Serverlose Multi-Tier-Architekturen in AWS

unter Verwendung von Amazon API Gateway und AWS Lambda

November 2015

Hinweise

Dieses Dokument wird nur zu Informationszwecken zur Verfügung gestellt. Es stellt das aktuelle Produktangebot und die Praktiken von AWS zum Erstellungsdatum dieses Dokuments dar. Änderungen vorbehalten. Kunden sind für ihre eigene unabhängige Einschätzung der Informationen in diesem Dokument und jedweder Nutzung der AWS-Services verantwortlich. Jeder Service wird ohne Gewähr und ohne Garantie jeglicher Art, weder ausdrücklich noch impliziert, bereitgestellt. Dieses Dokument gibt keine Garantien, Gewährleistungen, vertraglichen Verpflichtungen, Bedingungen oder Zusicherungen von AWS, seinen Partnern, Zulieferern oder Lizenzgebern. Die Verantwortung und Haftung von AWS gegenüber seinen Kunden werden durch AWS-Vereinbarungen geregelt. Dieses Dokument ist weder ganz noch teilweise Teil der Vereinbarungen von AWS mit seinen Kunden und ändert diese Vereinbarungen auch nicht.

Inhalt

Übersicht

In diesem Whitepaper erfahren Sie, welche neuen Möglichkeiten Ihnen durch Innovationen von Amazon Web Services (AWS) beim Entwurf von Multi-Tier-Architekturen für beliebte Anwendungsmuster wie Microservices, mobile Backends und öffentliche Websites zur Verfügung stehen. Architekten und Entwickler können nun ein Implementierungsmuster verwenden, das Amazon API Gateway sowie AWS Lambda umfasst, und so die Entwicklungs- und Operationszyklen reduzieren, die zum Erstellen und Betreiben von Multi-Tier-Anwendungen erforderlich sind.

Einführung

Die Multi-Tier-Anwendung (3-Schichten, n-Schichten usw.) ist seit Jahrzehnten ein grundlegendes Architekturmuster. Das Multi-Tier-Muster stellt eine gute Richtlinie dar, die für entkoppelte und skalierbare Anwendungskomponenten sorgt, die separat verwaltet und gepflegt werden können (oft durch verschiedene Teams). Bei der Erstellung von Multi-Tier-Anwendungen mit Web-Services wird häufig ein SOA-Ansatz (Service-Oriented Architecture, serviceorientierte Architektur) verwendet. Bei diesem Ansatz fungiert das Netzwerk als Grenze zwischen den Schichten. Allerdings sind bei der Erstellung einer Web-Service-Schicht als Teil einer Anwendung etliche undifferenzierte Aspekte zu berücksichtigen. Ein Großteil des Codes innerhalb einer Multi-Tier-Webanwendung resultiert aus dem Muster selbst. Dabei handelt es sich beispielsweise um Code, der eine Schicht in eine andere integriert oder der eine API und ein Datenmodell definiert, das die Schichten benötigen, um miteinander zu kommunizieren. Ein weiteres Beispiel ist sicherheitsrelevanter Code, der sicherstellt, dass die Integrationspunkte der Schichten nicht in unerwünschter Weise exponiert sind.

Amazon API Gateway[1], ein Service für die Erstellung und Verwaltung von APIs, und AWS Lambda[2], ein Service zur Ausführung beliebiger Codefunktionen, können zusammen verwendet werden, um die Erstellung von robusten Schichtenanwendungen zu vereinfachen.

Das Zusammenwirken von Amazon API Gateway und AWS Lambda ermöglicht, benutzerdefinierte Codefunktionen direkt über eine benutzerdefinierte HTTPS-Anforderung auszulösen. Unabhängig vom benötigten Anforderungsumfang werden sowohl API Gateway als auch Lambda automatisch so skaliert, wie es die Bedürfnisse Ihrer Anwendung erfordern. Durch die Kombination beider Services können Sie eine Schicht für Ihre Anwendung erstellen und sich auf den Code für diese Anwendung konzentrieren, *ohne* die verschiedenen anderen undifferenzierten Aspekte einer Multi-Tier-Architektur berücksichtigen zu müssen, wie Sicherstellen hoher Verfügbarkeit, Schreiben von Client-SDKs, Verwalten von Server und Betriebssystem sowie Skalieren und Implementieren eines Autorisierungsmechanismus für Clients.

Kürzlich hat AWS bekanntgegeben, dass die Erstellung von Lambda-Funktionen unterstützt wird, die in Ihrer Amazon Virtual Private Cloud (Amazon VPC)[3] ausführbar sind. Durch diese Möglichkeit sind die Vorteile der Kombination von API Gateway und Lambda auch in einer Vielzahl von Anwendungsfällen nutzbar, in denen Datenschutz im Netzwerk erforderlich ist. Ein Beispiel wäre ein Web-Service, der auf vertrauliche Daten in einer relationalen Datenbank zugreift. Die Integration von Lambda und Amazon VPC hat indirekt die Fähigkeiten von Amazon API Gateway erweitert, weil Entwickler nun die Möglichkeit haben, eigene, über das Internet zugängliche HTTPS-APIs vor einem Backend zu definieren, das als Teil von Amazon VPC privat und sicher bleibt. Die Vorteile dieses leistungsstarken Musters kommen auf jeder Schicht einer Multi-Tier-Architektur zum Tragen. In diesem Whitepaper konzentrieren wir uns auf die häufigste Multi-Tier-Architektur, die **Drei-Schichten-Webanwendung**. Dieses Schichtenmuster ist jedoch auch auf andere Multi-Tier-Webanwendungen anwendbar.

