Whitepaper zu AWS

AWS Serverless-Mehrschichtenarchitekturen mit Amazon API Gateway und AWS Lambda

AWS Serverless-Mehrschichtenarchitekturen mit Amazon API Gateway und AWS Lambda : Whitepaper zu AWS

Table of Contents

AWS Serverless-Mehrschichtenarchitekturen mit Amazon API Gateway und AWS Lambda

Veröffentlichungsdatum: 20. Oktober 2021 (Dokumentversionen)

Überblick

Dieses Whitepaper illustriert, wie Innovationen von Amazon Web Services (AWS) genutzt werden können, um die Art und Weise Ihrer Entwicklung von mehrschichtigen Architekturen zu ändern und beliebte Muster wie Microservices, mobile Backends und Einzelseitenanwendungen zu implementieren. Architekten und Entwickler können Amazon API Gateway, AWS Lambda und andere Services nutzen, um die erforderlichen Entwicklungs- und Produktionszyklen für die Erstellung und Verwaltung mehrschichtiger Anwendungen zu reduzieren.

Einführung

Die mehrschichtige Anwendung (dreischichtig, n-schichtig usw.) ist seit Jahrzehnten ein Eckpfeiler der Architektur und ist nach wie vor ein beliebtes Muster für benutzerorientierte Anwendungen. Obwohl mehrschichtige Architekturen mit unterschiedlichen Begriffen beschrieben werden, besteht eine mehrschichtige Anwendung im Allgemeinen aus den folgenden Komponenten:

- Präsentationsschicht: Komponente, mit der Benutzer direkt interagieren (z. B. Webseiten und Benutzeroberflächen mobiler Apps).
- Logikschicht: Code, der erforderlich ist, um Benutzeraktionen in Anwendungsfunktionen zu übersetzen (z. B. CRUD-Datenbankoperationen und Datenverarbeitung).
- Datenschicht: Speichermedien (z. B. Datenbanken, Objektspeicher, Caches und Dateisysteme), die die für die Anwendung relevanten Daten enthalten.

Das mehrschichtige Architekturmuster bietet ein allgemeines Framework, um sicherzustellen, dass entkoppelte und unabhängig skalierbare Anwendungskomponenten separat entwickelt, verwaltet und gewartet werden können (häufig von unterschiedlichen Teams).

Aufgrund dieses Musters, in dem das Netzwerk (eine Schicht interagiert über das Netzwerk mit einer anderen Schicht) als Grenze zwischen den Schichten fungiert, erfordert die Entwicklung einer mehrschichtigen Anwendung häufig die Erstellung vieler undifferenzierter Anwendungskomponenten. Einige dieser Komponenten beinhalten:

- Code, der eine Nachrichtenwarteschlange für die Kommunikation zwischen Schichten definiert
- Code, der eine Programmierschnittstelle (API) und ein Datenmodell definiert
- Sicherheitsbezogener Code, der einen angemessenen Zugriff auf die Anwendung gewährleistet

Alle diese Beispiele können als Standardkomponenten betrachtet werden, die in mehrschichtigen Anwendungen zwar erforderlich sind, sich in ihrer Implementierung von einer Anwendung zur anderen jedoch nicht stark unterscheiden.

AWS bietet eine Reihe von Services an, die die Erstellung von mehrschichtigen Serverless-Anwendungen ermöglichen. Dadurch wird die Bereitstellung solcher Anwendungen für die Produktion erheblich vereinfacht und der mit der herkömmlichen Serververwaltung verbundene Aufwand entfällt. Amazon API Gateway, ein Dienst zum Erstellen und Verwalten von APIs, und AWS Lambda, ein

Dienst zum Ausführen von Funktionen für beliebigen Code, können zusammen verwendet werden, um die Erstellung robuster mehrschichtiger Anwendungen zu vereinfachen.

