AWS Whitepaper

Implementierung von Microservices auf AWS

Implementierung von Microservices auf AWS: AWS Whitepaper

Table of Contents

Implementierung von Microservices auf AWS

Datum der Veröffentlichung: 31. Juli 2023 () Dokumentverlauf

Microservices bieten einen optimierten Ansatz für die Softwareentwicklung, der die Bereitstellung beschleunigt, Innovationen fördert, die Wartbarkeit verbessert und die Skalierbarkeit erhöht. Diese Methode basiert auf kleinen, lose gekoppelten Diensten, die über klar definierte APIs Dienste kommunizieren, die von autonomen Teams verwaltet werden. Die Einführung von Microservices bietet Vorteile wie verbesserte Skalierbarkeit, Belastbarkeit, Flexibilität und schnellere Entwicklungszyklen.

In diesem Whitepaper werden drei beliebte Microservice-Muster untersucht: API-gesteuert, ereignisgesteuert und Datenstreaming. Wir bieten einen Überblick über jeden Ansatz, skizzieren die wichtigsten Funktionen von Microservices, gehen auf die Herausforderungen bei ihrer Entwicklung ein und veranschaulichen, wie Amazon Web Services (AWS) Anwendungsteams dabei unterstützen kann, diese Hindernisse zu überwinden.

Angesichts der Komplexität von Themen wie Datenspeicherung, asynchrone Kommunikation und Serviceerkennung sollten Sie bei architektonischen Entscheidungen die spezifischen Anforderungen und Anwendungsfälle Ihrer Anwendung abwägen und dabei die bereitgestellten Hinweise beachten.

Einführung

Microservices-Architekturen kombinieren erfolgreiche und bewährte Konzepte aus verschiedenen Bereichen, wie z. B.:

- Agile Softwareentwicklung
- Serviceorientierte Architekturen
- API-orientiertes Design
- Kontinuierlich) Integration/Continuous Delivery (CI/CD

Häufig enthalten Microservices Entwurfsmuster aus der Twelve-Factor App.

Microservices bieten zwar viele Vorteile, es ist jedoch wichtig, die individuellen Anforderungen Ihres Anwendungsfalls und die damit verbundenen Kosten zu bewerten. Eine monolithische Architektur oder alternative Ansätze können in einigen Fällen besser geeignet sein. Die Entscheidung zwischen

Microservices oder Monolithen sollte auf einer case-by-case Grundlage getroffen werden, bei der Faktoren wie Umfang, Komplexität und spezifische Anwendungsfälle berücksichtigt werden.

Wir untersuchen zunächst eine hoch skalierbare, fehlertolerante Microservices-Architektur (Benutzeroberfläche, Implementierung von Microservices, Datenspeicher) und zeigen, wie diese mithilfe von Container-Technologien aufgebaut werden kann. AWS Anschließend schlagen wir AWS Dienste vor, um eine typische serverlose Microservices-Architektur zu implementieren und so die betriebliche Komplexität zu reduzieren.

Serverless zeichnet sich durch die folgenden Prinzipien aus:

- Keine Infrastruktur, die bereitgestellt oder verwaltet werden muss
- Automatische Skalierung nach Verbrauchseinheit
- Abrechnungsmodell mit nutzungsbasierter Bezahlung
- Integrierte Verfügbarkeit und Fehlertoleranz
- Event Driven Architecture (EDA)

Schließlich untersuchen wir das Gesamtsystem und erörtern dienstübergreifende Aspekte einer Microservices-Architektur, wie z. B. verteilte Überwachung, Protokollierung, Rückverfolgung, Prüfung, Datenkonsistenz und asynchrone Kommunikation.

Dieses Dokument konzentriert sich auf Workloads, die in Hybridszenarien ausgeführt werden AWS Cloud, und Migrationsstrategien ausgenommen. Informationen zu Migrationsstrategien finden Sie im Whitepaper Container Migration Methodology.

Sind Sie Well-Architected?

Das AWS Well-Architected Framework hilft Ihnen dabei, die Vor- und Nachteile der Entscheidungen zu verstehen, die Sie beim Aufbau von Systemen in der Cloud treffen. Die sechs Säulen des Frameworks ermöglichen es Ihnen, bewährte Architekturpraktiken für den Entwurf und Betrieb zuverlässiger, sicherer, effizienter, kostengünstiger und nachhaltiger Systeme kennenzulernen. Mithilfe des AWS Well-Architected Tool, das kostenlos im verfügbar ist AWS-Managementkonsole, können Sie Ihre Workloads anhand dieser bewährten Methoden überprüfen, indem Sie für jede Säule eine Reihe von Fragen beantworten.

Im Bereich Serverless Application Lens konzentrieren wir uns auf bewährte Methoden für die Architektur Ihrer serverlosen Anwendungen. AWS

Weitere Expertentipps und Best Practices für Ihre Cloud-Architektur — Referenzarchitekturbereitstellungen, Diagramme und Whitepapers — finden Sie im Architecture Center.AWS

Umstellung auf Microservices

Microservices sind im Wesentlichen kleine, unabhängige Einheiten, aus denen eine Anwendung besteht. Der Übergang von traditionellen monolithischen Strukturen zu Microservices kann verschiedenen Strategien folgen.

Dieser Übergang wirkt sich auch auf die Art und Weise aus, wie Ihr Unternehmen arbeitet:

- Er fördert eine agile Entwicklung, bei der Teams in schnellen Zyklen arbeiten.
- Teams sind in der Regel klein und werden manchmal auch als zwei Pizzateams bezeichnet — klein genug, dass zwei Pizzen das gesamte Team ernähren könnten.
- Teams übernehmen die volle Verantwortung für ihre Services, von der Erstellung über die Bereitstellung bis hin zur Wartung.

Einfache Microservices-Architektur auf AWS

Typische monolithische Anwendungen bestehen aus verschiedenen Schichten: einer Präsentationsschicht, einer Anwendungsschicht und einer Datenschicht. Microservices-Architekturen hingegen unterteilen Funktionen nach spezifischen Domänen und nicht nach technologischen Schichten in zusammenhängende vertikale Bereiche. Abbildung 1 zeigt eine Referenzarchitektur für eine typische Microservices-Anwendung auf. AWS

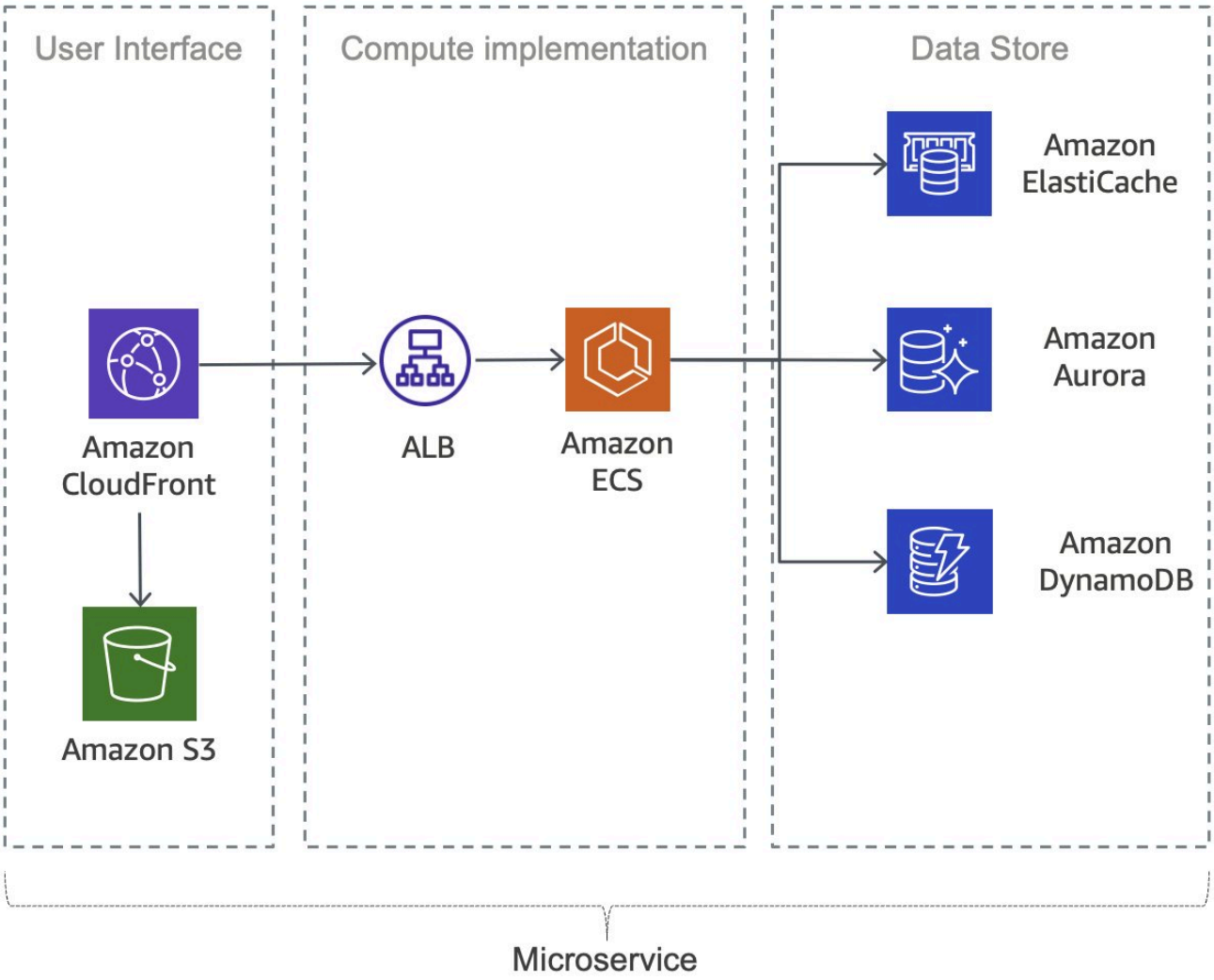

Abbildung 1: Typische Microservices-Anwendung auf AWS

Benutzeroberfläche

Moderne Webanwendungen verwenden häufig JavaScript Frameworks, um einseitige Anwendungen zu entwickeln, die mit dem Backend APIs kommunizieren. Diese APIs werden in der Regel mit Representational State Transfer (REST) oder RESTful APIs APIs GraphQL erstellt. Statische Webinhalte können mit Amazon Simple Storage Service (Amazon S3) und Amazon bereitgestellt CloudFront werden.

Microservices

APIs werden als Eingangstor für Microservices angesehen, da sie der Einstiegspunkt für die Anwendungslogik sind. In der Regel werden RESTful Webservice-API oder GraphQL APIs verwendet. Diese APIs verwalten und verarbeiten Client-Aufrufe und übernehmen Funktionen wie Verkehrsmanagement, Anforderungsfilterung, Routing, Caching, Authentifizierung und Autorisierung.

Microservices-Implementierungen

AWS bietet Bausteine für die Entwicklung von Microservices, darunter Amazon ECS und Amazon EKS als Wahl für Container-Orchestrierungs-Engines AWS Fargate und EC2 als Hosting-Optionen. AWS Lambda ist eine weitere serverlose Methode, auf der Microservices aufgebaut werden können. AWS Die Wahl zwischen diesen Hosting-Optionen hängt von den Anforderungen des Kunden an die Verwaltung der zugrunde liegenden Infrastruktur ab.

AWS Lambda ermöglicht es Ihnen, Ihren Code hochzuladen und dessen Ausführung automatisch mit hoher Verfügbarkeit zu skalieren und zu verwalten. Dadurch entfällt die Notwendigkeit einer Infrastrukturverwaltung, sodass Sie schnell handeln und sich auf Ihre Geschäftslogik konzentrieren können. Lambda unterstützt mehrere Programmiersprachen und kann durch andere AWS Dienste ausgelöst oder direkt aus Web- oder Mobilanwendungen aufgerufen werden.

Container-basierte Anwendungen haben aufgrund ihrer Portabilität, Produktivität und Effizienz an Beliebtheit gewonnen.AWS bietet verschiedene Dienste zum Erstellen, Bereitstellen und Verwalten von Containern.

- App2Container, ein Befehlszeilentool für die Migration und Modernisierung von Java- und .NET-Webanwendungen in das Containerformat. AWS A2C analysiert und erstellt ein Inventar von Anwendungen, die in Bare Metal, virtuellen Maschinen, Amazon Elastic Compute Cloud (EC2) - Instanzen oder in der Cloud ausgeführt werden.

- Amazon Elastic Container Service (Amazon ECS) und Amazon Elastic Kubernetes Service (Amazon EKS) verwalten Ihre Container-Infrastruktur und erleichtern so das Starten und Warten von containerisierten Anwendungen.
  - Amazon EKS ist ein verwalteter Kubernetes-Service zum Ausführen von Kubernetes in der AWS Cloud und in lokalen Rechenzentren (Amazon EKS Anywhere). Dadurch werden Cloud-Dienste auf lokale Umgebungen ausgedehnt, um niedrige Latenz, lokale Datenverarbeitung, hohe Datenübertragungskosten oder Anforderungen an die Datenresidenz zu gewährleisten (siehe das Whitepaper „Running Hybrid Container Workloads With Amazon EKS Anywhere“). Sie können alle vorhandenen Plug-ins und Tools der Kubernetes-Community mit EKS verwenden.
  - Amazon Elastic Container Service (Amazon ECS) ist ein vollständig verwalteter Container-Orchestrierungsservice, der Ihre Bereitstellung, Verwaltung und Skalierung von containerisierten Anwendungen vereinfacht. Kunden entscheiden sich aufgrund der Einfachheit und der umfassenden Integration mit Services für ECS. AWS

Weitere Informationen finden Sie im Blog Amazon ECS vs Amazon EKS: Making Sense of AWS Container Services.

- AWS App Runnerist ein vollständig verwalteter Container-Anwendungsservice, mit dem Sie containerisierte Webanwendungen und API-Services ohne vorherige Infrastruktur- oder Container-Erfahrung erstellen, bereitstellen und ausführen können.
- AWS Fargate, eine serverlose Compute-Engine, arbeitet sowohl mit Amazon ECS als auch mit Amazon EKS zusammen, um Rechenressourcen für Containeranwendungen automatisch zu verwalten.
- Amazon ECR ist eine vollständig verwaltete Container-Registry, die leistungsstarkes Hosting bietet, sodass Sie Anwendungsimages und Artefakte überall zuverlässig bereitstellen können.

Kontinuierliche Integration und kontinuierliche Bereitstellung (CI/CD)

Continuous Integration and Continuous Delivery (CI/CD) ist ein entscheidender Bestandteil einer DevOps Initiative für schnelle Softwareänderungen. AWS bietet Dienste zur Implementierung CI/CD für Microservices an, aber eine ausführliche Diskussion würde den Rahmen dieses Dokuments sprengen. Weitere Informationen finden Sie im AWS Whitepaper Practicing Continuous Integration and Continuous Delivery.

Private Vernetzung

AWS PrivateLink ist eine Technologie, die die Sicherheit von Microservices verbessert, indem sie private Verbindungen zwischen Ihrer Virtual Private Cloud (VPC) und unterstützten AWS Diensten ermöglicht. Sie hilft dabei, den Microservice-Verkehr zu isolieren und zu sichern und stellt so sicher, dass er niemals das öffentliche Internet durchquert. Dies ist besonders nützlich für die Einhaltung von Vorschriften wie PCI oder HIPAA.