Übersicht über die Drei-Schichten-Architektur

Die Drei-Schichten-Architektur ist ein beliebtes Muster für benutzerbezogene Anwendungen. Diese Architektur besteht aus der **Darstellungsschicht**, der **Logikschicht** und der **Datenschicht**. Die Darstellungsschicht ist die Komponente, mit der die Benutzer direkt interagieren, beispielsweise eine Webseite, die Benutzeroberfläche einer mobile App oder dergl. Die Logikschicht enthält den Code, der erforderlich ist, um Benutzeraktionen auf der Darstellungsschicht in die Funktionalität zu übersetzen, die das Verhalten der Anwendung steuert. Die Datenschicht umfasst die Speichermedien (Datenbanken, Objektspeicher, Cache-Speicher, Dateisysteme usw.), auf denen die für die Anwendung relevanten Daten enthalten sind. Abbildung 1 zeigt ein Beispiel einer einfachen Drei-Schichten-Anwendung.

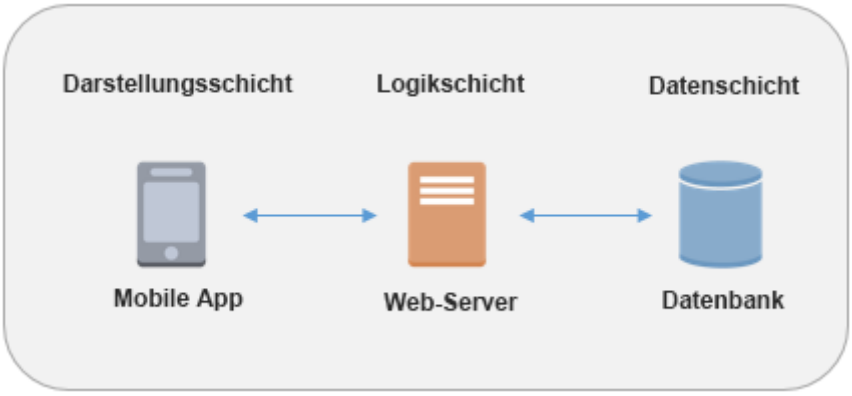

Abbildung 1: Architekturmuster für eine einfache Drei-Schichten-Anwendung

Sie finden online viele gute Ressourcen mit weiteren Informationen über das *allgemeine* Drei-Schichten-Architekturmuster. In diesem Whitepaper konzentrieren wir uns auf eine spezifische Implementierung dieser Architektur mithilfe von Amazon API Gateway und AWS Lambda.

Die serverlose Logikschicht

Die Logikschicht einer Drei-Schichten-Architektur stellt sozusagen das Gehirn der Anwendung dar. Aus diesem Grund kann die Einbeziehung von Amazon API Gateway und AWS Lambda bei der Erstellung der Logikschicht von großer Bedeutung sein. Die Funktionen der beiden Services ermöglichen Ihnen die Erstellung einer serverlosen Produktionsanwendung, die hochverfügbar, skalierbar und sicher ist. Selbst wenn Ihre Anwendung Tausende von Servern umfasst, müssen Sie bei der Verwendung dieses Musters nicht einen einzigen davon verwalten. Darüber hinaus bieten Ihnen diese verwalteten Services folgende Vorteile:

- Kein Auswählen, Sichern, Patchen oder Verwalten von Betriebssystemen
- Kein Dimensionieren, Überwachen oder Skalieren von Servern
- Kein Kostenrisiko durch unnötige Bereitstellungen
- Kein Leistungsrisiko durch unzureichende Bereitstellungen

Zusätzlich bietet jeder einzelne Service spezielle Funktionen, von denen ein Multi-Tier-Architekturmuster profitiert.

Amazon API Gateway

Amazon API Gateway ist ein vollständig verwalteter Service zum Definieren, Bereitstellen und Warten von APIs. Clients kommunizieren mit den APIs über Standard-HTTPS-Anforderungen. Die Anwendbarkeit auf eine Service-orientierte Multi-Tier-Architektur ist offensichtlich. Spezielle Funktionen und Qualitäten machen den Service zu einer leistungsfähigen Komponente der Logikschicht.

Zusammenwirken mit AWS Lambda

Amazon API Gateway ermöglicht Ihrer Anwendung auf einfache Art und Weise (HTTPS-Anforderungen), die Innovation von AWS Lambda direkt zu nutzen. API Gateway bildet die Brücke, die Ihre Darstellungsschicht mit den in AWS Lambda geschriebenen Funktionen verbindet. Nach der Definition der Client-Server-Beziehung mit Ihrer API wird der Inhalt der HTTPS-Anforderung des Clients zur Ausführung an die Lambda-Funktion übergeben. Diese Inhalte umfassen die Metadaten, Header und Nutzdaten der Anforderung.

Stabile API-Leistung weltweit

Jede Bereitstellung von Amazon API Gateway enthält eine Amazon CloudFront-Verteilung[4]. Amazon CloudFront ist ein Web-Service für die Bereitstellung von Inhalten, der das globale Amazon-Netz von Edge-Standorten nutzt und diese als Verbindungspunkte für Clients verwendet, die Ihre API integrieren. Das trägt dazu bei, die Gesamtlatenz der Reaktionszeit Ihrer API zu reduzieren. Durch die Verwendung von mehreren Edge-Standorten in der ganzen Welt bietet Amazon CloudFront auch Möglichkeiten zur Abwehr von Distributed-Denial-of-Service (DDoS)-Angriffen. Weitere Informationen finden Sie im Whitepaper AWS Best Practices for Combatting DDoS Attacks[5].

Sie können die Leistung bestimmter API-Anforderungen verbessern, indem Sie Amazon API Gateway verwenden, um Antworten in einem optionalen In-Memory-Cache zu speichern. Das bietet nicht nur Leistungsvorteile bei wiederholten API-Anforderungen, sondern reduziert auch die Backend-Ausführungen und damit möglicherweise Ihre Gesamtkosten.

Mut zu Innovationen

Die Entwicklungsarbeit für jede neue Anwendung ist eine Investition. Diese müssen Sie begründen, um das Projekt beginnen zu können. Durch die Reduzierung der erforderlichen Investitionen für Entwicklungsaufgaben und Arbeitszeit gewinnen Sie Freiraum für neue Entwicklungen und Innovationen.

