzung — Visualisieren Sie die Abfragehäufigkeit und die Datenzugriffsmuster tabellenübergreifend.
- Transaktionsflussdiagramme — Dokumentieren Sie Transaktionen mit mehreren Tabellen und deren Grenzen.
- Karten der Domänengrenzen — Skizzieren Sie potenzielle Servicegrenzen auf der Grundlage von Geschäftsbereichen.

Kombinieren Sie diese Artefakte in einem Dokument und aktualisieren Sie es regelmäßig, wenn die Zerlegung voranschreitet. Für Diagramme können Sie Tools wie draw.iooder verwenden. Lucidchart Erwägen Sie die Implementierung eines Wikis für den einfachen Zugriff und die Zusammenarbeit im Team. Dieser facettenreiche Dokumentationsansatz sorgt für ein klares, gemeinsames Verständnis von Systemkopplung und Kohäsion.

Wie priorisiere ich, welche Kopplungsprobleme zuerst angegangen werden sollen?

Priorisieren Sie Kopplungsprobleme auf der Grundlage einer ausgewogenen Bewertung der geschäftlichen und technischen Faktoren. Bewerten Sie jedes Problem anhand der Auswirkungen auf das Geschäft (z. B. Umsatz und Kundenerlebnis), des technischen Risikos (wie Systemstabilität und Datenintegrität), des Implementierungsaufwands und der Fähigkeiten des Teams. Erstellen Sie eine Priorisierungsmatrix, in der jedes Problem anhand dieser Dimensionen mit 1—5 bewertet wird. Diese Matrix hilft Ihnen dabei, die wertvollsten Chancen mit überschaubaren Risiken zu identifizieren.

Beginnen Sie mit Änderungen mit hoher Wirkung und geringem Risiko, die auf das vorhandene Fachwissen des Teams abgestimmt sind. Auf diese Weise können Sie organisatorisches Selbstvertrauen und Dynamik für komplexere Änderungen aufbauen. Dieser Ansatz fördert eine realistische Umsetzung und maximiert den Geschäftswert. Überprüfen Sie die Prioritäten regelmäßig und passen Sie sie an, um sie an die sich ändernden Geschäftsanforderungen und Teamkapazitäten anzupassen.

Wie gehe ich mit Transaktionen um, die sich über mehrere Operationen erstrecken?

Bewältigen Sie Transaktionen mit mehreren Operationen durch eine sorgfältig konzipierte Service-Level-Koordination. Implementieren Sie Saga-Muster für komplexe verteilte Transaktionen. Teilen Sie sie in kleinere, umkehrbare Schritte auf, die unabhängig voneinander verwaltet werden können. Beispielsweise kann ein Ablauf der Auftragsabwicklung in separate Schritte für die Inventarprüfung, die Zahlungsabwicklung und die Auftragserstellung aufgeteilt werden, von denen jeder über einen eigenen Vergütungsmechanismus verfügt.

Wenn möglich, sollten die Abläufe so umgestaltet werden, dass sie atomarer sind, wodurch die Notwendigkeit verteilter Transaktionen reduziert wird. Wenn verteilte Transaktionen unvermeidlich sind, sollten robuste Nachverfolgungs- und Kompensationsmechanismen eingeführt werden, um die Datenkonsistenz zu fördern. Überwachen Sie die Abschlussquoten von Transaktionen und implementieren Sie klare Verfahren zur Fehlerbehebung, um die Zuverlässigkeit des Systems zu gewährleisten.

FAQs über die Migration der Geschäftslogik auf die Anwendungsebene

Die Migration der Geschäftslogik von der Datenbank zur Anwendungsebene ist ein kritischer und komplexer Aspekt der Datenbankmodernisierung. Diese Migration der Geschäftslogik wird im Migration der Geschäftslogik von der Datenbank zur Anwendungsebene Abschnitt dieses Handbuchs erörtert. In diesem FAQ-Bereich werden häufig gestellte Fragen zur effektiven Verwaltung dieser Umstellung behandelt, von der Auswahl der ersten Kandidaten für die Migration bis hin zum Umgang mit komplexen gespeicherten Prozeduren und Triggern.