Datastore

Der Datenspeicher wird verwendet, um Daten zu speichern, die von den Microservices benötigt werden. Beliebte Speicher für Sitzungsdaten sind In-Memory-Caches wie Memcached oder Redis. AWS bietet beide Technologien als Teil des Managed Amazon ElastiCache Services an.

Das Platzieren eines Caches zwischen Anwendungsservern und einer Datenbank ist ein gängiger Mechanismus, um die Leselast in der Datenbank zu reduzieren, was wiederum dazu führen kann, dass Ressourcen genutzt werden können, um mehr Schreibvorgänge zu unterstützen. Caches können auch die Latenz verbessern.

Relationale Datenbanken sind immer noch sehr beliebt, um strukturierte Daten und Geschäftsobjekte zu speichern. AWS bietet sechs Datenbank-Engines (Microsoft SQL Server, Oracle, MySQL, MariaDB, PostgreSQL und Amazon Aurora) als verwaltete Services über Amazon Relational Database Service (Amazon RDS) an.

Relationale Datenbanken sind jedoch nicht für endlose Skalierbarkeit konzipiert, was die Anwendung von Techniken zur Unterstützung einer hohen Anzahl von Abfragen schwierig und zeitaufwändig machen kann.

NoSQL-Datenbanken wurden so konzipiert, dass Skalierbarkeit, Leistung und Verfügbarkeit der Konsistenz relationaler Datenbanken vorgezogen werden. Ein wichtiges Element von NoSQL-Datenbanken ist, dass sie in der Regel kein striktes Schema erzwingen. Daten werden über Partitionen verteilt, die horizontal skaliert werden können, und werden mithilfe von Partitionsschlüsseln abgerufen.

Da einzelne Microservices darauf ausgelegt sind, eine Sache gut zu machen, verfügen sie in der Regel über ein vereinfachtes Datenmodell, das für NoSQL-Persistenz gut geeignet sein könnte. Es ist wichtig zu verstehen, dass NoSQL-Datenbanken andere Zugriffsmuster haben als relationale Datenbanken. Es ist beispielsweise nicht möglich, Tabellen zu verknüpfen. Wenn dies erforderlich ist, muss die Logik in der Anwendung implementiert werden. Sie können Amazon DynamoDB

verwenden, um eine Datenbanktabelle zu erstellen, die beliebige Datenmengen speichern und abrufen und jede Ebene von Anforderungsverkehr verarbeiten kann. DynamoDB bietet eine Leistung im einstelligen Millisekundenbereich. Es gibt jedoch bestimmte Anwendungsfälle, die Reaktionszeiten in Mikrosekunden erfordern. DynamoDB Accelerator (DAX) bietet Caching-Funktionen für den Zugriff auf Daten.

DynamoDB bietet auch eine automatische Skalierungsfunktion, mit der die Durchsatzkapazität dynamisch an den tatsächlichen Verkehr angepasst werden kann. Es gibt jedoch Fälle, in denen die Kapazitätsplanung aufgrund großer, kurzzeitiger Aktivitätsspitzen in Ihrer Anwendung schwierig oder nicht möglich ist. Für solche Situationen bietet DynamoDB eine On-Demand-Option, die eine einfache pay-per-request Preisgestaltung bietet. DynamoDB On-Demand ist in der Lage, Tausende von Anfragen pro Sekunde sofort und ohne Kapazitätsplanung zu bearbeiten.

Weitere Informationen finden Sie unter Verteiltes Datenmanagement und So wählen Sie eine Datenbank aus.

Vereinfachung von Abläufen

Um den betrieblichen Aufwand für den Betrieb, die Wartung und die Überwachung von Microservices weiter zu vereinfachen, können wir eine vollständig serverlose Architektur verwenden.

Bereitstellung von Lambda-basierten Anwendungen

Sie können Ihren Lambda-Code bereitstellen, indem Sie ein `zip` Dateiarchiv hochladen oder ein Container-Image über die Konsolen-Benutzeroberfläche mit einer gültigen Amazon ECR-Image-URI erstellen und hochladen. Wenn eine Lambda-Funktion jedoch komplex wird, was bedeutet, dass sie Ebenen, Abhängigkeiten und Berechtigungen hat, kann das Hochladen über die Benutzeroberfläche für Codeänderungen unhandlich werden.

Die Verwendung von AWS CloudFormation und AWS Serverless Application Model (AWS SAM) oder Terraform optimiert den Prozess der Definition serverloser Anwendungen. AWS Cloud Development Kit (AWS CDK) AWS SAM, nativ unterstützt von CloudFormation, bietet eine vereinfachte Syntax für die Angabe serverloser Ressourcen.AWS Lambda Ebenen helfen dabei, gemeinsam genutzte Bibliotheken für mehrere Lambda-Funktionen zu verwalten, den Funktionsumfang zu minimieren, mandantenorientierte Bibliotheken zu zentralisieren und das Entwicklererlebnis zu verbessern. Lambda SnapStart für Java verbessert die Startleistung für latenzempfindliche Anwendungen.

Geben Sie zur Bereitstellung Ressourcen- und Berechtigungsrichtlinien in einer CloudFormation Vorlage an, packen Sie Artefakte für die Paketbereitstellung ein und stellen Sie die Vorlage bereit.

SAM Local, ein AWS CLI Tool, ermöglicht die lokale Entwicklung, das Testen und die Analyse serverloser Anwendungen vor dem Hochladen auf Lambda.

Die Integration mit Tools wie AWS Cloud9 IDE,, und AWS CodePipeline optimiert das Erstellen AWS CodeBuild AWS CodeDeploy, Testen, Debuggen und Bereitstellen SAM-basierter Anwendungen.

Das folgende Diagramm zeigt die Bereitstellung von AWS Serverless Application Model Ressourcen mithilfe CloudFormation von CI/CD-Tools. AWS

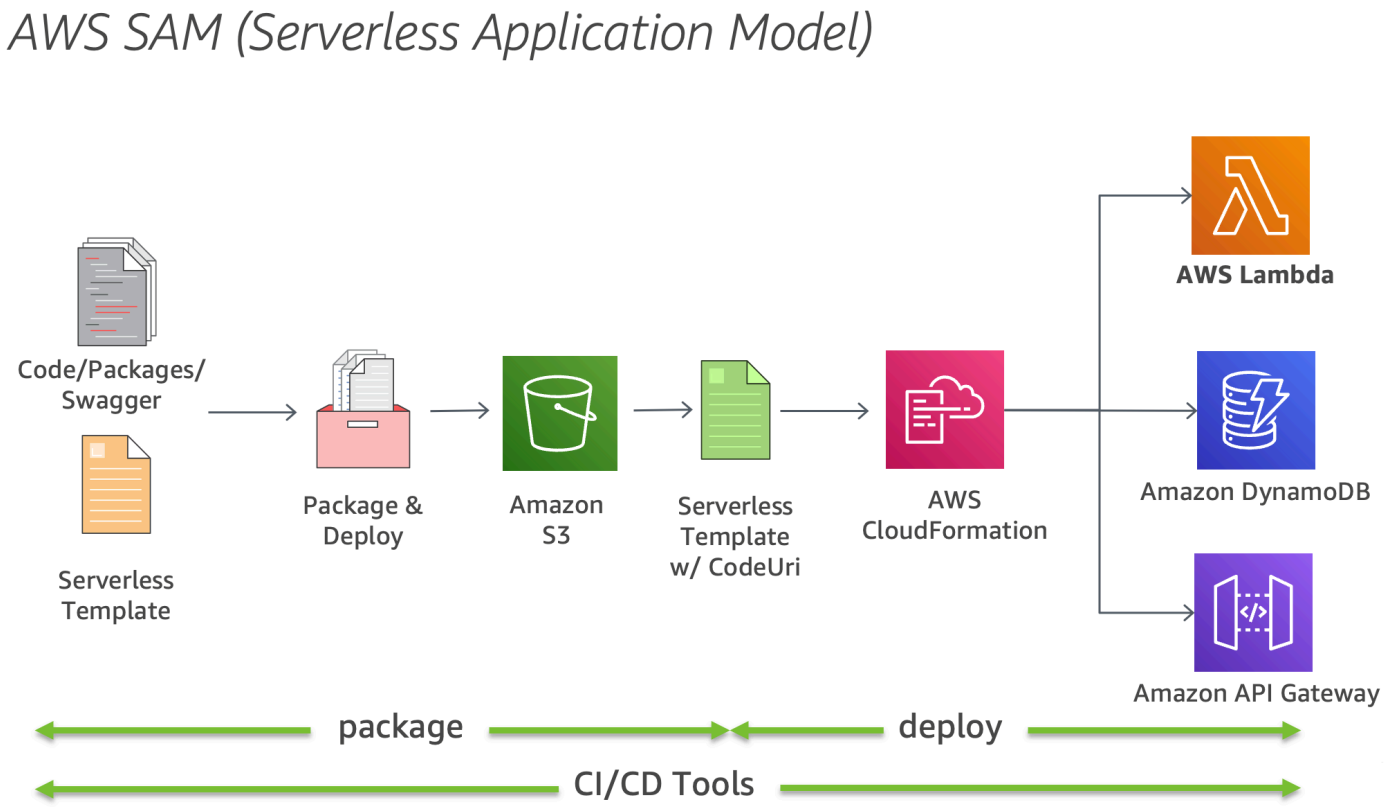

Abbildung 2: AWS Serverless Application Model (AWS SAM)

Abstraktion der Komplexität von Mehrmandantenverhältnissen

In einer Multi-Tenant-Umgebung wie SaaS-Plattformen ist es entscheidend, die Feinheiten im Zusammenhang mit Multi-Tenancy zu vereinfachen, sodass sich Entwickler auf die Entwicklung von Features und Funktionen konzentrieren können. Dies kann mit Tools wie AWS Lambda Layers erreicht werden, die gemeinsam genutzte Bibliotheken zur Lösung bereichsübergreifender Probleme bieten. Der Grund für diesen Ansatz ist, dass gemeinsam genutzte Bibliotheken und Tools, wenn sie richtig eingesetzt werden, den Mandantenkontext effizient verwalten.

Aufgrund der Komplexität und der Risiken, die sie mit sich bringen können, sollten sie sich jedoch nicht auf die Kapselung von Geschäftslogik erstrecken. Ein grundlegendes Problem bei gemeinsam genutzten Bibliotheken ist die zunehmende Komplexität im Zusammenhang mit Updates, wodurch sie im Vergleich zur Duplizierung von Standardcode schwieriger zu verwalten sind. Daher ist es wichtig, bei der Suche nach der effektivsten Abstraktion ein Gleichgewicht zwischen der Verwendung gemeinsam genutzter Bibliotheken und der Vervielfältigung zu finden.

API-Management

Die Verwaltung APIs kann zeitaufwändig sein, insbesondere wenn mehrere Versionen, Phasen des Entwicklungszyklus, Autorisierung und andere Funktionen wie Drosselung und Zwischenspeicherung berücksichtigt werden. Neben API Gateway verwenden einige Kunden auch ALB (Application Load Balancer) oder NLB (Network Load Balancer) für das API-Management. Amazon API Gateway trägt dazu bei, die betriebliche Komplexität der Erstellung und Wartung zu reduzieren RESTful APIs. Es ermöglicht Ihnen die APIs programmgesteuerte Erstellung, dient als „Eingangstür“ für den Zugriff auf Daten, Geschäftslogik oder Funktionen aus Ihren Backend-Services, Autorisierung und Zugriffskontrolle, Ratenbegrenzung, Caching, Überwachung und Datenverkehrsmanagement und läuft APIs ohne Serververwaltung.

Abbildung 3 zeigt, wie API Gateway API-Aufrufe verarbeitet und mit anderen Komponenten interagiert. Anfragen von Mobilgeräten, Websites oder anderen Backend-Diensten werden an den nächstgelegenen CloudFront Point of Presence (PoP) weitergeleitet, um die Latenz zu reduzieren und ein optimales Benutzererlebnis zu bieten.

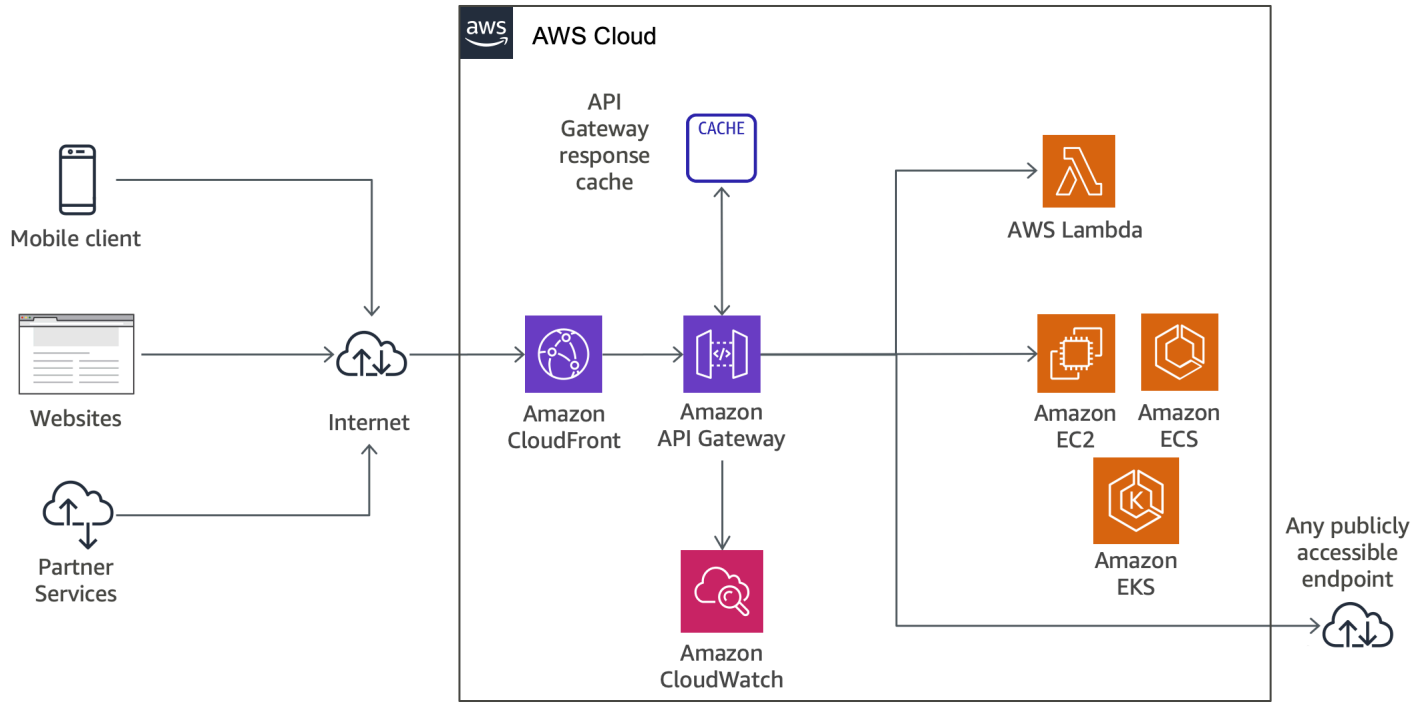

Abbildung 3: API-Gateway-Anruffluss

Microservices auf serverlosen Technologien

Die Verwendung von Microservices mit serverlosen Technologien kann die betriebliche Komplexität erheblich verringern. AWS Lambda und ermöglicht AWS Fargate, integriert in API Gateway, die Erstellung vollständig serverloser Anwendungen. Ab dem 7. April 2023 können Lambda-Funktionen die Antwort-Payloads schrittweise zurück zum Client streamen und so die Leistung von Web- und Mobilanwendungen verbessern. Zuvor mussten Lambda-basierte Anwendungen, die das herkömmliche Request-Response-Aufrufmodell verwendeten, die Antwort generieren und zwischenspeichern, bevor sie an den Client zurückgegeben wurde, was die Zeit bis zum ersten Byte verzögern konnte. Beim Antwort-Streaming können Funktionen Teilantworten an den Client zurücksenden, sobald sie bereit sind. Dadurch wird die Zeit bis zum ersten Byte erheblich verbessert. Web- und Mobilanwendungen reagieren besonders empfindlich darauf.