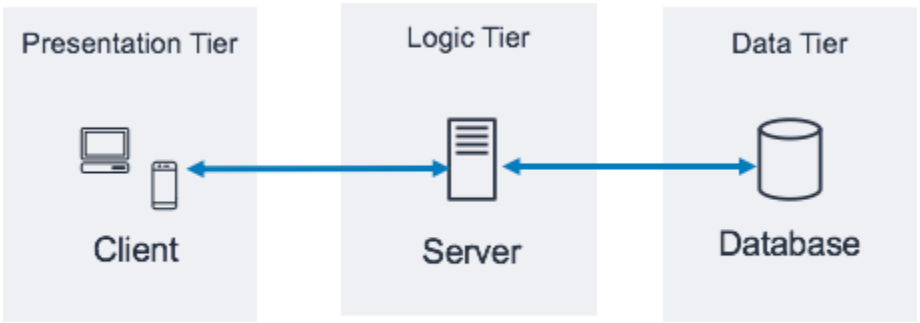
Durch die Integration von Amazon API Gateway mit AWS Lambda können benutzerdefinierte Codefunktionen direkt über HTTPS-Anforderungen initiiert werden. Unabhängig vom Anforderungsvolumen skalieren sowohl API Gateway als auch Lambda automatisch, um genau die Anforderungen Ihrer Anwendung zu unterstützen (siehe API Gateway – Amazon API Gateway-Kontingente und wichtige Hinweise für Informationen zur Skalierbarkeit). Durch die Kombination dieser beiden Dienste können Sie eine Schicht erstellen, die es Ihnen ermöglicht, sich ganz auf das Schreiben des Codes zu konzentrieren, der für Ihre Anwendung wichtig ist, anstatt auf verschiedene andere, nicht differenzierende Aspekte der Implementierung einer mehrschichtigen Architektur, z. B. Architektur für hohe Verfügbarkeit, Schreiben von Client-SDKs, Server- und Betriebssystemverwaltung, Skalierung und Implementierung eines Clientautorisierungsmechanismus.

API Gateway und Lambda ermöglichen die Erstellung einer Serverless-Logikschicht. Abhängig von Ihren Anwendungsanforderungen bietet AWS auch Optionen zum Erstellen einer Serverless-Präsentationsschicht (z. B. mit Amazon CloudFront und Amazon Simple Storage Service) und einer Serverless-Datenschicht (z. B. Amazon Aurora, Amazon DynamoDB).

Dieses Whitepaper konzentriert sich auf das am weitesten verbreitete Beispiel einer mehrschichtigen Architektur, die dreischichtige Webanwendung. Sie können dieses mehrschichtige Muster jedoch weit über eine typische dreischichtige Webanwendung hinaus anwenden.

Übersicht über die Drei-Schichten-Architektur

Die Drei-Schichten-Architektur ist die beliebteste Implementierung einer mehrschichtigen Architektur und besteht aus einer einzigen Präsentationsschicht, einer Logikschicht und einer Datenschicht. Die folgende Abbildung zeigt ein Beispiel für eine einfache, generische dreischichtige Anwendung.



Architekturmuster einer dreischichtigen Anwendung

Es gibt viele hilfreiche Online-Ressourcen, in denen Sie mehr über das allgemeine Drei-Schichten-Architekturmuster erfahren können. Dieses Whitepaper konzentriert sich auf ein bestimmtes Implementierungsmuster für diese Architektur unter Verwendung von Amazon API Gateway und AWS Lambda.

Serverless-Logikschicht

Die Logikschicht der Drei-Schichten-Architektur stellt das Gehirn der Anwendung dar. Hier kommt Amazon API Gateway zum Einsatz und AWS Lambda kann im Vergleich zu einer herkömmlichen, serverbasierten Implementierung die größte Wirkung haben. Mit den Funktionen dieser beiden Services können Sie eine Serverless-Anwendung entwickeln, die hochverfügbar, skalierbar und sicher ist. In einem herkömmlichen Modell könnte Ihre Anwendung Tausende von Servern benötigen. Wenn Sie jedoch Amazon API Gateway und AWS Lambda verwenden, sind Sie nicht für die Serververwaltung verantwortlich, ganz gleich, wie groß Ihre Anwendung ist. Wenn Sie diese verwalteten Services zusammen verwenden, profitieren Sie außerdem von den folgenden Vorteilen:

- AWS Lambda:
  - Sie müssen kein Betriebssystem auswählen, sichern, patchen oder verwalten
  - Sie müssen keine Server mit der richtigen Größe anschaffen, überwachen oder skalieren
  - Das Risiko, dass Sie wegen übermäßiger Bereitstellung zu hohe Kosten haben, ist geringer
  - Das Risiko einer schlechteren Leistung wegen unzureichender Bereitstellung ist geringer

- Amazon API Gateway:
  - Vereinfachte Mechanismen zur Bereitstellung, Überwachung und Sicherung von APIs
  - Verbesserte API-Leistung durch Caching und Bereitstellung von Inhalten

AWS Lambda

AWS Lambda ist ein Computing-Service, mit dem Sie beliebige Codefunktionen in jeder der unterstützten Sprachen (Node.js, Python, Ruby, Java, Go, .NET – weitere Informationen finden Sie in den häufig gestellten Fragen zu Lambda) ausführen können, ohne Server bereitstellen, verwalten oder skalieren zu müssen. Lambda-Funktionen werden in einem verwalteten, isolierten Container ausgeführt und als Reaktion auf ein Ereignis gestartet, das einer von mehreren programmatischen Auslösern sein kann, die AWS zur Verfügung stellt und die als Ereignisquelle bezeichnet werden. In den häufig gestellten Fragen zu Lambda sind alle Ereignisquellen aufgeführt.

Viele verbreitete Lambda-Anwendungsfälle stehen im Zusammenhang mit ereignisgesteuerten Datenverarbeitungs-Workflows, z. B. die Verarbeitung von in Amazon S3 gespeicherten Dateien oder das Streaming von Datensätzen von Amazon Kinesis. In Verbindung mit Amazon API Gateway erfüllt eine Lambda-Funktion die Funktionen eines typischen Webservice: Sie initiiert Code als Antwort auf

eine HTTPS-Anforderung des Clients; API-Gateway fungiert als Eingangstür zu Ihrer Logikschicht und AWS Lambda ruft den Anwendungscode auf.

Ihre Geschäftslogik wird hier implementiert, es sind keine Server erforderlich

Lambda erfordert das Schreiben von Codefunktionen, sogenannten Handlern, die ausgeführt werden, wenn sie von einem Ereignis initiiert werden. Um Lambda mit API Gateway zu verwenden, können Sie API Gateway so konfigurieren, dass Handler-Funktionen gestartet werden, wenn eine HTTPS-Anforderung an Ihre API erfolgt. In einer Serverless-Mehrschichtenarchitektur wird jede der APIs, die Sie in API Gateway erstellen, in eine Lambda-Funktion (und den darin enthaltenen Handler) integriert, die die erforderliche Geschäftslogik aufruft.

Durch die Verwendung von AWS Lambda-Funktionen zum Erstellen der Logikschicht können Sie die gewünschte Detailstufe für die Bereitstellung der Anwendungsfunktionalität definieren (eine Lambda-Funktion pro API oder eine Lambda-Funktion pro API-Methode). Innerhalb der Lambda-Funktion kann der Handler alle anderen Abhängigkeiten (z. B. andere Methoden, die Sie mit Ihrem Code hochgeladen haben, Bibliotheken, native Binärdateien und externe Webservices) oder sogar andere Lambda-Funktionen ansprechen.

Das Erstellen oder Aktualisieren einer Lambda-Funktion erfordert entweder das Hochladen von Code als Lambda-Bereitstellungspaket in einer ZIP-Datei an einen Amazon S3-Bucket oder das Verpacken von Code als Container-Image zusammen mit allen Abhängigkeiten. Die Funktionen können verschiedene Bereitstellungsmethoden nutzen, z. B. AWS-Managementkonsole, das Ausführen von AWS Command Line Interface (AWS CLI) oder das Ausführen von „Infrastructure as Code“-Vorlagen oder Frameworks wie AWS CloudFormation, AWS Serverless Application Model (AWS SAM) oder AWS Cloud Development Kit (AWS CDK). Wenn Sie Ihre Funktion mit einer dieser Methoden erstellen, geben Sie an, welche Methode in Ihrem Bereitstellungspaket als Anforderungs-Handler fungiert. Sie können dasselbe Bereitstellungspaket für mehrere Lambda-Funktionsdefinitionen wiederverwenden, wobei jede Lambda-Funktion möglicherweise über einen individuellen Handler innerhalb desselben Bereitstellungspakets verfügt.