Dieser Abschnitt enthält die folgenden Fragen:

- Wie identifiziere ich, welche gespeicherten Prozeduren zuerst migriert werden sollen?
- Was sind die Risiken einer Verlagerung der Logik auf die Anwendungsebene?
- Wie kann ich die Leistung aufrechterhalten, wenn ich Logik aus der Datenbank verlasse?
- Was sollte ich mit komplexen gespeicherten Prozeduren tun, die mehrere Tabellen beinhalten?
- Wie gehe ich mit Datenbankauslösern während der Migration um?
- Wie lässt sich die migrierte Geschäftslogik am besten testen?

- Wie verwalte ich den Übergangszeitraum, wenn sowohl Datenbank- als auch Anwendungslogik vorhanden sind?
- Wie gehe ich mit Fehlerszenarien in der Anwendungsebene um, die zuvor von der Datenbank verwaltet wurden?

Wie identifiziere ich, welche gespeicherten Prozeduren zuerst migriert werden sollen?

Identifizieren Sie zunächst gespeicherte Prozeduren, die die beste Kombination aus geringem Risiko und hohem Lernwert bieten. Konzentrieren Sie sich auf Verfahren mit minimalen Abhängigkeiten, klarer Funktionalität und unkritischen Auswirkungen auf das Geschäft. Diese eignen sich hervorragend für die erste Migration, da sie dem Team helfen, Selbstvertrauen aufzubauen und Muster zu etablieren. Wählen Sie beispielsweise Verfahren, die einfache Datenoperationen abwickeln, gegenüber Verfahren, die komplexe Transaktionen oder kritische Geschäftslogik verwalten.

Verwenden Sie Tools zur Datenbanküberwachung, um Nutzungsmuster zu analysieren und Verfahren, auf die selten zugegriffen wird, als frühe Kandidaten zu identifizieren. Dieser Ansatz minimiert das Geschäftsrisiko und bietet gleichzeitig wertvolle Erfahrungen für die spätere Bewältigung komplexerer Migrationen. Bewerten Sie jedes Verfahren nach Komplexität, geschäftlicher Wichtigkeit und Abhängigkeitsgrad, um eine priorisierte Migrationssequenz zu erstellen.

Was sind die Risiken einer Verlagerung der Logik auf die Anwendungsebene?

Die Verlagerung der Datenbanklogik auf die Anwendungsebene bringt mehrere wichtige Herausforderungen mit sich. Die Systemleistung kann sich aufgrund erhöhter Netzwerkaufrufe verschlechtern, insbesondere bei datenintensiven Vorgängen, die zuvor innerhalb der Datenbank abgewickelt wurden. Das Transaktionsmanagement wird immer komplexer und erfordert eine sorgfältige Koordination, um die Datenintegrität in verteilten Abläufen aufrechtzuerhalten. Die Sicherstellung der Datenkonsistenz wird zu einer Herausforderung, insbesondere bei Vorgängen, die zuvor auf Einschränkungen auf Datenbankebene beruhten.

Mögliche Betriebsunterbrechungen während der Migration und die Lernkurve für Entwickler geben ebenfalls Anlass zu großer Sorge. Mindern Sie diese Risiken durch gründliche Planung, umfangreiche Tests in gestaffelten Umgebungen und schrittweise Migration, die mit weniger

kritischen Komponenten beginnt. Implementieren Sie robuste Überwachungs- und Rollback-Verfahren, um Probleme in der Produktion schnell zu erkennen und zu beheben.

Wie kann ich die Leistung aufrechterhalten, wenn ich Logik aus der Datenbank verlasse?

Implementieren Sie geeignete Caching-Mechanismen für Daten, auf die häufig zugegriffen wird, optimieren Sie die Datenzugriffsmuster, um Netzwerkaufrufe zu minimieren, und verwenden Sie die Stapelverarbeitung für Massenvorgänge. Ziehen Sie bei non-time-critical Vorgängen die asynchrone Verarbeitung in Betracht, um die Reaktionsfähigkeit des Systems zu verbessern.

Überwachen Sie die Leistungskennzahlen der Anwendungen genau und passen Sie sie nach Bedarf an. Sie können beispielsweise mehrere einzeilige Operationen durch Massenverarbeitung ersetzen, Sie können Referenzdaten zwischenspeichern, die sich selten ändern, und Sie können Abfragemuster optimieren, um die Datenübertragung zu reduzieren. Regelmäßige Leistungstests und -optimierungen tragen dazu bei, dass das System akzeptable Reaktionszeiten beibehält und verbessert die Wartbarkeit und Skalierbarkeit.