Abbildung 4 zeigt eine serverlose Microservice-Architektur, die Dienste nutzt AWS Lambda und verwaltet. Diese serverlose Architektur macht es überflüssig, bei der Planung auf Skalierung und Hochverfügbarkeit zu achten, und reduziert den Aufwand für den Betrieb und die Überwachung der zugrunde liegenden Infrastruktur.

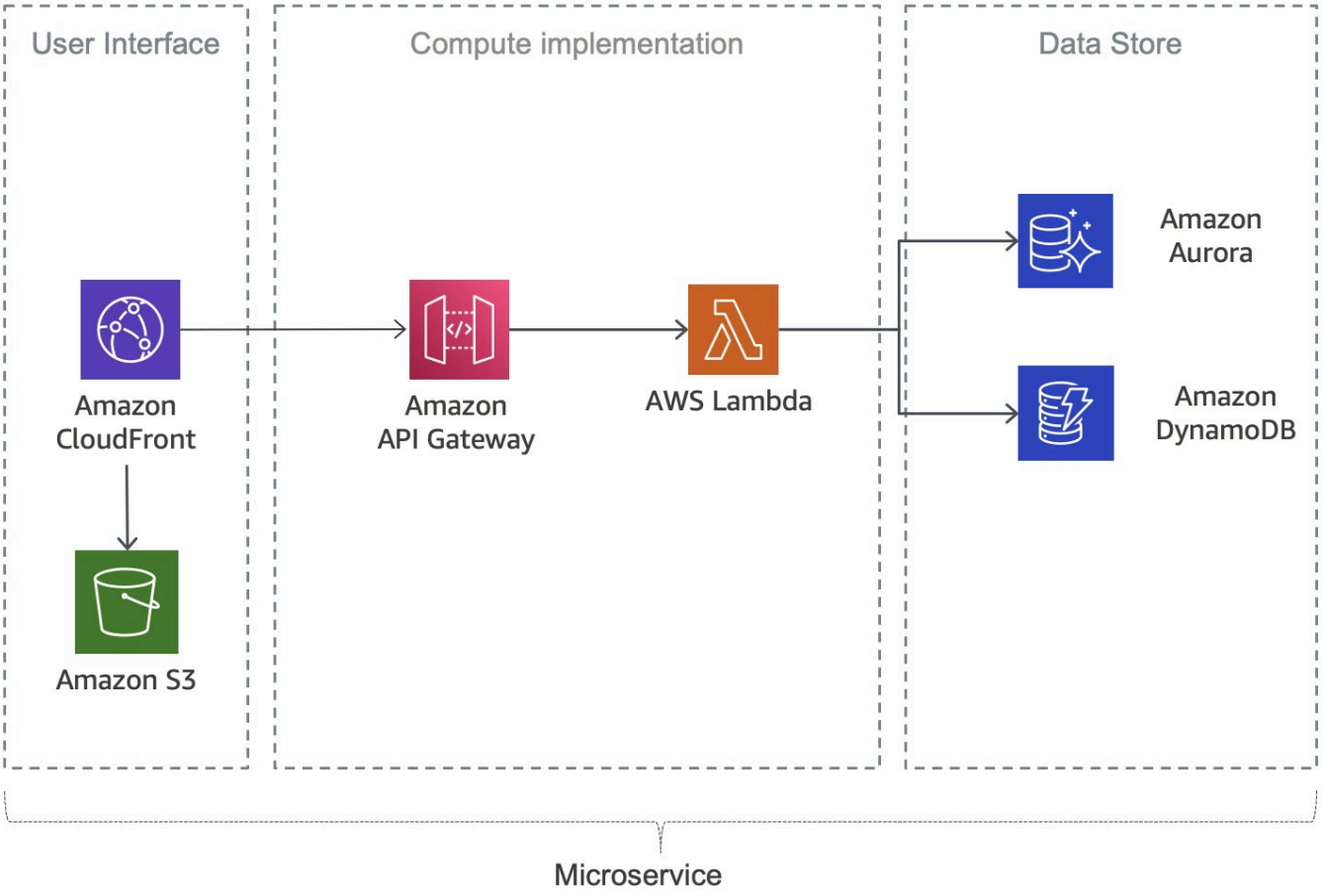

Abbildung 4: Serverloser Microservice unter Verwendung AWS Lambda

Abbildung 5 zeigt eine ähnliche serverlose Implementierung unter Verwendung von Containern mit AWS Fargate, wodurch Bedenken hinsichtlich der zugrunde liegenden Infrastruktur ausgeräumt werden. Es bietet auch Amazon Aurora Serverless, eine On-Demand-Datenbank mit auto-scaling, die die Kapazität automatisch an die Anforderungen Ihrer Anwendung anpasst.

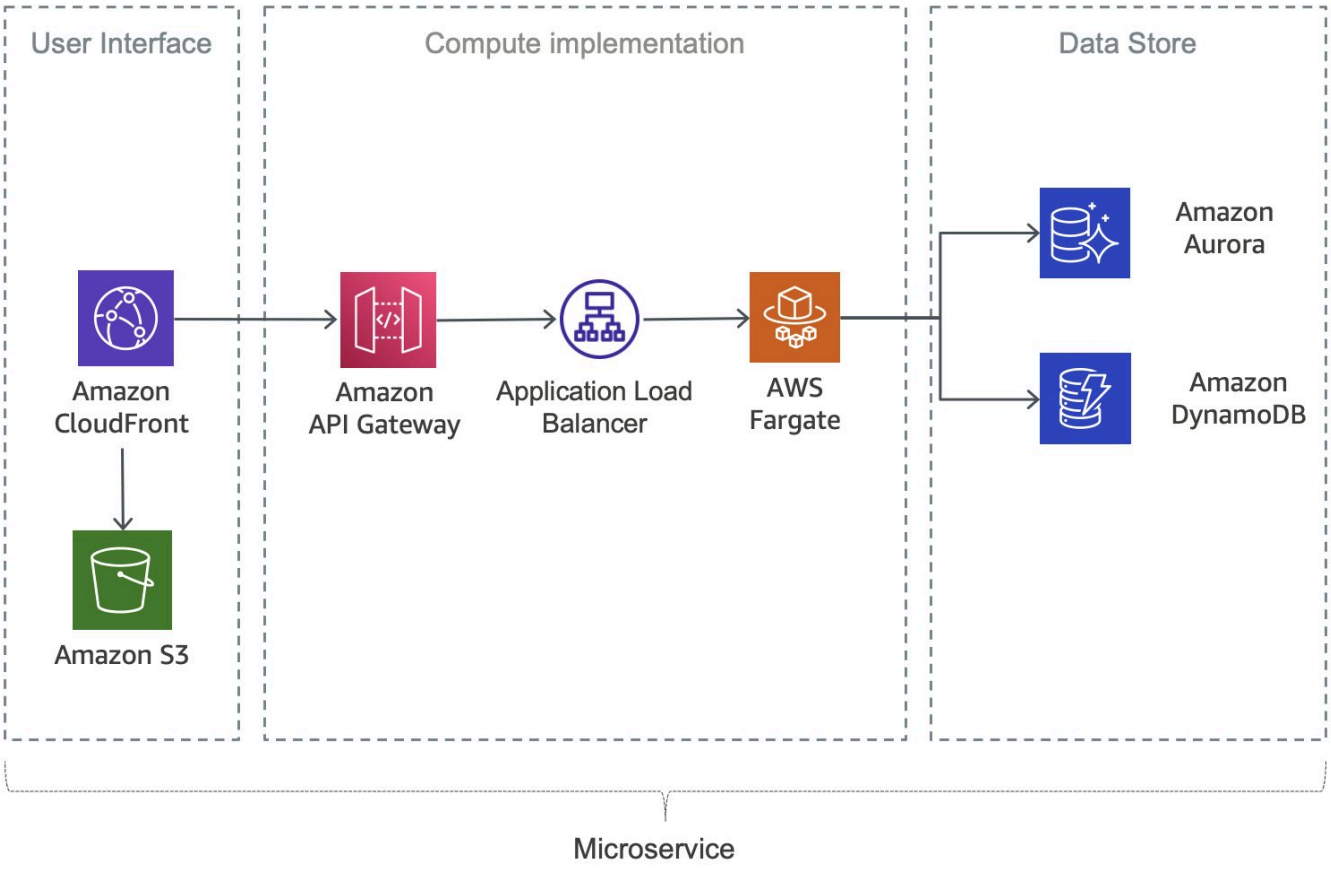

Abbildung 5: Serverloser Microservice unter Verwendung AWS Fargate

Resiliente und effiziente Systeme

Notfallwiederherstellung (DR)

Microservices-Anwendungen folgen häufig den Twelve-Factor-Anwendungsmustern, bei denen Prozesse zustandslos sind und persistente Daten in statusbehafteten Backing-Diensten wie Datenbanken gespeichert werden. Dies vereinfacht die Notfallwiederherstellung (DR), denn wenn ein Dienst ausfällt, ist es einfach, neue Instanzen zu starten, um die Funktionalität wiederherzustellen.

Disaster-Recovery-Strategien für Microservices sollten sich auf nachgelagerte Dienste konzentrieren, die den Status der Anwendung aufrechterhalten, wie Dateisysteme, Datenbanken oder Warteschlangen. Organizations sollten Recovery Time Objective (RTO) und Recovery Point Objective (RPO) einplanen. RTO ist die maximal zulässige Verzögerung zwischen Dienstunterbrechung und Wiederherstellung, während RPO die maximale Zeit seit dem letzten Datenwiederherstellungspunkt ist.

Weitere Informationen zu Strategien für die Notfallwiederherstellung finden Sie im Whitepaper Disaster Recovery of Workloads unter AWS: Recovery in the Cloud.

Hohe Verfügbarkeit (HA)

Wir werden die Hochverfügbarkeit (HA) für verschiedene Komponenten einer Microservices-Architektur untersuchen.

Amazon EKS bietet Hochverfügbarkeit, indem Kubernetes-Steuerungs- und Datenebeneninstanzen in mehreren Availability Zones ausgeführt werden. Es erkennt und ersetzt automatisch fehlerhafte Instanzen auf der Kontrollebene und bietet automatisierte Versions-Upgrades und Patches.

Amazon ECR verwendet Amazon Simple Storage Service (Amazon S3) für die Speicherung, um Ihre Container-Images hochverfügbar und zugänglich zu machen. Es funktioniert mit Amazon EKS, Amazon ECS und und AWS Lambda vereinfacht so den Arbeitsablauf von der Entwicklung bis zur Produktion.

Amazon ECS ist ein regionaler Service, der den Betrieb von Containern auf hochverfügbare Weise in mehreren Availability Zones innerhalb einer Region vereinfacht. Er bietet mehrere Planungsstrategien, bei denen Container nach Ressourcen- und Verfügbarkeitsanforderungen platziert werden.

AWS Lambda arbeitet in mehreren Availability Zones und gewährleistet so die Verfügbarkeit bei Serviceunterbrechungen in einer einzigen Zone. Wenn Sie Ihre Funktion mit einer VPC verbinden, geben Sie Subnetze in mehreren Availability Zones an, um eine hohe Verfügbarkeit zu gewährleisten.

Komponenten verteilter Systeme

In einer Microservices-Architektur bezieht sich Service Discovery auf den Prozess der dynamischen Lokalisierung und Identifizierung der Netzwerkstandorte (IP-Adressen und Ports) einzelner Microservices innerhalb eines verteilten Systems.

Berücksichtigen Sie bei der Auswahl eines Ansatzes unter AWS anderem folgende Faktoren:

- Codeänderung: Können Sie die Vorteile nutzen, ohne den Code zu ändern?
- VPC- oder kontenübergreifender Verkehr: Benötigt Ihr System bei Bedarf ein effizientes Kommunikationsmanagement zwischen verschiedenen oder? VPCs AWS-Konten
- Bereitstellungsstrategien: Verwendet Ihr System erweiterte Bereitstellungsstrategien wie Blue-Green- oder Canary-Implementierungen oder plant solche?
- Überlegungen zur Leistung: Wenn Ihre Architektur häufig mit externen Diensten kommuniziert, welche Auswirkungen hat das auf die Gesamtleistung?

AWS bietet mehrere Methoden zur Implementierung von Service Discovery in Ihrer Microservices-Architektur:

- Amazon ECS Service Discovery: Amazon ECS unterstützt Service Discovery mithilfe seiner DNS-basierten Methode oder durch Integration mit AWS Cloud Map (siehe ECS Service Discovery). ECS Service Connect verbessert das Verbindungsmanagement weiter, was besonders für größere Anwendungen mit mehreren interagierenden Diensten von Vorteil sein kann.
- Amazon Route 53: Route 53 lässt sich in ECS und andere AWS Dienste wie EKS integrieren, um die Serviceerkennung zu erleichtern. In einem ECS-Kontext kann Route 53 die ECS Service Discovery-Funktion verwenden, die die Auto Naming API nutzt, um Dienste automatisch zu registrieren und zu deregistrieren.
- AWS Cloud Map: Diese Option bietet eine dynamische API-basierte Serviceerkennung, die Änderungen an Ihren Diensten weitergibt.

Für anspruchsvollere Kommunikationsanforderungen ist Amazon VPC Lattice ein Anwendungsnetzwerk-Service, der die Kommunikation zwischen Ihren Services konsistent verbindet, überwacht und sichert und so zur Steigerung der Produktivität beiträgt, sodass sich Ihre Entwickler auf die Entwicklung von Funktionen konzentrieren können, die für Ihr Unternehmen von Bedeutung sind. Sie können Richtlinien für die Verwaltung, den Zugriff und die Überwachung des

Netzwerkverkehrs definieren, um Rechendienste auf vereinfachte und konsistente Weise über Instances, Container und serverlose Anwendungen hinweg miteinander zu verbinden.

Falls Sie bereits Software von Drittanbietern wie HashiCorp Consul oder Netflix Eureka für die Diensterkennung verwenden, ziehen Sie es möglicherweise vor, diese bei der Migration weiter zu verwenden AWS, um einen reibungsloseren Übergang zu ermöglichen.

Die Wahl zwischen diesen Optionen sollte Ihren spezifischen Bedürfnissen entsprechen. Für einfachere Anforderungen könnten DNS-basierte Lösungen wie Amazon ECS oder Amazon ausreichend AWS Cloud Map sein. Für komplexere oder größere Systeme sind Service Meshes wie Amazon VPC Lattice möglicherweise besser geeignet.