Lambda-Sicherheit

Damit eine Lambda-Funktion ausgeführt werden kann, muss sie von einem Ereignis oder Service aufgerufen werden, das/der durch eine AWS Identity and Access Management (IAM)-Richtlinie zugelassen ist. Mithilfe von IAM-Richtlinien können Sie eine Lambda-Funktion erstellen, die überhaupt nicht initiiert werden kann, sofern sie nicht von einer von Ihnen definierten API-Gateway-

Ressource aufgerufen wird. Eine solche Richtlinie kann mithilfe ressourcenbasierter Richtlinien für verschiedene AWS-Services definiert werden.

Jede Lambda-Funktion nimmt eine IAM-Rolle an, die zugewiesen wird, wenn die Lambda-Funktion bereitgestellt wird. Diese IAM-Rolle definiert die anderen AWS-Services und -Ressourcen, mit denen Ihre Lambda-Funktion interagieren kann (z. B. Amazon DynamoDB Amazon S3). Im Zusammenhang mit der Lambda-Funktion wird dies als Ausführungsrolle bezeichnet.

Speichern Sie keine vertraulichen Informationen in einer Lambda-Funktion. IAM verarbeitet den Zugriff auf AWS-Services über die Lambda-Ausführungsrolle. Wenn Sie in Ihrer Lambda-Funktion auf andere Anmeldeinformationen (z. B. Datenbank-Anmeldeinformationen und API-Schlüssel) zugreifen müssen, können Sie AWS Key Management Service (AWS KMS) mit Umgebungsvariablen verwenden oder einen Dienst wie AWS Secrets Manager nutzen, um diese Informationen zu schützen, wenn sie nicht verwendet werden.

Skalierbare Leistung

Code, der als Container-Image aus der Amazon Elastic Container Registry (Amazon ECR) oder aus einer auf Amazon S3 hochgeladenen ZIP-Datei abgerufen wird, wird in einer isolierten Umgebung ausgeführt, die von AWS verwaltet wird. Sie müssen Ihre Lambda-Funktionen nicht skalieren – jedes Mal, wenn eine Ereignisbenachrichtigung von Ihrer Funktion empfangen wird, sucht AWS Lambda die verfügbare Kapazität innerhalb ihrer Computing-Flotte und führt Ihren Code mit von Ihnen definierten Laufzeit-, Speicher-, Festplatten- und Timeout-Konfigurationen aus. Mit diesem Muster kann AWS so viele Kopien Ihrer Funktion wie nötig starten.

Eine Lambda-basierte Logikschicht hat immer die richtige Größe für Ihre Kundenanforderungen. Dank der Fähigkeit, Datenverkehrsspitzen durch verwaltete Skalierung und gleichzeitige Codeinitiierung in Kombination mit den nutzungsabhängigen Preisen von Lambda schnell zu absorbieren, können Sie Kundenwünsche stets erfüllen und vermeiden gleichzeitig, für ungenutzte Rechenkapazität zu zahlen.