In vielen Multi-Tier-Anwendungen, die auf einem Web-Service basieren, ist die Darstellungsschicht im Hinblick auf die Benutzer (separate mobile Geräte, Web-Browser usw.) fragmentiert. Zudem befinden sich diese Benutzer an unterschiedlichen geografischen Standorten. Dagegen wird eine entkoppelte Logikschicht von den Benutzern nicht physisch fragmentiert. Alle Benutzer sind abhängig von derselben Infrastruktur, in der Ihre Logikschicht ausgeführt wird, was die Bedeutung der Infrastruktur erhöht. Um eine neue Anwendung schneller liefern zu können, wird oftmals mit Argumenten wie „wir müssen beim ersten Start keine Metriken erfassen“ oder „die Auslastung wird zunächst gering sein, um eine Skalierung kümmern wir uns später“ dafür plädiert, die Logikschicht für die erstmalige Implementierung zu vereinfachen. Dies kann zu technischen Schulden und operationellen Risiken führen, wenn Sie die notwendigen Änderungen in einer Anwendung bereitstellen müssen, die bereits in der Produktionsphase arbeitet. Amazon API Gateway erlaubt *Ihnen*, durch derartige Vereinfachungen schneller zu liefern, da der Service sie bereits für Sie implementiert hat.

Möglicherweise ist die Gesamtlebensdauer einer Anwendung unbekannt, oder es steht fest, dass sie kurz sein wird. Aus diesen Gründen kann es schwierig sein, ein Business Case für eine neue Multi-Tier-Anwendung zu erstellen. Das fällt jedoch leichter, wenn Ihr Ausgangspunkt bereits die verwalteten Funktionen enthält, die Amazon API Gateway bietet, und wenn Infrastrukturkosten erst dann anfallen, nachdem Ihre APIs erste Anforderungen verarbeitet haben. Weitere Informationen dazu finden Sie unter Amazon API Gateway – Preise[6].

Schnell reagieren und agil bleiben

Für neue Anwendungen kann die Benutzerbasis bezüglich Umfang, Nutzungsverhalten usw. noch sehr undefiniert sein. Deshalb muss die Logikschicht agil bleiben, während die Benutzerbasis Formen annimmt. Ihre Anwendung und Ihr Unternehmen müssen in der Lage sein, die sich ändernden Erwartungen der Early Adopters aufzunehmen und zu integrieren. Amazon API Gateway reduziert die Anzahl der Entwicklungszyklen, die für eine API von der Konzeption bis zur Bereitstellung erforderlich sind. Amazon API Gateway unterstützt die Erstellung von Mock Integrations[7], die es ermöglichen, API-Antworten direkt in API Gateway zu generieren. Mithilfe dieser Antworten können Sie Client-Anwendungen und parallel dazu die volle Backend-Logik entwickeln. Dies ist nicht nur für eine erste API-Bereitstellung von Vorteil, sondern auch für den Fall, dass im Unternehmen entschieden wird, die Anwendung (und die vorhandene API) schnell an das Feedback der Benutzer anzupassen. API Gateway und AWS Lambda unterstützen Versioning, sodass vorhandene Funktionalität und Client-Abhängigkeiten erhalten bleiben, während neue Funktionalität in Form einer separaten API oder Funktion freigegeben wird.

Sicherheit

Bei der Implementierung der Logikschicht einer öffentlichen Drei-Schichten-Webanwendung steht sofort das Thema Sicherheit im Vordergrund. Die Anwendung muss sicherzustellen, dass nur autorisierte Clients Zugriff auf die Logikschicht haben (die über das Netzwerk exponiert ist). Mit Amazon API Gateway können Sie darauf vertrauen, dass Ihr Backend sicher ist, denn der Service stellt etliche Sicherheitsmaßnahmen bereit. Bezüglich der Zugriffskontrolle sollten Sie vermeiden, für Ihre Client-Anwendungen statische API-Schlüsselzeichenfolgen bereitzustellen, denn diese können auf dem Client ausgelesen und anderswo verwendet werden. Amazon API Gateway kann auf unterschiedlichen Weise dazu beitragen, Ihre Logikschicht zu sichern:

- Alle an Ihre APIs gerichteten Anforderungen können über HTTPS erfolgen, um eine verschlüsselte Übertragung zu ermöglichen.
- Ihre AWS Lambda-Funktionen können den Zugriff beschränken, sodass eine Vertrauensbeziehung nur zwischen einer bestimmten API innerhalb von Amazon API Gateway und einer bestimmten Funktion in AWS Lambda besteht. Diese Lambda-Funktion kann ausschließlich mithilfe der API aufgerufen werden, in der Sie die Funktion bereitstellen.

- Amazon API Gateway ermöglicht Ihnen, Client-SDKs zu generieren, die Ihre APIs integrieren. Ein solches SDK verwaltet auch das Signieren von Anforderungen, wenn APIs Authentifizierung erfordern. Diese auf dem Client zur Authentifizierung verwendeten API-Anmeldeinformationen werden direkt an Ihre AWS Lambda-Funktion übergeben, die bei Bedarf zur weiteren Authentifizierung Code ausführen kann, den Sie geschrieben haben und besitzen.
- Jede Kombination von Ressourcen und Methoden, die Sie als Teil Ihrer API erstellen, erhält einen eigenen spezifischen Amazon Resource Name (ARN), auf den in AWS Identity und Access Management (IAM)[8]-Richtlinien verwiesen werden kann.
  - Somit werden Ihre APIs zusammen mit den anderen AWS-eigenen APIs sozusagen als Bürger erster Klasse behandelt. IAM-Richtlinien können so detailliert sein, dass sie spezifische Ressourcen/Methoden einer API referenzieren, die mit Amazon API Gateway erstellt wurde.
  - Der API-Zugriff wird durch die IAM-Richtlinien erzwungen, die Sie außerhalb des Kontexts Ihres Anwendungscodes erstellen. Somit müssen Sie keinen Code schreiben, der diese Zugriffsebenen beachtet oder eine Beachtung erzwingt. Und Code, der nicht vorhanden ist, kann keine Fehler enthalten oder für Angriffe ausgenutzt werden.
  - Das Autorisieren von Clients mithilfe von AWS Signature Version 4 (SigV4)[9] und IAM-Richtlinien für den API-Zugriff ermöglichen, mit denselben Anmeldeinformationen je nach Bedarf auch den Zugriff auf andere Services und Ressourcen von AWS (beispielsweise Amazon S3-Buckets oder Amazon DynamoDB-Tabellen) zu beschränken oder zu erlauben.