Zusammenfassend lässt sich sagen, dass es beim Entwerfen einer Microservices-Architektur vor AWS allem darum geht, die richtigen Tools für Ihre spezifischen Anforderungen auszuwählen. Wenn Sie die besprochenen Überlegungen berücksichtigen, können Sie sicherstellen, dass Sie fundierte Entscheidungen treffen, um die Serviceerkennung und die Kommunikation zwischen den Diensten Ihres Systems zu optimieren.

Verteiltes Datenmanagement

In herkömmlichen Anwendungen teilen sich alle Komponenten häufig eine einzige Datenbank. Im Gegensatz dazu verwaltet jede Komponente einer auf Microservices basierenden Anwendung ihre eigenen Daten, was Unabhängigkeit und Dezentralisierung fördert. Dieser Ansatz, bekannt als verteiltes Datenmanagement, bringt neue Herausforderungen mit sich.

Eine solche Herausforderung ergibt sich aus dem Kompromiss zwischen Konsistenz und Leistung in verteilten Systemen. Es ist oft praktischer, geringfügige Verzögerungen bei Datenaktualisierungen in Kauf zu nehmen (letztendliche Konsistenz), als auf sofortigen Aktualisierungen zu bestehen (sofortige Konsistenz).

Manchmal ist es für den Geschäftsbetrieb erforderlich, dass mehrere Microservices zusammenarbeiten. Wenn ein Teil ausfällt, müssen Sie möglicherweise einige abgeschlossene Aufgaben rückgängig machen. Das Saga-Muster hilft dabei, dies zu bewältigen, indem es eine Reihe von Ausgleichsmaßnahmen koordiniert.

Damit Microservices synchron bleiben, kann ein zentraler Datenspeicher verwendet werden. Dieser Shop, der mit Tools wie AWS Lambda, und Amazon verwaltet wird AWS Step Functions EventBridge, kann bei der Bereinigung und Deduplizierung von Daten helfen.

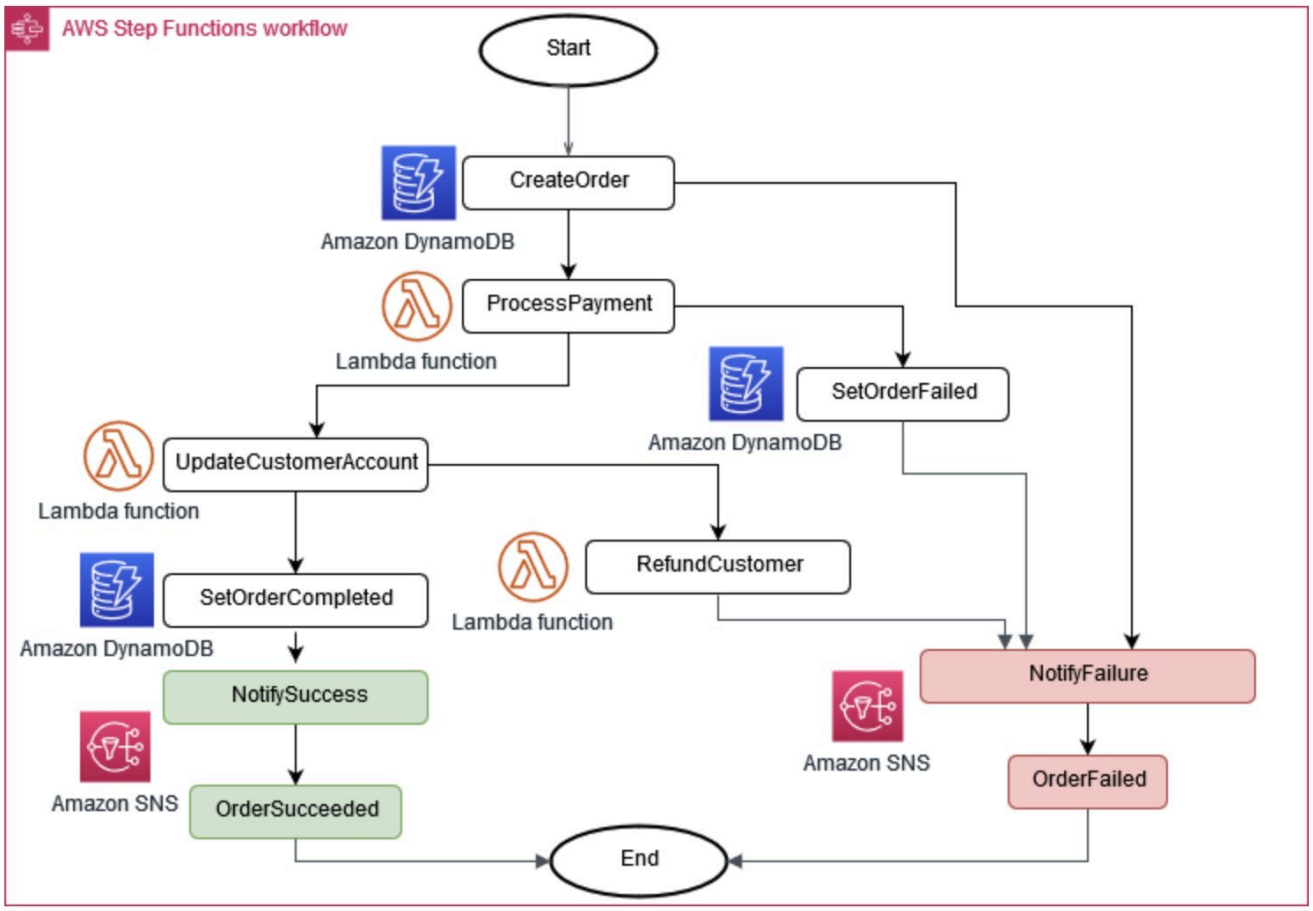

Abbildung 6: Saga-Ausführungskoordinator

Ein gängiger Ansatz bei der Verwaltung von Änderungen in Microservices ist das Event Sourcing. Jede Änderung an der Anwendung wird als Ereignis aufgezeichnet, wodurch ein Zeitplan für den Systemstatus erstellt wird. Dieser Ansatz hilft nicht nur beim Debuggen und Audit, sondern ermöglicht auch, dass verschiedene Teile einer Anwendung auf dieselben Ereignisse reagieren.

Event Sourcing arbeitet häufig hand-in-hand mit dem CQRS-Muster (Command Query Responsibility Segregation), das Datenänderung und Datenabfrage in verschiedene Module unterteilt, um Leistung und Sicherheit zu verbessern.

Bei „On AWS“ können Sie diese Muster mithilfe einer Kombination von Diensten implementieren. Wie Sie in Abbildung 7 sehen können, kann Amazon Kinesis Data Streams als Ihr zentraler Ereignisspeicher dienen, während Amazon S3 einen dauerhaften Speicher für alle Ereignisaufzeichnungen bietet. AWS Lambda, Amazon DynamoDB und Amazon API Gateway arbeiten zusammen, um diese Ereignisse zu behandeln und zu verarbeiten.

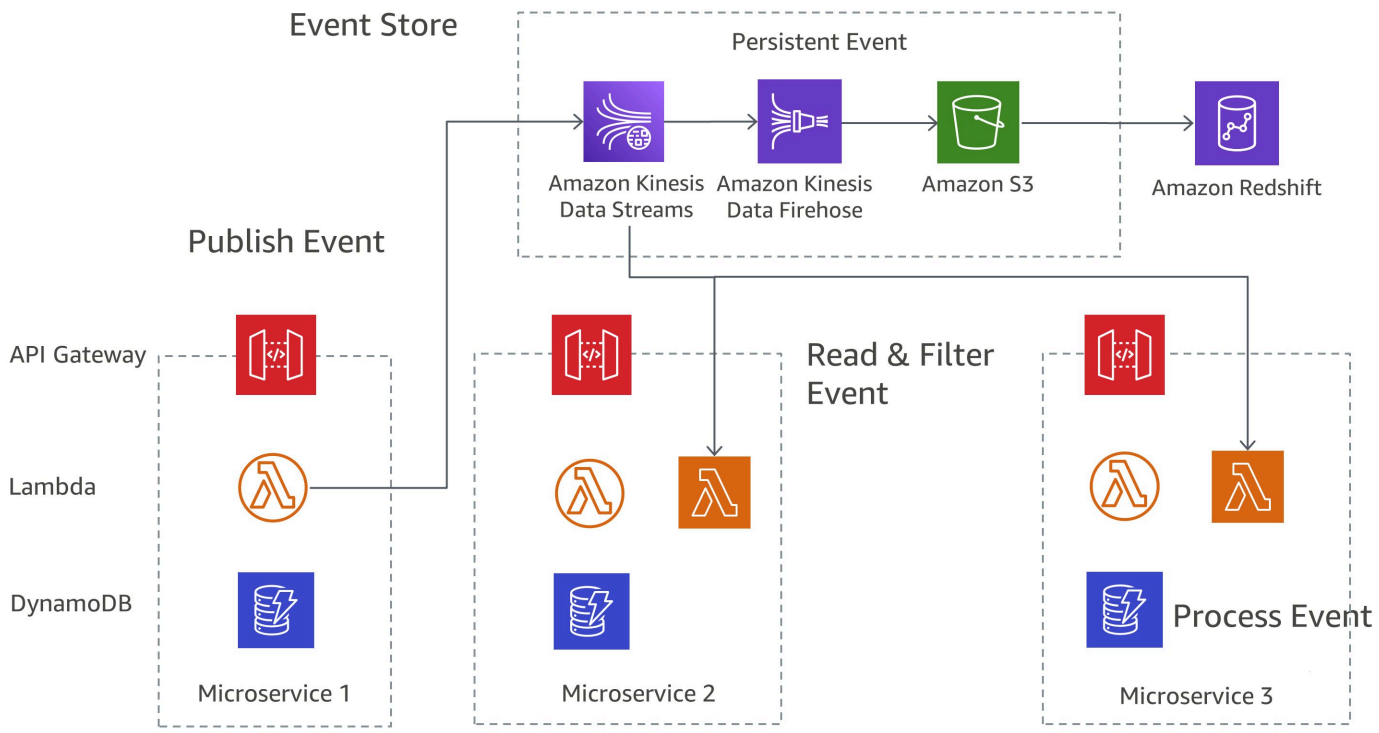

Abbildung 7: Muster für die Ereignisbeschaffung aktiviert AWS

Denken Sie daran, dass in verteilten Systemen Ereignisse aufgrund von Wiederholungsversuchen möglicherweise mehrfach zugestellt werden. Daher ist es wichtig, Ihre Anwendungen so zu gestalten, dass sie dies handhaben.

Konfigurationsmanagement

In einer Microservices-Architektur interagiert jeder Dienst mit verschiedenen Ressourcen wie Datenbanken, Warteschlangen und anderen Diensten. Eine konsistente Methode zur Konfiguration der Verbindungen und der Betriebsumgebung der einzelnen Dienste ist von entscheidender Bedeutung. Im Idealfall sollte sich eine Anwendung an neue Konfigurationen anpassen, ohne dass ein Neustart erforderlich ist. Dieser Ansatz ist Teil der Twelve-Factor App-Prinzipien, die empfehlen, Konfigurationen in Umgebungsvariablen zu speichern.

Ein anderer Ansatz ist die Verwendung von AWS App Config. Es handelt sich um eine Funktion von AWS Systems Manager, die es Kunden leicht macht, Feature-Flags und Anwendungskonfigurationen schnell und sicher zu konfigurieren, zu validieren und bereitzustellen. Ihre Feature-Flags- und Konfigurationsdaten können in der Phase vor der Bereitstellung syntaktisch oder semantisch validiert werden. Sie können überwacht und automatisch zurückgesetzt werden, wenn ein von Ihnen konfigurierter Alarm ausgelöst wird. AppConfig kann mithilfe des AWS AppConfig Agenten in Amazon ECS und Amazon EKS integriert werden. Der Agent fungiert als Sidecar-Container, der neben Ihren Amazon ECS- und Amazon EKS-Containeranwendungen ausgeführt wird. Wenn Sie AWS AppConfig Feature-Flags oder andere dynamische Konfigurationsdaten in einer Lambda-Funktion verwenden, empfehlen wir Ihnen, die AWS AppConfig Lambda-Erweiterung als Ebene zu Ihrer Lambda-Funktion hinzuzufügen.

GitOpsist ein innovativer Ansatz für das Konfigurationsmanagement, der Git als Informationsquelle für alle Konfigurationsänderungen verwendet. Das bedeutet, dass alle an deinen Konfigurationsdateien vorgenommenen Änderungen automatisch über Git verfolgt, versioniert und geprüft werden.

Verwaltung von Secrets

Sicherheit ist von größter Bedeutung, daher sollten Anmeldeinformationen nicht im Klartext weitergegeben werden. AWS bietet dafür sichere Dienste wie AWS Systems Manager Parameter Store und AWS Secrets Manager. Diese Tools können Geheimnisse als Volumes an Container in Amazon EKS oder als Umgebungsvariablen an Amazon ECS senden. AWS Lambda In werden Umgebungsvariablen automatisch für Ihren Code verfügbar gemacht. Bei Kubernetes-Workflows ruft der Operator External Secrets Secrets Secrets direkt von Diensten ab und erstellt beispielsweise AWS Secrets Manager entsprechende Kubernetes-Geheimnisse. Dies ermöglicht eine nahtlose Integration mit Kubernetes-nativen Konfigurationen.

Kostenoptimierung und Nachhaltigkeit

Die Microservices-Architektur kann die Kostenoptimierung und Nachhaltigkeit verbessern. Indem Sie eine Anwendung in kleinere Teile aufteilen, können Sie nur die Dienste skalieren, die mehr Ressourcen benötigen, wodurch Kosten und Verschwendung reduziert werden. Dies ist besonders nützlich, wenn Sie mit variablem Verkehr zu tun haben. Microservices werden unabhängig voneinander entwickelt. So können Entwickler kleinere Updates durchführen und den Ressourcenaufwand für End-to-End-Tests reduzieren. Bei der Aktualisierung müssen sie im Gegensatz zu Monolithen nur einen Teil der Funktionen testen.

Zustandslose Komponenten (Dienste, die den Status in einem externen Datenspeicher statt in einem lokalen Datenspeicher speichern) in Ihrer Architektur können Amazon EC2 Spot-Instances nutzen, die ungenutzte EC2 Kapazität in der AWS Cloud bieten. Diese Instances sind kostengünstiger als On-Demand-Instances und eignen sich perfekt für Workloads, die Unterbrechungen bewältigen können. Dies kann die Kosten weiter senken und gleichzeitig die hohe Verfügbarkeit aufrechterhalten.

Bei isolierten Diensten können Sie kostenoptimierte Rechenoptionen für jede Gruppe mit auto-scaling verwenden. AWS Graviton bietet beispielsweise kostengünstige, leistungsstarke Rechenoptionen für Workloads, die für ARM-basierte Instanzen geeignet sind.

Die Optimierung der Kosten und des Ressourcenverbrauchs trägt auch zur Minimierung der Umweltbelastung bei und steht im Einklang mit der Nachhaltigkeitssäule des Well-Architected Framework. Mit dem AWS Customer Carbon Footprint Tool können Sie Ihre Fortschritte bei der Reduzierung der CO2-Emissionen überwachen. Dieses Tool bietet Einblicke in die Umweltauswirkungen Ihrer AWS Nutzung.