Serverless-Bereitstellung und -Verwaltung

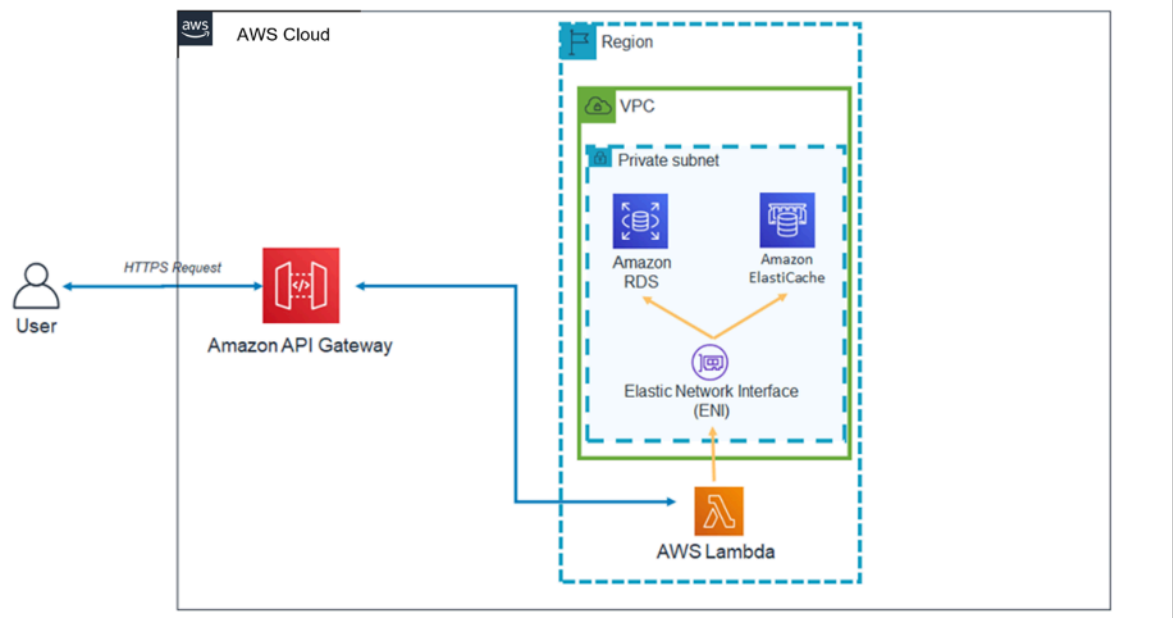
Verwenden Sie zur Unterstützung bei der Bereitstellung und Verwaltung Ihrer Lambda-Funktionen AWS Serverless Application Model (AWS SAM), ein Open-Source-Framework, das Folgendes umfasst:

- AWS SAM-Vorlagenspezifikation – Syntax zur Definition Ihrer Funktionen und zur Beschreibung ihrer Umgebungen, Berechtigungen, Konfigurationen und Ereignisse für ein vereinfachtes Hochladen und Bereitstellen.
- AWS SAM CLI – Befehle, mit denen Sie die SAM-Vorlagensyntax überprüfen, Funktionen lokal aufrufen, Lambda-Funktionen debuggen und Paketfunktionen bereitstellen können.

Sie können auch AWS CDK verwenden, ein Softwareentwicklungsframework zum Definieren der Cloud-Infrastruktur mithilfe von Programmiersprachen und ihrer Bereitstellung über CloudFormation. CDK bietet eine zwingende Methode zur Definition von AWS-Ressourcen, während AWS SAM eine deklarative Methode bietet.

Wenn Sie eine Lambda-Funktion bereitstellen, wird sie normalerweise mit Berechtigungen aufgerufen, die durch ihre zugewiesene IAM-Rolle definiert sind, und kann mit dem Internet verbundene Endpunkte erreichen. Als Kern Ihrer Logikschicht ist AWS Lambda die Komponente, die direkt mit der Datenschicht integriert wird. Wenn Ihre Datenschicht vertrauliche Geschäfts- oder Benutzerinformationen enthält, müssen Sie dafür sorgen, dass diese Datenschicht angemessen isoliert ist (in einem privaten Subnetz).

Sie können eine Lambda-Funktion für die Verbindung mit privaten Subnetzen in einer Virtual Private Cloud (VPC) in Ihrem AWS-Konto konfigurieren, wenn Sie möchten, dass die Lambda-Funktion auf Ressourcen zugreift, die Sie nicht öffentlich verfügbar machen können, z. B. eine private Datenbankinstanz. Wenn Sie eine Funktion mit einer VPC verbinden, erstellt Λ für jedes Subnetz in der VPC-Konfiguration Ihrer Funktion eine Elastic-Network-Schnittstelle, die verwendet wird, um privat auf Ihre internen Ressourcen zuzugreifen.



Lambda-Architekturmuster in einer VPC

Wenn Sie Lambda mit einer VPC verwenden, können Sie dafür sorgen, dass Datenbanken und andere Speichermedien, von denen Ihre Geschäftslogik abhängt, nicht über das Internet zugänglich sind. Die VPC stellt außerdem sicher, dass die einzige Möglichkeit, über das Internet mit Ihren Daten zu interagieren, die von Ihnen definierten APIs und die von Ihnen geschriebenen Lambda-Codefunktionen sind.