AWS Lambda

Im Grunde ermöglicht AWS Lambda als Reaktion auf ein Ereignis die Ausführung von beliebigem Code, der in einer der unterstützten Sprachen (Node auf JVM-Basis und Python ab November 2015) geschrieben ist. Ein solches Ereignis wird von einem der programmgesteuerten Trigger ausgelöst, die AWS verfügbar macht und die als **Ereignisquellen** bezeichnet werden (die unterstützten Ereignisquellen finden Sie hier[10]). Viele der gängigen Anwendungsfälle für AWS Lambda enthalten ereignisgesteuerte Datenverarbeitungsabläufe wie das Verarbeiten von Dateien, die in Amazon Simple Storage Service (Amazon S3)[11] gespeichert sind, oder das Streamen von Datensätzen aus Amazon Kinesis[12].

In Verbindung mit Amazon API Gateway kann eine AWS Lambda-Funktion im Kontext eines typischen Web-Service enthalten sein und durch eine HTTPS-Anforderung direkt ausgelöst werden. Amazon API Gateway fungiert als Eingangstür zu Ihrer Logikschicht, aber jetzt müssen Sie die Logik hinter diesen APIs auszuführen. Hier kommt AWS Lambda ins Spiel.

Hier steht die Geschäftslogik

AWS Lambda ermöglicht Ihnen, Codefunktionen zu schreiben (sog. **Handler**), deren Ausführung durch ein Ereignis ausgelöst wird. Zum Beispiel können Sie einen Handler schreiben, der ausgelöst wird, wenn ein Ereignis wie eine HTTPS-Anforderung für Ihre API eintritt. Lambda ermöglicht Ihnen, modulare Handler auf der jeweils benötigten Stufe (einen pro API oder einen pro API-Methode) zu erstellen, die unabhängig voneinander aktualisiert, aufgerufen und geändert werden können. Der Handler kann dann alle seine anderen Abhängigkeiten (wie weitere Funktionen, die Sie mit Ihrem Code hochgeladen haben, Bibliotheken, native Binärdateien oder sogar externe Web-Services) erreichen. Mit Lambda können Sie bei der Erstellung einer Funktion alle erforderlichen Abhängigkeiten in der Funktionsdefinition verpacken. Dazu geben Sie bei der Erstellung an, welche Methode in Ihrem Bereitstellungspaket als Anforderungs-Handler fungiert. Es steht Ihnen frei, dasselbe Bereitstellungspaket für mehrere Lambda-Funktionsdefinitionen zu verwenden, wobei jede Lambda-Funktion in einem Bereitstellungspaket einen eigenen Handler besitzen kann. In einem serverlosen Multi-Tier-Architekturmuster integriert jede der APIs, die Sie in Amazon API Gateway erstellen, eine Lambda-Funktion (und den zugehörigen Handler), welche die erforderliche Geschäftslogik ausführt.

Integration von Amazon VPC

AWS Lambda, der Kern Ihrer Logikschicht, ist die Komponente, die direkt mit der Datenschicht zusammenarbeitet. Da die Datenschicht häufig sensible Geschäfts- oder Benutzerinformationen enthält, sollte sie gut gesichert sein. Für AWS-Services, die über eine Lambda-Funktion erreichbar sind, können Sie die Zugriffskontrolle mit IAM-Richtlinien verwalten. Zu diesen Services gehören Amazon S3, Amazon DynamoDB, Amazon Kinesis, Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), andere AWS Lambda-Funktionen und weitere. Möglicherweise verfügen Sie jedoch über eine Komponente, die ihre eigene Zugriffskontrolle steuert, wie beispielsweise eine relationale Datenbank. Für solche Komponenten wird die Sicherheit erhöht, wenn sie in einer privaten Netzwerkumgebung bereitgestellt werden – einer Amazon Virtual Private Cloud (Amazon VPC)[13].

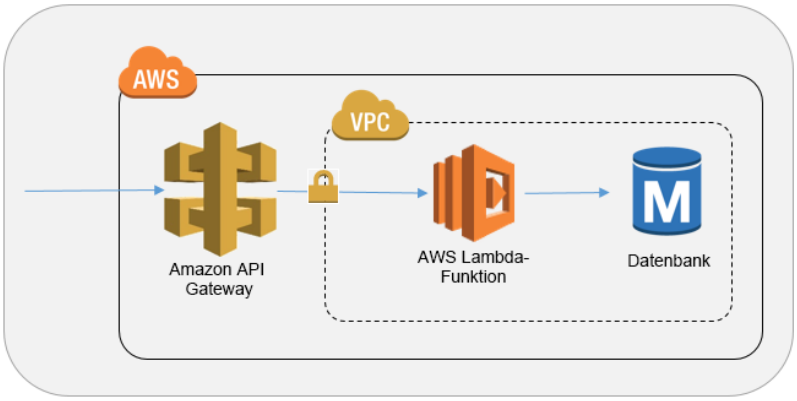

Abbildung 2: Architekturmuster mit einer VPC

Die Verwendung einer VPC bedeutet, dass die Datenbanken und andere Speichermedien, von denen Ihre Geschäftslogik abhängt, über das Internet zugänglich gemacht werden können. Zudem stellt eine VPC sicher, dass der Zugriff auf Ihre Daten aus dem Internet *ausschließlich* über die von Ihnen definierten APIs und die von Ihnen geschriebenen Lambda-Codefunktionen möglich ist.