Was sollte ich mit komplexen gespeicherten Prozeduren tun, die mehrere Tabellen beinhalten?

Gehen Sie komplexe gespeicherte Prozeduren mit mehreren Tabellen durch systematische Zerlegung an. Teilen Sie sie zunächst in kleinere, logisch kohärente Komponenten auf und identifizieren Sie klare Transaktionsgrenzen und Datenabhängigkeiten. Erstellen Sie Serviceschnittstellen für jede logische Komponente. Auf diese Weise können Sie schrittweise migrieren, ohne die bestehende Funktionalität zu beeinträchtigen.

Implementieren Sie eine step-by-step Migration, beginnend mit den am wenigsten gekoppelten Komponenten. Bei sehr komplizierten Verfahren sollten Sie erwägen, sie vorübergehend in der Datenbank zu belassen und gleichzeitig einfachere Teile zu migrieren. Dieser hybride Ansatz gewährleistet die Systemstabilität, während Sie Ihren architektonischen Zielen näher kommen. Überwachen Sie kontinuierlich Leistung und Funktionalität während der Migration und bereiten Sie sich darauf vor, Ihre Strategie auf der Grundlage der Ergebnisse anzupassen.

Wie gehe ich mit Datenbankauslösern während der Migration um?

Verwandeln Sie Datenbank-Trigger in Event-Handler auf Anwendungsebene und behalten Sie gleichzeitig die Systemfunktionalität bei. Ersetzen Sie synchrone Trigger durch ereignisgesteuerte

Muster, die Meldungen für asynchrone Operationen in die Warteschlangen stellen. Erwägen Sie, Amazon Simple Notification Service (Amazon SNS) oder Amazon Simple Queue Service (Amazon SQS) für die Nachrichtenwarteschlangen zu verwenden. Implementieren Sie für Prüfanforderungen die Protokollierung auf Anwendungsebene oder verwenden Sie Funktionen zur Erfassung von Datenbankänderungen (CDC).

Analysieren Sie den Zweck und die Wichtigkeit jedes Triggers. Einige Trigger lassen sich möglicherweise besser durch die Anwendungslogik bedienen, und bei anderen sind möglicherweise Muster zur Erfassung von Ereignissen erforderlich, um die Datenkonsistenz zu gewährleisten. Beginnen Sie mit einfachen Triggern wie Audit-Logs, bevor Sie sich mit komplexen Triggern befassen, die Geschäftsregeln oder die Datenintegrität verwalten. Überwachen Sie die Migration sorgfältig, um sicherzustellen, dass die Funktionalität oder Datenkonsistenz nicht beeinträchtigt wird.

Wie lässt sich die migrierte Geschäftslogik am besten testen?

Implementieren Sie einen mehrschichtigen Testansatz, bevor Sie die migrierte Geschäftslogik bereitstellen. Beginnen Sie mit Komponententests für neuen Anwendungscode und fügen Sie dann Integrationstests hinzu, die end-to-end Geschäftsabläufe abdecken. Führen Sie alte und neue Implementierungen parallel aus und vergleichen Sie dann die Ergebnisse, um die funktionale Gleichwertigkeit zu überprüfen. Führen Sie Leistungstests unter verschiedenen Lastbedingungen durch, um sicherzustellen, dass das Systemverhalten den vorherigen Funktionen entspricht oder diese übertrifft.

Verwenden Sie Feature-Flags, um die Bereitstellung zu steuern, sodass Sie bei Problemen schnell ein Rollback durchführen können. Binden Sie Geschäftsanwender in die Validierung ein, insbesondere bei kritischen Workflows. Überwachen Sie die wichtigsten Kennzahlen bei der ersten Implementierung und erhöhen Sie schrittweise den Traffic für die neue Implementierung. Behalten Sie die Möglichkeit bei, bei Bedarf zur ursprünglichen Datenbanklogik zurückzukehren.

Wie verwalte ich den Übergangszeitraum, wenn sowohl Datenbank- als auch Anwendungslogik vorhanden sind?

Wenn sowohl die Datenbank als auch die Anwendungslogik verwendet werden, implementieren Sie Feature-Flags, die den Datenfluss steuern und einen schnellen Wechsel zwischen alten und neuen Implementierungen ermöglichen. Sorgen Sie für eine strenge Versionskontrolle und dokumentieren Sie beide Implementierungen und ihre jeweiligen Verantwortlichkeiten klar und deutlich. Richten Sie eine umfassende Überwachung für beide Systeme ein, um Unstimmigkeiten oder Leistungsprobleme schnell zu erkennen.