Amazon API Gateway

Amazon API Gateway ist ein vollständig verwalteter Service. Er ermöglicht Entwicklern das Erstellen, Veröffentlichen, Warten, Überwachen und Sichern von APIs in jeder Größenordnung.

Clients (d. h. Präsentationsschichten) lassen sich mithilfe von Standard-HTTPS-Anforderungen in die APIs integrieren, die über API Gateway bereitgestellt werden. Die Anwendbarkeit von APIs, die über API Gateway für eine serviceorientierte mehrschichtige Architektur bereitgestellt werden, ist die Fähigkeit, einzelne Teile der Anwendungsfunktionalität zu trennen und diese Funktionalität über REST-Endpunkte verfügbar zu machen. Amazon API Gateway verfügt über spezifische Funktionen und Qualitäten, die Ihrer Logikschicht leistungsstarke Fähigkeiten hinzufügen können.

Integration mit AWS Lambda

Amazon API Gateway unterstützt sowohl APIs des REST- als auch des HTTP-Typs. Eine API Gateway-API setzt sich aus Ressourcen und Methoden zusammen. Eine Ressource ist eine logische

Entität, auf die eine App über einen Ressourcenpfad (z. B. `/tickets`) zugreifen kann. Eine Methode entspricht einer API-Anforderung, die an eine API-Ressource gesendet wird (z. B. `GET /tickets`). Mit API Gateway können Sie jede Methode mit einer Lambda-Funktion sichern. Wenn Sie also die API über den in API Gateway bereitgestellten HTTPS-Endpunkt aufrufen, ruft API Gateway die Lambda-Funktion auf.

Sie können API Gateway und Lambda-Funktionen mithilfe von Proxy-Integrationen und Nicht-Proxy-Integrationen verbinden.

Proxy-Integrationen

Bei einer Proxy-Integration wird die gesamte Client-HTTPS-Anforderung unverändert an die Lambda-Funktion gesendet. API Gateway übergibt die gesamte Clientanforderung als Ereignisparameter der Lambda-Handler-Funktion, und die Ausgabe der Lambda-Funktion wird direkt an den Client zurückgegeben (einschließlich Statuscode, Header usw.).

Nicht-Proxy-Integrationen

In einer Nicht-Proxy-Integration konfigurieren Sie, wie die Parameter, Header und der Hauptteil der Clientanforderung an den Ereignisparameter der Lambda-Handler-Funktion übergeben werden. Außerdem konfigurieren Sie, wie die Lambda-Ausgabe zurück an den Benutzer übertragen wird.

> **Note**
>
> API Gateway kann auch als Proxy zu zusätzlichen Serverless-Ressourcen außerhalb von AWS Lambda fungieren, z. B. Pseudointegrationen (nützlich für die anfängliche Anwendungsentwicklung), und direkt zu S3-Objekten.

Stabile API-Leistung über Regionen hinweg

Jede Bereitstellung von Amazon API Gateway umfasst eine Amazon CloudFront-Verteilung. CloudFront ist ein Service zur Bereitstellung von Inhalten, der Amazons globales Netzwerk von Edge-Standorten als Verbindungspunkte für Clients nutzt, die Ihre API verwenden. Dies trägt dazu bei, die Reaktionslatenz Ihrer API zu verringern. Durch die Verwendung mehrerer Edge-Standorte auf der ganzen Welt bietet Amazon CloudFront auch Funktionen zur Bekämpfung von DDoS-Angriffsszenarien (Distributed Denial of Service). Weitere Informationen finden Sie im Whitepaper Bewährte AWS-Methoden für DDoS-Ausfallsicherheit.

Sie können die Leistung bestimmter API-Anforderungen verbessern, indem Sie API Gateway verwenden, um Antworten in einem optionalen In-Memory-Cache zu speichern. Dieser Ansatz bietet nicht nur Leistungsvorteile für wiederholte API-Anforderungen, sondern reduziert auch die Anzahl der Aufrufe Ihrer Lambda-Funktionen, was Ihre Gesamtkosten senken kann.

