Verständnis und Implementierung von Mikrofrontends auf AWS

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Verständnis und Implementierung von Mikrofrontends auf AWS

Table of Contents

Mikrofrontends verstehen und implementieren auf AWS

Amazon Web Services (Mitwirkende)

Juli 2024 (Verlauf der Dokumente)

Da Unternehmen nach Agilität und Skalierbarkeit streben, wird die konventionelle monolithische Architektur oft zu einem Engpass, der eine schnelle Entwicklung und Implementierung behindert. Mikro-Frontends mildern dieses Problem, indem sie komplexe Benutzeroberflächen in kleinere, unabhängige Komponenten aufteilen, die eigenständig entwickelt, getestet und bereitgestellt werden können. Dieser Ansatz verbessert die Effizienz der Entwicklungsteams und erleichtert die Zusammenarbeit zwischen Backend und Frontend, wodurch eine Abstimmung verteilter Systeme gefördert wird. end-to-end

Diese präskriptiven Leitlinien sind darauf zugeschnitten, IT-Führungskräften, Produkteigentümern und Architekten verschiedener Fachbereiche dabei zu helfen, die Mikro-Frontend-Architektur zu verstehen und Mikro-Frontend-Anwendungen auf Amazon Web Services zu entwickeln ().AWS

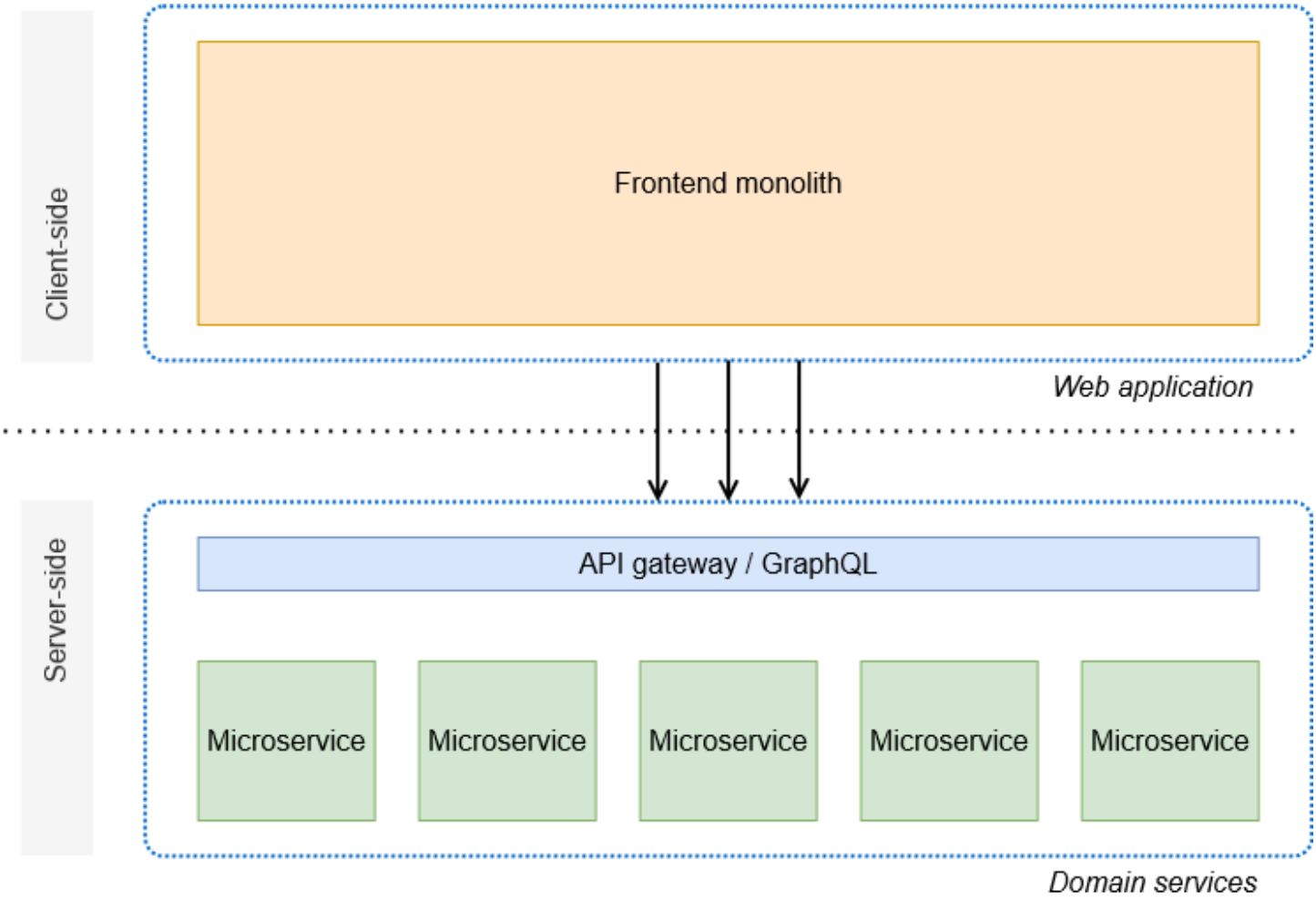
Übersicht

Mikro-Frontends sind eine Architektur, die auf der Zerlegung von Anwendungs-Frontends in unabhängig entwickelte und bereitgestellte Artefakte basiert. Wenn Sie große Frontends in autonome Softwareartefakte aufteilen, können Sie die Geschäftslogik kapseln und Abhängigkeiten reduzieren. Dies unterstützt eine schnellere und häufigere Bereitstellung von Produktinkrementen.

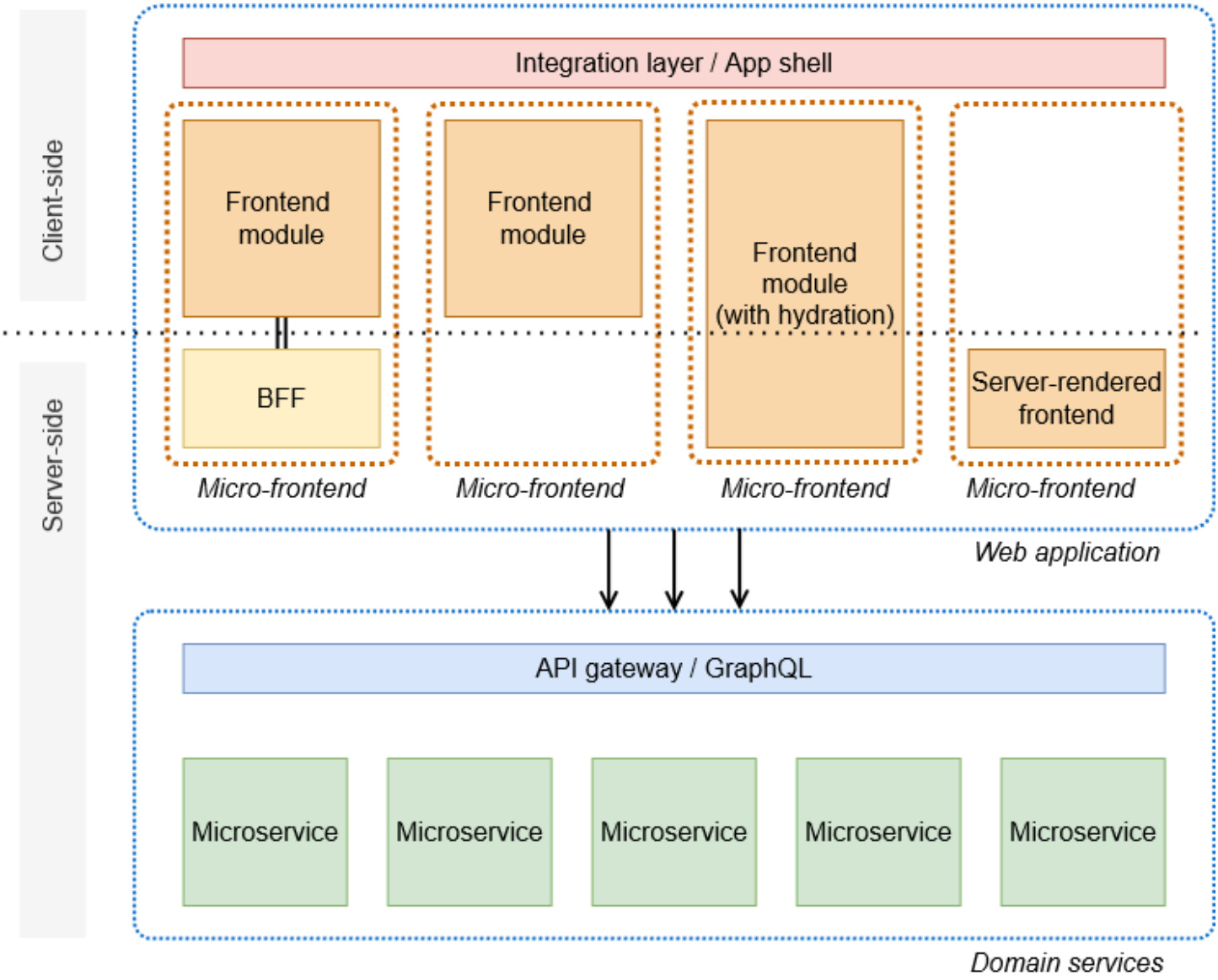
Mikro-Frontends ähneln Microservices. Tatsächlich leitet sich der Begriff Mikrofrontend vom Begriff Microservice ab und zielt darauf ab, den Begriff eines Microservices als Frontend zu vermitteln. Während eine Microservices-Architektur typischerweise ein verteiltes System im Backend mit einem monolithischen Frontend kombiniert, sind Mikro-Frontends in sich geschlossene verteilte Frontend-Dienste. Diese Dienste können auf zwei Arten eingerichtet werden:

- Nur Frontend, Integration mit einer gemeinsamen API-Schicht, hinter der eine Microservices-Architektur läuft
- Full-Stack, was bedeutet, dass jedes Mikro-Frontend seine eigene Backend-Implementierung hat.

Das folgende Diagramm zeigt eine traditionelle Microservices-Architektur mit einem Frontend-Monolith, der ein API-Gateway verwendet, um eine Verbindung zu Backend-Microservices herzustellen.



Das folgende Diagramm zeigt eine Mikro-Frontend-Architektur mit verschiedenen Implementierungen von Microservices.



Wie im vorherigen Diagramm gezeigt, können Sie Mikro-Frontends mit clientseitigem Rendering oder serverseitigem Rendering verwenden:

- Clientseitig gerenderte Mikrofrontends können APIs, die von einem zentralen API Gateway bereitgestellt werden, direkt nutzen.
- Das Team kann innerhalb des begrenzten Kontextes eine backend-for-frontend (BFF) erstellen, um die Konversation des Frontends gegenüber den APIs zu reduzieren.
- Auf der Serverseite können Mikro-Frontends durch einen serverseitigen Ansatz ausgedrückt werden, der auf der Client-Seite durch die Verwendung einer Technik namens Hydration erweitert wird. Wenn eine Seite vom Browser gerendert wird, JavaScript wird die zugehörige Seite

hydratisiert, um Interaktionen mit Benutzeroberflächenelementen, wie z. B. das Klicken auf eine Schaltfläche, zu ermöglichen.

- Mikro-Frontends können im Backend gerendert werden und Hyperlinks verwenden, um zu einem neuen Teil einer Website weiterzuleiten.

Mikro-Frontends eignen sich hervorragend für Unternehmen, die Folgendes tun möchten:

- Skalieren Sie mit mehreren Teams, die an demselben Projekt arbeiten.
- Nutzen Sie die Dezentralisierung der Entscheidungsfindung und ermöglichen Sie es Entwicklern, innerhalb der identifizierten Systemgrenzen innovativ zu sein.

Dieser Ansatz reduziert die kognitive Belastung der Teams erheblich, da sie für bestimmte Teile des Systems verantwortlich sind. Es erhöht die geschäftliche Flexibilität, da Änderungen an einem Teil des Systems vorgenommen werden können, ohne den Rest zu stören.

Mikro-Frontends sind ein eigenständiger architektonischer Ansatz. Obwohl es verschiedene Möglichkeiten gibt, Mikrofrontends zu erstellen, weisen sie alle gemeinsame Merkmale auf:

- Eine Mikro-Frontend-Architektur besteht aus mehreren unabhängigen Elementen. Die Struktur ähnelt der Modularisierung, die bei Microservices im Backend stattfindet.
- Ein Mikro-Frontend ist vollständig für die Frontend-Implementierung innerhalb seines begrenzten Kontextes verantwortlich, der Folgendes umfasst:
  - Benutzeroberfläche
  - Daten
  - Status oder Sitzung
  - Geschäftslogik
  - Flow

Ein begrenzter Kontext ist ein intern konsistentes System mit sorgfältig entworfenen Grenzen, die vermitteln, was ein- und ausgehen kann. Ein Mikrofrontend sollte so wenig Geschäftslogik und Daten wie möglich mit anderen Mikrofrontends teilen. Wo auch immer die gemeinsame Nutzung erfolgen muss, erfolgt sie über klar definierte Schnittstellen wie benutzerdefinierte Ereignisse oder reaktive Streams. Wenn es jedoch um bereichsübergreifende Anliegen geht, wie etwa ein Designsystem oder Logging-Bibliotheken, ist absichtliches Teilen willkommen.

Ein empfohlenes Muster besteht darin, Mikro-Frontends mithilfe funktionsübergreifender Teams zu erstellen. Das bedeutet, dass jedes Mikro-Frontend von demselben Team entwickelt wird, das vom Backend bis zum Frontend arbeitet. Die Eigenverantwortung des Teams ist von entscheidender Bedeutung, von der Codierung bis zur Operationalisierung des Systems in der Produktion.

Mit diesen Leitlinien soll kein bestimmter Ansatz empfohlen werden. Stattdessen werden verschiedene Muster, bewährte Verfahren, Kompromisse sowie architektonische und organisatorische Überlegungen erörtert.

Grundlegende Konzepte

Die Micro-Frontend-Architektur ist stark von drei früheren Architekturkonzepten inspiriert:

- Domänengesteuertes Design ist das mentale Modell zur Strukturierung komplexer Anwendungen in kohärente Bereiche.
- Verteilte Systeme sind ein Ansatz zur Erstellung von Anwendungen als lose gekoppelte Subsysteme, die unabhängig voneinander entwickelt werden und auf ihrer eigenen dedizierten Infrastruktur ausgeführt werden.
- Cloud Computing ist ein Ansatz für den Betrieb von IT-Infrastrukturen als Dienste mit einem pay-as-you-go Modell.

Domänengesteuertes Design

Domain-Driven Design (DDD) ist ein von Eric Evans entwickeltes Paradigma. In seinem 2003 erschienenen Buch Domain-Driven Design: Tackling Complexity in the Heart of Software postuliert Evans, dass sich die Softwareentwicklung eher an geschäftlichen als an technischen Belangen orientieren sollte. Evans schlägt vor, dass IT-Projekte zunächst eine allgegenwärtige Sprache entwickeln, die Fach- und Fachexperten hilft, ein gemeinsames Verständnis zu finden. Auf der Grundlage dieser Sprache können sie ein für alle Seiten verständliches Modell der Geschäftsrealität formulieren.

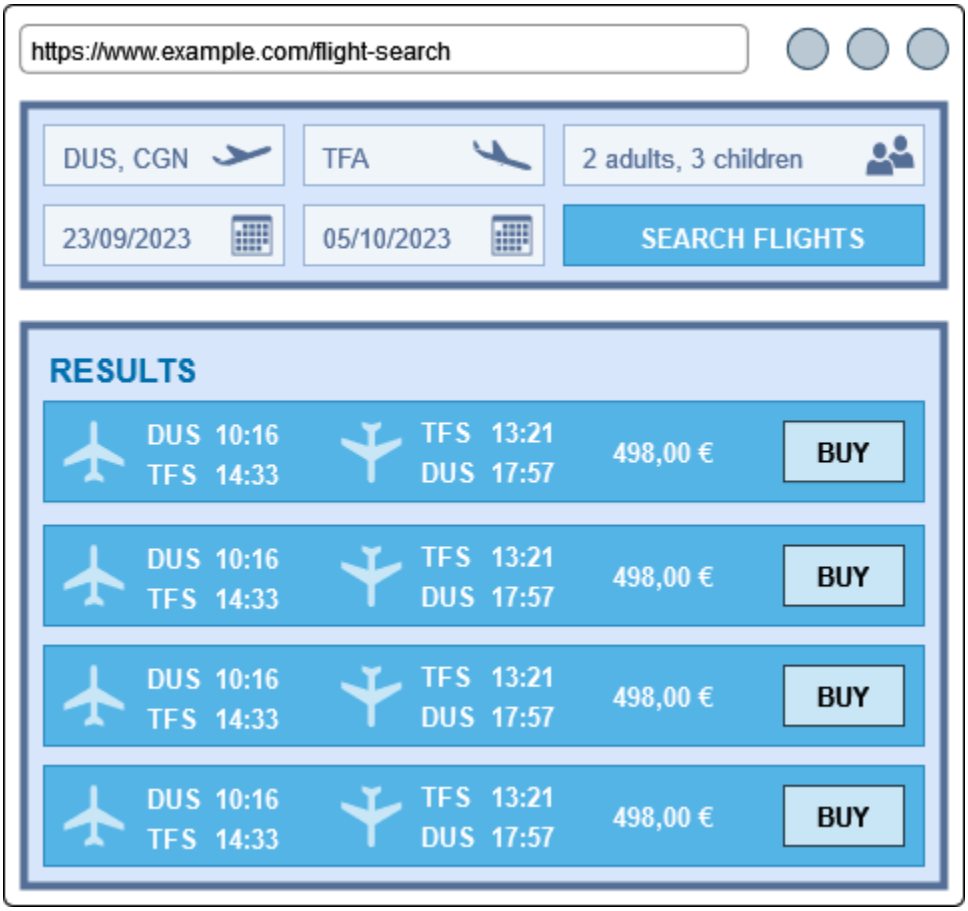
So offensichtlich dieser Ansatz auch sein mag, viele Softwareprojekte leiden unter einer Diskrepanz zwischen Geschäft und IT. Diese Unterbrechungen führen häufig zu erheblichen Missverständnissen, die zu Budgetüberschreitungen, Qualitätseinbußen oder zum Scheitern von Projekten führen.

Evans führt mehrere weitere wichtige Begriffe ein, von denen einer den begrenzten Kontext ist. Ein begrenzter Kontext ist ein in sich geschlossenes Segment einer großen IT-Anwendung, das die Lösung oder Implementierung für genau ein Geschäftsproblem enthält. Eine große Anwendung besteht aus mehreren begrenzten Kontexten, die durch Integrationsmuster lose miteinander verknüpft sind. Diese begrenzten Kontexte können sogar ihre eigenen Dialekte der allgegenwärtigen Sprache haben. Beispielsweise könnte ein Benutzer im Zahlungskontext einer Anwendung andere Aspekte haben als ein Benutzer im Lieferkontext, weil der Begriff Versand bei der Zahlung irrelevant wäre.

Evans definiert nicht, wie klein oder groß ein begrenzter Kontext sein sollte. Die Größe wird durch das Softwareprojekt bestimmt und kann sich im Laufe der Zeit ändern. Gute Indikatoren für die Grenzen

eines Kontextes sind der Grad der Kohäsion zwischen den Entitäten (Domänenobjekten) und der Geschäftslogik.

Im Kontext von Mikro-Frontends lässt sich domänengesteuertes Design am Beispiel einer komplexen Webseite wie einer Flugbuchungsseite veranschaulichen.



Die Hauptbausteine auf dieser Seite sind ein Suchformular, ein Filterfenster und die Ergebnisliste. Um die Grenzen zu identifizieren, müssen Sie unabhängige funktionale Kontexte identifizieren. Berücksichtigen Sie außerdem nicht funktionale Aspekte wie Wiederverwendbarkeit, Leistung und Sicherheit. Der wichtigste Indikator dafür, dass „Dinge zusammengehören“, sind ihre Kommunikationsmuster. Wenn einige Elemente in einer Architektur häufig kommunizieren und komplexe Informationen austauschen müssen, haben sie wahrscheinlich denselben begrenzten Kontext.