Richten Sie klare Rollback-Verfahren für jede migrierte Komponente ein, sodass Sie bei Bedarf zur ursprünglichen Logik zurückkehren können. Kommunizieren Sie regelmäßig mit allen Beteiligten über den Status der Umstellung, mögliche Auswirkungen und Eskalationsverfahren. Dieser Ansatz hilft Ihnen bei der schrittweisen Migration unter Wahrung der Systemstabilität und des Vertrauens der Stakeholder.

Wie gehe ich mit Fehlerszenarien in der Anwendungsebene um, die zuvor von der Datenbank verwaltet wurden?

Ersetzen Sie die Fehlerbehandlung auf Datenbankebene durch robuste Mechanismen auf Anwendungsebene. Implementieren Sie Schutzschalter und Wiederholungslogik für vorübergehende Ausfälle. Verwenden Sie kompensierende Transaktionen, um die Datenkonsistenz in verteilten Abläufen aufrechtzuerhalten. Wenn beispielsweise eine Zahlungsaktualisierung fehlschlägt, sollte die Anwendung es innerhalb definierter Grenzen automatisch wiederholen und bei Bedarf Ausgleichsmaßnahmen einleiten.

Richten Sie umfassende Überwachungs- und Warnmeldungen ein, um Probleme schnell zu erkennen, und führen Sie detaillierte Prüfprotokolle zur Behebung von Problemen. Gestalten Sie die Fehlerbehandlung so automatisiert wie möglich und definieren Sie klare Eskalationspfade für Szenarien, die menschliches Eingreifen erfordern. Dieser vielschichtige Ansatz sorgt für Systemstabilität und gewährleistet gleichzeitig die Datenintegrität und die Kontinuität der Geschäftsprozesse.

Nächste Schritte für die Datenbankzerlegung auf AWS

Nach der Implementierung der ersten Strategien zur Zerlegung von Datenbanken mithilfe von Datenbank-Wrapper-Services und der Verlagerung der Geschäftslogik auf die Anwendungsebene müssen Unternehmen ihre nächste Entwicklung planen. In diesem Abschnitt werden die wichtigsten Überlegungen zur Fortsetzung Ihrer Modernisierung dargelegt.

In diesem Abschnitt werden folgende Themen behandelt:

- Inkrementelle Strategien für die Zerlegung von Datenbanken
- Technische Überlegungen für verteilte Datenbankumgebungen
- Organisatorische Änderungen zur Unterstützung verteilter Architekturen

Inkrementelle Strategien für die Zerlegung von Datenbanken

Die Zerlegung von Datenbanken folgt einer schrittweisen Entwicklung in drei verschiedenen Phasen. Die Teams versehen die monolithische Datenbank zunächst mit einem Datenbank-Wrapper-Service, um den Zugriff zu kontrollieren. Anschließend beginnen sie, die Daten in dienstspezifische Datenbanken aufzuteilen und gleichzeitig die Primärdatenbank für ältere Anforderungen beizubehalten. Schließlich schließen sie die Migration der Geschäftslogik ab, um auf vollständig unabhängige Servicedatenbanken umzustellen.

Während dieser Reise müssen die Teams sorgfältige Datensynchronisierungsmuster implementieren und die Konsistenz zwischen den Diensten kontinuierlich überprüfen. Die Leistungsüberwachung wird entscheidend, um potenzielle Probleme frühzeitig zu erkennen und zu beheben. Da sich Dienste unabhängig voneinander weiterentwickeln, sollten ihre Schemas auf der Grundlage der tatsächlichen Nutzungsmuster optimiert werden, und Sie sollten redundante Strukturen entfernen, die sich im Laufe der Zeit angesammelt haben.

Dieser schrittweise Ansatz trägt dazu bei, Risiken zu minimieren und gleichzeitig die Systemstabilität während des gesamten Transformationsprozesses aufrechtzuerhalten.

Technische Überlegungen für verteilte Datenbankumgebungen

In einer verteilten Datenbankumgebung ist die Leistungsüberwachung unverzichtbar, um Engpässe frühzeitig zu erkennen und zu beheben. Teams müssen umfassende Überwachungssysteme und

Caching-Strategien implementieren, um das Leistungsniveau aufrechtzuerhalten. Read/write Durch die Aufteilung können die Lasten im System effektiv ausgeglichen werden.