Sicherheit

Die Ausführung einer Lambda-Funktion kann nur durch Ereignisse oder Services ausgelöst werden, die dazu aufgrund einer IAM-Richtlinie berechtigt sind. Es ist möglich, eine Lambda-Funktion zu erstellen, die *nur dann* ausgeführt wird, wenn sie durch eine von Ihnen definierte API Gateway-Anforderung aufgerufen wird. Ihr Code wird nur als Teil Ihres gültigen Anwendungsfalls verarbeitet; der durch die von Ihnen erstellte API definiert ist.

Jeder Lambda-Funktion ist eine IAM-Rolle zugeordnet, die über eine IAM-Vertrauensbeziehung gewährt werden muss. Das IAM-Rolle definiert die anderen Services und Ressourcen in AWS, mit denen Ihre Lambda-Funktion interagieren kann (z. B. eine Amazon DynamoDB-Tabelle oder ein Amazon S3-Bucket). Die Services, auf die Ihre Funktion zugreifen kann, werden außerhalb der Funktion definiert und gesteuert. Dies ist subtil, aber wirkungsvoll. Es ermöglicht, dass der von Ihnen geschriebene Code keine AWS-Anmeldeinformationen speichern oder abrufen muss. Dies bedeutet, dass Ihr Code keine API-Schlüssel enthalten muss, und dass es nicht erforderlich ist, Code zu schreiben, der solche Schlüssel abruft oder im Arbeitsspeicher ablegt. Die Aktivierung Ihrer Lambda-Funktion zum Aufrufen der Services, die in der IAM-Rolle der Funktion definiert sind, wird für Sie durch den Service selbst verwaltet.

Die Datenschicht

Durch die Verwendung von AWS Lambda als Logikschicht verfügen Sie über eine große Anzahl von Datenspeicheroptionen für Ihre Datenschicht. Diese Optionen gliedern sich in zwei große Kategorien: In Amazon VPC gehostete Datenspeicher und IAM-fähige Datenspeicher. AWS Lambda kann mit beiden Datenspeicherarten zusammenarbeiten.

In Amazon VPC gehostete Datenspeicher

Das Zusammenwirken von AWS Lambda und Amazon VPC ermöglicht Funktionen, eine Vielzahl von Datenspeichertechnologien in einer privaten und sicheren Art und Weise zu verwenden.

- Amazon RDS[14]

  Verwenden Sie jede der Engines, die Amazon Relational Database Service (Amazon RDS) bereitstellt. Sie stellen die Verbindung mit Amazon RDS direkt aus dem Code her, den Sie in Lambda geschrieben haben, und zwar in gleicher Weise wie außerhalb von Lambda, aber mit dem Vorteil der einfachen Integration des AWS Key Management Service (AWS KMS) zur Verschlüsselung von Anmeldeinformationen für Datenbanken.

- Amazon ElastiCache[15]

  Verwenden Sie Ihre Lambda-Funktionen zusammen mit einem verwalteten In-Memory-Cache, um die Leistung Ihrer Anwendung zu steigern.

- Amazon RedShift[16]

  Schreiben Sie beispielsweise sichere Abfragefunktionen für ein Enterprise Data Warehouse, um Berichte oder Dashboards zu erstellen oder Ad-hoc-Ergebnisse abzurufen.

- Von Amazon Elastic Compute Cloud (Amazon EC2)[17] gehostete private Web-Services

  Möglicherweise verfügen Sie über Anwendungen, die als private Web-Services in einer VPC ausgeführt werden. Veranlassen Sie HTTP-Anforderungen über Ihr logisches privates VPC-Netzwerk mithilfe einer Lambda-Funktion.

IAM-fähige Datenspeicher

Da IAM in AWS Lambda integriert ist, kann dieser Service IAM verwenden, um eine sichere Zusammenarbeit mit jedem AWS Service zu ermöglichen, der mithilfe der AWS-APIs direkt angesprochen werden kann.

- Amazon DynamoDB[18]

  Amazon DynamoDB ist die unbegrenzt skalierbare NoSQL-Datenbank von AWS. Ziehen Sie Amazon DynamoDB in Erwägung, wenn Sie Datensätze bis zu einer Größe von 400 KB (Maximalwert, als dieses Whitepaper geschrieben wurde) mit einer Leistung im einstelligen Millisekundenbereich abrufen möchten, unabhängig vom Umfang der Datenbank. Dank der fein justierbaren Zugriffssteuerung von Amazon DynamoDB können Ihre Lambda-Funktionen beim Abfragen bestimmter Daten in DynamoDB die bewährte Methode der geringsten Rechte anwenden.

- Amazon S3[19]

  Amazon Simple Storage Service (Amazon S3) ist ein skalierbarer Objektspeicher für das Internet. Amazon S3 ist für eine Dauerhaftigkeit der Objekte von 99,999999999 % ausgelegt. Ziehen Sie diesen Service in Erwägung, wenn Ihre Anwendung kostengünstigen, sehr robusten Speicher benötigt. Zudem bietet Amazon S3 eine Verfügbarkeit der Objekte von 99,99 % pro Jahr. Beachten Sie dies, wenn Ihre Anwendung hochverfügbaren Speicher benötigt. Auf die in Amazon S3 gespeicherten Objekte (Dateien, Bilder, Protokolle, beliebige Binärdaten) kann direkt über HTTP zugegriffen werden. Lambda-Funktionen können über virtuelle private Endpunkte sicher mit Amazon S3 kommunizieren, und der Zugriff auf Daten in S3 kann gemäß der IAM-Richtlinie eingeschränkt werden, die jeder Lambda-Funktion zugeordnet ist.

- Amazon Elasticsearch Service[20]

  Amazon Elasticsearch Service (Amazon ES) ist eine verwaltete Version der beliebten Such- und Analyse-Engine Elasticsearch. Amazon ES bietet verwaltete Bereitstellung von Clustern, Fehlererkennung und den Austausch von Knoten. Sie können mit IAM-Richtlinien den Zugriff auf die API von Amazon ES beschränken.