Einzelne Benutzeroberflächenelemente wie Schaltflächen sind keine begrenzten Kontexte, da sie nicht funktionell unabhängig sind. Außerdem eignet sich die gesamte Seite nicht gut für einen begrenzten Kontext, da sie in kleinere unabhängige Kontexte unterteilt werden kann. Ein vernünftiger Ansatz besteht darin, das Suchformular als einen begrenzten Kontext und die Ergebnisliste als

zweiten begrenzten Kontext zu behandeln. Jeder dieser beiden begrenzten Kontexte kann nun als separates Mikrofrontend implementiert werden.

Verteilte Systeme

Um die Wartung zu vereinfachen und die Fähigkeit zur Weiterentwicklung zu unterstützen, sind die meisten nicht trivialen IT-Lösungen modular aufgebaut. Modular bedeutet in diesem Fall, dass IT-Systeme aus identifizierbaren Bausteinen bestehen, die durch Schnittstellen voneinander getrennt sind, um eine Trennung der einzelnen Bereiche zu erreichen.

Verteilte Systeme sollten nicht nur modular sein, sondern auch eigenständige, unabhängige Systeme sein. In einem rein modularen System ist jedes Modul idealerweise gekapselt und stellt seine Funktionen über Schnittstellen zur Verfügung, aber es kann nicht unabhängig voneinander eingesetzt werden oder auch nur eigenständig funktionsfähig sein. Außerdem folgen Module im Allgemeinen demselben Lebenszyklus wie andere Module, die Teil desselben Systems sind. Die Bausteine eines verteilten Systems haben dagegen jeweils ihre eigenen Lebenszyklen. Unter Anwendung des domänengesteuerten Entwurfsparadigmas adressiert jeder Baustein eine Geschäftsdomäne oder Subdomäne und lebt in seinem eigenen begrenzten Kontext.

Wenn verteilte Systeme während der Erstellungszeit interagieren, besteht ein gängiger Ansatz darin, Mechanismen zur schnellen Identifizierung von Problemen zu entwickeln. Sie könnten beispielsweise typisierte Sprachen verwenden und viel in Komponententests investieren. Mehrere Teams können bei der Entwicklung und Wartung von Modulen zusammenarbeiten, die häufig als Bibliotheken verteilt werden, damit Systeme sie mit Tools wie npm, Apache Maven und pip nutzen können. NuGet

Während der Laufzeit gehören interagierende verteilte Systeme in der Regel einzelnen Teams. Die Nutzung von Abhängigkeiten führt aufgrund von Fehlerbehandlung, Leistungsausgleich und Sicherheit zu betrieblicher Komplexität. Investitionen in Integrationstests und Beobachtbarkeit sind für die Reduzierung von Risiken von grundlegender Bedeutung.

Die beliebtesten Beispiele für verteilte Systeme sind heute Microservices. In Microservice-Architekturen sind Backend-Services domänengesteuert (und nicht von technischen Belangen wie Benutzeroberfläche oder Authentifizierung abhängig) und gehören autonomen Teams. Mikro-Frontends basieren auf denselben Prinzipien und erweitern den Lösungsumfang auf das Frontend.

Cloud-Computing

Cloud Computing ist eine Möglichkeit, IT-Infrastruktur als Service mit einem pay-as-you-go Modell zu erwerben, anstatt eigene Rechenzentren aufzubauen und Hardware zu kaufen, um sie vor Ort zu betreiben. Cloud Computing bietet mehrere Vorteile:

- Ihr Unternehmen gewinnt erheblich an geschäftlicher Flexibilität, da es in der Lage ist, mit neuen Technologien zu experimentieren, ohne im Voraus große, langfristige finanzielle Verpflichtungen eingehen zu müssen.
- Durch die Nutzung eines Cloud-Anbieters wie AWS kann Ihr Unternehmen auf ein breites Portfolio wartungsarmer und hochintegrierbarer Dienste (wie API-Gateways, Datenbanken, Container-Orchestrierung und Cloud-Funktionen) zugreifen. Durch den Zugriff auf diese Dienste können sich Ihre Mitarbeiter auf Aufgaben konzentrieren, die Ihr Unternehmen von der Konkurrenz abheben.
- Wenn Ihr Unternehmen bereit ist, eine Lösung weltweit einzuführen, können Sie die Lösung in der Cloud-Infrastruktur auf der ganzen Welt einsetzen.

Cloud Computing unterstützt Mikro-Frontends, indem es eine hochgradig verwaltete Infrastruktur bereitstellt. Dies erleichtert funktionsübergreifenden Teams die end-to-end Eigenverantwortung. Das Team sollte zwar über fundierte Betriebskenntnisse verfügen, aber die manuellen Aufgaben der Infrastrukturbereitstellung, Betriebssystemaktualisierungen und Netzwerke würden ablenkend sein.

Da Mikro-Frontends in begrenzten Kontexten existieren, können Teams den am besten geeigneten Dienst für ihren Betrieb auswählen. Teams können beispielsweise zwischen Cloud-Funktionen und Containern für die Datenverarbeitung wählen, und sie können zwischen verschiedenen Varianten von SQL- und NoSQL-Datenbanken oder In-Memory-Caches wählen. Teams können ihre Mikro-Frontends sogar auf einem hochintegrierten Toolkit aufbauen AWS Amplify, wie z. B., das vorkonfigurierte Bausteine für eine serverlose Infrastruktur enthält.

Vergleich von Mikro-Frontends mit alternativen Architekturen

Wie bei allen Architekturstrategien muss die Entscheidung für die Einführung von Mikro-Frontends auf Bewertungskriterien basieren, die sich an den Prinzipien Ihres Unternehmens orientieren. Mikro-Frontends haben Vor- und Nachteile. Wenn sich Ihr Unternehmen für den Einsatz von Mikro-Frontends entscheidet, müssen Sie über Strategien verfügen, um die Herausforderungen verteilter Systeme zu bewältigen

Bei der Auswahl einer Anwendungsarchitektur sind Monolithen, n-Tier-Anwendungen und Microservices in Kombination mit einem SPA-Frontend (Single Page Application) die beliebtesten Alternativen zu Mikro-Frontends. Dies sind alles gültige Ansätze, und jeder von ihnen hat Vor- und Nachteile.

Monolithen

Eine kleine Anwendung, die nicht häufig geändert werden muss, kann sehr schnell als Monolith bereitgestellt werden. Selbst in Situationen, in denen ein erhebliches Wachstum erwartet wird, ist ein Monolith ein natürlicher erster Schritt. Später kann der Monolith entweder ausgemustert oder zu einer flexibleren Struktur umgebaut werden. Wenn Sie mit einem Monolithen beginnen, kann Ihr Unternehmen schneller auf den Markt gehen, Kundenfeedback einholen und das Produkt schneller verbessern.

Monolithische Anwendungen neigen jedoch dazu, sich zu verschlechtern, wenn sie nicht sorgfältig gewartet werden oder wenn die Codebasis im Laufe der Zeit an Größe zunimmt. Wenn mehrere Teams wesentlich zu derselben Codebasis beitragen, tragen sie selten alle zu deren Wartung und Betrieb bei. Dies führt zu einem Ungleichgewicht der Zuständigkeiten, was sich auf die Geschwindigkeit auswirkt und zu Ineffizienzen führt. Gleichzeitig führt eine unbeabsichtigte Kopplung zwischen den Modulen eines Monolithen zu unbeabsichtigten Nebenwirkungen, wenn sich die Codebasis weiterentwickelt. Diese Nebenwirkungen können zu Fehlfunktionen und Ausfällen führen.

N-Tier-Anwendungen

Eine komplexere Anwendung mit einem relativ statischen Entwicklungstempo kann als dreistufige Architektur (Präsentation, Anwendung, Daten) mit einer REST- oder GraphQL-Schicht zwischen Frontend und Backend erstellt werden. Dies ist viel flexibler, und Teams für die verschiedenen Stufen können sich bis zu einem gewissen Grad unabhängig voneinander entwickeln. Der Nachteil einer n-

Tier-Anwendung besteht darin, dass es viel schwieriger ist, Funktionen bereitzustellen. Frontend und Backend sind durch einen API-Vertrag entkoppelt, sodass grundlegende Änderungen zusammen implementiert werden müssen oder die API versioniert werden muss.

Stellen Sie sich das folgende gängige Szenario vor: Wenn die Veröffentlichung einer neuen Funktion eine Änderung des Datenschemas erfordert, kann es Tage dauern, bis sich die Produktbesitzer mit einem Frontend-Team auf eine Reihe von Funktionen geeinigt haben. Dann wird das Frontend-Team das Backend-Team bitten, die Funktionalität seinerseits zu entwickeln und zu veröffentlichen. Das Backend-Team wird mit den Datenbesitzern zusammenarbeiten, um ein Datenbankschema-Update zu veröffentlichen. Als Nächstes wird das Backend-Team eine neue Version der API veröffentlichen, damit das Frontend-Team seine Änderungen entwickeln und veröffentlichen kann. In diesem Szenario kann die Übertragung aller Änderungen an der Produktion Wochen oder sogar Monate dauern, da jedes Team seinen eigenen Backlog, seine eigenen Prioritäten und Mechanismen für die Entwicklung, das Testen und die Veröffentlichung von Änderungen hat.

Microservices

In einer Microservices-Architektur ist das Backend in kleine Dienste unterteilt, von denen jeder ein bestimmtes Geschäftsproblem innerhalb eines begrenzten Kontextes adressiert. Jeder Microservice ist zudem stark von anderen Diensten abgekoppelt, da er über einen klar definierten Schnittstellenvertrag verfügt.

Es ist erwähnenswert, dass begrenzte Kontexte und Schnittstellenverträge auch in gut gestalteten Monolithen und n-Tier-Architekturen existieren sollten. In einer Microservices-Architektur erfolgt die Kommunikation jedoch über das Netzwerk, normalerweise über das HTTP-Protokoll, und Dienste verfügen über eine eigene Laufzeitinfrastruktur. Dies unterstützt die unabhängige Entwicklung, Bereitstellung und den Betrieb der einzelnen Back-End-Dienste.

Wählen Sie den Ansatz für Ihre Anforderungen

Monolithen und n-Tier-Architekturen bündeln Probleme mehrerer Domänen in einem technischen Artefakt. Dadurch lassen sich Aspekte wie Abhängigkeiten und interner Datenfluss leicht verwalten, aber die Bereitstellung neuer Funktionen wird dadurch erschwert. Um eine kohärente Codebasis aufrechtzuerhalten, investiert ein Team aufgrund der großen Codebasis, mit der es umgehen muss, oft Zeit in Refactoring und Entkopplung.

Anwendungen, die von wenigen Teams entwickelt wurden, benötigen möglicherweise nicht die zusätzliche Komplexität, die mit der Umstellung auf Mikro-Frontends einhergeht. Dies gilt

insbesondere dann, wenn die Teams nicht die Strafen zahlen müssen, die sich aus einer hohen Kopplung und langen Vorlaufzeiten für die Veröffentlichung von Änderungen ergeben.

Zusammenfassend lässt sich sagen, dass komplexere und verteilte Architekturen oft die richtige Wahl für komplexe und schnelllebige Anwendungen sind. Für kleine bis mittelgroße Anwendungen ist eine verteilte Architektur einer monolithischen Architektur nicht unbedingt überlegen, insbesondere wenn sich die Anwendung innerhalb kurzer Zeit nicht dramatisch weiterentwickelt.

Architekturentscheidungen in Mikrofrontends

Teams, die ein Mikro-Frontend-Architekturmuster für ihre Anwendungen anwenden, müssen frühzeitig mehrere Architekturentscheidungen treffen:

- Identifizierung von Mikrofrontends und Definition von Grenzen
- Verfassen von Seiten und Ansichten mit Mikrofrontends
- Routing, Statusverwaltung und Kommunikation über Mikrofrontends hinweg
- Verwaltung von Abhängigkeiten bei bereichsübergreifenden Anliegen

In den folgenden Abschnitten werden diese Themen eingehender behandelt.

Bei Architekturentscheidungen ist es wichtig, über die richtigen Kennzahlen zu verfügen und die Nutzungsmuster, Anwendungsmerkmale und Kompromisse zu verstehen. Beispielsweise weist eine E-Commerce-Website im Vergleich zu einem Videobearbeitungstool oder Observability-Dashboards andere Merkmale und Nutzungsmuster auf.

Für die Öffentlichkeit zugängliche Anwendungen mit hohem Traffic und kurzer Sitzungstiefe können für Metriken wie Time to Interactive (TTI) und First Contentful Paint (FCP) optimiert werden. Im Gegensatz dazu kann eine Anwendung, bei der sich Benutzer zu Beginn ihres Tages anmelden und mit der sie den ganzen Tag über interagieren, möglicherweise für das anwendungsinterne Erlebnis optimiert werden. Das Anwendungsteam optimiert möglicherweise nach jeder Navigation die Kennzahl First Input Delay (FID), anstatt die Seite beim ersten Laden zu laden.

Öffentliche Websites müssen für verschiedene Browserumgebungen geeignet sein. Unternehmensanwendungen mit bekannten Einschränkungen in der Client-Umgebung können ihre Mikro-Frontend-Zusammensetzung entsprechend ihren Einschränkungen optimieren.

Es gibt keine einzige richtige Wahl für Architekturentscheidungen. Verstehen Sie die Kompromisse, den Kontext, in dem das Unternehmen tätig ist, die Nutzungsmuster und die Kennzahlen, um Entscheidungen zu treffen, die für jede einzelne Anwendung geeignet sind.

Identifizieren Sie die Grenzen des Mikro-Frontends

Um die Autonomie des Teams zu verbessern, können die von einer Anwendung bereitgestellten Geschäftsfunktionen in mehrere Mikrofrontends mit minimalen Abhängigkeiten voneinander aufgeteilt werden.

Gemäß der zuvor erörterten DDD-Methodik können Teams eine Anwendungsdomäne in Geschäftssubdomänen und begrenzte Kontexte unterteilen. Autonome Teams können dann die Funktionalität ihrer begrenzten Kontexte selbst in die Hand nehmen und diese Kontexte als Mikro-Frontends bereitstellen. Weitere Informationen zur Trennung von Problemen finden Sie im Serverless Land-Diagramm.

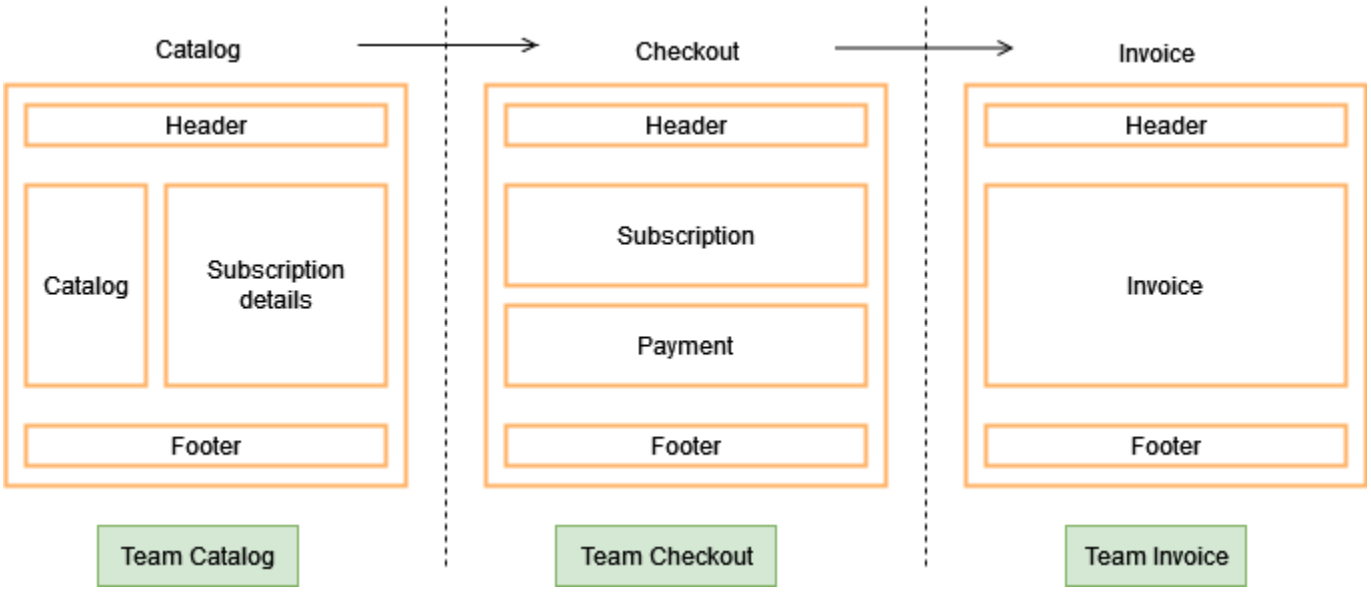
Ein klar definierter, begrenzter Kontext sollte funktionale Überschneidungen und den Bedarf an kontextübergreifender Laufzeitkommunikation minimieren. Die erforderliche Kommunikation kann mit ereignisgesteuerten Methoden implementiert werden. Dies unterscheidet sich nicht von einer ereignisgesteuerten Architektur für die Entwicklung von Microservices.

Eine gut konzipierte Anwendung sollte auch die Bereitstellung zukünftiger Erweiterungen durch neue Teams unterstützen, um den Kunden ein einheitliches Erlebnis zu bieten.

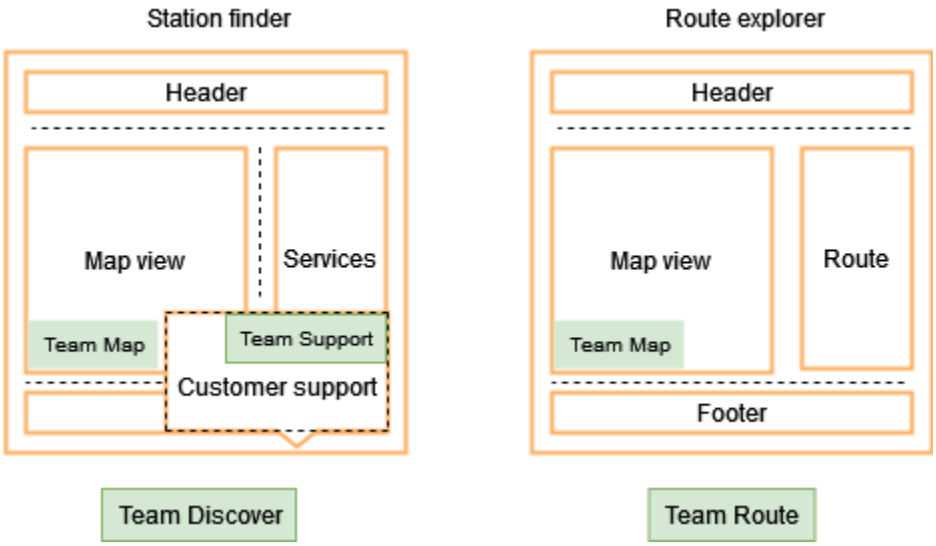
Wie unterteilt man eine monolithische Anwendung in Mikro-Frontends

Der Abschnitt Überblick enthielt ein Beispiel für die Identifizierung unabhängiger funktionaler Kontexte auf einer Webseite. Es ergeben sich mehrere Muster für die Aufteilung der Funktionen auf der Benutzeroberfläche.

Wenn die Geschäftsdomänen beispielsweise Phasen einer Benutzerreise bilden, kann eine vertikale Aufteilung im Frontend angewendet werden, bei der eine Sammlung von Ansichten innerhalb der Benutzerreise als Mikro-Frontends bereitgestellt wird. Das folgende Diagramm zeigt eine vertikale Aufteilung, bei der die Schritte „Katalog“, „Checkout“ und „Rechnung“ von separaten Teams als separate Mikrofrontends bereitgestellt werden.