Datenkonsistenz erfordert eine sorgfältige Orchestrierung über verteilte Dienste hinweg. Teams sollten gegebenenfalls Konsistenzmuster implementieren und klare Grenzen für den Datenbesitz festlegen. Eine robuste Überwachung fördert die Datenintegrität in allen Diensten.

Darüber hinaus muss die Sicherheit weiterentwickelt werden, um der verteilten Architektur Rechnung zu tragen. Jeder Service benötigt detaillierte Sicherheitskontrollen, und Ihre Zugriffsmuster müssen regelmäßig überprüft werden. Eine verbesserte Überwachung und Prüfung wird in dieser verteilten Umgebung immer wichtiger.

Organisatorische Änderungen zur Unterstützung verteilter Architekturen

Die Teamstruktur sollte sich an den Dienstgrenzen orientieren, um klare Verantwortlichkeiten und Verantwortlichkeiten zu definieren. Organizations müssen neue Kommunikationsmuster etablieren und zusätzliche technische Fähigkeiten innerhalb von Teams aufbauen. Diese Struktur sollte sowohl die Wartung vorhandener Dienste als auch Ihre kontinuierliche architektonische Weiterentwicklung unterstützen.

Sie müssen Ihre Betriebsprozesse aktualisieren, um die verteilte Architektur handhaben zu können. Teams müssen die Bereitstellungsverfahren ändern, die Prozesse zur Reaktion auf Vorfälle anpassen und die Verfahren für das Änderungsmanagement weiterentwickeln, um sie über mehrere Dienste hinweg zu koordinieren.

Ressourcen

Die folgenden zusätzlichen Ressourcen und Tools können Ihrem Unternehmen bei der Zerlegung von Datenbanken helfen.

AWS Präskriptive Leitlinien

- Migrieren von Datenbanken Oracle auf das AWS Cloud
- Optionen zur Neuplattformierung für ein Oracle DatabaseAWS
- Entwurfsmuster, Architekturen und Implementierungen der Cloud

AWS Blog-Beiträge

- Migrieren Sie die Geschäftslogik von der Datenbank zur Anwendung, um Innovationen und Flexibilität zu beschleunigen

AWS-Services

- AWS Application Migration Service
- AWS Database Migration Service (AWS DMS)
- Migration Evaluator
- AWS Schema Conversion Tool (AWS SCT)
- AWS Transform

Andere Tools

- AppEngine (Dynatrace Website)
- Oracle Automatic Workload Repository (Oracle Website)
- CAST Imaging (CAST Website)
- Kiro (Kiro Website)
- pgAdmin (pgAdmin Website)
- pg_stat_statements (PostgreSQL Website)

- SchemaSpy (SchemaSpy Website)
- SQL Developer (Oracle Website)
- SQLWays (Ispirer Website)
- vFunction (vFunction Website)

Sonstige Ressourcen

- Von Monolith zu Microservices (Website) O'Reilly

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
Häufig gestellte Fragen zu Mainframes und KI-Tools	Wir haben Folgendes hinzugefügt: Gelten diese Empfehlungen für monolithische Mainframe-Datenbanken? Häufig gestellte Fragen, und wir haben zusätzliche Informationen zu KI-Tools hinzugefügt, die Sie bei der Datenbankzerlegung verwenden können.	14. Oktober 2025
Erste Veröffentlichung	—	30. September 2025

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.

autoritative Datenquelle

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

Availability Zone

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

AWS Framework für die 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, wie 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 Benutzeranmeldeinformationen, die Sie in verschlüsselter Form speichern. Es besteht aus dem geheimen Wert und seinen Metadaten. Der geheime Wert kann binär, eine einzelne Zeichenfolge oder mehrere Zeichenketten sein. Weitere Informationen finden Sie unter Was ist in einem Secrets Manager Manager-Geheimnis? in der Secrets Manager Manager-Dokumentation.

Sicherheit durch Design

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

Sicherheitskontrolle

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

Härtung der Sicherheit

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

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

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

Automatisierung von Sicherheitsreaktionen

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

Serverseitige Verschlüsselung

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

Service-Kontrollrichtlinie (SCP)

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

Service-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.