Die Darstellungsschicht

Amazon API Gateway bietet eine Vielzahl von Möglichkeiten auf der Darstellungsschicht. Ein über das Internet erreichbares HTTPS-API kann von jedem Client verwendet werden, der die HTTPS-Kommunikation beherrscht. Die folgende Liste enthält gängige Beispiele, die auf der Darstellungsschicht Ihrer Anwendung verwendet werden können:

- Mobile App: Neben der Einbeziehung eigener Geschäftslogik über Amazon API Gateway und AWS Lambda könnten Sie Amazon Cognito[21] als Mechanismus zum Erstellen und Verwalten von Benutzeridentitäten verwenden.
- Statische Website-Inhalte (z. B. in Amazon S3 gehostete Dateien): Sie können ermöglichen, dass Ihre APIs für Amazon API Gateway CORS-konform (Cross-Origin Resource Sharing) sind. Dies ermöglicht Web-Browsern, Ihre APIs direkt aus den statischen Webseiten aufzurufen.
- Jedes andere HTTPS-fähige Client-Gerät: Viele angeschlossene Geräte sind in der Lage, über HTTPS zu kommunizieren. Es gibt bezüglich der Art und Weise, wie Clients mit den von Ihnen mithilfe von Amazon API Gateway erstellten APIs kommunizieren, keine Besonderheiten oder Vorgaben. Es wird reines HTTPS erwartet. Spezielle Client-Software oder Lizenzen sind nicht erforderlich.

Einfache Architekturmuster

Sie können die folgenden beliebten Architekturmuster implementieren und mit Amazon API Gateway und AWS Lambda die verbindende Logikschicht bilden. Für jedes Beispiel verwenden wir nur AWS-Services, sodass Benutzer keine eigene Infrastruktur verwalten müssen.

Mobiles Backend

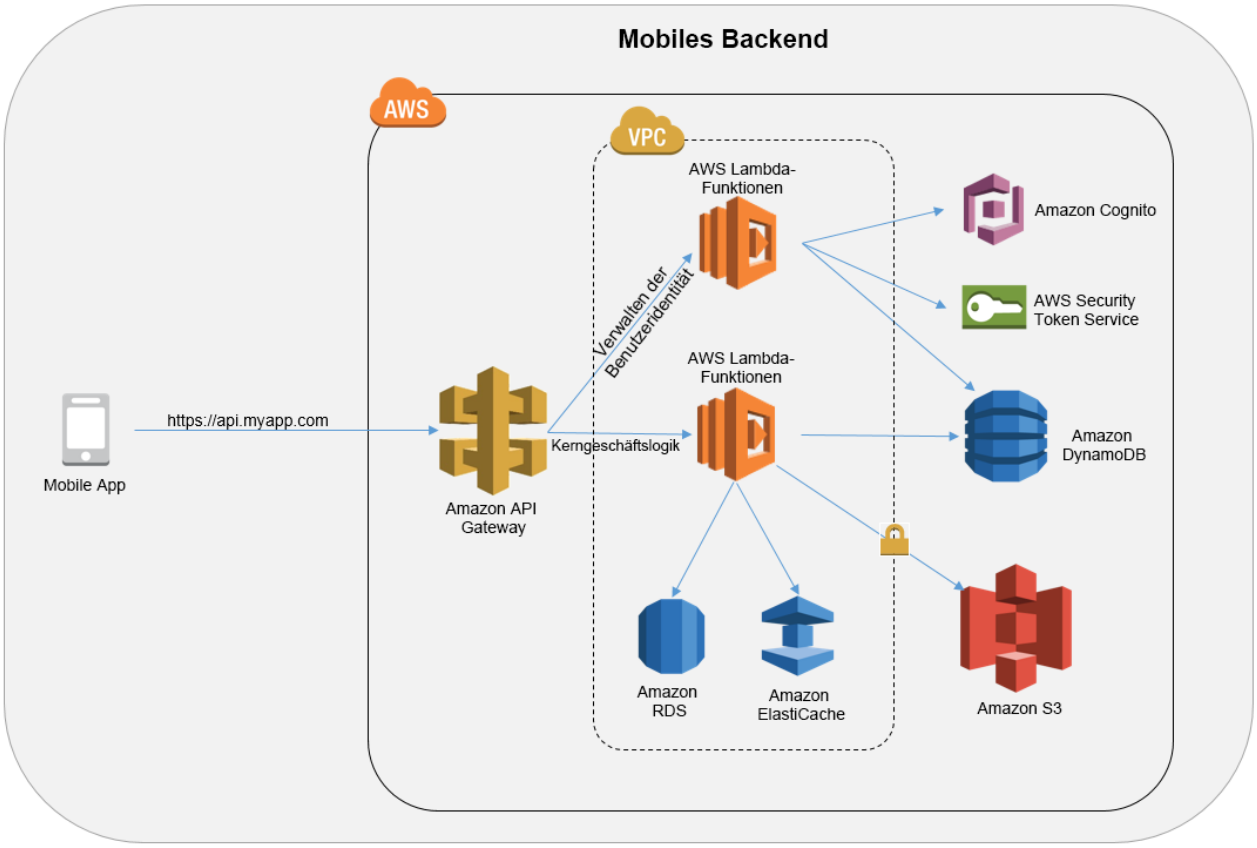

Abbildung 3: Architekturmuster für ein mobiles Backend

- **Darstellungsschicht:** Eine mobile Anwendung, die auf dem Smartphone jedes Benutzers läuft.
- **Logikschicht:** Amazon API Gateway und AWS Lambda Die Logikschicht wird weltweit durch die Amazon CloudFront-Verteilung bereitgestellt, die als Teil jeder Amazon API Gateway-API erstellt wird. Eine Reihe von Lambda-Funktionen kann speziell für die Identitätsverwaltung von Benutzern und Geräten sowie zur Authentifizierung ausgelegt sein und von Amazon Cognito verwaltet werden. Dadurch wird IAM für temporäre Benutzeranmeldeinformationen sowie für populäre Identitäts-Provider von Drittanbietern zur Verfügung gestellt. Andere Lambda-Funktionen können die grundlegende Geschäftslogik für Ihr mobiles Backend definieren.