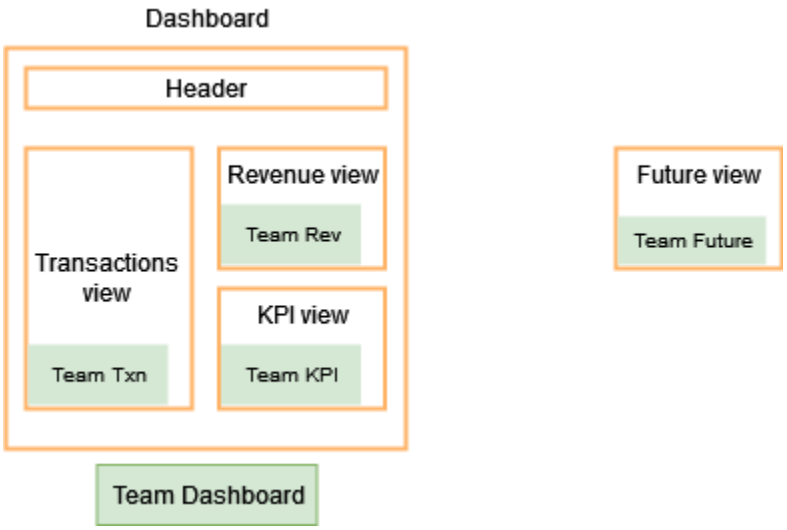
Für einige Anwendungen reicht die vertikale Aufteilung allein möglicherweise nicht aus. Beispielsweise müssen einige Funktionen möglicherweise in vielen Ansichten bereitgestellt werden. Für diese Anwendungen können Sie eine gemischte Aufteilung anwenden. Das folgende Diagramm zeigt eine gemischte Split-Lösung, bei der Mikro-Frontends für Station Finder und Route Explorer beide die Kartenansichtsfunktion verwenden.



Anwendungen vom Typ Portal oder Dashboard fassen in der Regel Frontend-Funktionen in einer einzigen Ansicht zusammen. Bei diesen Arten von Anwendungen kann jedes Widget als Mikro-Frontend bereitgestellt werden, und die Hosting-Anwendung definiert die Einschränkungen und Schnittstellen, die die Mikro-Frontends implementieren sollen.

Dieser Ansatz bietet Mikro-Frontends einen Mechanismus, mit dem Probleme wie die Größe von Viewports, Authentifizierungsanbieter, Konfigurationseinstellungen und Metadaten berücksichtigt werden können. Diese Arten von Anwendungen sind im Hinblick auf ihre Erweiterbarkeit optimiert. Neue Funktionen können von neuen Teams entwickelt werden, um die Dashboard-Funktionen zu skalieren.

Das folgende Diagramm zeigt eine Dashboard-Anwendung, die von drei einzelnen Teams entwickelt wurde, die Teil von Team Dashboard sind.



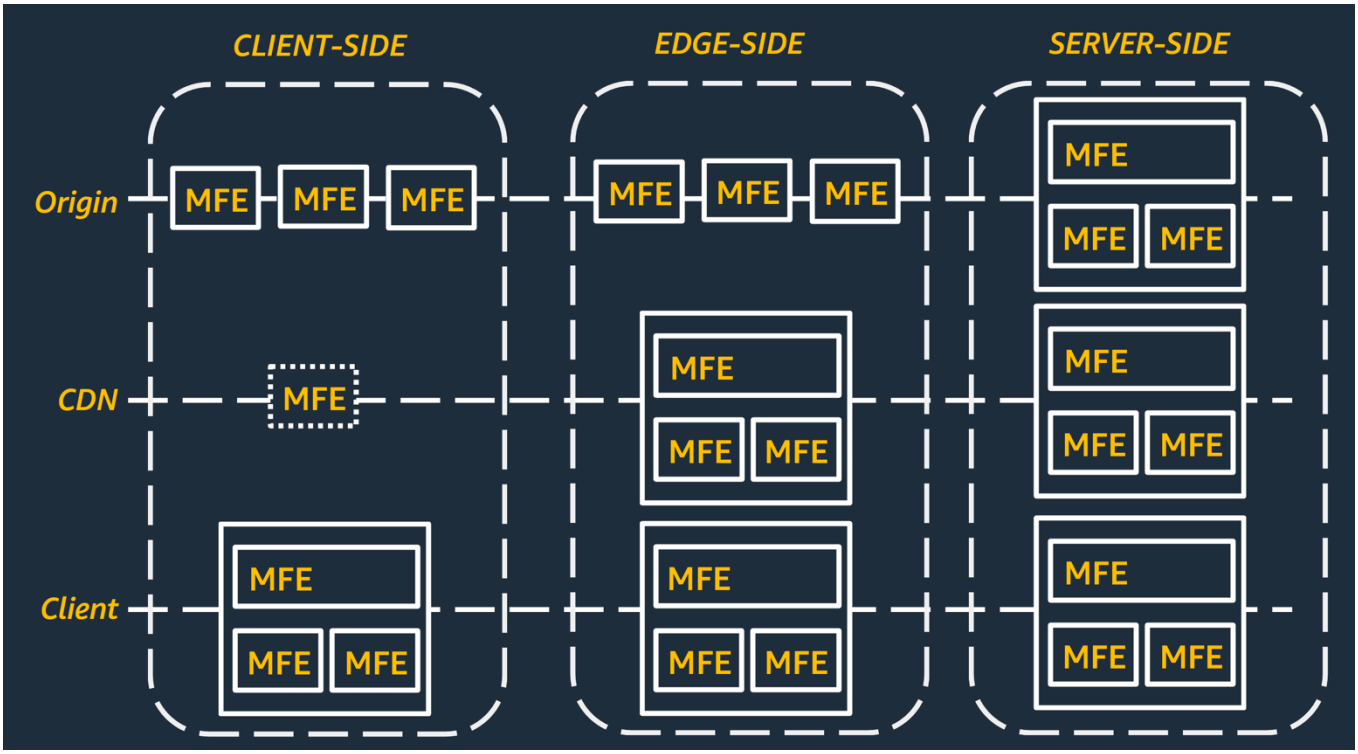
In dem Diagramm stellt die future Ansicht neue Funktionen dar, die von neuen Teams entwickelt wurden, um das Team-Dashboard und die Dashboard-Funktionen zu skalieren.

Portal- und Dashboard-Anwendungen stellen in der Regel Funktionen zusammen, indem sie eine gemischte Aufteilung der Benutzeroberfläche verwenden. Die Mikrofrontends sind mit genau definierten Einstellungen, einschließlich Positions- und Größenbeschränkungen, konfigurierbar.

Verfassen von Seiten und Ansichten mit Mikrofrontends

Sie können Ansichten einer Anwendung mit clientseitiger, randseitiger und serverseitiger Komposition erstellen. Die Kompositionsmuster weisen unterschiedliche Merkmale in Bezug auf die erforderlichen Teamfähigkeiten, Fehlertoleranz, Leistung und Cache-Verhalten auf.

Das folgende Diagramm zeigt, wie die Zusammensetzung auf den clientseitigen, edge-seitigen und serverseitigen Ebenen einer Micro-Frontend-Architektur erfolgt.



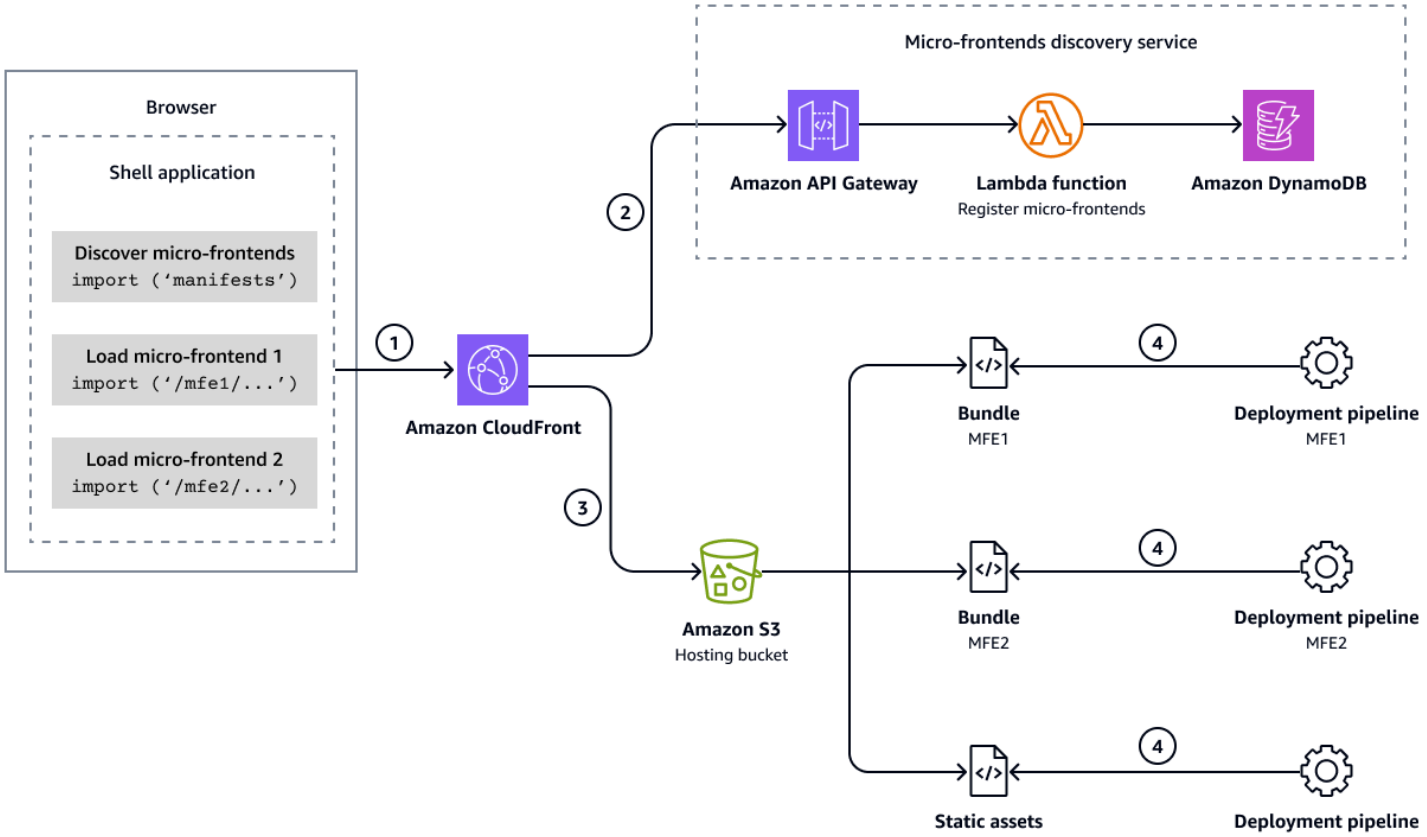
Die clientseitigen, edge-seitigen und serverseitigen Schichten werden in den folgenden Abschnitten behandelt.

Clientseitige Zusammensetzung

Dynamisches Laden und Anhängen von Mikrofrontends als DOM-Fragmente (Document Object Model) auf dem Client (Browser oder mobile Webansicht). Die Mikrofrontend-Artefakte, wie JavaScript z. B. CSS-Dateien, können aus Content Delivery Networks (CDNs) geladen werden, um die Latenz zu reduzieren. Für die clientseitige Zusammenstellung ist Folgendes erforderlich:

- Ein Team, das eine Shell-Anwendung oder ein Micro-Frontend-Framework besitzt und verwaltet, um das Auffinden, Laden und Rendern von Mikro-Frontend-Komponenten zur Laufzeit im Browser zu ermöglichen
- Hohe Kenntnisse in Frontend-Technologien wie HTML, CSS und tiefes Verständnis von JavaScript Browserumgebungen
- Optimierung der Anzahl der auf einer Seite JavaScript geladenen Dateien und Disziplin zur Vermeidung globaler Namespace-Konflikte

Das folgende Diagramm zeigt eine AWS Beispielarchitektur für serverlose clientseitige Komposition.



Die clientseitige Zusammenstellung erfolgt in der Browserumgebung über eine Shell-Anwendung. Das Diagramm zeigt die folgenden Details:

1. Nachdem die Shell-Anwendung geladen wurde, sendet sie eine erste Anfrage an Amazon, um die Mikro-Frontends CloudFront zu ermitteln, die über einen Manifest-Endpunkt geladen werden sollen.
2. Manifeste enthalten Informationen zu jedem Mikro-Frontend (z. B. Name, URL, Version und Fallback-Verhalten). Die Manifeste werden vom Micro-Frontend-Discovery-Service bereitgestellt. In der Abbildung wird dieser Discovery-Service durch Amazon API Gateway, eine AWS Lambda Funktion, und Amazon DynamoDB dargestellt. Die Shell-Anwendung verwendet die Manifestinformationen, um einzelne Mikrofrontends aufzufordern, die Seite innerhalb eines bestimmten Layouts zusammenzustellen.
3. Jedes Micro-Frontend-Bundle besteht aus statischen Dateien (wie CSS JavaScript und HTML). Die Dateien werden in einem Amazon Simple Storage Service (Amazon S3) -Bucket gehostet und über bereitgestellt CloudFront.
4. Teams können neue Versionen ihrer Mikrofrontends bereitstellen und die Manifestinformationen aktualisieren, indem sie Bereitstellungspipelines verwenden, die ihnen gehören.

Randseitige Zusammensetzung

Verwenden Sie Transklusionstechniken wie Edge Side Includes (ESI) oder Server Side Includes (SSI), die von einigen CDNs und Proxys vor den Ursprungsservern unterstützt werden, um eine Seite zu erstellen, bevor Sie sie drahtgebunden an die Clients senden. ESI erfordert Folgendes:

- Ein CDN mit ESI-Fähigkeit oder eine Proxybereitstellung vor serverseitigen Mikrofrontends. Proxy-Implementierungen wie HAProxy, Varnish und NGINX unterstützen SSI.
- Ein Verständnis der Verwendung und der Einschränkungen von ESI- und SSI-Implementierungen.

Teams, die neue Anwendungen starten, wählen für ihr Kompositionsmuster in der Regel keine randseitige Zusammensetzung. Dieses Muster könnte jedoch einen Weg für ältere Anwendungen bieten, die auf Transklusion angewiesen sind.

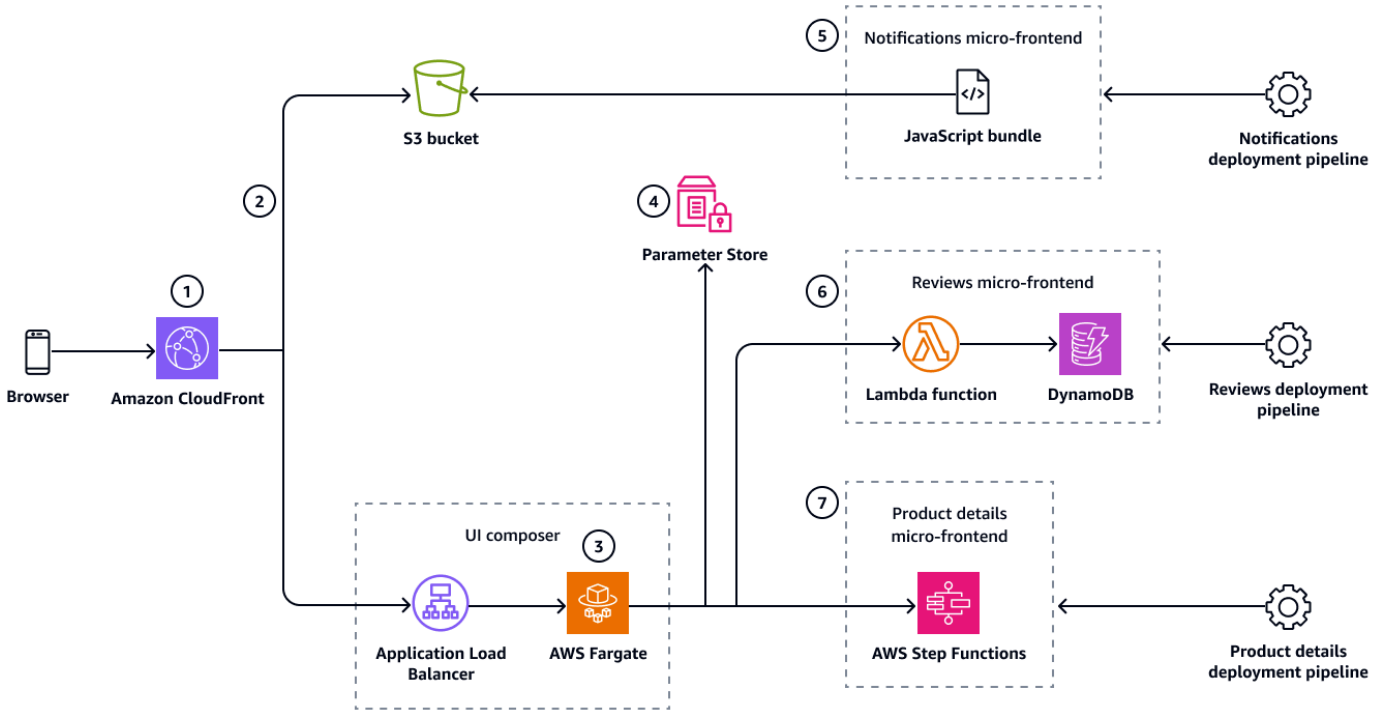
Serverseitige Zusammensetzung

Verwenden Sie Originalserver, um Seiten zu erstellen, bevor sie am Edge zwischengespeichert werden. Dies kann mit herkömmlichen Technologien wie PHP, Jakarta Server Pages (JSP) oder Templating-Bibliotheken geschehen, um die Seiten mithilfe von Fragmenten von Mikrofrontends zusammenzustellen. Sie können auch JavaScript Frameworks wie Next.js verwenden, die auf dem Server ausgeführt werden, um Seiten auf dem Server mit serverseitigem Rendering (SSR) zu erstellen.

Nachdem die Seiten auf dem Server gerendert wurden, können sie auf CDNs zwischengespeichert werden, um die Latenz zu reduzieren. Wenn neue Versionen von Mikro-Frontends bereitgestellt werden, müssen die Seiten neu gerendert und der Cache aktualisiert werden, um den Kunden die neuesten Versionen zur Verfügung zu stellen.

Die serverseitige Zusammenstellung erfordert ein tiefes Verständnis der Serverumgebung, um Muster für die Bereitstellung, die Erkennung serverseitiger Mikrofrontends und die Cache-Verwaltung festzulegen.

Das folgende Diagramm zeigt die serverseitige Zusammensetzung.



Das Diagramm umfasst die folgenden Komponenten und Prozesse:

1. Amazon CloudFront bietet einen einzigartigen Einstiegspunkt für die Anwendung. Die Distribution hat zwei Ursprünge: den ersten für statische Dateien und den zweiten für den UI Composer.
2. Statische Dateien werden in einem Amazon S3 S3-Bucket gehostet. Sie werden vom Browser und vom UI Composer für HTML-Vorlagen verwendet.
3. Der UI Composer läuft auf einem Container-Cluster in AWS Fargate. Mit einer containerisierten Lösung können Sie bei Bedarf Streaming-Funktionen und Multithread-Rendering verwenden.
4. Parameter Store, eine Funktion von AWS Systems Manager, wird als grundlegendes System zur Erkennung von Mikro-Frontends verwendet. Diese Funktion stellt einen Schlüsselwertspeicher bereit, der vom UI Composer zum Abrufen der zu verwendenden Mikro-Frontend-Endpunkte verwendet wird.
5. Das Mikrofrontend für Benachrichtigungen speichert das optimierte Paket im S3-Bucket. JavaScript Dies wird auf dem Client gerendert, da dieser auf Benutzerinteraktionen reagieren muss.
6. Das Micro-Frontend für Bewertungen besteht aus einer Lambda-Funktion, und die Benutzerrezensionen werden in DynamoDB gespeichert. Das Mikrofrontend für Bewertungen wird vollständig serverseitig gerendert und gibt ein HTML-Fragment aus.