Kommunikationsmechanismen

Im Microservices-Paradigma müssen verschiedene Komponenten einer Anwendung über ein Netzwerk kommunizieren. Zu den gängigen Ansätzen hierfür gehören REST-basiertes, GraphQL-basiertes, GRPC-basiertes und asynchrones Messaging.

REST-basierte Kommunikation

Das HTTP/S Protokoll, das häufig für die synchrone Kommunikation zwischen Microservices verwendet wird, funktioniert häufig über. RESTful APIs API Gateway bietet eine optimierte Möglichkeit, eine API zu erstellen, die als zentraler Zugangspunkt zu Backend-Diensten dient und Aufgaben wie Verkehrsmanagement, Autorisierung, Überwachung und Versionskontrolle übernimmt.

GraphQL-basierte Kommunikation

In ähnlicher Weise ist GraphQL eine weit verbreitete Methode für synchrone Kommunikation, die dieselben Protokolle wie REST verwendet, aber die Exposition auf einen einzelnen Endpunkt beschränkt. Mit AWS AppSync können Sie GraphQL-Anwendungen erstellen und veröffentlichen, die direkt mit AWS Diensten und Datenspeichern interagieren, oder Lambda-Funktionen für die Geschäftslogik integrieren.

GRPC-basierte Kommunikation

gRPC ist ein synchrones, leichtes, leistungsstarkes Open-Source-RPC-Kommunikationsprotokoll. gRPC verbessert die zugrunde liegenden Protokolle, indem es HTTP/2 verwendet und mehr Funktionen wie Komprimierung und Stream-Priorisierung ermöglicht. Es verwendet die Protobuf Interface Definition Language (IDL), die binär codiert ist und somit die Vorteile des binären HTTP/2-Framings nutzt.

Asynchrone Nachrichtenübermittlung und Ereignisübergabe

Asynchrones Messaging ermöglicht Diensten die Kommunikation, indem sie Nachrichten über eine Warteschlange senden und empfangen. Auf diese Weise können Dienste lose miteinander verknüpft bleiben und die Diensterkennung wird gefördert.

Es können die folgenden drei Typen von Nachrichten definiert werden:

- Nachrichtenwarteschlangen: Eine Nachrichtenwarteschlange fungiert als Puffer, der Absender (Produzenten) und Empfänger (Verbraucher) von Nachrichten entkoppelt. Produzenten stellen Nachrichten in die Warteschlange, und Verbraucher stellen sie aus der Warteschlange und verarbeiten sie. Dieses Muster ist nützlich für asynchrone Kommunikation, Lastausgleich und die Verarbeitung von Datenverkehrsspitzen.
- Publish-Subscribe: Beim Publish-Subscribe-Muster wird eine Nachricht zu einem Thema veröffentlicht, und mehrere interessierte Abonnenten erhalten die Nachricht. Dieses Muster ermöglicht die asynchrone Übertragung von Ereignissen oder Nachrichten an mehrere Verbraucher.
- Ereignisgesteuertes Messaging: Beim ereignisgesteuerten Messaging werden Ereignisse erfasst und darauf reagiert, die im System auftreten. Ereignisse werden in einem Nachrichtenbroker veröffentlicht, und interessierte Dienste abonnieren bestimmte Ereignistypen. Dieses Muster ermöglicht eine lose Kopplung und ermöglicht es Diensten, ohne direkte Abhängigkeiten auf Ereignisse zu reagieren.

Um jeden dieser Nachrichtentypen zu implementieren, AWS bietet verschiedene verwaltete Dienste wie Amazon SQS, Amazon SNS, Amazon EventBridge, Amazon MQ und Amazon MSK an. Diese Dienste verfügen über einzigartige Funktionen, die auf spezifische Bedürfnisse zugeschnitten sind:

- Amazon Simple Queue Service (Amazon SQS) und Amazon Simple Notification Service (Amazon SNS): Wie Sie in Abbildung 8 sehen können, ergänzen sich diese beiden Dienste gegenseitig. Amazon SQS bietet Speicherplatz für Nachrichten und Amazon SNS ermöglicht die Zustellung von Nachrichten an mehrere Abonnenten. Sie sind wirksam, wenn dieselbe Nachricht an mehrere Ziele zugestellt werden muss.

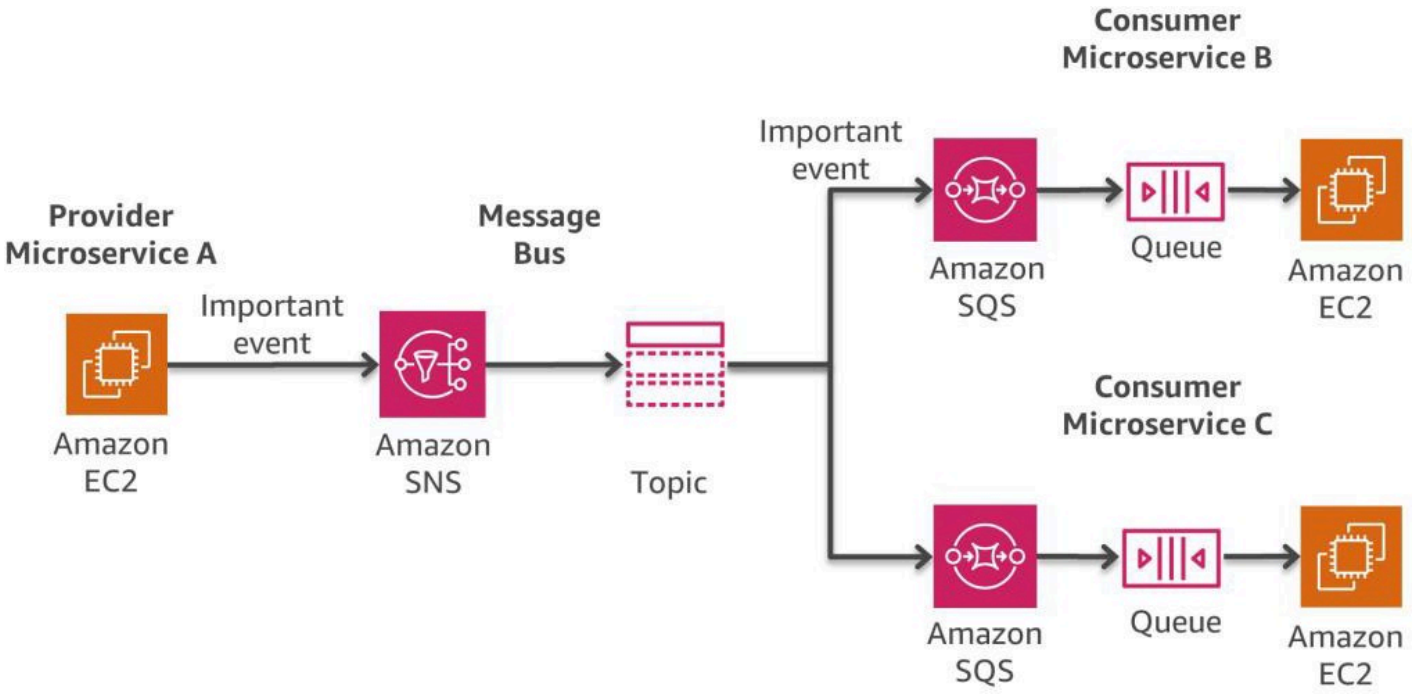

Abbildung 8: Nachrichtenbusmuster aktiviert AWS

- Amazon EventBridge: Ein serverloser Service, der Ereignisse verwendet, um Anwendungskomponenten miteinander zu verbinden, sodass Sie leichter skalierbare, ereignisgesteuerte Anwendungen erstellen können. Verwenden Sie ihn, um Ereignisse aus Quellen wie selbst entwickelten Anwendungen, AWS Diensten und Software von Drittanbietern an Verbraucheranwendungen in Ihrem Unternehmen weiterzuleiten. EventBridge bietet eine einfache und konsistente Methode zum Erfassen, Filtern, Transformieren und Bereitstellen von Ereignissen, sodass Sie schnell neue Anwendungen erstellen können. EventBridge Event-Busse eignen sich hervorragend für die many-to-many Weiterleitung von Ereignissen zwischen ereignisgesteuerten Diensten.
- Amazon MQ: Eine gute Wahl, wenn Sie bereits ein Messaging-System haben, das Standardprotokolle wie JMS, AMQP oder ähnliches verwendet. Dieser verwaltete Service bietet einen Ersatz für Ihr System, ohne den Betrieb zu unterbrechen.
- Amazon MSK (Managed Kafka): Ein Nachrichtensystem zum Speichern und Lesen von Nachrichten, nützlich für Fälle, in denen Nachrichten mehrfach verarbeitet werden müssen. Es unterstützt auch Nachrichtenstreaming in Echtzeit.
- Amazon Kinesis: Verarbeitung und Analyse von Streaming-Daten in Echtzeit. Dies ermöglicht die Entwicklung von Echtzeitanwendungen und bietet eine nahtlose Integration in das AWS Ökosystem.

Denken Sie daran, dass der beste Service für Sie von Ihren spezifischen Bedürfnissen abhängt. Daher ist es wichtig zu verstehen, was die einzelnen Angebote bieten und wie sie Ihren Anforderungen entsprechen.

Orchestrierung und Statusverwaltung

Microservices-Orchestrierung bezieht sich auf einen zentralisierten Ansatz, bei dem eine zentrale Komponente, der sogenannte Orchestrator, für die Verwaltung und Koordination der Interaktionen zwischen Microservices verantwortlich ist. Die Orchestrierung von Workflows über mehrere Microservices hinweg kann eine Herausforderung sein. Von der direkten Einbettung von Orchestrierungscode in Dienste wird abgeraten, da dies zu einer engeren Kopplung führt und den Ersatz einzelner Dienste behindert.

Step Functions bietet eine Workflow-Engine zur Verwaltung der komplexen Service-Orchestrierung, wie z. B. Fehlerbehandlung und Serialisierung. Auf diese Weise können Sie Anwendungen schnell skalieren und ändern, ohne Koordinationscode hinzufügen zu müssen. Step Functions ist Teil der AWS serverlosen Plattform und unterstützt Lambda-Funktionen, Amazon EC2, Amazon EKS, Amazon ECS AWS Glue, SageMaker KI und mehr.

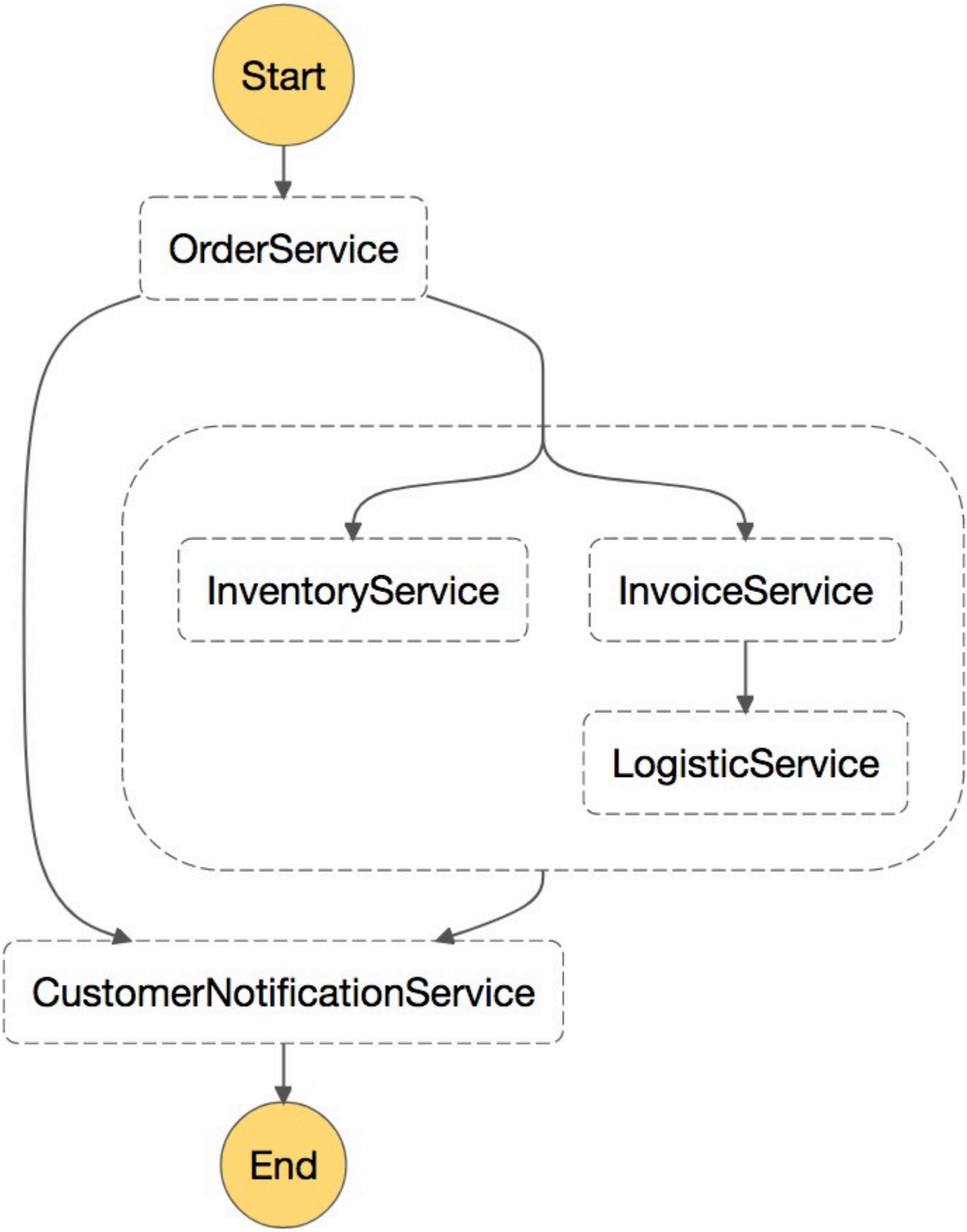
Start
OrderService
InventoryService
InvoiceService
LogisticService
CustomerNotificationService
End

Abbildung 9: Ein Beispiel für einen Microservices-Workflow mit parallel und sequentiellen Schritten, aufgerufen von AWS Step Functions

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) ist eine Alternative zu Step Functions. Sie sollten Amazon MWAA verwenden, wenn Sie Wert auf Open Source und Portabilität legen. Airflow verfügt über eine große und aktive Open-Source-Community, die regelmäßig neue Funktionen und Integrationen beisteuert.

Beobachtbarkeit

Da Microservices-Architekturen grundsätzlich aus vielen verteilten Komponenten bestehen, ist die Beobachtbarkeit all dieser Komponenten von entscheidender Bedeutung. Amazon CloudWatch ermöglicht dies, indem es Metriken sammelt und verfolgt, Protokolldateien überwacht und auf Änderungen in Ihrer AWS Umgebung reagiert. Es kann AWS Ressourcen und benutzerdefinierte Metriken überwachen, die von Ihren Anwendungen und Diensten generiert werden.

Themen

Überwachen

CloudWatch bietet systemweiten Einblick in die Ressourcennutzung, die Anwendungsleistung und den Betriebsstatus. In einer Microservices-Architektur CloudWatch ist die Überwachung benutzerdefinierter Metriken von Vorteil, da Entwickler selbst entscheiden können, welche Metriken erfasst werden sollen. Dynamische Skalierung kann auch auf diesen benutzerdefinierten Metriken basieren.