- **Datenschicht:** Die verschiedenen Datenspeicher-Services können je nach Bedarf verwendet werden. Die entsprechenden Optionen werden weiter oben in diesem Papier erläutert.

In Amazon S3 gehostete Website

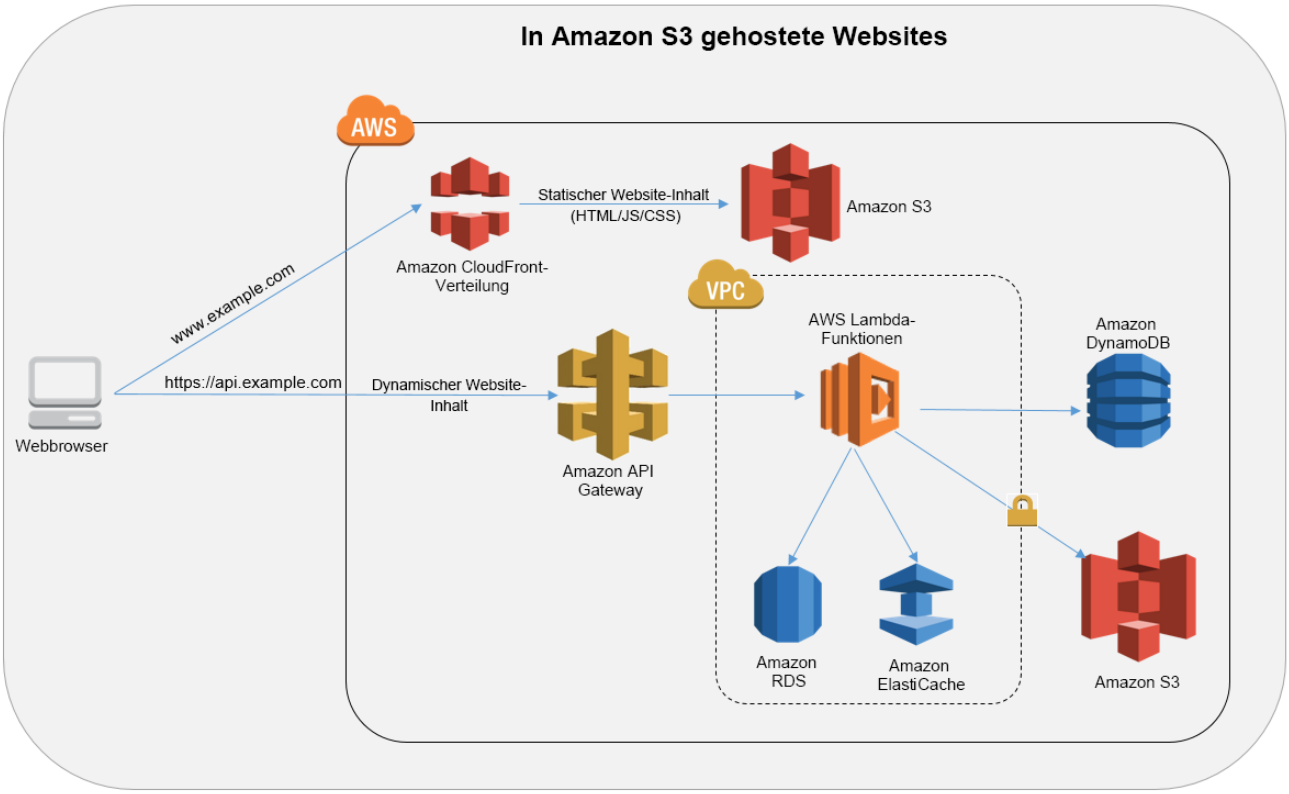

Abbildung 4: Architekturmuster für eine in Amazon S3 gehostete statische Website

- **Darstellungsschicht:** In Amazon S3 gehosteter Inhalt der statischen Website, verteilt mit Amazon CloudFront. Das Hosten statischer Website-Inhalte auf Amazon S3 ist eine kostengünstige Alternative zum Hosten von Inhalten in einer serverbasierten Infrastruktur. Enthält eine Website jedoch aufwändige Funktionen, ist für den statischen Inhalt oft ein dynamisches Backend erforderlich.
- **Logikschicht:** Amazon API Gateway und AWS Lambda In Amazon S3 gehosteter statischer Web-Inhalt kann direkt und CORS-konform mit Amazon API Gateway zusammenwirken.

- **Datenschicht:** Die verschiedenen Datenspeicher-Services können je nach Bedarf verwendet werden. Die entsprechenden Optionen werden weiter oben in diesem Whitepaper erläutert.