7. Das Mikrofrontend für Produktdetails ist ein Low-Code-Mikrofrontend, das verwendet. AWS Step Functions Der Express-Workflow kann synchron aufgerufen werden und enthält die Logik für das Rendern des HTML-Fragments und eine Caching-Ebene.

Weitere Informationen zur serverseitigen Komposition finden Sie im Blogbeitrag Mikrofrontends für serverseitiges Rendern — die Architektur.

Routing und Kommunikation zwischen Mikrofrontends

Die Routing-Optionen hängen vom Kompositionsansatz ab. Die Kommunikation kann optimiert werden, indem die Kopplung zwischen den Frontend-Komponenten reduziert wird.

Routing

Anwendungen, die eine clientseitige Zusammensetzung mit vertikaler Aufteilung verwenden, können serverseitiges Routing (mehrseitige Anwendung) oder clientseitiges Routing (einseitige Anwendung) verwenden. Wenn sie eine gemischte Aufteilung für die Gestaltung der Benutzeroberfläche verwenden, ist clientseitiges Routing erforderlich, um tiefere Routing-Hierarchien von Mikrofrontends auf einer Seite zu unterstützen.

Anwendungen, die Edge-side Composition und serverseitige Komposition verwenden, passen besser zu serverseitigem Routing oder Routing mit Edge-Compute wie Lambda @Edge mit Amazon. CloudFront

Kommunikation zwischen Mikro-Frontends

Bei Mikro-Frontend-Architekturen empfehlen wir, die Kopplung zwischen Frontend-Komponenten zu reduzieren. Ein Ansatz zur Reduzierung der Kopplung besteht darin, von synchronen Funktionsaufrufen zu asynchronem Messaging überzugehen.

Browser-Laufzeiten und Benutzerinteraktionen sind naturgemäß asynchron. Ereignisse können zwischen Produzenten und Verbrauchern durch Nachrichten ausgetauscht werden. Die Veranstaltungen bieten eine klar definierte Schnittstelle für die Kommunikation zwischen Mikrofrontends.

Wenn Sie DDD-Verfahren anwenden, um Ihre begrenzten Kontexte für Mikro-Frontends zu identifizieren, besteht der nächste Schritt darin, Ereignisse zu identifizieren, die grenzüberschreitend kommuniziert werden müssen.

Als Nachrichtenmechanismus für Ereignisse können native DOM-Ereignisse (`CustomEvents`), JavaScript Event-Emitter oder reaktive Stream-Bibliotheken verwendet werden, die von den Plattformteams bereitgestellt werden. Mikro-Frontends veröffentlichen Ereignisse und abonnieren Ereignisse, die für ihren begrenzten Kontext relevant sind. Bei dieser Methode müssen sich Herausgeber und Abonnenten nicht gegenseitig bewusst sein. Der Vertrag ist die Definition des Ereignisses. Eine visuelle Darstellung davon finden Sie im Abschnitt Kommunizieren mit Ereignissen des Diagramms Eingeschränkter Kontext mit Ereignisarchitekturen.

Verwaltung von Abhängigkeiten aus bereichsübergreifenden Gründen

Ein bewusstes Abhängigkeitsmanagement ist entscheidend für den Erfolg einer verteilten Architektur wie Mikro-Frontends. Das Abhängigkeitsmanagement ist einer der schwierigsten Bereiche der Mikro-Frontend-Entwicklung.

In einer Micro-Frontend-Architektur sind zwei wichtige Aspekte des Abhängigkeitsmanagements die Leistungseinbußen, die durch die Übertragung großer Codeartefakte auf den Client entstehen, und der Overhead an Rechenressourcen. Im Idealfall muss Ihr Unternehmen festlegen, wie Abhängigkeiten in einer verteilten Frontend-Architektur aufrechterhalten werden.

Drei praktikable Strategien, um die Aufrechterhaltung von Abhängigkeiten vorzuschreiben, sind „Share Nothing“ und verwenden Webstandards wie Importzuordnungen und Modulverbund. Andere Ansätze sind gegen Pattern gerichtet, weil sie gegen die Grundprinzipien verteilter Architekturen verstoßen.

Teilen Sie nichts, wo immer möglich

Der Share-Nothing-Ansatz postuliert, dass Abhängigkeiten zwischen unabhängigen Softwareartefakten überhaupt nicht gemeinsam genutzt werden sollten, zumindest nicht bei der Integration oder Laufzeit. Das heißt, wenn zwei Mikro-Frontends von derselben Bibliothek abhängen, muss jedes zum Zeitpunkt der Erstellung in der Bibliothek gebacken und separat ausgeliefert werden. Außerdem muss jedes Mikro-Frontend sicherstellen, dass die Bibliothek globale Namespaces und gemeinsam genutzte Ressourcen nicht verschmutzt.

Dies führt zu Redundanzen, ist aber ein bewusster Kompromiss mit maximaler Agilität. Da es keine gemeinsamen Laufzeitabhängigkeiten gibt, haben Teams maximale Flexibilität, um die Software so weiterzuentwickeln, wie sie es für sinnvoll halten, solange sie dies im Rahmen ihrer Lösung tun und keine Schnittstellenverträge verletzen.

Auf einer Plattform, auf der Mikrofrontends dem Share-Nothing-Prinzip folgen, ist es wichtig, Mikro-Frontends so leicht wie möglich zu halten. Es erfordert Entwickler, die ihre Mikrofrontends kompetent und gewissenhaft im Hinblick auf die Leistung optimieren und die Benutzererfahrung nicht der Entwicklererfahrung opfern.

Wenn du Code teilst

Wenn Sie sich entscheiden, Code gemeinsam zu nutzen, können Sie ihn als Bibliotheken oder Laufzeitmodule teilen. Das Frontend-Kernteam stellt beispielsweise Bibliotheken für die Nutzung im Mikro-Frontend über CDNs bereit. Die Business Value Teams können die Bibliotheken zur Laufzeit laden oder sie können Paket-Repositorys verwenden, um ihre Bibliotheken zu veröffentlichen. Micro-Frontend-Teams können zum Zeitpunkt der Erstellung anhand einer bestimmten Version der Paketbibliothek entwickeln, ähnlich wie mobile Anwendungen, die hybride Frameworks verwenden.

Eine dritte Option ist die Verwendung einer privaten Paketregistrierung, um die Integration gängiger Bibliotheken während der Erstellung zu unterstützen. Dadurch wird das Risiko verringert, dass eine grundlegende Änderung im Bibliotheksvertrag Fehler zur Laufzeit auslöst. Dieser konservativere Ansatz erfordert jedoch mehr Governance, um alle Mikrofrontends mit neueren Bibliotheksversionen zu synchronisieren.

Um die Ladezeiten der Seiten zu verbessern, können Mikrofrontends die Bibliotheksabhängigkeiten externalisieren, sodass sie aus zwischengespeicherten Chunks von einem CDN wie Amazon geladen werden. CloudFront

Um Laufzeitabhängigkeiten zu verwalten, können Mikrofrontends Import-Maps (oder Bibliotheken wie`System.js`) verwenden, um anzugeben, von wo jedes Modul zur Laufzeit geladen wird. Webpack Module Federation ist ein weiterer Ansatz, um auf eine gehostete Version eines Remote-Moduls zu verweisen und gemeinsame Abhängigkeiten zwischen unabhängigen Mikrofrontends aufzulösen.

Ein anderer Ansatz besteht darin, das dynamische Laden von Import-Maps mit einer ersten Anfrage an einen Discovery-Endpunkt zu erleichtern.

Gemeinsamer Status

Um die Kopplung von Mikrofrontends zu reduzieren, ist es wichtig, ein globales State-Management zu vermeiden, auf das alle Mikrofrontends in derselben Ansicht zugreifen können, ähnlich wie bei monolithischen Architekturen. Ein globaler Redux-Store, auf den beispielsweise von allen Mikrofrontends aus zugegriffen werden kann, erhöht die Kopplung.

Ein Muster zur Eliminierung des gemeinsamen Zustands besteht darin, ihn in Mikro-Frontends einzukapseln und mit asynchronen Nachrichten zu kommunizieren, wie bereits beschrieben.

Wenn es absolut notwendig ist, führen Sie klar definierte Schnittstellen für den globalen Status ein und entscheiden Sie sich für die gemeinsame Nutzung nur zum Lesen, um unerwartetes Verhalten zu vermeiden:

- Wenn eine vertikale Aufteilung vorhanden ist, können Sie URL-Komponenten und Browserspeicher verwenden, um auf Informationen aus der Host-Umgebung zuzugreifen.
- Bei einer gemischten Aufteilung können Sie auch benutzerdefinierte DOM-Standardereignisse oder JavaScript Bibliotheken wie Emitter oder bidirektionale Streams verwenden, um Informationen an Mikrofrontends weiterzuleiten.

Wenn Sie mehrere Informationen über Mikro-Frontends hinweg gemeinsam nutzen müssen, empfehlen wir, die Grenzen der Mikro-Frontends erneut zu überprüfen. Die Notwendigkeit des Austauschs könnte auf die Geschäftsentwicklung oder ein unterdurchschnittliches anfängliches Design zurückzuführen sein.

Es ist auch möglich, serverseitige Sitzungen zu verwenden, bei denen jedes Mikro-Frontend die erforderlichen Daten mithilfe einer Sitzungs-ID abruft. Um die Kopplung zu reduzieren, ist es wichtig, den gemeinsamen Status zu eliminieren und die spezifischen Sitzungsdaten für das Mikrofrontend getrennt zu halten.

Frameworks und Tools

Es gibt keinen Mangel an Frontend-Frameworks wie Angular und Next.js, aber die meisten von ihnen wurden nicht mit Blick auf Mikro-Frontends entwickelt. Daher fehlen ihnen manchmal Mechanismen, um den Herausforderungen der Mikro-Frontend-Architektur zu begegnen.

Allgemeine Überlegungen zu den Rahmenbedingungen

Dieser Leitfaden zielt nicht darauf ab, einzelne Frameworks zu empfehlen oder zu vergleichen.
Da häufig mehrere Mikro-Frontends auf derselben Webanwendungsseite ausgeführt werden, sind Ladevorgänge und Laufzeitleistung ein großes Problem. Es ist wichtig, ein Framework zu wählen, das so wenig Overhead wie möglich verursacht.

Frameworks werden basierend auf der Rendering-Ebene unterteilt:

- Clientseitiges Rendern (CSR)
- Serverseitiges Rendern (SSR)

Frontend-Architekturen beinhalten weitere Funktionen, wie z. B. die Generierung statischer Websites (SSG). SSG wird jedoch nur einmal ausgeführt. Mikro-Frontends werden hauptsächlich zur Laufzeit erstellt, daher sind CSR und SSR die Hauptoptionen.

Clientseitiges Rendern

Für CSR gibt es zwei beliebte Optionen:

- Einheitliches SPA-Framework
- Modul Federation

Single SPA ist eine einfache Wahl für die Zusammenstellung von Mikro-Frontends. Es löst die häufigsten Herausforderungen in Mikro-Frontend-Architekturen, wie z. B. die Zusammenstellung mehrerer Mikro-Frontends auf derselben Seite und die Vermeidung von Abhängigkeitskonflikten.

Module Federation begann als Plugin, das von Webpack 5 angeboten wurde. Es löst die meisten Herausforderungen in Mikro-Frontend-Architekturen, einschließlich der Verwaltung von Abhängigkeiten zwischen verschiedenen Artefakten. Module Federation 2.0 funktioniert nativ mit Rspack, Webpack, Esbuild und jetzt mit. JavaScript

Erwägen Sie, überhaupt kein Framework zu verwenden. Moderne Browser, die laut caniuse.com einen Marktanteil von insgesamt 98 Prozent haben, bieten Funktionen wie benutzerdefinierte Elemente nativ an und sind für eine Micro-Frontend-Anwendung ausreichend. Kombinieren Sie bei Bedarf benutzerdefinierte Elemente mit einfachen Bibliotheken für die Übertragung von Ereignissen, die Internationalisierung oder andere spezifische Probleme.

Serverseitiges Rendern

Auf der SSR-Seite sind die beiden Hauptoptionen komplizierter:

- Nutzen Sie ein vorhandenes Framework wie Next.js und wenden Sie ein Mikro-Frontend-Prinzip an, das Module Federation verwendet.
- Verwenden Sie HTML-over-the-wire , um HTML-Fragmente auszutauschen, die Mikrofrontends darstellen, und fügen Sie diese Fragmente zur Laufzeit in einer Vorlage zusammen. Ein Beispiel für diesen Ansatz ist Podium.

API-Integration – Backend für Frontend

Das Muster Backends for Frontends (BFF) wird typischerweise in Microservices-Umgebungen verwendet. Im Kontext von Mikro-Frontends ist ein BFF ein serverseitiger Dienst, der zu einem Mikro-Frontend gehört. Nicht alle Mikrofrontends müssen über ein BFF verfügen. Wenn Sie jedoch ein BFF verwenden, muss es innerhalb desselben begrenzten Kontextes ausgeführt werden und darf nicht von anderen begrenzten Kontexten gemeinsam genutzt werden.

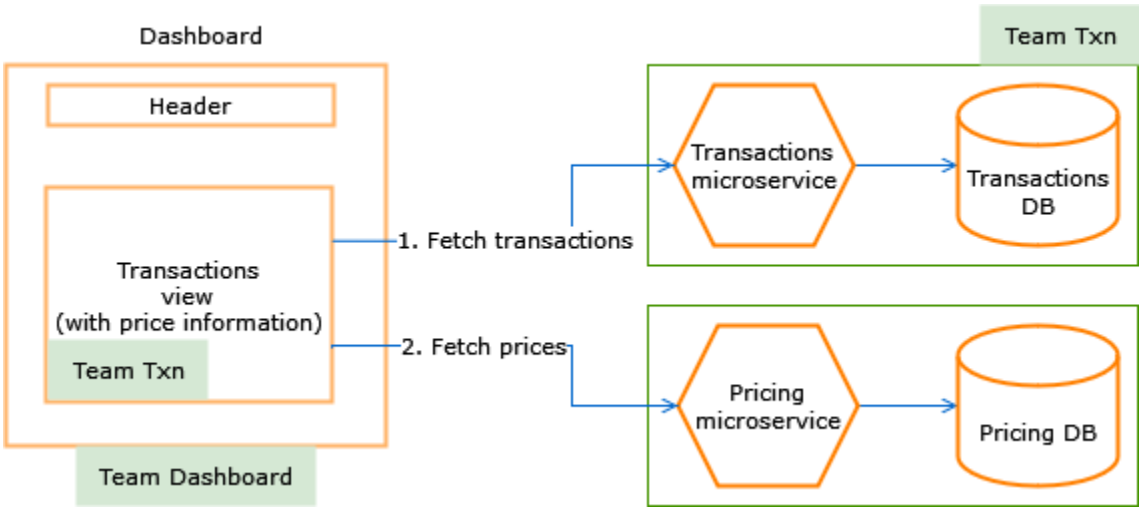
Im Gegensatz zu einem herkömmlichen Dienst folgt ein BFF keinem Domänenmodell. Stattdessen handelt es sich um eine API-Ebene für das Mikro-Frontend, um Daten vorzuverarbeiten, bevor sie den Client erreichen. Zu den Bereichen, in denen dies nützlich ist, gehören die folgenden:

- Autorisierung für private APIs
- Aggregation von Daten aus verschiedenen Quellen
- Transformation von Daten zur Reduzierung der Netzwerklast und zur Erleichterung des Datenverbrauchs durch den Kunden

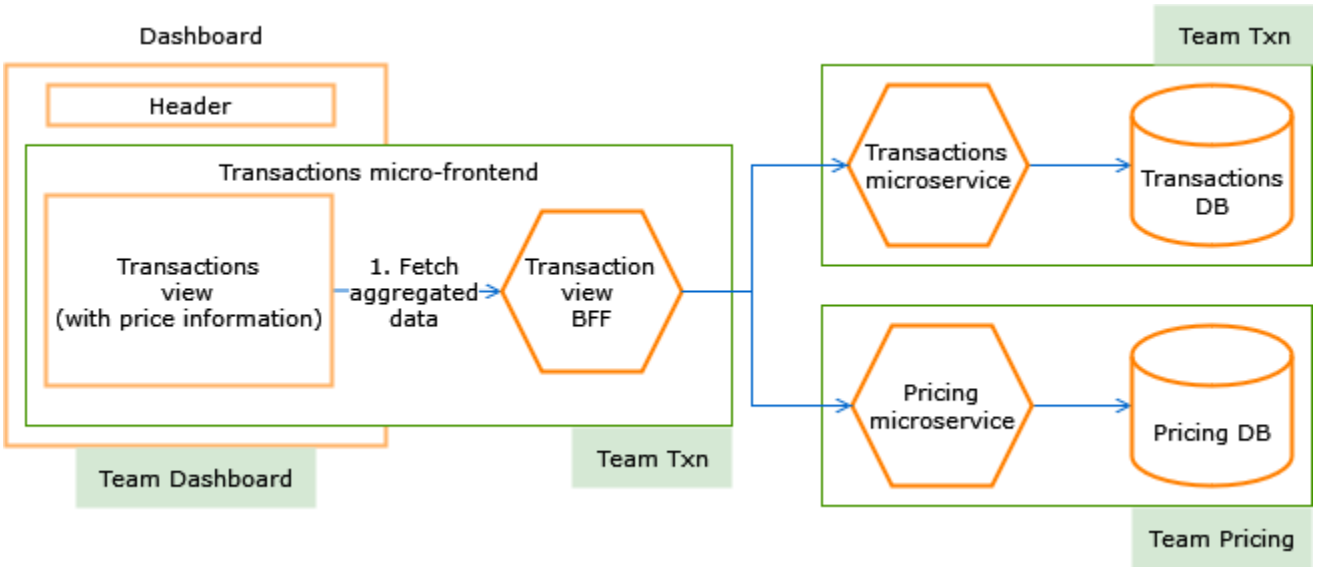
Somit gehört ein BFF dem Micro-Frontend und nicht dem Domain-Servicetier. BFFs können wie folgt bereitgestellt werden:

- AppSync AWS-GraphQL-APIs
- Eine Reihe von AWS Lambda Lambda-Funktionen
- Als Container, der auf Amazon ECS, Amazon EKS oder AWS läuft AppRunner

Das folgende Diagramm zeigt, dass Mikro-Frontends ohne das BFF-Muster eine Verbindung zu einzelnen Microservice-API-Endpunkten herstellen müssen, um Daten abzurufen und zu aggregieren.



Stattdessen können Mikro-Frontends anhand des BFF-Musters in der folgenden Abbildung mit ihrem eigenen Backend kommunizieren und aggregierte Daten abrufen.



Teams können BFFs für verschiedene Kanäle entwickeln, z. B. für Mobilgeräte, das Internet oder bestimmte Ansichten. Dabei müssen die Interaktionen im Backend optimiert werden, indem die Chataktivität reduziert wird.

Styling und CSS

Cascading Style Sheets (CSS) ist eine Sprache, mit der die Präsentation eines Dokuments zentral festgelegt werden kann, anstatt die Formatierung von Text und Objekten fest zu codieren. Die Cascading Funktion der Sprache wurde entwickelt, um mithilfe von Vererbung Prioritäten zwischen Stilen zu steuern. Wenn Sie an Mikro-Frontends arbeiten und eine Strategie zur Verwaltung von Abhängigkeiten entwickeln, kann die kaskadierende Funktion der Sprache eine Herausforderung sein.

Beispielsweise existieren zwei Mikro-Frontends nebeneinander auf derselben Seite, von denen jedes sein eigenes Styling für das HTML-Element definiert. `body` Wenn jede Datei ihre eigene CSS-Datei abruft und sie mithilfe eines `style` Tags an das DOM anhängt, überschreibt die CSS-Datei die erste, wenn beide Definitionen für gemeinsame HTML-Elemente, Klassennamen oder Element-IDs haben. Es gibt verschiedene Strategien, um mit diesen Problemen umzugehen, je nachdem, welche Abhängigkeitsstrategie Sie für die Verwaltung von Stilen wählen.