Mit integrierten Funktionen Innovationen fördern und Aufwand verringern

Die Entwicklungskosten einer neuen Anwendung sind eine Investition. Die Verwendung von API Gateway kann den Zeitaufwand für bestimmte Entwicklungsaufgaben reduzieren und die Gesamtentwicklungskosten senken, sodass Organisationen freier experimentieren und Innovationen entwickeln können.

In den ersten Phasen der Anwendungsentwicklung werden die Implementierung der Protokollierung und das Erfassen von Metriken häufig vernachlässigt, um die neue Anwendung schneller bereitzustellen. Dies kann zu technischen Nachlässigkeiten und Betriebsrisiken führen, wenn diese Funktionen in einer in Produktion laufenden Anwendung bereitgestellt werden. Amazon API Gateway lässt sich nahtlos in Amazon CloudWatch integrieren, das Rohdaten von API Gateway zu lesbaren, nahezu in Echtzeit verfügbaren Metriken zur Überwachung der API-Ausführung sammelt und verarbeitet. API Gateway unterstützt auch die Zugriffsprotokollierung mit konfigurierbaren Berichten sowie die AWS X-Ray-Ablaufverfolgung für das Debuggen. Für keine dieser Funktionen muss Code geschrieben werden, und sie können in Anwendungen angepasst werden, die in einer Produktionsumgebung laufen – ohne Risiko für die Kerngeschäftslogik.

Die Gesamtlebensdauer einer Anwendung ist möglicherweise unbekannt oder bekanntermaßen kurz. Das Erstellen eines Business Case für die Entwicklung solcher Anwendungen kann vereinfacht werden, wenn Ihr Startpunkt bereits die verwalteten Funktionen enthält, die API Gateway bereitstellt, und wenn Infrastrukturkosten erst anfallen, nachdem Ihre APIs Anfragen erhalten. Weitere Informationen finden Sie unter Amazon API Gateway – Preise.

Schnell iterieren, agil bleiben

Durch die Verwendung von Amazon API Gateway und AWS Lambda zum Aufbau der Logikschicht Ihrer API können Sie sich schnell an die sich ändernden Anforderungen Ihrer Benutzerbasis anpassen, da API-Bereitstellung und Versionsverwaltung vereinfacht werden.

Staging der Bereitstellung

Wenn Sie eine API in API Gateway bereitstellen, müssen Sie die Bereitstellung einer API-Gateway-Phase zuordnen – jede Phase ist ein Snapshot der API und wird Client-Apps zum Aufrufen

zur Verfügung gestellt. Mit dieser Konvention können Sie Apps einfach für die Entwicklungs-, Test-, Staging- oder Produktionsphase bereitstellen und Bereitstellungen zwischen den Phasen verschieben. Jedes Mal, wenn Sie Ihre API für eine Phase bereitstellen, erstellen Sie eine andere Version der API, die bei Bedarf rückgängig gemacht werden kann. Dadurch können vorhandene Funktionen und Clientabhängigkeiten ungestört weitergeführt werden, während neue Funktionen als separate API-Version veröffentlicht werden.

Entkoppelte Integration mit Lambda

Die Integration zwischen der API in API Gateway und der Lambda-Funktion kann mithilfe von API-Gateway-Phasenvariablen und einem Lambda-Funktionsalias entkoppelt werden. Dies vereinfacht und beschleunigt die API-Bereitstellung. Anstatt den Namen oder Alias der Lambda-Funktion direkt in der API zu konfigurieren, können Sie die Phasenvariable in der API konfigurieren, die auf einen bestimmten Alias in der Lambda-Funktion verweisen kann. Ändern Sie während der Bereitstellung den Wert der Phasenvariable, sodass er auf einen Lambda-Funktionsalias verweist, und die API führt die Lambda-Funktionsversion hinter dem Lambda-Alias für eine bestimmte Phase aus.

Canary-Release-Bereitstellung

Canary-Release ist eine Softwareentwicklungsstrategie, bei der eine neue Version einer API zu Testzwecken eingesetzt wird und die Basisversion weiterhin als Produktionsrelease für den normalen Betrieb in derselben Phase bereitgestellt wird. Bei einer Canary-Release-Bereitstellung wird der gesamte API-Datenverkehr mit einem vorkonfigurierten Verhältnis nach dem Zufallsprinzip in ein Produktionsrelease und ein Canary-Release getrennt. APIs in API Gateway können für die Canary-Release-Bereitstellung konfiguriert werden, um neue Funktionen mit einer begrenzten Anzahl von Benutzern zu testen.