CloudWatch Container Insights erweitert diese Funktionalität und sammelt automatisch Metriken für viele Ressourcen wie CPU, Arbeitsspeicher, Festplatte und Netzwerk. Es hilft bei der Diagnose von Container-Problemen und bei der Optimierung der Problembehebung.

Für Amazon EKS ist Prometheus, eine Open-Source-Plattform, die umfassende Überwachungs- und Warnfunktionen bietet, eine häufig bevorzugte Wahl. Es ist in der Regel mit Grafana für eine intuitive Metrikvisualisierung gekoppelt. Amazon Managed Service for Prometheus (AMP) bietet einen vollständig mit Prometheus kompatiblen Überwachungsservice, mit dem Sie containerisierte Anwendungen mühelos überwachen können. Darüber hinaus vereinfacht Amazon Managed Grafana (AMG) die Analyse und Visualisierung Ihrer Kennzahlen, sodass Sie die zugrunde liegende Infrastruktur nicht mehr verwalten müssen.

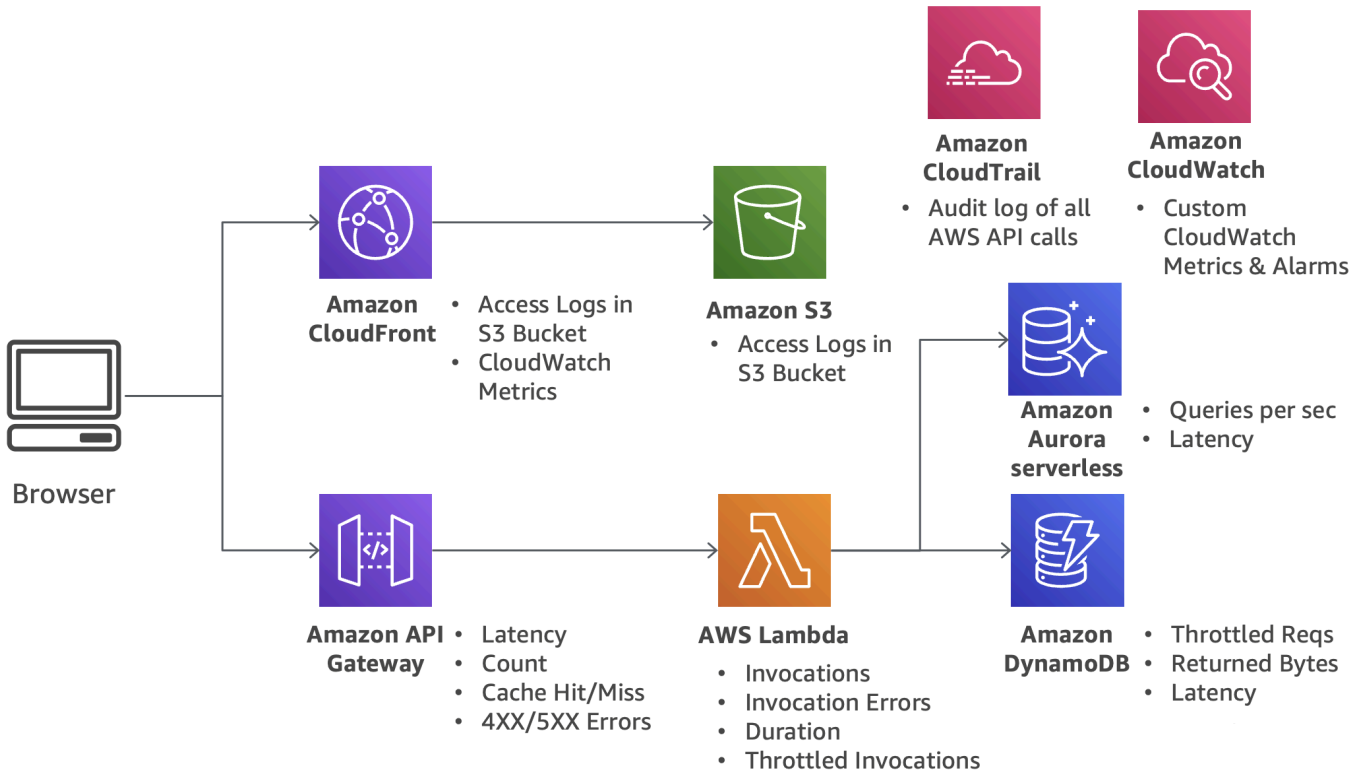

Abbildung 10: Eine serverlose Architektur mit Überwachungskomponenten

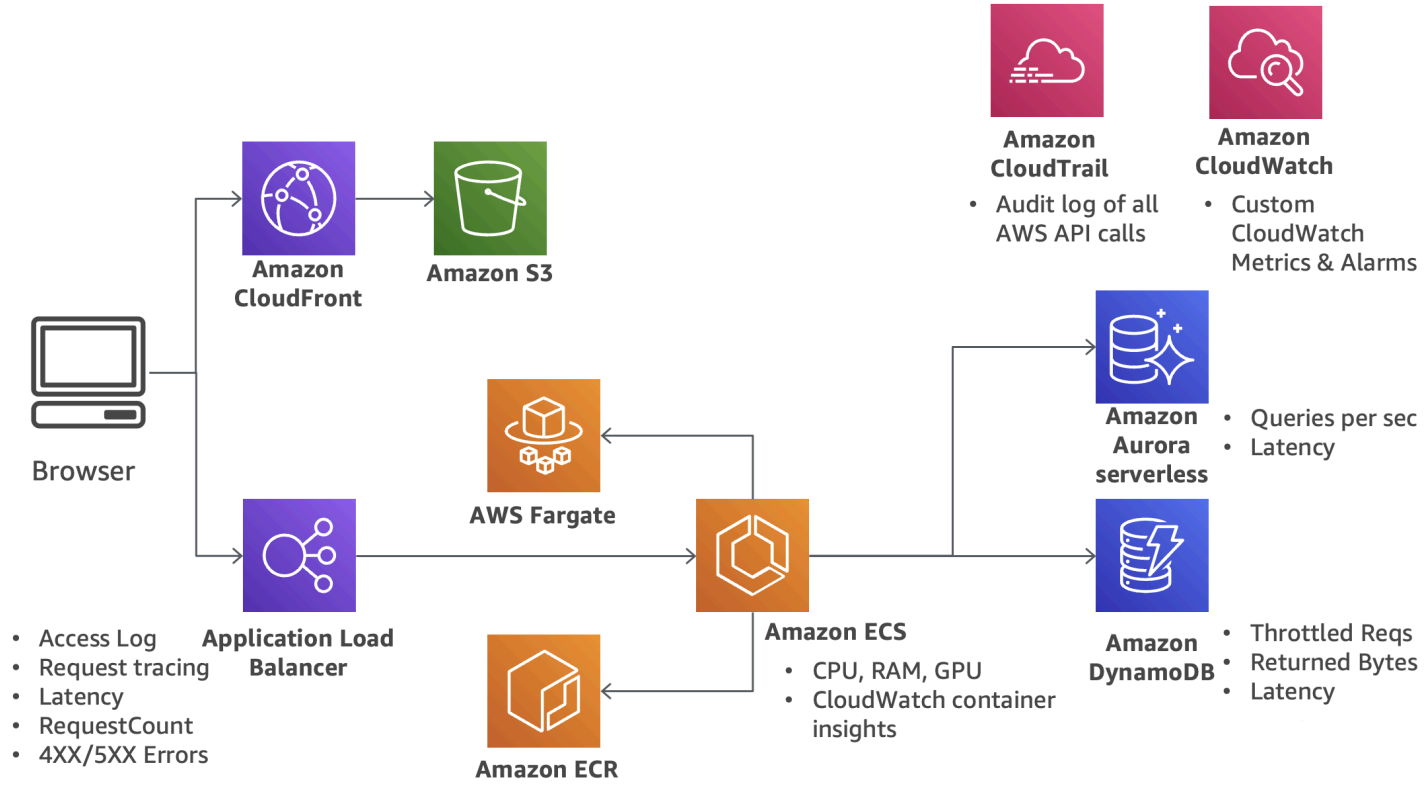

Abbildung 11: Eine containerbasierte Architektur mit Überwachungskomponenten

Zentralisierung von Protokollen

Die Protokollierung ist entscheidend, um Probleme zu lokalisieren und zu lösen. Mit Microservices können Sie häufiger Releases veröffentlichen und mit neuen Funktionen experimentieren. AWS bietet Dienste wie Amazon S3, CloudWatch Logs und Amazon OpenSearch Service zur Zentralisierung von Protokolldateien. Amazon EC2 verwendet einen Daemon zum Senden von Protokollen an CloudWatch, während Lambda und Amazon ECS ihre Protokollausgabe nativ dorthin senden. Für Amazon EKS können entweder Fluent Bit oder Fluentd verwendet werden, um Protokolle CloudWatch zur Berichterstattung über OpenSearch Kibana weiterzuleiten. Aufgrund des geringeren Platzbedarfs und der Leistungsvorteile wird Fluent Bit jedoch gegenüber Fluentd empfohlen.

Abbildung 12 zeigt, wie Protokolle von verschiedenen AWS Diensten an Amazon S3 und weitergeleitet werden CloudWatch. Diese zentralisierten Protokolle können mit Amazon OpenSearch Service, einschließlich Kibana zur Datenvisualisierung, weiter analysiert werden. Amazon Athena kann auch für Ad-hoc-Abfragen der in Amazon S3 gespeicherten Protokolle verwendet werden.

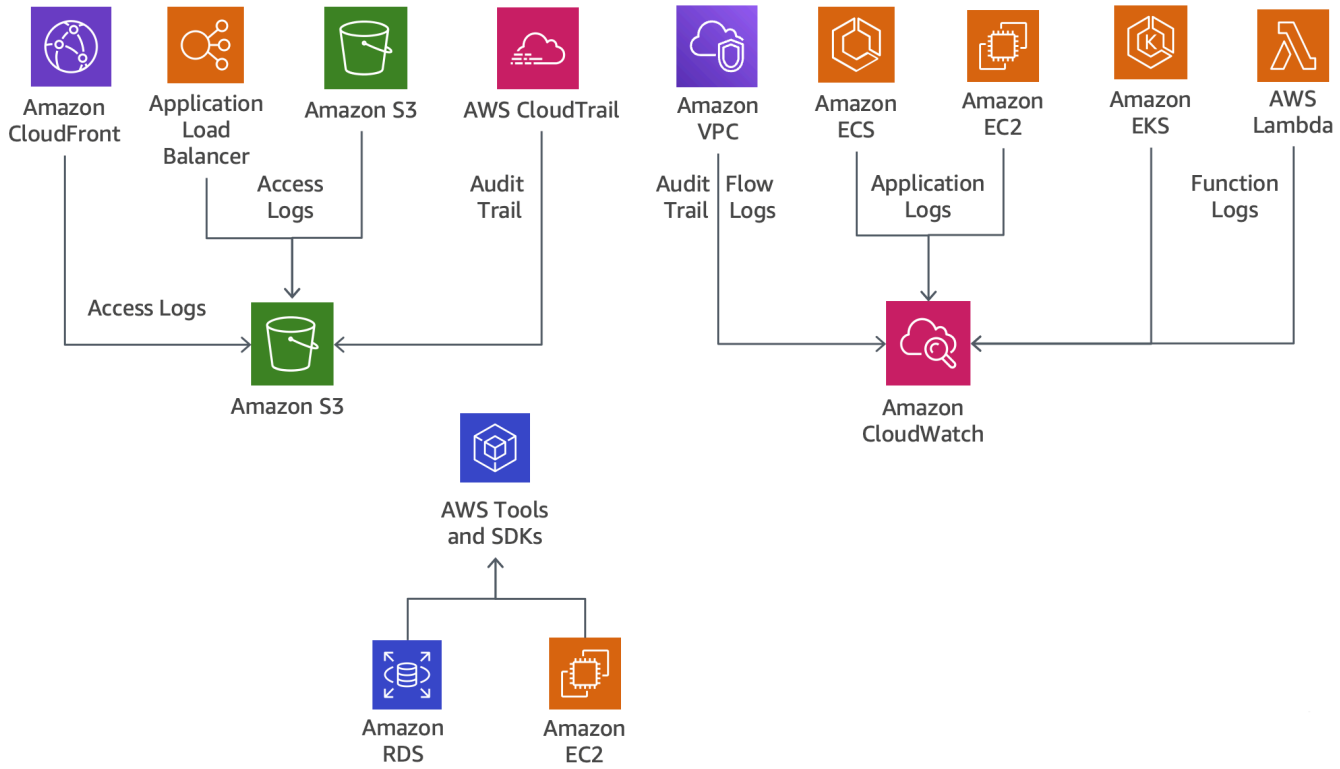

Abbildung 12: Protokollierungsfunktionen von AWS Diensten

Verteilte Ablaufverfolgung

Microservices arbeiten häufig zusammen, um Anfragen zu bearbeiten. AWS X-Ray verwendet Korrelations-IDs, um Anfragen über diese Dienste hinweg zu verfolgen. X-Ray funktioniert mit Amazon EC2, Amazon ECS, Lambda und Elastic Beanstalk.

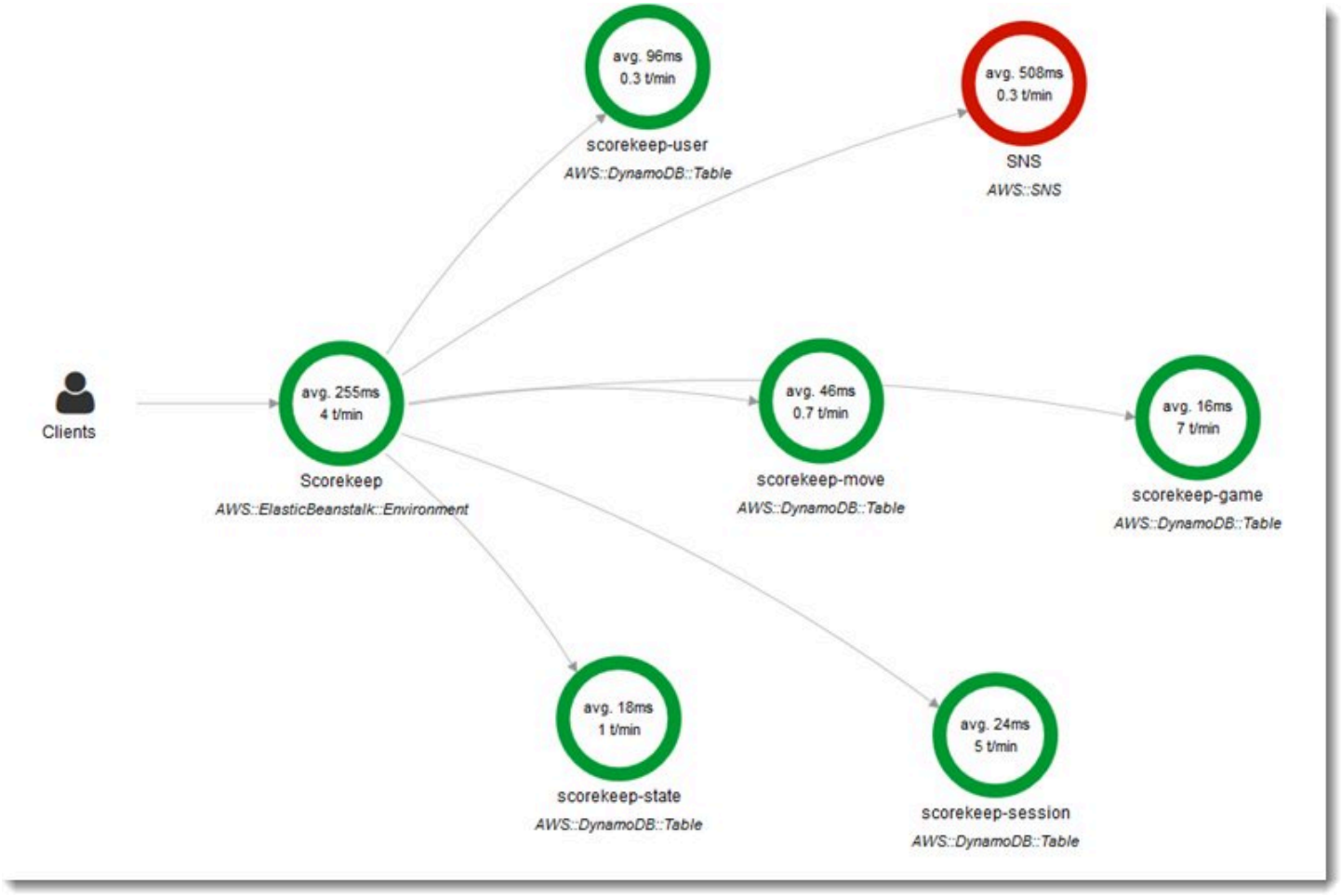

Abbildung 13: AWS X-Ray Servicekarte

AWS Distro for OpenTelemetry ist Teil des OpenTelemetry Projekts und bietet Open-Source-Software und Agenten zur Erfassung verteilter Traces APIs und Metriken, wodurch Ihre Anwendungsüberwachung verbessert wird. Es sendet Metriken und Traces an mehrere Überwachungslösungen AWS und Partnerlösungen. Durch die Erfassung von Metadaten aus Ihren AWS Ressourcen gleicht es die Anwendungsleistung mit den zugrunde liegenden Infrastrukturdaten ab und beschleunigt so die Problemlösung. Darüber hinaus ist es mit einer Vielzahl von AWS Diensten kompatibel und kann vor Ort verwendet werden.