Derzeit besteht der beliebteste Ansatz, um Leistung, Konsistenz und Entwicklererfahrung in Einklang zu bringen, in der Entwicklung und Wartung eines Designsystems.

Designsysteme – Ein Ansatz, bei dem man etwas teilen kann

Bei diesem Ansatz wird ein System verwendet, das bei Bedarf das Design gemeinsam nutzt und gleichzeitig gelegentliche Abweichungen unterstützt, um ein Gleichgewicht zwischen Konsistenz, Leistung und Entwicklererfahrung herzustellen. Ein Designsystem ist eine Sammlung wiederverwendbarer Komponenten, die sich an klaren Standards orientieren. Die Entwicklung eines Designsystems wird in der Regel von einem Team mit Beiträgen und Beiträgen von vielen Teams vorangetrieben. In der Praxis ist ein Designsystem eine Möglichkeit, Elemente auf niedriger Ebene gemeinsam zu nutzen, die als JavaScript Bibliothek exportiert werden können. Mikro-Frontend-Entwickler können die Bibliothek als Abhängigkeit verwenden, um einfache Schnittstellen zu erstellen, indem sie vorgefertigte verfügbare Ressourcen zusammenstellen, und als Ausgangspunkt für die Erstellung neuer Schnittstellen.

Stellen Sie sich das Beispiel eines Mikro-Frontends vor, das ein Formular benötigt. Die typische Entwicklererfahrung besteht darin, vorgefertigte Komponenten zu verwenden, die im Designsystem verfügbar sind, um Textfelder, Schaltflächen, Dropdownlisten und andere Benutzeroberflächenelemente zusammenzustellen. Der Entwickler muss kein Styling für die

eigentlichen Komponenten schreiben, sondern nur dafür, wie sie zusammen aussehen. Das System, das erstellt und veröffentlicht werden soll, kann Webpack Module Federation oder einen ähnlichen Ansatz verwenden, um das Designsystem als externe Abhängigkeit zu deklarieren, sodass die Logik des Formulars verpackt wird, ohne das Designsystem einzubeziehen.

Mehrere Mikrofrontends können dann dasselbe tun, um gemeinsame Anliegen zu lösen. Wenn Teams neue Komponenten entwickeln, die von mehreren Mikrofrontends gemeinsam genutzt werden können, werden diese Komponenten dem Designsystem hinzugefügt, sobald sie ausgereift sind.

Ein Hauptvorteil des Designsystem-Ansatzes ist das hohe Maß an Konsistenz. Mikro-Frontends können zwar Stile schreiben und gelegentlich solche aus dem Designsystem überschreiben, aber das ist kaum erforderlich. Die wichtigsten Elemente auf niedriger Ebene ändern sich nicht oft und bieten grundlegende Funktionen, die standardmäßig erweiterbar sind. Ein weiterer Vorteil ist die Leistung. Mit einer guten Strategie für die Erstellung und Veröffentlichung können Sie eine minimale Anzahl gemeinsam genutzter Bundles erstellen, die von der Anwendungsshell instrumentiert werden. Sie können sich sogar noch weiter verbessern, wenn mehrere Micro-Frontend-spezifische Bundles bei Bedarf asynchron geladen werden, und das bei minimalem Platzbedarf in Bezug auf die Netzwerkbandbreite. Nicht zuletzt ist das Entwicklererlebnis ideal, da sich die Benutzer darauf konzentrieren können, umfangreiche Benutzeroberflächen zu erstellen, ohne das Rad neu zu erfinden (wie das Schreiben JavaScript und CSS jedes Mal, wenn einer Seite eine Schaltfläche hinzugefügt werden muss).

Der Nachteil ist, dass ein Designsystem jeglicher Art eine Abhängigkeit ist und daher gewartet und manchmal aktualisiert werden muss. Wenn mehrere Mikro-Frontends eine neue Version einer gemeinsamen Abhängigkeit erfordern, können Sie eine der folgenden Optionen verwenden:

- Ein Orchestrierungsmechanismus, der gelegentlich mehrere Versionen dieser gemeinsamen Abhängigkeit ohne Konflikte abrufen kann
- Eine gemeinsame Strategie, um alle abhängigen Benutzer auf eine neue Version umzustellen

Wenn beispielsweise alle Mikro-Frontends von der Version 3.0 eines Designsystems abhängen und es eine neue Version namens 3.1 gibt, die gemeinsam verwendet werden soll, können Sie Feature-Flags für alle Mikrofrontends implementieren, um sie mit minimalem Risiko zu migrieren. Weitere Informationen finden Sie im Abschnitt Feature-Flags. Ein weiterer potenzieller Nachteil besteht darin, dass Designsysteme in der Regel mehr als nur das Styling berücksichtigen. Sie beinhalten auch JavaScript Praktiken und Tools. Diese Aspekte erfordern, dass durch Debatte und Zusammenarbeit ein Konsens erzielt wird.

Die Implementierung eines Designsystems ist eine gute langfristige Investition. Es ist ein beliebter Ansatz und sollte von jedem in Betracht gezogen werden, der an einer komplexen Frontend-Architektur arbeitet. In der Regel müssen Frontend-Ingenieure sowie Produkt- und Designteams zusammenarbeiten und Mechanismen definieren, um miteinander zu interagieren. Es ist wichtig, Zeit einzuplanen, bis der gewünschte Zustand erreicht ist. Es ist auch wichtig, von der Unternehmensleitung unterstützt zu werden, damit die Mitarbeiter etwas aufbauen können, das zuverlässig, gut gewartet und langfristig leistungsfähig ist.

Vollständig gekapseltes CSS – Ein Share-Nothing-Ansatz

Jedes Mikro-Frontend verwendet Konventionen und Tools, um die kaskadierende Funktion von CSS zu überwinden. Ein Beispiel ist die Sicherstellung, dass der Stil jedes Elements immer mit einem Klassennamen und nicht mit der ID des Elements verknüpft ist und Klassennamen immer eindeutig sind. Auf diese Weise ist alles auf individuelle Mikrofrontends zugeschnitten und das Risiko unerwünschter Konflikte wird minimiert. Die Anwendungsshell ist in der Regel dafür zuständig, die Stile der Mikrofrontends zu laden, nachdem sie in das DOM geladen wurden. Einige Tools bündeln die Stile jedoch mithilfe von JavaScript

Der Hauptvorteil, wenn nichts geteilt wird, ist das geringere Risiko, dass Konflikte zwischen Mikrofrontends entstehen. Ein weiterer Vorteil ist die Erfahrung des Entwicklers. Jedes Mikro-Frontend hat nichts mit anderen Micro-Frontends gemeinsam. Isoliertes Releasen und Testen ist einfacher und schneller.

Ein Hauptnachteil des Share-Nothing-Ansatzes ist der potenzielle Mangel an Konsistenz. Es gibt kein System zur Bewertung der Kohärenz. Auch wenn das Ziel darin besteht, gemeinsam genutzte Inhalte zu duplizieren, wird es schwierig, die Geschwindigkeit der Veröffentlichung und die Zusammenarbeit unter einen Hut zu bringen. Eine gängige Abhilfemaßnahme besteht darin, Tools zur Messung der Konsistenz zu entwickeln. Sie können beispielsweise ein System erstellen, mit dem automatisierte Screenshots mehrerer Mikrofrontends erstellt werden, die auf einer Seite mit einem Headless-Browser gerendert werden. Sie können die Screenshots dann vor einer Veröffentlichung manuell überprüfen. Dies erfordert jedoch Disziplin und Steuerung. Weitere Informationen finden Sie im Abschnitt Balance zwischen Autonomie und Ausrichtung.

Je nach Anwendungsfall ist ein weiterer potenzieller Nachteil die Leistung. Wenn von allen Mikrofrontends eine große Menge an Styling verwendet wird, muss der Kunde viel doppelten Code herunterladen. Das wird sich negativ auf die Benutzererfahrung auswirken.

Dieser Share-Nothing-Ansatz sollte nur für Mikro-Frontend-Architekturen in Betracht gezogen werden, an denen nur wenige Teams beteiligt sind, oder für Mikro-Frontends, die eine geringe

Konsistenz vertragen. Es kann auch ein natürlicher erster Schritt sein, während eine Organisation an einem Designsystem arbeitet.

Gemeinsames globales CSS – Ein Ansatz, bei dem alle gemeinsam genutzt werden

Bei diesem Ansatz wird der gesamte Code, der sich auf das Styling bezieht, in einem zentralen Repository gespeichert, in dem die Mitwirkenden CSS für alle Mikrofrontends schreiben, indem sie an CSS-Dateien arbeiten oder Präprozessoren wie Sass verwenden. Wenn Änderungen vorgenommen werden, erstellt ein Build-System ein einzelnes CSS-Bundle, das in einem CDN gehostet und von der Anwendungsshell in jedes Mikro-Frontend aufgenommen werden kann. Micro-Frontend-Entwickler können ihre Anwendungen entwerfen und erstellen, indem sie ihren Code über eine lokal gehostete Anwendungsshell ausführen.

Neben dem offensichtlichen Vorteil, das Risiko von Konflikten zwischen Mikrofrontends zu verringern, liegen die Vorteile dieses Ansatzes in Konsistenz und Leistung. Durch die Entkopplung von Stilen von Markup und Logik ist es für Entwickler jedoch schwieriger zu verstehen, wie Stile verwendet werden, wie sie sich weiterentwickeln können und wie sie veraltet sein können. Zum Beispiel könnte es schneller sein, einen neuen Klassennamen einzuführen, als etwas über die bestehende Klasse und die Folgen der Bearbeitung ihrer Eigenschaften zu erfahren. Zu den Nachteilen der Erstellung eines neuen Klassennamens gehören die Zunahme der Paketgröße, was sich auf die Leistung auswirkt, und die Gefahr von Inkonsistenzen in der Benutzererfahrung.

Ein gemeinsames globales CSS kann zwar der Ausgangspunkt einer monolith-to-micro-frontends Migration sein, ist aber für Mikro-Frontend-Architekturen, bei denen mehr als ein oder zwei Teams zusammenarbeiten, selten von Vorteil. Wir empfehlen, so bald wie möglich in ein Designsystem zu investieren und während der Entwicklung des Designsystems einen Share-Nothing-Ansatz zu implementieren.

Organisation und Arbeitsweise

Wie bei allen Architekturstrategien haben Mikro-Frontends Auswirkungen, die weit über die Technologie hinausgehen, für deren Implementierung sich ein Unternehmen entscheidet. Die Entscheidung, Mikro-Frontend-Anwendungen zu entwickeln, muss auf das Geschäft, das Produkt, die Organisation, den Betrieb und sogar die Kultur abgestimmt sein (z. B. die Stärkung von Teams und die Dezentralisierung der Entscheidungsfindung). Im Gegenzug unterstützt diese Art von Micro-Frontend-Architektur eine wirklich agile, produktorientierte Entwicklung, da sie den Kommunikationsaufwand zwischen ansonsten unabhängigen Teams erheblich reduziert.

Agile Entwicklung

Die Idee der agilen Softwareentwicklung ist in den letzten Jahren so allgegenwärtig geworden, dass praktisch jedes Unternehmen behauptet, agil zu arbeiten. Eine abschließende Definition von Agilität würde zwar den Rahmen dieser Strategie sprengen, es lohnt sich jedoch, die wichtigsten Elemente zu überprüfen, die für die Mikro-Frontend-Entwicklung relevant sind.

Die Grundlage des agilen Paradigmas ist das Agile Manifest (2001), das vier Hauptprinzipien (zum Beispiel „Individuen und Interaktionen über Prozesse und Tools“) und zwölf Prinzipien postulierte. Prozess-Frameworks wie Scrum und das Scaled Agile Framework (SAFe) sind rund um das Agile Manifest entstanden und haben ihren Weg in die tägliche Praxis gefunden. Die Philosophie, die ihnen zugrunde liegt, wird jedoch weitgehend missverstanden oder ignoriert.

Im Kontext der Micro-Frontend-Architektur ist es wichtig, die folgenden agilen Prinzipien zu berücksichtigen:

- „Liefern Sie regelmäßig funktionierende Software, von ein paar Wochen bis zu ein paar Monaten, und bevorzugen Sie dabei kürzere Zeiträume.“

  Dieses Prinzip unterstreicht, wie wichtig es ist, schrittweise zu arbeiten und Software so regelmäßig und so oft wie möglich für die Produktion bereitzustellen. Aus technologischer Sicht bezieht sich dies auf kontinuierliche Integration und kontinuierliche Lieferung (CI/CD). Bei CI/CD sind die Tools und Prozesse zum Erstellen, Testen und Bereitstellen integraler Bestandteil jedes Softwareprojekts. Das Prinzip beinhaltet auch, dass die Laufzeitinfrastruktur und die operativen Verantwortlichkeiten dem Team obliegen müssen. Diese Eigenverantwortung ist besonders wichtig in verteilten Systemen, in denen unabhängige Subsysteme möglicherweise erheblich unterschiedliche Anforderungen an Infrastruktur und Betrieb stellen.

- „Bauen Sie Projekte rund um motivierte Einzelpersonen auf. Geben Sie ihnen das Umfeld und die Unterstützung, die sie benötigen, und vertrauen Sie darauf, dass sie ihre Arbeit erledigen.“

  „Die besten Architekturen, Anforderungen und Designs entstehen durch sich selbst organisierende Teams.“

  Beide Prinzipien betonen die Vorteile von Eigenverantwortung, Unabhängigkeit und end-to-end Verantwortung. Eine Mikro-Frontend-Architektur wird erfolgreich sein, wenn (und nur wenn) die Teams ihre Micro-Frontends wirklich selbst in der Hand haben. Die nd-to-end Verantwortung von der Konzeption über das Design und die Implementierung bis hin zu Lieferung und Betrieb stellt sicher, dass das Team tatsächlich Verantwortung übernehmen kann. Diese Unabhängigkeit ist sowohl technisch als auch organisatorisch erforderlich, damit das Team die strategische Ausrichtung selbst bestimmen kann. Wir empfehlen nicht, eine Micro-Frontend-Plattform in einer zentralisierten Organisation zu verwenden, die ein Wasserfall-Entwicklungsmodell verwendet.

Zusammensetzung und Größe des Teams

Damit ein Softwareteam Verantwortung übernehmen kann, muss es sich innerhalb der von der Organisation auferlegten Grenzen selbst verwalten, einschließlich der Art und Weise, wie und was das Team leistet.

Um effektiv zu sein, müssen Teams in der Lage sein, Software unabhängig zu liefern, und die Befugnis haben, zu entscheiden, wie sie am besten bereitgestellt wird. Ein Team, das Funktionsanforderungen von einem externen Produktmanager oder UI-Designs von einem externen Designer erhält, ohne an der Planung dieser Elemente beteiligt zu sein, kann nicht als eigenständig betrachtet werden. Die Funktionen könnten gegen bestehende Verträge oder Funktionen verstoßen. Solche Verstöße erfordern weitere Diskussionen und Verhandlungen, wobei die Gefahr besteht, dass sich die Umsetzung verzögert und unnötige Konflikte zwischen den Teams entstehen.

Gleichzeitig sollten die Teams nicht zu groß werden. Während ein größeres Team über mehr Ressourcen verfügt und individuelle Abwesenheiten bewältigen kann, nimmt die Komplexität der Kommunikation mit jedem neuen Mitglied exponentiell zu. Es ist nicht möglich, eine allgemein gültige maximale Teamgröße anzugeben. Die Anzahl der Personen, die für ein Projekt benötigt werden, hängt von Faktoren wie Teamreife, technischer Komplexität, Innovationstempo und Infrastruktur ab. Amazon folgt beispielsweise der Zwei-Pizza-Regel: Ein Team, das zu groß ist, um mit zwei Pizzen gefüttert zu werden, sollte in kleinere Teams aufgeteilt werden. Das kann eine Herausforderung sein. Die Aufteilung sollte entlang natürlicher Grenzen erfolgen und jedem Team Autonomie und Eigenverantwortung für seine Arbeit geben.

DevOps Kultur

DevOps bezieht sich auf eine Praxis der Softwareentwicklung, bei der die Schritte des Entwicklungszyklus aus organisatorischer und technischer Sicht eng miteinander verknüpft sind. Entgegen der landläufigen Meinung geht DevOps es hier sehr viel um Kultur und Denkweise und sehr wenig um Rollen und Tools.

Traditionell verfügte eine Softwareorganisation über Spezialistenteams, z. B. für Design, Implementierung, Tests, Bereitstellung und Betrieb. Immer wenn ein Team seine Arbeit abschloss, übergab es das Projekt an das nächste Team. Die Bereitstellung von Software durch Silos spezialisierter Teams führt jedoch zu Reibungsverlusten bei der Übergabe. Wenn Spezialisten gezwungen sind, mit einem engen Fokus zu arbeiten, fehlt es ihnen gleichzeitig an Wissen in benachbarten Bereichen und sie haben keine systemische Sicht auf das Produkt. Diese Defizite können zu einer geringen Kohärenz des Softwareprodukts führen.

Wenn ein Softwarearchitekt beispielsweise eine Lösung entwirft, die von jemandem in einem anderen Team implementiert werden soll, übersieht er möglicherweise inhärente Aspekte der Implementierung (z. B. ein Missverhältnis zwischen Abhängigkeiten). Entwickler verwenden dann Abkürzungen (z. B. einen Monkey Patch), oder es back-and-forth wird eine formelle Vereinbarung zwischen dem Architekten und dem Entwicklungsteam eingeleitet. Aufgrund des hohen Verwaltungsaufwands für diese Prozesse ist die Entwicklung nicht mehr agil (im Sinne von flexibel, adaptiv, inkrementell und informell).

Obwohl sich der Begriff DevOps hauptsächlich auf Kultur bezieht, beinhaltet er die Technologien und Prozesse, die in der Praxis DevOps möglich sind. DevOps ist eng mit CI/CD verwandt. Wenn ein Entwickler die Implementierung eines Softwareinkrements abgeschlossen hat, überträgt er es an ein Versionskontrollsystem wie Git. Traditionell würde ein Build-System dann die Software erstellen und integrieren und sie in einem mehr oder weniger einheitlichen und zentralisierten Prozess testen und veröffentlichen lassen. Bei CI/CD erfolgt die Erstellung, Integration, Prüfung und Veröffentlichung der Software inhärent und automatisiert. Im Idealfall ist der Prozess mithilfe von Konfigurationsdateien, die speziell auf das jeweilige Projekt zugeschnitten sind, Teil des Softwareprojekts selbst.

So viele Schritte wie möglich werden automatisiert. Beispielsweise sollten manuelle Testpraktiken reduziert werden, da fast alle Arten von Tests automatisiert werden können. Wenn das Projekt auf diese Weise eingerichtet ist, können Updates für ein Softwareprodukt mehrmals täglich mit hoher Zuverlässigkeit bereitgestellt werden. Eine weitere Technologie, die dies unterstützt, DevOps ist Infrastructure as Code (IaC).