Microservices-Umgebung

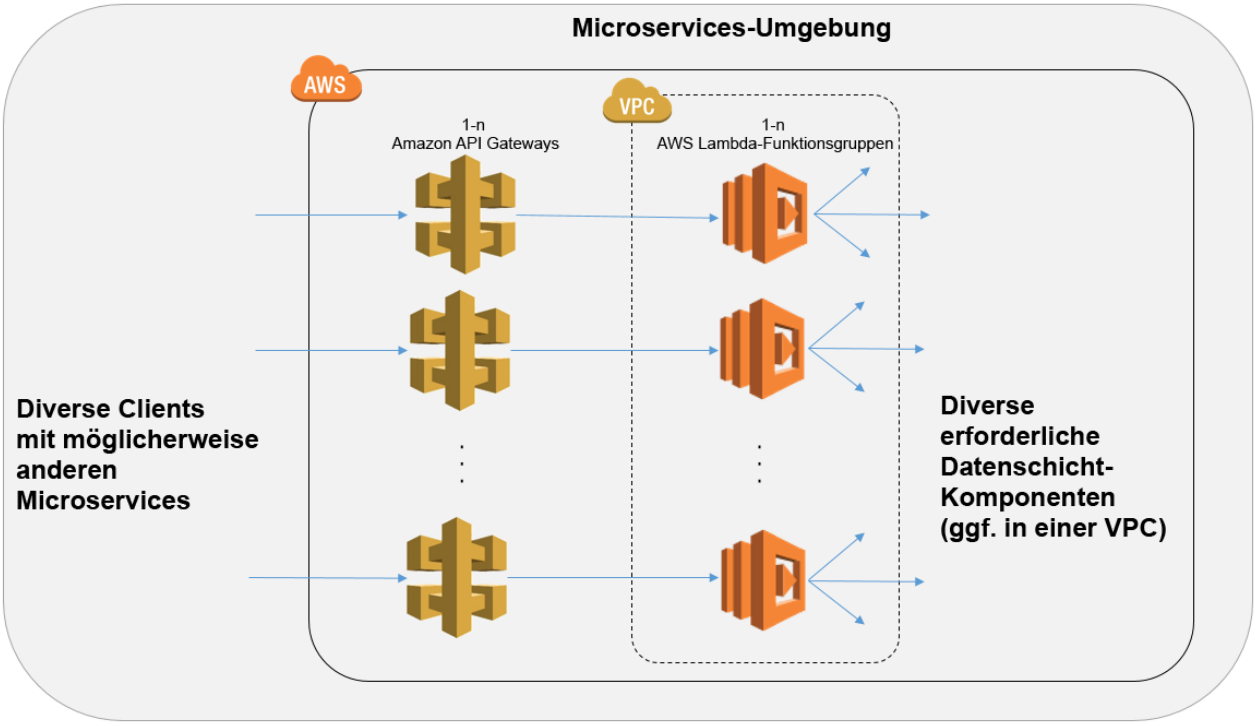

Abbildung 5: Architekturmuster für eine Microservices-Umgebung

Das **Microservices**-Architekturmuster nicht an die typische Drei-Schichten-Architektur gebunden, die wir in diesem Whitepaper behandelt haben. Eine Microservices-Architektur zeichnet sich durch eine starke Entkopplung der Software-Komponenten aus, sodass die Vorteile der Multi-Tier-Architektur durchgehend verstärkt werden. Um einen Microservice zu erstellen, benötigen Sie nur eine mit Amazon API Gateway erstellte API sowie Funktionen, die anschließend von AWS Lambda ausgeführt werden. Ihr Team kann dieses Services verwenden, um Ihre Umgebung so weit wie gewünscht zu entkoppeln und zu fragmentieren.

Im Allgemeinen kann eine Microservices-Umgebung folgende Probleme verursachen: Wiederholter Overhead bei der Erstellung jedes neuen Microservice, Schwierigkeiten bei der Optimierung der Serverdichte und Servernutzung, Komplexität durch gleichzeitige Ausführung mehrerer Versionen mehrerer Microservices, Zunahme von clientseitigem Bedarf an Code zur Integration vieler einzelner Services.

Wenn Sie Microservices jedoch mit dem serverlosen AWS-Muster erstellen, sind diese Probleme einfacher zu lösen oder treten in einigen Fällen überhaupt nicht auf. Das Microservices-Muster von AWS senkt die Barriere für die Erstellung jedes nachfolgenden Microservice (Amazon API Gateway ermöglicht sogar das Klonen bestehender APIs). Eine Optimierung der Serverauslastung ist mit diesem Muster nicht mehr relevant. Sowohl API Gateway als auch Lambda ermöglichen einfache Versioning-Funktionen. Und schließlich bietet Amazon API Gateway die programmgesteuerte Generierung von Client-SDKs in vielen populären Sprachen, um den Integrationsaufwand zu reduzieren.

Zusammenfassung

Die Verwendung des Multi-Tier-Architekturmusters ist eine bewährte Methode zur Erstellung von Anwendungskomponenten, die skalierbar, entkoppelt und leicht zu pflegen sind. Sie können diese Methode ohne großen Aufwand anwenden, indem Sie eine Logikschicht erstellen, in der die Integration über Amazon API Gateway und die Datenverarbeitung in AWS Lambda erfolgt. Zusammen bieten diese Services ein HTTPS-API-Frontend für Ihre Clients und innerhalb Ihrer VPC eine sichere Umgebung zur Ausführung von Geschäftslogik. Indem Sie diese verwalteten Services verwenden, statt eine typische serverbasierte Infrastruktur selbst zu verwalten, können Sie die Vorteile vieler beliebter Szenarien nutzen.

Mitwirkende

Dieses Dokument ist unter der Mitarbeit folgender Personen und Organisationen entstanden:

Andrew Baird, AWS-Lösungsarchitekt

Stefano Buliani, Senior Product Manager, Tech, AWS Mobile

Vyom Nagrani, Senior Product Manager, AWS Mobile

Ajay Nair, Senior Product Manager, AWS Mobile

Anmerkungen

[1] http://aws.amazon.com/api-gateway/

[2] http://aws.amazon.com/lambda/

[3] https://aws.amazon.com/vpc/

[4] https://aws.amazon.com/cloudfront/

[5] https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf

[6] https://aws.amazon.com/api-gateway/pricing/

[7] http://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-mock-integration.html

[8] http://aws.amazon.com/iam/

[9] http://docs.aws.amazon.com/general/latest/gr/signature-version-4.html

[10] http://docs.aws.amazon.com/lambda/latest/dg/intro-core-components.html#intro-core-components-event-sources

[11] https://aws.amazon.com/s3/

[12] https://aws.amazon.com/kinesis/

[13] https://aws.amazon.com/vpc/

[14] https://aws.amazon.com/rds/

[15] https://aws.amazon.com/elasticache/

[16] https://aws.amazon.com/redshift/

[17] https://aws.amazon.com/ec2/

[18] https://aws.amazon.com/dynamodb/

[19] https://aws.amazon.com/s3/storage-classes/

[20] https://aws.amazon.com/elasticsearch-service/

[21] https://aws.amazon.com/cognito/