Loggen Sie die Analyse ein AWS

Amazon CloudWatch Logs Insights ermöglicht die Erkundung, Analyse und Visualisierung von Protokollen in Echtzeit. Für die weitere Analyse von Protokolldateien ist Amazon OpenSearch Service, zu dem auch Kibana gehört, ein leistungsstarkes Tool. CloudWatch Protokolle können Protokolleinträge in Echtzeit an den OpenSearch Service streamen. Kibana ist nahtlos in Kibana integriert OpenSearch, visualisiert diese Daten und bietet eine intuitive Suchoberfläche.

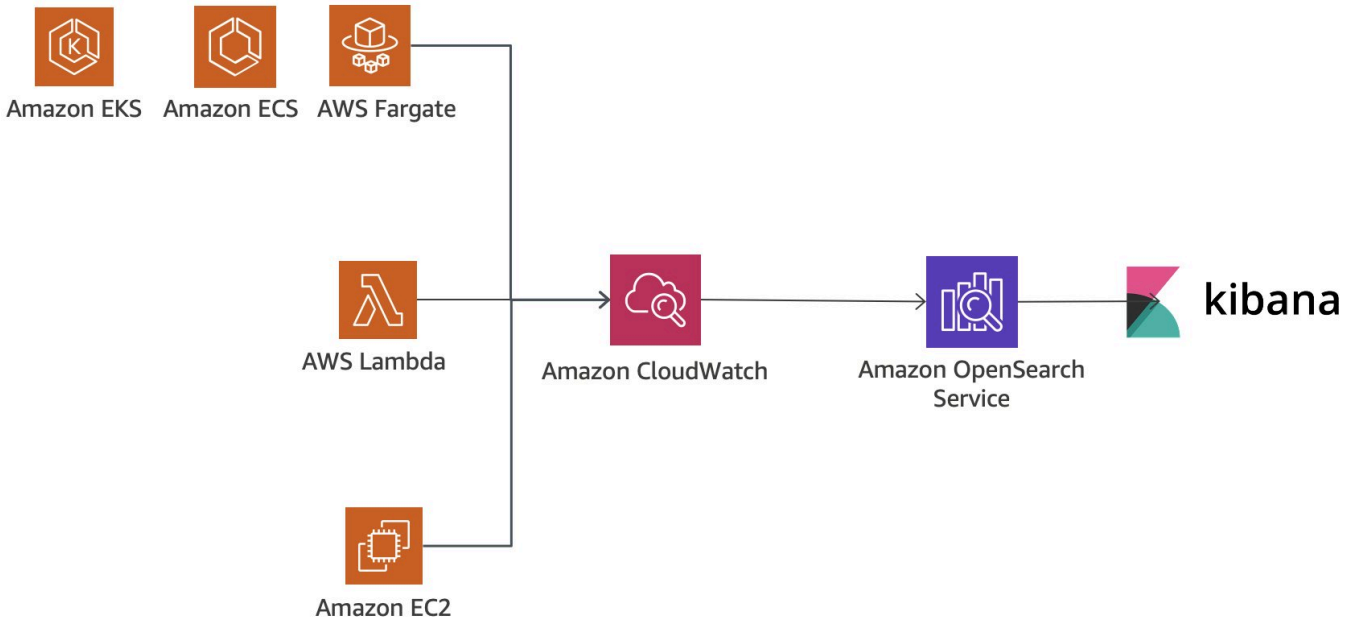

Abbildung 14: Protokollanalyse mit Amazon OpenSearch Service

Andere Optionen für die Analyse

Für weitere Protokollanalysen bieten Amazon Redshift, ein vollständig verwalteter Data Warehouse-Service, und Quick, ein skalierbarer Business Intelligence-Service, effektive Lösungen. QuickSight bietet einfache Konnektivität zu verschiedenen AWS Datendiensten wie Redshift, RDS, Aurora, EMR, DynamoDB, Amazon S3 und Kinesis und vereinfacht so den Datenzugriff.

CloudWatch Logs können Protokolleinträge an Amazon Data Firehose streamen, einen Service zur Bereitstellung von Echtzeit-Streaming-Daten. QuickSight verwendet dann die in Redshift gespeicherten Daten für umfassende Analysen, Berichte und Visualisierungen.

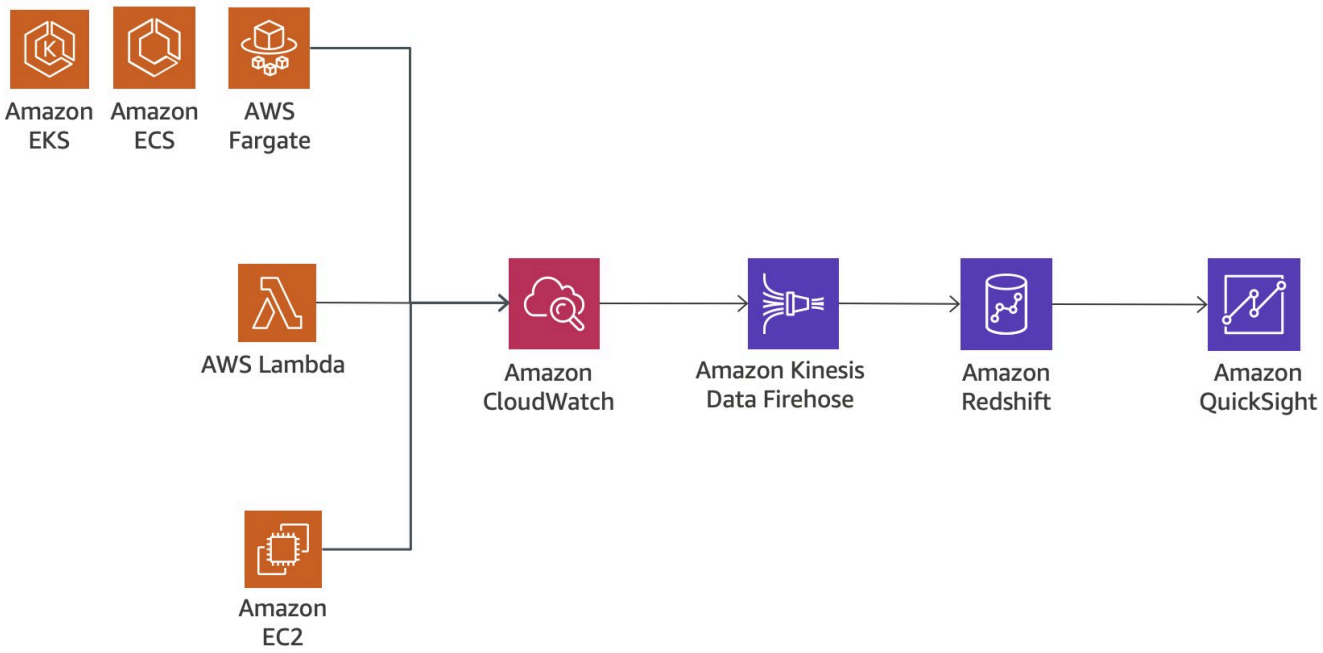

Abbildung 15: Protokollanalyse mit Amazon Redshift und Quick

Wenn Protokolle in S3-Buckets, einem Objektspeicherdienst, gespeichert werden, können die Daten außerdem in Dienste wie Redshift oder EMR, eine cloudbasierte Big-Data-Plattform, geladen werden, was eine gründliche Analyse der gespeicherten Protokolldaten ermöglicht.

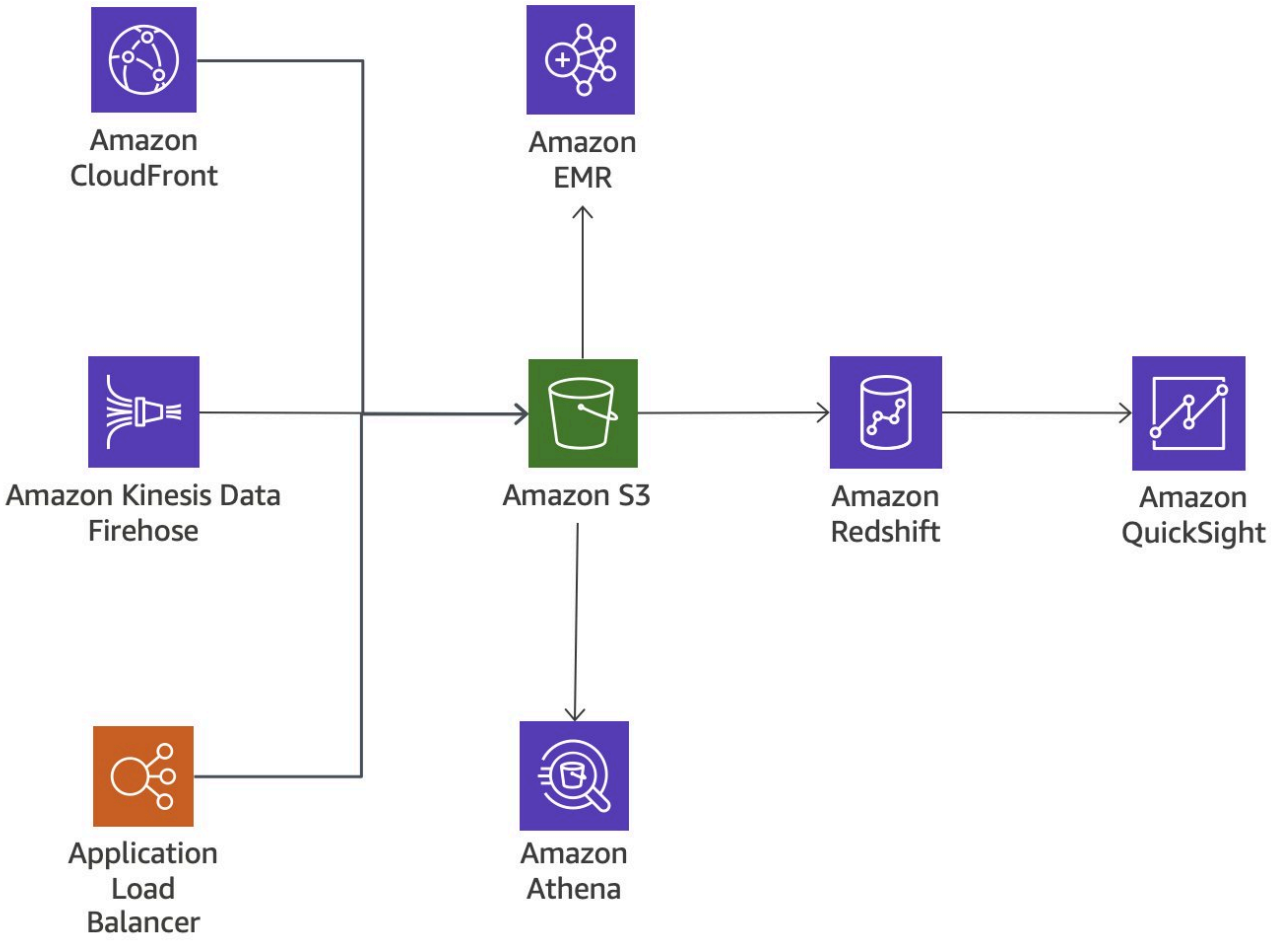

Abbildung 16: Rationalisierung der Protokollanalyse: Von AWS Diensten zu QuickSight

Umgang mit Chattiness in der Microservices-Kommunikation

Chattiness bezieht sich auf eine übermäßige Kommunikation zwischen Microservices, die aufgrund der erhöhten Netzwerklatenz zu Ineffizienz führen kann. Für ein gut funktionierendes System ist es unerlässlich, Chatnachrichten effektiv zu verwalten.

Einige wichtige Tools für die Verwaltung von Chattiness sind REST APIs, HTTP APIs und gRPC. APIs REST APIs bietet eine Reihe erweiterter Funktionen wie API-Schlüssel, Drosselung pro Client, Überprüfung von Anfragen, AWS WAF Integration oder private API-Endpunkte. HTTP APIs sind mit minimalen Funktionen konzipiert und daher zu einem niedrigeren Preis erhältlich. Weitere Informationen zu diesem Thema und eine Liste der Kernfunktionen, die in REST APIs und HTTP verfügbar sind APIs, finden Sie unter Auswahl zwischen REST APIs und HTTP APIs.

Aufgrund der weit verbreiteten Verwendung verwenden Microservices häufig REST über HTTP für die Kommunikation. In Situationen mit hohem Datenvolumen kann der Mehraufwand von REST jedoch zu Leistungsproblemen führen. Das liegt daran, dass die Kommunikation den TCP-Handshake verwendet, der für jede neue Anfrage erforderlich ist. In solchen Fällen ist die gRPC-API die bessere Wahl. gRPC reduziert die Latenz, da es mehrere Anfragen über eine einzige TCP-Verbindung ermöglicht. gRPC unterstützt auch bidirektionales Streaming, sodass Clients und Server Nachrichten gleichzeitig senden und empfangen können. Dies führt zu einer effizienteren Kommunikation, insbesondere bei großen Datenübertragungen oder Datenübertragungen in Echtzeit.