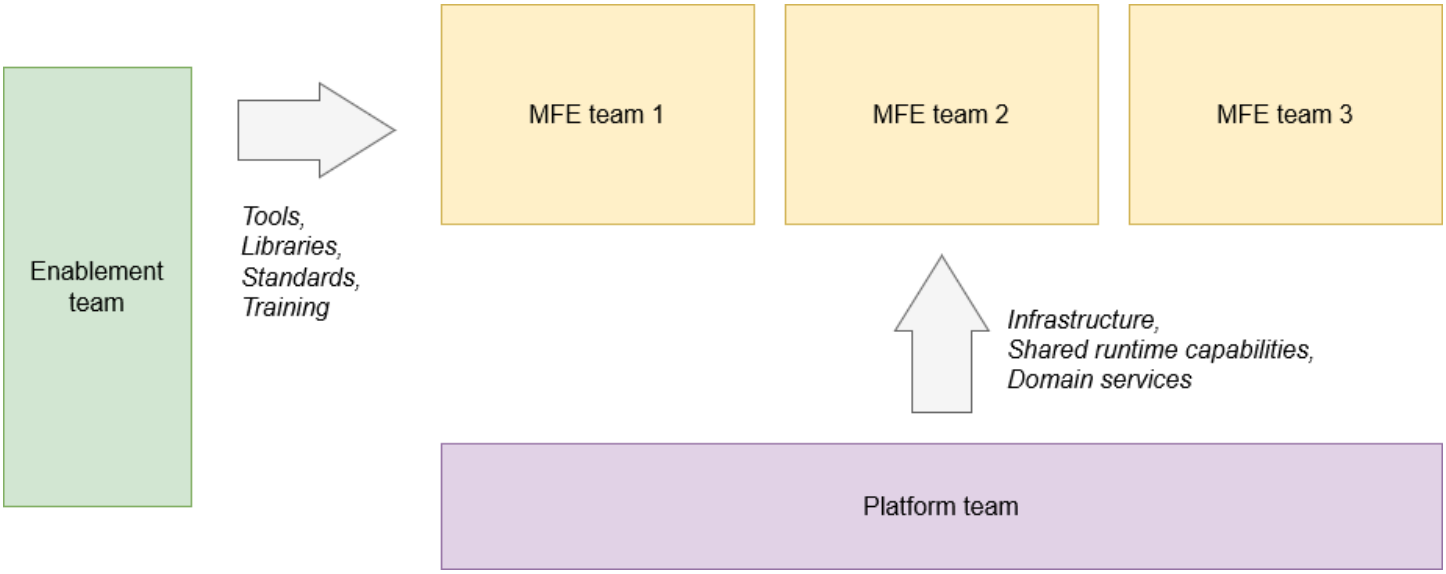
Traditionell erforderten die Einrichtung und Wartung der IT-Infrastruktur die manuelle Installation und Wartung von Hardware (Einrichtung von Kabeln und Servern in einem Rechenzentrum) und Betriebssoftware. Das war notwendig, hatte aber zahlreiche Nachteile. Die Einrichtung war zeitaufwändig und fehleranfällig. Hardware war oft über- oder unterausgestattet, was entweder zu hohen Ausgaben oder zu Leistungseinbußen führte. Mithilfe von IaC können Sie die Infrastrukturanforderungen für ein IT-System anhand einer Konfigurationsdatei beschreiben, aus der Cloud-Dienste automatisch bereitgestellt und aktualisiert werden können.

Was hat das alles mit Mikro-Frontends zu tun? DevOps, CI/CD und IaC sind ideale Ergänzungen zu einer Micro-Frontend-Architektur. Die Vorteile von Micro-Frontends beruhen auf schnellen und reibungslosen Lieferprozessen. Eine DevOps Unternehmenskultur kann nur in Umgebungen gedeihen, in denen Teams verantwortungsbewusst Softwareprojekte übernehmen. end-to-end

Orchestrierung der Mikro-Frontend-Entwicklung in mehreren Teams

Bei der Skalierung der Mikro-Frontend-Entwicklung auf mehrere funktionsübergreifende Teams treten zwei Probleme auf: Erstens beginnen die Teams, ihre eigene Interpretation des Paradigmas zu entwickeln, Framework- und Bibliotheksentscheidungen zu treffen und ihre eigenen Tools und Hilfsbibliotheken zu erstellen. Zweitens müssen vollständig autonome Teams die Verantwortung für generische Funktionen wie das Infrastrukturmanagement auf niedriger Ebene übernehmen. Daher ist es sinnvoll, zwei zusätzliche Teams in einer Micro-Frontend-Organisation mit mehreren Teams einzuführen: ein Enablement-Team und ein Plattformteam. Diese Konzepte sind in modernen IT-Organisationen mit verteilten Systemen weit verbreitet und in Teamtopologien ausführlich dokumentiert.

Das folgende Diagramm zeigt, wie das Enablement-Team drei Micro-Frontend-Teams Tools, Bibliotheken, Standards und Tests zur Verfügung stellt. Das Plattformteam stellt Infrastruktur, gemeinsame Runtime-Funktionen und Domain-Services für dieselben drei Micro-Frontend-Teams bereit.



Das Plattformteam unterstützt die Micro-Frontend-Teams, indem es sie von undifferenzierter Schwerstarbeit befreit. Diese Unterstützung kann Infrastrukturdienste wie Container-Laufzeiten, CI/CD-Pipelines, Tools für die Zusammenarbeit und Überwachung umfassen. Die Einrichtung eines Plattformteams sollte jedoch nicht zu einer Organisation führen, in der die Entwicklung vom Betrieb getrennt ist. Das Gegenteil ist der Fall: Das Plattformteam bietet ein technisches Produkt an, und die Micro-Frontend-Teams sind für ihre Dienste auf der Plattform verantwortlich und tragen die Verantwortung für deren Laufzeit.

Das Enablement-Team bietet Unterstützung, indem es sich auf die Steuerung konzentriert und die Konsistenz zwischen den Micro-Frontend-Teams sicherstellt. (Das Plattformteam sollte daran nicht beteiligt sein.) Das Enablement-Team verwaltet gemeinsam genutzte Ressourcen wie eine UI-Bibliothek und erstellt Standards wie die Wahl des Frameworks, Leistungsbudgets und Interoperabilitätskonventionen. Gleichzeitig werden neue Teams oder Teammitglieder darin geschult, die von der Unternehmensleitung festgelegten Standards und Tools anzuwenden.

Bereitstellen

Das Nonplusultra für Autonomie in Micro-Frontend-Teams ist eine automatisierte Pipeline mit einem Weg zur Produktion, der unabhängig von anderen Micro-Frontend-Teams ist. Teams, die dem Share-Nothing-Prinzip folgen, können unabhängige Pipelines implementieren. Teams, die Bibliotheken gemeinsam nutzen oder von einem Plattformteam abhängig sind, müssen entscheiden, wie Abhängigkeiten in Bereitstellungspipelines verwaltet werden sollen.

In der Regel macht jede Pipeline Folgendes:

- Baut Frontend-Assets auf
- Stellt die Ressourcen zur Nutzung auf dem Hosting bereit
- Stellt sicher, dass Registries und Caches aktualisiert werden, sodass die neuen Versionen an Kunden ausgeliefert werden können

Die tatsächlichen Pipeline-Schritte variieren je nach Technologie-Stack und Ansatz zur Seitenzusammensetzung.

Für die clientseitige Zusammenstellung bedeutet dies, dass Anwendungspakete in Hosting-Buckets hochgeladen und durch Caching in einem CDN für den Nutzer freigegeben werden. Anwendungen, die Browser-Caching mit Servicemitarbeitern verwenden, sollten auch Methoden zur Aktualisierung der Servicemitarbeiter-Caches implementieren.

Bei serverseitiger Zusammenstellung bedeutet dies in der Regel, die neue Version der Serverkomponente bereitzustellen und die Micro-Frontend-Registrierung zu aktualisieren, damit die neue Version auffindbar ist. Sie können die Bereitstellungsmuster Blau/Grün oder Canary verwenden, um neue Versionen schrittweise einzuführen.

Governance

In der Regel arbeiten mehrere Personen an Mikro-Frontends, und jede Person arbeitet unter unterschiedlichen Einschränkungen, um gemeinsame Geschäftsziele zu erreichen. Kommunikation und Zusammenarbeit zwischen Menschen sind zwar der Schlüssel zum Erfolg, aber zu viel Kommunikation und Implementierung übermäßig komplizierter Prozesse verlangsamen den Entwicklungszyklus. Dies führt zu einer verminderten Arbeitsmoral und senkt die Qualitätsmaßstäbe.

Die erfolgreichsten Unternehmen, die Mikro-Frontends mithilfe mehrerer Teams implementieren, schaffen Mechanismen, um Autonomie und Abstimmung in Einklang zu bringen. Sie ermöglichen es Entscheidungsträgern, lokal zu handeln und nur dann hierarchisch zu eskalieren, wenn dies erforderlich ist. Zu den Mechanismen gehören die folgenden:

- API-Verträge
- Interaktive Nutzung von Ereignissen
- Balance zwischen Autonomie und Ausrichtung
- Feature-Flaggen
- Serviceerkennung

API-Verträge

Jedes Mikro-Frontend ist ein System, das Meinungen, Logik und Komplexität zusammenfassen kann. Zu den bereichsübergreifenden Anliegen gehören in der Regel die folgenden:

- Designsysteme – Tools zur Entwicklung von Benutzeroberflächen, die als Bibliotheken vertrieben werden
- Zusammensetzung – Die Art und Weise, wie ein Mikro-Frontend mit einer Anwendungsshell interagieren muss, um ihren Kontext zu rendern und zu erben
- Umgang mit Logik – Interaktion mit APIs zur Behandlung von persistenten Zuständen
- Interaktivität mit anderen Mikrofrontends – Szenarien wie die Veröffentlichung und Nutzung von Ereignissen oder das Navigieren von einem Mikrofrontend zum anderen

Um die Nutzung und Problembehebung zu beschleunigen, ist es üblich, in die Standardisierung der Art und Weise zu investieren, wie diese Schnittstellen deklariert und dokumentiert werden, einschließlich der Abhängigkeiten von Mikrofrontends. Von Menschen kuratierte Wikis sind ein guter

Anfang. Ein skalierbarerer Ansatz besteht darin, diese Informationen als strukturierte Metadaten im Code zu speichern. Sie können sie dann für die Nutzung zentralisieren, indem Sie mithilfe von Automatisierung historische Änderungen verfolgen und eine Volltextsuche bereitstellen.

Wenn an Mikro-Frontends eine große Anzahl von Teams beteiligt ist, benötigen Sie eine Strategie zur Koordination zwischen den Teams. Die einheitliche gemeinsame Nutzung von API-Verträgen wird zu einem Muss, da dadurch der Kommunikationsaufwand reduziert und die Entwicklererfahrung verbessert wird.

OpenAPI ist eine Spezifikationssprache für HTTP-APIs, die die einheitliche Definition von API-Schnittstellen und Verträgen unterstützt. Sie können REST-APIs mithilfe von OpenAPI in Amazon API Gateway implementieren. Sie können auch eine Vielzahl von Open-Source-Frameworks verwenden, die Sie in Containern oder virtuellen Maschinen hosten können. Ein wesentlicher Vorteil besteht darin, dass OpenAPI automatisch Dokumentation in einem konsistenten Format generieren kann, sodass mehrere Teams ihr Wissen mit einer minimalen Anfangsinvestition austauschen können.

Wenn mehrere Teams an Mikro-Frontends arbeiten, bilden sie oft Gruppen. In diesen Gruppen können sich Menschen treffen und voneinander lernen, während sie gleichzeitig über das Gesamtbild nachdenken und dazu beitragen. In diesen Initiativen werden in der Regel Eigentumsgrenzen definiert und dokumentiert, bereichsübergreifende Anliegen erörtert und Doppelarbeit bei der Lösung gemeinsamer Probleme frühzeitig erkannt.

Interaktive Nutzung von Ereignissen

In einigen Szenarien müssen möglicherweise mehrere Mikrofrontends miteinander interagieren, um auf Statusänderungen oder Benutzeraktionen zu reagieren. Beispielsweise können mehrere Mikrofrontends auf der Seite zusammenklappbare Menüs enthalten. Ein Menü erscheint, wenn der Benutzer eine Schaltfläche auswählt. Das Menü wird ausgeblendet, wenn der Benutzer auf eine andere Stelle klickt, einschließlich eines anderen Menüs, das in einem anderen Mikrofrontend gerendert wird.

Technisch gesehen kann eine gemeinsam genutzte Statusbibliothek wie Redux von mehreren Mikrofrontends verwendet und von einer Shell koordiniert werden. Dies führt jedoch zu einer erheblichen Kopplung zwischen Anwendungen, was dazu führt, dass Code schwieriger zu testen ist und die Leistung beim Rendern beeinträchtigen kann.

Ein gängiger, effektiver Ansatz besteht darin, einen Event-Bus zu entwickeln, der als Bibliothek verteilt, von der Anwendungsshell orchestriert und von mehreren Mikro-Frontends verwendet wird. Auf diese Weise veröffentlicht und überwacht jedes Mikro-Frontend bestimmte Ereignisse asynchron,

wobei sein Verhalten nur auf seinem eigenen internen Status basiert. Anschließend können mehrere Teams eine gemeinsame Wiki-Seite verwalten, auf der Ereignisse beschrieben und Verhaltensweisen dokumentiert werden, auf die sich die User Experience Designer geeinigt haben.

In einer Implementierung des Event-Bus-Beispiels verwendet eine Dropdownkomponente den gemeinsam genutzten Bus, um ein Ereignis zu veröffentlichen, das `drop-down-open-menu` mit einer Nutzlast von aufgerufen wurde. `{"id": "homepage-aboutus-button"}` Die Komponente fügt dem Ereignis einen Listener hinzu, um sicherzustellen, dass, wenn ein `drop-down-open-menu` Ereignis für eine neue ID ausgelöst wird, die Dropdownkomponente so gerendert wird, dass ihr zusammenklappbarer Abschnitt ausgeblendet wird. Auf diese Weise kann das Mikro-Frontend asynchron auf Änderungen reagieren, was die Leistung erhöht und die Kapselung verbessert, was es mehreren Teams erleichtert, Verhaltensweisen zu entwerfen und zu testen.

Wir empfehlen die Verwendung von Standard-APIs, die von modernen Browsern nativ implementiert werden, um die Einfachheit und Wartbarkeit zu verbessern. Die MDN-Ereignisreferenz enthält Informationen zur Verwendung von Ereignissen mit clientseitig gerenderten Anwendungen.

Balance zwischen Autonomie und Ausrichtung

Mikro-Frontend-Architekturen sind stark auf Teamautonomie ausgerichtet. Es ist jedoch wichtig, zwischen Bereichen zu unterscheiden, die Flexibilität und unterschiedliche Lösungsansätze unterstützen können, und Bereichen, in denen Standardisierung erforderlich ist, um eine Abstimmung zu erreichen. Führungskräfte und Architekten müssen diese Bereiche frühzeitig identifizieren und Investitionen priorisieren, um ein Gleichgewicht zwischen Sicherheit, Leistung, betrieblicher Exzellenz und Zuverlässigkeit von Mikrofrontends zu finden. Um dieses Gleichgewicht zu finden, sind folgende Aufgaben erforderlich: Erstellung, Testen, Veröffentlichung und Protokollierung, Überwachung und Alarmierung von Mikro-Frontends.

Erstellung von Mikro-Frontends

Im Idealfall sind alle Teams darauf ausgerichtet, die Vorteile in Bezug auf die Leistung der Endbenutzer zu maximieren. In der Praxis kann dies schwierig sein und möglicherweise mehr Aufwand erfordern. Wir empfehlen, mit einigen schriftlichen Richtlinien zu beginnen, zu denen mehrere Teams im Rahmen einer offenen und transparenten Debatte beitragen können. Teams können dann schrittweise das Cookiecutter-Softwarepuster übernehmen, das die Entwicklung von Tools unterstützt, die eine einheitliche Methode zur Gestaltung eines Projekts bieten.

Mit diesem Ansatz können Sie Meinungen und Einschränkungen berücksichtigen. Der Nachteil ist, dass diese Tools erhebliche Investitionen in die Erstellung und Wartung erfordern und sicherstellen

müssen, dass Hindernisse schnell behoben werden, ohne die Produktivität der Entwickler zu beeinträchtigen.

nd-to-end E-Tests für Mikro-Frontends

Unit-Tests können den Besitzern überlassen werden. Wir empfehlen, frühzeitig eine Strategie zu implementieren, um Mikro-Frontends, die auf einer einzigartigen Shell laufen, übergreifend zu testen. Die Strategie beinhaltet die Möglichkeit, Anwendungen vor und nach einer Produktionsversion zu testen. Wir empfehlen, Prozesse und Dokumentationen für technische und nichttechnische Mitarbeiter zu entwickeln, um kritische Funktionen manuell testen zu können.

Es ist wichtig sicherzustellen, dass Änderungen weder das funktionale noch das nicht funktionierende Kundenerlebnis beeinträchtigen. Eine ideale Strategie besteht darin, schrittweise in automatisierte Tests zu investieren, sowohl für wichtige Funktionen als auch für Architekturmerkmale wie Sicherheit und Leistung.

Veröffentlichung von Mikro-Frontends

Jedes Team hat möglicherweise seine eigene Methode, seinen Code bereitzustellen, Meinungen einzubeziehen und seine eigene Infrastruktur zu nutzen. Die Kosten der Komplexität bei der Wartung solcher Systeme wirken in der Regel abschreckend. Stattdessen empfehlen wir, frühzeitig in die Umsetzung einer gemeinsamen Strategie zu investieren, die durch gemeinsam genutzte Tools durchgesetzt werden kann.

Entwickeln Sie Vorlagen mit der CI/CD-Plattform Ihrer Wahl. Teams können dann die vorab genehmigten Vorlagen und die gemeinsame Infrastruktur verwenden, um Änderungen an der Produktion zu veröffentlichen. Sie können frühzeitig damit beginnen, in diese Entwicklungsarbeit zu investieren, da diese Systeme nach einer anfänglichen Test- und Konsolidierungsphase selten nennenswerte Updates benötigen.

Protokollierung und Überwachung

Jedes Team kann über unterschiedliche Geschäfts- und Systemkennzahlen verfügen, die es für betriebliche oder analytische Zwecke verfolgen möchte. Das Cookiecutter-Softwaremuster kann auch hier angewendet werden. Die Bereitstellung von Ereignissen kann abstrakt und als Bibliothek zur Verfügung gestellt werden, die von mehreren Mikrofrontends genutzt werden kann. Um ein Gleichgewicht zwischen Flexibilität und Autonomie zu erreichen, sollten Sie Tools für die Protokollierung benutzerdefinierter Metriken und die Erstellung benutzerdefinierter Dashboards

oder Berichte entwickeln. Die Berichterstattung fördert die enge Zusammenarbeit mit den Produkteigentümern und reduziert die Feedback-Schleife der Endkunden.

Durch die Standardisierung der Bereitstellung können mehrere Teams zusammenarbeiten, um Kennzahlen zu verfolgen. Beispielsweise kann eine E-Commerce-Website die Benutzerreise vom Mikrofrontend „Produktdetails“ über das Mikrofrontend „Warenkorb“ bis hin zum Mikrofrontend „Kauf“ verfolgen, um Interaktionen, Kundenabwanderung und Probleme zu messen. Wenn jedes Mikro-Frontend Ereignisse mithilfe einer einzigen Bibliothek protokolliert, können Sie diese Daten als Ganzes nutzen, sie ganzheitlich untersuchen und aufschlussreiche Trends identifizieren.

Warnfunktion

Ähnlich wie bei der Protokollierung und Überwachung profitieren Warnmeldungen von einer Standardisierung, die ein gewisses Maß an Flexibilität bietet. Verschiedene Teams reagieren möglicherweise unterschiedlich auf funktionale und nicht funktionale Warnmeldungen. Wenn jedoch alle Teams über eine konsolidierte Methode verfügen, um Warnmeldungen auf der Grundlage von Kennzahlen auszulösen, die auf einer gemeinsamen Plattform gesammelt und analysiert werden, kann das Unternehmen teamübergreifende Probleme identifizieren. Diese Funktion ist bei Ereignissen im Incident-Management nützlich. Warnmeldungen können beispielsweise wie folgt ausgelöst werden:

- Erhöhte Anzahl von JavaScript clientseitigen Ausnahmen in einer bestimmten Browserversion
- Die Zeit bis zum Rendern hat sich bei Überschreitung eines bestimmten Schwellenwerts erheblich verschlechtert
- Erhöhte Anzahl von 5xx-Statuscodes bei der Nutzung einer bestimmten API

Je nach Reifegrad Ihres Systems können Sie Ihre Anstrengungen auf verschiedene Teile Ihrer Infrastruktur verteilen, wie in der folgenden Tabelle dargestellt.

Annahme	Forschung und Entwicklung	Aufstieg	Reife
Erstellen Sie Mikro-Frontends.	Experimentieren Sie, dokumentieren Sie und teilen	Investieren Sie in Tools, um neue Mikro-Frontends zu entwickeln. Evangelisiert die Adoption.	Konsolidieren Sie die Werkzeuge für den Gerüstbau. Drängen Sie auf Akzeptanz.

Annahme	Forschung und Entwicklung	Aufstieg	Reife
	Sie Erkenntnisse.		
Testen Sie Mikro-Frontends von Ende zu Ende.	Implementieren Sie Mechanismen zum manuellen Testen aller zugehörigen Mikrofrontends.	Investieren Sie in Tools für automatisierte Sicherheits- und Leistungstests. Untersuchen Sie Feature-Flags und die Serviceerkennung.	Konsolidieren Sie die Tools für Serviceerkennung, Tests in der Produktion und automatisierte end-to-end Tests.
Veröffentlichen Sie Mikro-Frontends.	Investieren Sie in eine gemeinsame CI/CD-Infrastruktur und automatisierte Releases für mehrere Umgebungen. Verkündet die Adoption.	Konsolidierung der Tools für die CI/CD-Infrastruktur Implementieren Sie manuelle Rollback-Mechanismen. Drängen Sie auf Akzeptanz.	Schaffen Sie Mechanismen zur Einleitung automatisierter Rollbacks zusätzlich zu System- und Geschäftskennzahlen und Warnmeldungen.

Annahme	Forschung und Entwicklung	Aufstieg	Reife
Beobachten Sie die Leistung des Mikro-Frontends.	Investieren Sie in eine gemeinsame Überwachungsinfrastruktur und Bibliothek für die konsistente Protokollierung von System- und Geschäftsereignissen.	Konsolidieren Sie die Tools für Überwachung und Warnmeldungen. Implementieren Sie teamübergreifende Dashboards, um den allgemeinen Zustand zu überwachen und das Incident-Management zu verbessern.	Standardisieren Sie die Protokollierungsschemas. Optimieren Sie im Hinblick auf die Kosten. Implementieren Sie Warnmeldungen auf der Grundlage komplexer Geschäftskennzahlen.

Feature-Flags

Feature-Flags können in Mikro-Frontends implementiert werden, um die Koordination von Tests und Freigaben von Funktionen in mehreren Umgebungen zu erleichtern. Die Feature-Flag-Technik besteht darin, Entscheidungen in einem booleschen Speicher zu zentralisieren und das Verhalten darauf aufbauend zu steuern. Sie wird häufig verwendet, um Änderungen, die bis zu einem bestimmten Zeitpunkt verborgen bleiben können, im Hintergrund zu verbreiten und gleichzeitig neue Versionen für neue Funktionen freizuschalten, die andernfalls blockiert würden, wodurch die Teamgeschwindigkeit reduziert wird.

Stellen Sie sich das Beispiel von Teams vor, die an einer Micro-Frontend-Funktion arbeiten, die an einem bestimmten Datum veröffentlicht wird. Die Funktion ist fertig, muss aber zusammen mit einer Änderung an einem anderen Mikrofrontend veröffentlicht werden, das unabhängig veröffentlicht wird. Eine Blockierung der Veröffentlichung beider Mikrofrontends würde als Anti-Pattern angesehen werden und würde das Risiko erhöhen, wenn sie eingesetzt werden.

Stattdessen können die Teams in einer Datenbank ein boolesches Feature-Flag erstellen, das beide während der Renderzeit verwenden (vielleicht durch einen HTTP-Aufruf an eine gemeinsam genutzte Feature-Flags-API). Die Teams können die Änderung sogar in einer Testumgebung veröffentlichen, in der der boolesche Wert so eingestellt ist, dass er projektübergreifende funktionale und nichtfunktionale `True` Anforderungen verifiziert, bevor mit der Produktion begonnen wird.

Ein weiteres Beispiel für die Verwendung von Feature-Flags ist die Implementierung eines Mechanismus zum Überschreiben des Werts eines Flags, indem ein bestimmter Wert über den `QueryString` Parameter festgelegt oder eine bestimmte Testzeichenfolge in einem Cookie gespeichert wird. Produkteigentümer können an Funktionen iterieren, ohne die Veröffentlichung anderer Funktionen oder Bugfixes bis zum Startdatum zu blockieren. Wenn der Flaggenwert in der Datenbank am angegebenen Datum geändert wird, wird die Änderung sofort in der Produktion sichtbar, ohne dass teamübergreifend koordinierte Releases erforderlich sind. Nach der Veröffentlichung einer Funktion bereinigen die Entwicklungsteams den Code, um das alte Verhalten zu entfernen.

Zu den weiteren Anwendungsfällen gehört die Veröffentlichung eines kontextbasierten Feature-Flag-Systems. Wenn beispielsweise eine einzelne Website Kunden in mehreren Sprachen bedient, ist eine Funktion möglicherweise nur für Besucher eines bestimmten Landes verfügbar. Das Feature-Flag-System kann davon abhängen, dass der Verbraucher den Länderkontext sendet (z. B. durch Verwendung des `Accept-Language` HTTP-Headers), und je nach Kontext kann es zu einem unterschiedlichen Verhalten kommen.

Feature-Flags sind zwar ein leistungsstarkes Tool, um die Zusammenarbeit zwischen Entwicklern und Produktbesitzern zu erleichtern, sie sind jedoch auf die Sorgfalt der Mitarbeiter angewiesen, um eine signifikante Verschlechterung der Codebasis zu vermeiden. Wenn Flags für mehrere Funktionen aktiviert bleiben, kann dies die Komplexität bei der Behebung von Problemen erhöhen, die JavaScript Paketgröße erhöhen und letztendlich technische Schulden anhäufen. Zu den üblichen Maßnahmen zur Schadensbegrenzung gehören:

- Unit-Tests für jedes Feature hinter einem Flag, um die Wahrscheinlichkeit von Fehlern zu verringern, was zu längeren Feedback-Schleifen in den automatisierten CI/CD-Pipelines führen kann, in denen die Tests ausgeführt werden
- Entwicklung von Tools zur Messung der Zunahme der Paketgröße bei Codeänderungen, die bei Code-Reviews abgemildert werden können

AWS bietet eine Reihe von Lösungen zur Optimierung von A/B-Tests am Edge mithilfe von CloudFront Amazon-Funktionen oder Lambda @Edge. Diese Ansätze tragen dazu bei, die Komplexität der Integration einer Lösung oder des vorhandenen SaaS-Produkts, das Sie zur Bestätigung Ihrer Annahmen verwenden, zu reduzieren. Weitere Informationen finden Sie unter A/B-Tests.

Serviceerkennung

Das Frontend-Discovery-Pattern verbessert die Entwicklungserfahrung bei der Entwicklung, dem Testen und der Bereitstellung von Mikro-Frontends. Das Pattern verwendet eine gemeinsam nutzbare Konfiguration, die den Einstiegspunkt von Mikro-Frontends beschreibt. Die gemeinsam nutzbare Konfiguration umfasst auch zusätzliche Metadaten, die mithilfe von Canary-Releases für sichere Bereitstellungen in jeder Umgebung verwendet werden.

Moderne Frontend-Entwicklung beinhaltet die Verwendung einer Vielzahl von Tools und Bibliotheken, um die Modularität während der Entwicklung zu unterstützen. Traditionell bestand dieser Prozess aus der Bündelung von Code in einzelnen Dateien, die in einem CDN gehostet werden konnten, mit dem Ziel, die Netzwerkaufrufe während der Laufzeit auf ein Minimum zu beschränken, einschließlich des anfänglichen Ladens (wenn eine App in einem Browser geöffnet wird) und der Nutzung (wenn ein Kunde Aktionen wie das Auswählen von Schaltflächen oder das Einfügen von Informationen ausführt).

Bündel aufteilen

Mikro-Frontend-Architekturen lösen die Leistungsprobleme, die durch sehr große Pakete verursacht werden, die durch die individuelle Bündelung einer Vielzahl von Funktionen generiert werden. Beispielsweise kann eine sehr große E-Commerce-Website in einer 6-MB-Datei gebündelt werden. JavaScript Trotz der Komprimierung kann sich die Größe dieser Datei negativ auf die Benutzererfahrung beim Laden der App und beim Herunterladen der Datei von einem Edge-optimierten CDN auswirken.

Wenn Sie die App in Startseite, Produktdetails und Warenkorb-Mikrofrontends aufteilen, können Sie mithilfe eines Bündelungsmechanismus drei individuelle 2-MB-Bundles erstellen. Durch diese Änderung kann die Leistung beim ersten Laden um 300 Prozent verbessert werden, wenn Benutzer die Startseite aufrufen. Die Produkt- oder Warenkorb-Mikrofrontend-Pakete werden nur dann asynchron geladen, wenn der Benutzer die Produktseite für einen Artikel besucht und sich für den Kauf entscheidet.

Viele Frameworks und Bibliotheken basieren auf diesem Ansatz, und es gibt sowohl für Kunden als auch für Entwickler Vorteile. Um Geschäftsgrenzen zu identifizieren, die zur Entkopplung von Abhängigkeiten im Code führen können, können Sie verschiedene Geschäftsfunktionen mehreren Teams zuordnen. Die verteilte Eigentümerschaft sorgt für Unabhängigkeit und Agilität.

Wenn Sie Build-Pakete aufteilen, können Sie eine Konfiguration verwenden, um Mikro-Frontends zuzuordnen und die Orchestrierung für das erste Laden und für die Navigation nach dem Laden

zu steuern. Dann kann die Konfiguration während der Laufzeit und nicht während der Build-Zeit verwendet werden. Beispielsweise kann der clientseitige Frontend-Code oder der serverseitige Backend-Code einen ersten Netzwerkaufruf an eine API tätigen, um die Liste der Mikro-Frontends dynamisch abzurufen. Es ruft auch die Metadaten ab, die für die Zusammenstellung und Integration erforderlich sind. Sie können Failover-Strategien und Caching konfigurieren, um Zuverlässigkeit und Leistung zu gewährleisten. Die Zuordnung der Mikro-Frontends trägt dazu bei, dass einzelne Bereitstellungen von Mikro-Frontends von zuvor bereitgestellten Mikro-Frontends, die von einer Shell-App orchestriert werden, auffindbar sind.

Veröffentlichungen auf Canary

Eine Canary-Version ist ein etabliertes und beliebtes Muster für die Bereitstellung von Mikrodiensten. Bei Canary-Releases werden die Zielbenutzer einer Version in mehrere Gruppen aufgeteilt. Release-Änderungen werden schrittweise vorgenommen, im Gegensatz zu einem sofortigen Ersatz (auch bekannt als blaue/grüne Implementierung). Ein Beispiel für eine Release-Strategie von Canary besteht darin, eine neue Änderung für 10 Prozent der Zielbenutzer einzuführen und jede Minute 10 Prozent hinzuzufügen, mit einer Gesamtdauer von 10 Minuten, um 100 Prozent zu erreichen.

Das Ziel einer Canary-Version besteht darin, frühzeitig Feedback zu den Änderungen zu erhalten und das System zu überwachen, um die Auswirkungen von Problemen zu reduzieren. Sobald die Automatisierung eingeführt ist, können Geschäfts- oder Systemkennzahlen durch ein internes System überwacht werden, das die Bereitstellung stoppen oder ein Rollback einleiten kann.

Beispielsweise kann eine Änderung zu einem Fehler führen, der in den ersten Minuten einer Version zu Umsatzeinbußen oder Leistungseinbußen führt. Durch automatisiertes Monitoring kann ein Alarm ausgelöst werden. Mit dem Diensterkennungsmuster kann dieser Alarm die Bereitstellung beenden und sofort rückgängig machen, sodass nur 20 Prozent der Benutzer betroffen sind, anstatt 100 Prozent. Das Unternehmen profitiert vom geringeren Umfang des Problems.

Eine Beispielarchitektur, die DynamoDB als Speicher für die Implementierung einer REST-Admin-API verwendet, finden Sie in der Frontend Service Discovery on AWS-Lösung unter. GitHub Verwenden Sie die AWS CloudFormation Vorlage, um die Architektur in Ihre eigenen CI/CD-Pipelines zu integrieren. Die Lösung umfasst eine REST-Consumer-API zur Integration der Lösung in Ihre Frontend-Anwendungen.

Benötigen Sie ein Plattformteam?

Einige Unternehmen haben ein Team, das für den Besitz und die Wartung von Code, Infrastruktur und Prozessen verantwortlich ist, die von anderen Teams für die Arbeit an Mikro-Frontends übernommen werden. Zu den gemeinsamen Aufgaben gehören:

- Erstellen und verwalten Sie eine CI/CD-Pipeline, die mit Repositorys verwendet werden kann, die Mikro-Frontends enthalten. Erstellen und testen Sie Codeänderungen und veröffentlichen Sie sie in mehreren Umgebungen.
- Erstellen und verwalten Sie Tools im Zusammenhang mit der Beobachtbarkeit, z. B. gemeinsam genutzte Dashboards, Warnmechanismen und Systeme zur Reaktion auf Probleme.
- Erstellen und verwalten Sie gemeinsam genutzte Bibliotheken für die Bearbeitung von Ereignissen, die Nutzung gemeinsam genutzter Dienste und Abhängigkeiten von Drittanbietern.
- Entwickeln und verwalten Sie Tools, die ununterbrochen nicht funktionierende Eigenschaften wie Leistung, Sicherheit und Zuverlässigkeit des Systems überwachen.
- Erstellen und verwalten Sie Designsysteme.
- Erstellen, warten und unterstützen Sie die Anwendungsshell für das Micro-Frontend-System.

Je nach Umfang des Projekts können Sie diese Aufgaben mit einem der folgenden Ansätze verwalten:

- Stellen Sie ein eigenes Plattformteam zusammen, dessen einzige Verantwortung darin besteht, an gemeinsam genutzten Tools zu arbeiten.
- Erstellen Sie eine Gruppe, die sich aus Mitgliedern mehrerer Teams zusammensetzt. Die Gruppenmitglieder teilen ihre Zeit zwischen der Arbeit an Mikro-Frontends und der Arbeit an gemeinsam genutzten Tools auf. Dies wird auch als Tigerteam bezeichnet.

Der Tiger-Team-Ansatz ist zwar eine effektive Methode, um kundenorientiert zu bleiben, aber ein Tiger-Team entwickelt sich oft zu einem Plattformteam, wenn das Projekt an Bedeutung gewinnt und Verantwortung übernimmt. Sowohl für Plattformteams als auch für Tiger-Teams bilden die erfolgreichsten Unternehmen, die an Mikro-Frontends arbeiten, diese Teams, sodass mehrere Personen mit unterschiedlichen Hintergründen und Fähigkeiten dazu beitragen können. Zu den Teammitgliedern können Backend-Ingenieure, Frontend-Ingenieure, User Experience (UX) -Designer und technische Produktmanager gehören. Diese Vielfalt zwingt die Menschen dazu, kontinuierlich gesunde Debatten zu führen und bei der Gestaltung stets auf Einfachheit zu achten.

Nächste Schritte

Dieser Leitfaden befasste sich mit architektonischen und organisatorischen Mustern, Kompromissen bei wichtigen Entscheidungen und Verwaltungsfragen im Zusammenhang mit Mikrofrontends. In den Tabellen sind die Kompromisse der in diesem Dokument erörterten Verfahren anhand der folgenden Dimensionen zusammengefasst:

- Autonomie – Die Fähigkeit jedes Micro-Frontend-Teams, seine Implementierung und Veröffentlichung für Endbenutzer unabhängig weiterzuentwickeln.
- Konsistenz – Die Gesamterfahrung der Anwendung, bei der sich jedes Mikro-Frontend wie erwartet verhält. Hohe Konsistenz bedeutet, dass Mikro-Frontends mit dem Rest der Anwendung konsistent sind und sich nicht negativ auf die Benutzererfahrung der gesamten Anwendung auswirken.
- Komplexität – Der Umfang der Infrastruktur, des Codes und des Aufwands, der für die Implementierung und das Testen von Mikro-Frontends, der Gesamtanwendung und der Verwaltungskontrollen erforderlich ist.

Praxis	Autonomie	Konsistenz	Komplexität
Bauen mit Mikrofrontends statt mit monolithischen Anwendungen	Hoch	Medium	Hoch

Praktiker der gemeinsa en Nutzung von Code	Autonomie	Konsistenz	Komplexität
Nichts teilen	Hoch	Niedrig	Niedrig
Teilen Sie bereichsü bergreife nde Anliegen	Medium	Hoch	Mittelschwer
Teilen Sie die Geschäftl logik	Niedrig	Hoch	Mittelschwer
Teilen Sie Inhalte während der Erstellun g über Bibliothe ken	Medium	Hoch	Niedrig
Zur Laufzeit teilen	Hoch	Hoch	Hoch

Methode zur Erkennu von Mikrofror tends	Autonomie	Konsistenz	Komplexität
Konfigura tion während der Anwendu serstellu ng	Niedrig	Hoch	Niedrig
Serverse tige Erkennu	Hoch	Hoch	Mittelschwer
Clientsei tige Erkennu (Laufzeit)	Hoch	Hoch	Mittelschwer

Komposi onsprakt ken anzeiger	Autonomie	Konsistenz	Komplexität
Serverse tige Zusamm etzung	Hoch	Medium	Hoch

Kompositionspraktiken anzeiger	Autonomie	Konsistenz	Komplexität
Randseitige Zusammensetzung	Mittelschwer	Medium	Hoch
Clientseitige Zusammensetzung	Hoch	Mittelschwer	Mittelschwer

Weitere Informationen zu den in diesem Leitfaden vorgestellten Konzepten finden Sie im Abschnitt Ressourcen.

Ressourcen

- Mikro-Frontends im Kontext
- Domänengesteuertes Design
- EDA-Bildmaterial
- Frontend-Entdeckung
- Frontend Service Discovery aktiviert AWS
- Agiles Manifest
- MDN-Ereignisreferenz
- OpenAPI

Mitwirkende

Die folgenden Personen haben zu diesem Leitfaden beigetragen.

- Matteo Figus, leitender Lösungsarchitekt, AWS
- Alexander Guensche, leitender Lösungsarchitekt, AWS
- Harun Hasdal, leitender Lösungsarchitekt, AWS
- Luca Mezzalira, Principal Go-to-Market Specialist Solutions Architect Serverless UK, AWS

Dokumentverlauf

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

Änderung	Beschreibung	Datum
Erste Veröffentlichung	—	12. Juli 2024

AWS Glossar zu Prescriptive Guidance

Die folgenden Begriffe werden häufig in Strategien, Leitfäden und Mustern verwendet, die von AWS Prescriptive Guidance bereitgestellt werden. 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 On-Premises-Oracle-Datenbank zu der SQL Postgre-kompatible Amazon-Aurora-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 On-Premises-Oracle-Datenbank zu Amazon Relational Database Service (AmazonRDS) für Oracle im. 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 (CRM) 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 On-Premises-Oracle-Datenbank zu Oracle auf einer EC2 Instance im 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 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.

abstrakte Dienste

Siehe Managed Services.

ACID

Siehe Atomizität, Konsistenz, Isolierung, 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. Sie ist flexibler, erfordert aber mehr Arbeit als die aktiv-passive Migration.

Aktiv-Passiv-Migration

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

Aggregationsfunktion

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 Leitfaden zur Betriebsintegration. 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 ABACfür 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 eigener Standort in einer AWS-Region , der isoliert und somit vor Ausfällen in anderen Availability Zones geschützt ist und kostengünstige Netzwerkkonnektivität mit geringer Latenz zu anderen Availability Zones in der gleichen Region bereitstellt.