Wenn die Konversation trotz der Wahl des richtigen API-Typs anhält, kann es notwendig sein, Ihre Microservices-Architektur neu zu bewerten. Durch die Konsolidierung von Diensten oder die Überarbeitung Ihres Domänenmodells könnten Sie die Gesprächsfrequenz verringern und die Effizienz verbessern.

Verwendung von Protokollen und Caching

Microservices verwenden häufig Protokolle wie gRPC und REST für die Kommunikation (siehe den vorherigen Abschnitt unter)Kommunikationsmechanismen. gRPC verwendet HTTP/2 für den Transport, während REST normalerweise HTTP/1.1 verwendet. gRPC verwendet Protokollpuffer für die Serialisierung, während REST normalerweise JSON oder XML verwendet. Um die Latenz und den Kommunikationsaufwand zu reduzieren, kann Caching angewendet werden. Dienste wie Amazon ElastiCache oder die Caching-Schicht in API Gateway können dazu beitragen, die Anzahl der Aufrufe zwischen Microservices zu reduzieren.

Prüfung

In einer Microservices-Architektur ist es entscheidend, Einblick in die Benutzeraktionen aller Dienste zu haben. AWS bietet Tools wie AWS CloudTrail, das alle API-Aufrufe protokolliert und AWS CloudWatch das zur Erfassung von Anwendungsprotokollen verwendet wird. AWS Auf diese Weise können Sie Änderungen verfolgen und das Verhalten Ihrer Microservices analysieren. Amazon EventBridge kann schnell auf Systemänderungen reagieren, indem es die richtigen Personen benachrichtigt oder sogar automatisch Workflows zur Problemlösung startet.

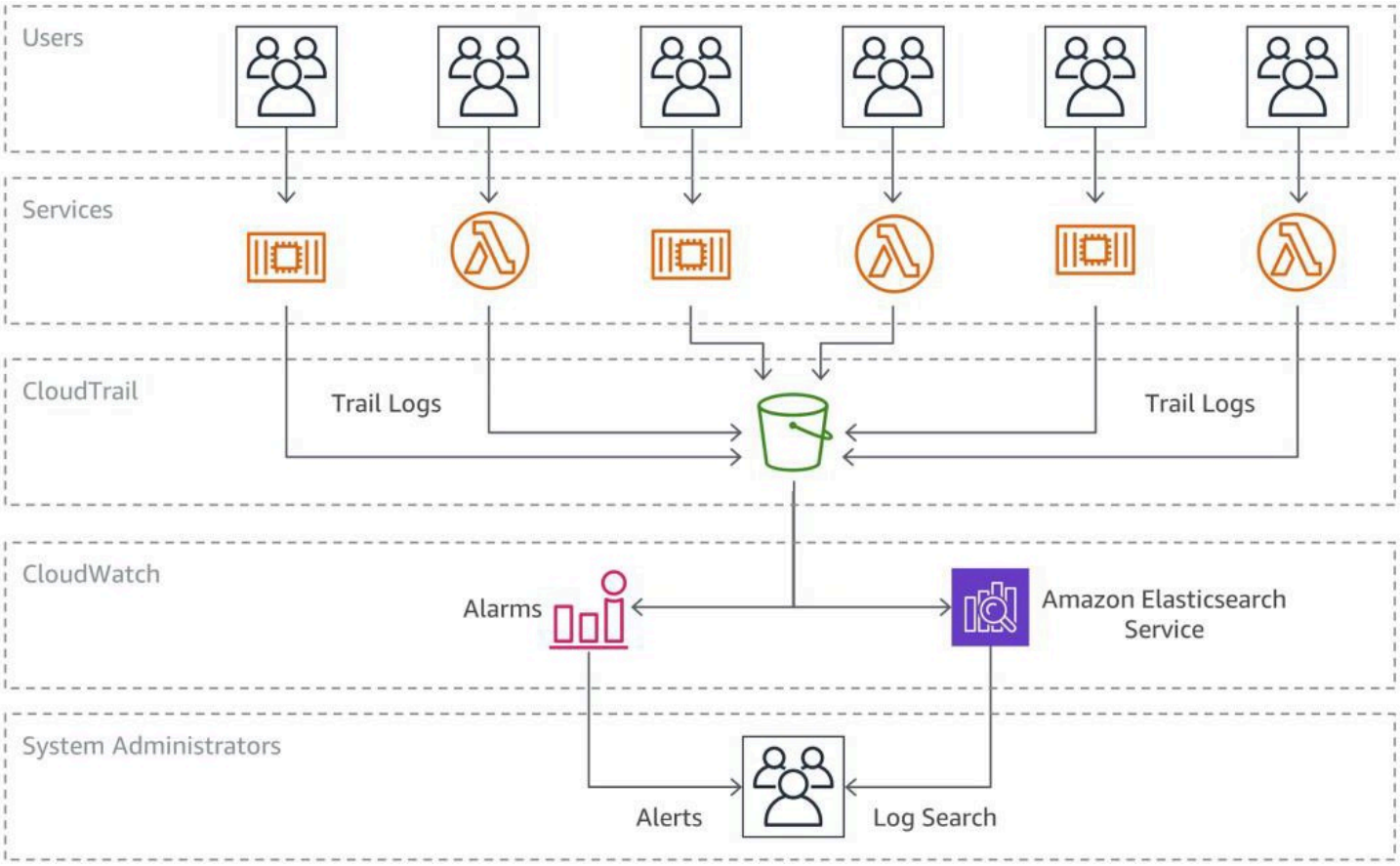

Abbildung 17: Prüfung und Problembehebung für Ihre Microservices

Ressourceninventar und Änderungsmanagement

In einer agilen Entwicklungsumgebung mit sich schnell entwickelnden Infrastrukturkonfigurationen sind automatisierte Prüfungen und Kontrollen von entscheidender Bedeutung. AWS-Config-Regeln bieten einen verwalteten Ansatz zur Überwachung dieser Änderungen in allen Microservices. Sie ermöglichen die Definition spezifischer Sicherheitsrichtlinien, mit denen Richtlinienverstöße automatisch erkannt, nachverfolgt und Warnmeldungen gesendet werden.

Wenn beispielsweise eine API-Gateway-Konfiguration in einem Microservice so geändert wird, dass sie eingehenden HTTP-Verkehr statt nur HTTPS-Anfragen akzeptiert, kann eine vordefinierte AWS Config Regel diese Sicherheitsverletzung erkennen. Sie protokolliert die Änderung zur Prüfung und löst eine SNS-Benachrichtigung aus, wodurch der konforme Status wiederhergestellt wird.

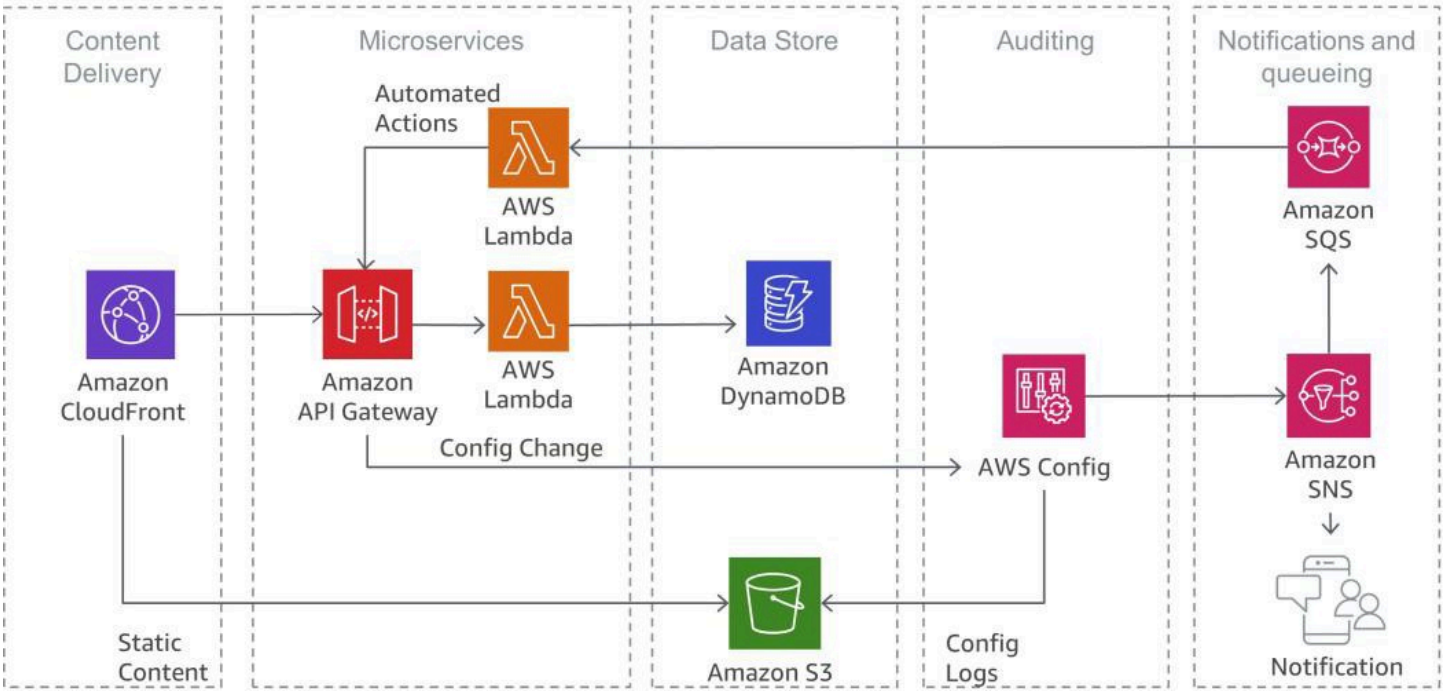

Abbildung 18: Erkennen von Sicherheitsverletzungen mit AWS Config

Schlussfolgerung

Die Microservices-Architektur, ein vielseitiger Entwurfsansatz, der eine Alternative zu herkömmlichen monolithischen Systemen bietet, hilft bei der Skalierung von Anwendungen, beschleunigt die Entwicklungsgeschwindigkeit und fördert das Unternehmenswachstum. Aufgrund seiner Anpassungsfähigkeit kann es mithilfe von Containern, serverlosen Ansätzen oder einer Mischung aus beidem implementiert werden, um auf spezifische Bedürfnisse zugeschnitten zu werden.

Es ist jedoch keine Lösung. one-size-fits-all Jeder Anwendungsfall erfordert angesichts der potenziellen Zunahme der architektonischen Komplexität und der betrieblichen Anforderungen eine sorgfältige Bewertung. Bei strategischer Herangehensweise können die Vorteile von Microservices diese Herausforderungen jedoch erheblich überwiegen. Der Schlüssel liegt in der proaktiven Planung, insbesondere in den Bereichen Beobachtbarkeit, Sicherheit und Änderungsmanagement.

Es ist auch wichtig zu beachten, dass es neben Microservices auch ganz andere Architektur-Frameworks wie generative KI-Architekturen wie Retrieval Augmented Generation (RAG) gibt, die eine Reihe von Optionen bieten, die Ihren Anforderungen am besten entsprechen.

AWS, mit seiner robusten Suite von Managed Services, ermöglicht es Teams, effiziente Microservices-Architekturen aufzubauen und die Komplexität effektiv zu minimieren. Dieses Whitepaper soll Sie durch die entsprechenden AWS Services und die Implementierung der wichtigsten Muster führen. Ziel ist es, Ihnen das Wissen zu vermitteln, mit dem Sie das Potenzial von Microservices optimal nutzen können AWS, sodass Sie deren Vorteile nutzen und Ihre Anwendungsentwicklung transformieren können.

Mitwirkende

Folgende Personen und Organisationen haben zu diesem Dokument beigetragen:

- Sascha Möllering, Lösungsarchitektur, Amazon Web Services
- Christian Müller, Lösungsarchitektur, Amazon Web Services
- Matthias Jung, Lösungsarchitektur, Amazon Web Services
- Peter Dalbhanjan, Lösungsarchitektur, Amazon Web Services
- Peter Chapman, Lösungsarchitektur, Amazon Web Services
- Christoph Kassen, Lösungsarchitektur, Amazon Web Services
- Umair Ishaq, Lösungsarchitektur, Amazon Web Services
- Rajiv Kumar, Lösungsarchitektur, Amazon Web Services
- Ramesh Dwarakanath, Lösungsarchitektur, Amazon Web Services
- Andrew Watkins, Lösungsarchitektur, Amazon Web Services
- Yann Stoneman, Lösungsarchitektur, Amazon Web Services
- Mainak Chaudhuri, Lösungsarchitektur, Amazon Web Services
- Gaurav Acharya, Lösungsarchitektur, Amazon Web Services

Dokumentverlauf

Abonnieren Sie den RSS-Feed, um über Aktualisierungen des Whitepapers benachrichtigt zu werden.

Änderung	Beschreibung	Datum
Größere Aktualisierung	Es wurden Informationen über das AWS Customer Carbon Footprint Tool, Amazon EventBridge, AWS AppSync (GraphQL), AWS Lambda Layers, Lambda, Large Language Models (LLMs) SnapStart, Amazon Managed Streaming for Apache Kafka (MSK), Amazon Managed Workflows für Apache Airflow (MWAA), Amazon VPC Lattice, hinzugefügt. AWS AppConfig Ein separater Abschnitt zu Kostenopt imierung und Nachhaltigkeit wurde hinzugefügt.	31. Juli 2023
Kleinere Updates	Well-Architected wurde zur Zusammenfassung hinzugefü gt.	13. April 2022
Whitepaper aktualisiert	Integration von Amazon EventBridge, AWS OpenTelemetry, AMP, AMG, Container Insights, geringfüg ige Textänderungen.	9. November 2021
Kleinere Updates	Das Seitenlayout wurde angepasst	30. April 2021

Kleinere Updates	Geringfügige Textänderungen.	1. August 2019
Whitepaper aktualisiert	Integration von Amazon EKS, AWS Fargate, Amazon MQ, AWS PrivateLink, AWS App Mesh und AWS Cloud Map	1. Juni 2019
Whitepaper aktualisiert	Integration von AWS Step Functions, AWS X-Ray und ECS-Event-Streams.	1. September 2017
Erste Veröffentlichung	Implementierung von Microservices auf AWS veröffentlicht.	1. Dezember 2016

> **Note**
>
> Um RSS-Updates zu abonnieren, muss für den von Ihnen verwendeten Browser ein RSS-Plug-In aktiviert sein.

Hinweise

Kunden sind dafür verantwortlich, Ihre eigene unabhängige Bewertung der Informationen in diesem Dokument vorzunehmen. Dieses Dokument: (a) dient nur zu Informationszwecken, (b) stellt aktuelle AWS Produktangebote und Praktiken dar, die ohne vorherige Ankündigung geändert werden können, und (c) stellt keine Verpflichtungen oder Zusicherungen von AWS und seinen verbundenen Unternehmen, Lieferanten oder Lizenzgebern dar. AWS Produkte oder Dienstleistungen werden „wie sie sind“ ohne ausdrückliche oder stillschweigende Garantien, Zusicherungen oder Bedingungen jeglicher Art bereitgestellt. Die Verantwortlichkeiten und Verbindlichkeiten AWS gegenüber seinen Kunden werden durch AWS Vereinbarungen geregelt, und dieses Dokument ist weder Teil einer Vereinbarung zwischen AWS und seinen Kunden noch ändert es diese.

AWS Glossar

Die neueste AWS Terminologie finden Sie im AWS Glossar in der AWS-Glossar Referenz.

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.