AWS Cloud Adoption Framework (AWS CAF)

Ein Rahmen von Leitlinien und bewährten Verfahren von AWS , um Organisationen bei der Entwicklung eines effizienten und effektiven Plans für eine erfolgreiche Umstellung auf die Cloud zu unterstützen. AWS CAFunterteilt die Beratung in sechs Schwerpunktbereiche, die als Perspektiven bezeichnet werden: Geschäft, Menschen, 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. Für diese Perspektive AWS CAF bietet Beratung für Personalentwicklung, Training und Kommunikation, um die Organisation auf eine erfolgreiche Cloud-Einführung vorzubereiten. Weitere Informationen finden Sie AWS CAFauf der Website und im AWS CAFWhitepaper.

AWS Workload Qualification Framework (AWS WQF)

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

B

Bad 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 Anrufe 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, die als bösartige Bots bezeichnet werden, sollen Einzelpersonen oder Organisationen stören oder ihnen Schaden zufügen.

Bot-Netz

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

branch

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

Zugang durch Glasbruch

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

Brownfield-Strategie

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

Puffer-Cache

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

Geschäftsfähigkeit

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

Planung der Geschäftskontinuität (BCP)

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

C

CAF

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

Canary-Bereitstellung

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 unter 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 es 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

Weitere Informationen finden Sie unter Kontinuierliche Bereitstellung und kontinuierliche Bereitstellung.

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 der Ziel- sie AWS-Service empfängt.

Cloud-Kompetenzzentrum (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 hier: CCoEBeiträge auf dem -Blog zur AWS Cloud Unternehmensstrategie.

Cloud Computing

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

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 Organisationen normalerweise durchlaufen, wenn sie auf AWS Cloud die

- Projekt – Durchführung einiger Cloud-bezogener Projekte zu Machbarkeitsnachweisen und zu Lernzwecken
- Fundament — Grundlegende Investitionen tätigen, um Ihre Cloud-Einführung zu skalieren (z. B. Einrichtung einer landing zoneCCoE, 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 Der Weg zu Cloud-First und die Phasen der Einführung im -Blog zur AWS Cloud Unternehmensstrategie definiert. Informationen darüber, wie sie sich auf die AWS -Migrationsstrategie beziehen finden Sie im Leitfaden zur Vorbereitung der Migration.

CMDB

Siehe Datenbank für das Konfigurationsmanagement.

Code-Repository

Ein Ort, an dem Quellcode und andere Komponenten wie Dokumentation, Beispiele und Skripts gespeichert und im Rahmen von Versionskontrollprozessen aktualisiert werden. Zu den gängigen Cloud-Repositorys gehören GitHub oder AWS CodeCommit. 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 in leistungsschwächere und kostengünstigere Speicherstufen oder -klassen können die 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. AWS Panorama Bietet beispielsweise Geräte an, die CV zu lokalen Kameranetzwerken hinzufügen, und Amazon SageMaker stellt Bildverarbeitungsalgorithmen für CV bereit.

Konfigurations-Drift

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 Phase der Portfolioerkennung und -analyse der Migration.

Konformitätspaket

Eine Sammlung von AWS Config -Regeln und Abhilfemaßnahmen, die Sie zusammenstellen können, um Ihre Compliance- und Sicherheitsprüfungen individuell anzupassen. Mit Hilfe einer YAML Vorlage können Sie ein Konformitätspaket als einzelne Einheit in einer AWS-Konto und Region oder in einer gesamten Organisation bereitstellen. Weitere Informationen finden Sie unter Konformitätspakete in der AWS Config -Dokumentation.

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.

Daten-Drift

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

Daten während der Übertragung

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

Daten-Mesh

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.

Daten-Perimeter

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.

Datensubjekt

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 an historischen 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 in anwenden AWS, fügen Sie mehrere Steuerelemente auf verschiedenen Ebenen der AWS Organizations -Struktur hinzu, um das Backup von Ressourcen zu unterstützen. Ein defense-in-depth Ansatz könnte beispielsweise Multi-Faktor-Authentifizierung, Netzwerksegmentierung und Verschlüsselung kombinieren.

delegierter Administrator

In AWS Organizations kann ein kompatibler Service ein AWS -Mitgliedskonto registrieren, um die Konten der Organisation zu verwalten und Berechtigungen für diesen Service zu

verwalten. Dieses Konto wird als delegierter Administrator für diesen Service bezeichnet. Weitere Informationen und eine Liste kompatibler Services finden Sie unter Services, die mit AWS Organizations funktionieren in der AWS Organizations -Dokumentation.

Bereitstellung

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

Entwicklungsumgebung

Siehe Umgebung.

Detektivische Kontrolle

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

Abbildung des Wertstroms in der Entwicklung (DVSM)

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

digitaler Zwilling

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

Maßtabelle

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

Katastrophe

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

Disaster Recovery (DR)

Die Strategie und der Prozess, mit denen Sie Ausfallzeiten und Datenverluste aufgrund einer Katastrophe minimieren. Weitere Informationen finden Sie unter Notfallwiederherstellung von Workloads unter Wiederherstellung AWS 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 Modernisieren älterer Microsoft. ASP NET(ASMX) schrittweise Webservices mithilfe von Containern und Amazon API Gateway.

DR

Siehe Disaster Recovery.

Erkennung

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 der Wertströme in der Entwicklung.

E

EDA

Siehe explorative Datenanalyse.

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.

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) Prinzipalen erstellen AWS PrivateLink und diesen Berechtigungen gewähren. Diese Konten oder Prinzipale können sich privat mit Ihrem Endpunktservice verbinden, indem sie VPC Schnittstellen-Endpunkte erstellen. Weitere Informationen finden Sie unter Einen Endpunkt-Service erstellen in der Amazon Virtual Private Cloud (AmazonVPC) -Dokumentation.

Unternehmensressourcenplanung (ERP)

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

Envelope-Verschlüsselung

Der Prozess der Verschlüsselung eines Verschlüsselungsschlüssels mit einem anderen Verschlüsselungsschlüssel. Weitere Informationen finden Sie unter Umschlagverschlüsselung 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 AWS CAF Sicherheitsepics 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

Weitere Informationen finden Sie unter 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. EDAwird durchgeführt, indem zusammenfassende Statistiken berechnet und Datenvisualisierungen erstellt werden.

F

Fakten-Tabelle

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.

scheitern

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

Fehlerisolationsgrenze

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 AWS Fehler-Isolationsgrenzen.

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 Machine Learning 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.

FGAC

Siehe differenzierte Zugriffskontrolle.

differenzierte Zugriffskontrolle () FGAC

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

Flash-Cut-Migration

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

G

geografische Blockierung

Siehe geografische Einschränkungen.

Geografische Einschränkungen (Geoblocking)

In Amazon eine Option CloudFront, die verhindert, 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 unter Einschränken der geografischen Verteilung von Inhalt in der CloudFront -Dokumentation.

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.

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

H

HEKTAR

Siehe Hochverfügbarkeit.

Heterogene Datenbankmigration

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

hohe Verfügbarkeit (HA)

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

Modernisierung der Geschichte

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.

Homogene Datenbankmigration

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

heiße Daten

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

Hotfix

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

Hypercare-Phase

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

I

IaC

Sehen Sie sich Infrastruktur als Code an.

Identitätsbasierte Richtlinie

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

Leerlaufanwendung

Eine Anwendung mit einer durchschnittlichen CPU 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 unter AWS Well-Architected Framework.

Eingehende (ingress) VPC

In einer AWS -Multi-Konto-Architektur, eine, VPC die Netzwerkverbindungen von außerhalb einer Anwendung akzeptiert, überprüft und weiterleitet. Die -Referenzarchitektur für die AWS Sicherheit empfiehlt, Ihr Netzwerkkonto mit eingehenden und ausgehenden Daten und Inspektionssäule einzurichten, VPCs um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet 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 (IIoT)

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).

Inspektion VPC

In einer AWS -Multi-Konto-Architektur, eine zentralisierte, VPC die Inspektionen des Netzwerkverkehrs zwischen VPCs (in demselben oder unterschiedlichen AWS-Regionen), das Internet und On-Premises-Netzwerke verwaltet. Die -Referenzarchitektur für die AWS Sicherheit empfiehlt, Ihr Netzwerkkonto mit eingehenden und ausgehenden Daten und Inspektionssäule einzurichten, VPCs um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet 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 von Modellen für Machine Learning mit AWS.

IoT

Siehe Internet der Dinge.

IT-Informationsbibliothek (ITIL)

Eine Reihe von bewährten Methoden für die Bereitstellung von IT-Services und die Abstimmung dieser Services auf die Geschäftsanforderungen. ITILbietet die Grundlage fürITSM.

IT-Servicemanagement (ITSM)

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.

ITIL

Weitere Informationen finden Sie in der IT-Informationsbibliothek.

ITSM

Siehe IT-Servicemanagement.

L

Labelbasierte Zugriffskontrolle () LBAC

Eine Implementierung der obligatorischen Zugriffskontrolle (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, skalierbare und sichere AWS -Umgebung mit mehreren Konten. 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..

Große Migration

Eine Migration von 300 oder mehr Servern.

LBAC

Weitere Informationen finden Sie unter Label-basierte Zugriffskontrolle.

Geringste Berechtigung

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

Lift and Shift

Siehe 7 Rs.

Little-Endian-System

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

Niedrigere Umgebungen

Siehe Umwelt.

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.

Managed Services

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.

Fertigungsleitsystem () MES

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

MAP

Siehe Migration Acceleration Program.

Mechanismus

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

Mitgliedskonto

Alle mit AWS-Konten Ausnahme des Verwaltungskontos, die Teil einer Organisation in sind AWS Organizations. Ein Konto kann jeweils nur einer Organisation angehören.

MES

Siehe Manufacturing Execution System.

Message Queuing Telemetry Transport () 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 Service, der klar definiert 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 Integrieren von Microservices mithilfe von AWS -Serverless-Services.

Microservices-Architekturen

Ein Ansatz zur Erstellung einer Anwendung mit unabhängigen Komponenten, die jeden Anwendungsprozess als Microservice ausführen. Diese Microservices kommunizieren über eine klar definierte Schnittstelle mithilfe von Lightweight. 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 Implementieren von Microservices in AWS.

Migration Acceleration Program (MAP)

Ein AWS -Programm, das Beratung, Unterstützung, Training und Services bietet, um Organisationen dabei zu unterstützen, eine solide betriebliche Grundlage für den Umstieg auf die Cloud zu schaffen und die anfänglichen Kosten von Migrationen auszugleichen. MAPumfasst 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 der Migrationsfabrik gehören in der Regel Betrieb, 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 das AWS -Konto.

Migrationsmuster

Eine wiederholbare Migrationsaufgabe, in der die Migrationsstrategie, das Migrationsziel und die verwendete Migrationsanwendung oder der verwendete Migrationsservice detailliert beschrieben werden. Beispiel: Hostwechsel-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 zu. AWS Cloud MPAbietet eine detaillierte Portfoliobewertung (richtige Servergröße, TCO Preisgestaltung, Migrationskostenanalyse) sowie Migrationsplanung (Anwendungsdatenanalyse und Datenerfassung, Anwendungsgruppierung, Migrationspriorisierung und Wellenplanung). Das MPATool (erfordert Anmeldung) ist für alle AWS -Berater und APN Partnerberater kostenlos verfügbar.

Migration Readiness Assessment (MRA)

Der Prozess der Gewinnung von Erkenntnissen über die Cloud-Bereitschaft einer Organisation, der Identifizierung von Stärken und Schwächen und der Erstellung eines Aktionsplans zur Schließung identifizierter Lücken unter Verwendung des AWS CAF. Weitere Informationen finden Sie im Benutzerhandbuch für Migration Readiness. MRAist die erste Phase der AWS - Migrationsstrategie.

Migrationsstrategie

Der Ansatz, der verwendet wird, um einen Workload auf den zu migrieren AWS Cloud. Weitere Informationen finden Sie unter 7 Rs in diesem Glossar und Ihre Organisation mobilisieren, um groß angelegte 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 im 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 Bewertung der Modernisierungsbereitschaft von Anwendungen im 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 Telemetry Transport.

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

Weitere Informationen finden Sie unter Origin Access Control.

OAI

Siehe Zugriffsidentität von Origin.

OCM

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 Betriebsebene.

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 Offene Prozesskommunikation — Einheitliche Architektur.

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 zu unterstützen ()SLA.

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

Eine Spur, AWS CloudTrail die von erstellt wird und alle Ereignisse für alle AWS-Konten in einer Organisation in protokolliert 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 unter Erstellen eines Trails für eine Organisation in der CloudTrail -Dokumentation.

Organisatorisches Veränderungsmanagement (OCM)

Ein Framework für das Management wichtiger, disruptiver Geschäftstransformationen aus Sicht der Mitarbeiter, der Kultur und der Führung. OCMhilft 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 heißt dieser Rahmen Beschleunigung der Menschen genannt, aufgrund der Geschwindigkeit des Wandels, der bei Projekten zur Cloud-Einführung erforderlich ist. Weitere Informationen finden Sie im OCM-Handbuch.

Ursprungszugriffskontrolle (OAC)

In CloudFront, eine erweiterte Option für die Einschränkung des Zugriffs zur Sicherung Ihrer Amazon Simple Storage Service (Amazon S3) -Inhalte. OACunterstützt 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 Einschränkung des Zugriffs zur Sicherung Ihrer Amazon-S3-Inhalte. Wenn Sie verwendenOAI, CloudFront erstellt einen Prinzipal, mit dem sich Amazon S3 authentifizieren kann. Authentifizierte Prinzipale können nur über eine bestimmte CloudFront Distribution auf Inhalte in einem S3-Bucket zugreifen. Siehe auch OAC, das eine detailliertere und verbesserte Zugriffskontrolle bietet.

ORR

Siehe Überprüfung der Betriebsbereitschaft.

NICHT

Siehe Betriebstechnologie.

Ausgehende (egress) VPC

In einer AWS -Multi-Konto-Architektur, eine, VPC die Netzwerkverbindungen verarbeitet, die von einer Anwendung aus initiiert werden. Die -Referenzarchitektur für die AWS Sicherheit empfiehlt, Ihr Netzwerkkonto mit eingehenden und ausgehenden Daten und Inspektionssäule einzurichten, VPCs um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet 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 Berechtigungen in der IAM -Dokumentation.

Personenbezogene Daten (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 hierfür PII sind Namen, Adressen und Kontaktinformationen.

PII

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 kann (siehe identitätsbasierte Richtlinie), geben Sie die Zugriffsbedingungen an (siehe ressourcenbasierte Richtlinie), oder definieren Sie die maximalen Berechtigungen für alle Konten in einer Organisation in AWS Organizations (siehe Richtlinie zur Servicekontrolle).

Polyglotte Beharrlichkeit

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

Portfoliobewertung

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

predicate

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

Prädikat Push-Down

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 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 in AWS , die Aktionen durchführen und auf Ressourcen zugreifen kann. Diese Entität ist normalerweise ein Root-Benutzer für ein AWS-Konto, eine IAM Rolle oder ein Benutzer. Weitere Informationen finden Sie in der IAM Dokumentation unter Principal in Roles (Begriffe und Konzepte).

Datenschutz durch Design

Ein Ansatz in der Systemtechnik, der den Datenschutz während des gesamten Engineering-Prozesses berücksichtigt.

Privat gehostete Zonen

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

proaktive Steuerung

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

Produktlebenszyklusmanagement (PLM)

Das Management von Daten und Prozessen für ein Produkt während seines gesamten Lebenszyklus, von der Konstruktion, Entwicklung und Markteinführung über Wachstum und Reife bis hin zu Verkauf und Verkauf.

Produktionsumgebung

Siehe Umgebung.

programmierbare Logiksteuerung (PLC)

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

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.

veröffentlichen/abonnieren (pub/sub)

Ein Muster, das asynchrone Kommunikation zwischen Microservices ermöglicht, um die Skalierbarkeit und Reaktionsfähigkeit zu verbessern. In einem Microservice-basierten System kann ein Microservice beispielsweise Ereignismeldungen in einem Kanal veröffentlichen MES, 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 SQL relationalen 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

RACImatrix

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

Ransomware

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

RASCImatrix

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.

architektonwechsel

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.

Ziel für die Wiederherstellungszeit (RTO)

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

Faktorwechsel

Siehe 7 Rs.

Region

Eine Sammlung von AWS -Ressourcen in einem geografischen Bereich. Jede 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.

Hostwechsel

Siehe 7 Rs.

Veröffentlichung

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

umziehen

Siehe 7 Rs.

Wechsel

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 Ausfallsicherheit.

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.

Matrix (verantwortlich, rechenschaftspflichtig, konsultiert, informiertRACI)

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 RASCIMatrix bezeichnet, und wenn Sie sie ausschließen, wird sie als RACIMatrix 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.

Drehung

Der Prozess, bei dem Sie das Secret in regelmäßigen Abständen aktualisieren.

Zugriffskontrolle für Zeilen und Spalten (RCAC)

Die Verwendung einfacher, flexibler SQL Ausdrücke, die über definierte Zugriffsregeln verfügen. RCACbesteht aus Zeilenberechtigungen und Spaltenmasken.

RPO

Siehe Recovery Point Objective.

RTO

Siehe Ziel der Wiederherstellungszeit.

Runbook

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

S

SAML2.0

Ein offener Standard, den viele Identitätsanbieter (IdPs) verwenden. Dieses Feature ermöglicht verbundenes Single Sign-On (SSO), sodass Benutzer sich bei der anmelden AWS Management Console oder die AWS API Operationen aufrufen können, ohne dass Sie eine Benutzeranmeldung IAM für jedes Mitglied der Organisation erstellen müssen. Weitere Informationen zum SAML 2.0-basierten Verbund finden Sie unter Über den SAML 2.0-basierten Verbund in der IAM Dokumentation.

SCADA

Siehe Aufsichtskontrolle und Datenerfassung.

SCP

Siehe Richtlinie zur Dienstkontrolle.

Secret

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

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 (SIEM)

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

Automatisierung von Sicherheitsreaktionen

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

Serverseitige Verschlüsselung

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

Service-Kontrollrichtlinie (SCP)

Eine Richtlinie, die eine zentrale Kontrolle über die Berechtigungen für alle Konten in einer Organisation in AWS Organizations ermöglicht. SCPsdefinieren Sie Integritätsschutz 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 Services oder Aktionen zulässig oder verboten sind. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien in der AWS Organizations -Dokumentation.

Service-Endpunkt

Der 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.

Indikator für das Serviceniveau () SLI

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

Ziel auf Serviceniveau () 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, wohingegen Sie für die Sicherheit in der Cloud verantwortlich sind. Weitere Informationen finden Sie unter Modell der geteilten Verantwortung.

SIEM

Siehe System zur Verwaltung von Sicherheitsinformationen und Ereignissen.

zentraler Fehlerpunkt (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.

SLO

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 im. AWS Cloud

SPOF

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 Modernisieren älterer MicrosoftASP. NET(ASMX) schrittweise Webservices mithilfe von Containern und Amazon API Gateway.

Subnetz

Ein Bereich an IP-Adressen in IhremVPC. 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.

Synthetische Tests

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.

T

tags

Schlüssel-Wert-Paare, die als Metadaten zur Organisation Ihrer AWS -Ressourcen dienen. 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 Transit-Gateway ist ein Netzwerk-Transit-Hub, mit dem Sie Ihre Netzwerke VPCs und On-Premises-Netzwerke miteinander verbinden können. Weitere Informationen finden Sie unter Was ist ein Transit-Gateway? in der AWS Transit Gateway -Dokumentation.

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

Erteilen von Berechtigungen für einen Service, den Sie für die Ausführung von Aufgaben in AWS Organizations und in ihren Konten und in Ihrem Namen in Ihrer Organisation angeben. 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.

VPCPeering

Eine Verbindung zwischen zweiVPCs, mit der Sie den Datenverkehr mithilfe von privaten IP-Adressen weiterleiten können. Weitere Informationen finden Sie unter Was ist VPC Peering? in der VPC Amazon-Dokumentation.

Schwachstelle

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

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.

WORM

Sehen Sie einmal, schreiben Sie einmal, lesen Sie viele.

WQF

Siehe AWSWorkload-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 wird als unveränderlich angesehen.

Z

Null-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.

Zombie-Anwendung

Eine Anwendung, deren durchschnittliche CPU 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.