Benutzerdefinierte Domänennamen

Sie können anstelle der von API Gateway bereitgestellten URL einen intuitiven unternehmensfreundlichen URL-Namen zur API angeben. API Gateway bietet Funktionen zum Konfigurieren einer benutzerdefinierten Domäne für die APIs. Mit benutzerdefinierten Domänennamen können Sie den Hostnamen Ihrer API einrichten und einen mehrstufigen Basispfad (z. B. `myservice`, `myservice/cat/v1` oder `myservice/dog/v2`) auswählen, um die alternative URL Ihrer API zuzuordnen.

API-Sicherheit priorisieren

Alle Anwendungen müssen sicherstellen, dass nur autorisierte Clients Zugriff auf ihre API-Ressourcen haben. Beim Entwerfen einer mehrschichtigen Anwendung können Sie verschiedene Möglichkeiten nutzen, mit denen Amazon API Gateway zur Sicherung Ihrer Logikschicht beiträgt:

Übertragungssicherheit

Alle Anfragen an Ihre APIs können über HTTPS gestellt werden, um die Verschlüsselung während der Übertragung zu ermöglichen.

API Gateway bietet integrierte SSL-/TLS-Zertifikate. Wenn Sie die Option für benutzerdefinierte Domänennamen für öffentlich zugängliche APIs verwenden, können Sie mit AWS Certificate Manager Ihr eigenes SSL-/TLS-Zertifikat bereitstellen. API Gateway unterstützt auch die gegenseitige TLS-Authentifizierung (mTLS). Gegenseitige TLS erhöht die Sicherheit Ihrer API und schützt Ihre Daten vor Angriffen wie Client-Spoofing oder Man-in-the-Middle-Angriffen.

API-Autorisierung

Jeder Ressourcen-/Methodenkombination, die Sie als Teil Ihrer API erstellen, wird ein individueller Amazon-Ressourcenname (ARN) zugewiesen, auf den in AWS Identity and Access Management (IAM)-Richtlinien verwiesen werden kann.

Es gibt drei allgemeine Methoden, um einer API in API Gateway eine Autorisierung hinzuzufügen:

- IAM-Rollen und -Richtlinien: Clients verwenden die Autorisierung mit AWS Signature Version 4 (SigV4) sowie IAM-Richtlinien für den API-Zugriff. Dieselben Anmeldeinformationen können den Zugriff auf andere AWS-Services und -Ressourcen nach Bedarf einschränken oder zulassen (z. B. Amazon S3-Buckets oder Amazon DynamoDB-Tabellen).
- Amazon Cognito-Benutzerpools: Clients melden sich über einen Amazon Cognito-Benutzerpool an und erhalten Token, die im Autorisierungs-Header einer Anforderung enthalten sind.
- Lambda-Genehmiger: Definieren Sie eine Lambda-Funktion, die ein benutzerdefiniertes Autorisierungsschema implementiert, das eine Träger-Token-Strategie (z. B. OAuth und SAML) oder Anforderungsparameter verwendet, um Benutzer zu identifizieren.

Zugriffsbeschränkungen

API Gateway unterstützt die Generierung von API-Schlüsseln und die Verknüpfung dieser Schlüssel mit einem konfigurierbaren Nutzungsplan. Sie können die Verwendung von API-Schlüsseln mit CloudWatch überwachen.

API Gateway unterstützt Drosselung, Ratenbegrenzungen und Burst-Ratenbegrenzungen für jede Methode in Ihrer API.

Private APIs

Mit API Gateway können Sie private REST-APIs erstellen, auf die nur von Ihrer Virtual Private Cloud in Amazon VPC aus zugegriffen werden kann, indem Sie einen Schnittstellen-VPC-Endpunkt verwenden. Dies ist eine Endpunkt-Netzwerkschnittstelle, die Sie in Ihrer VPC erstellen.

Mithilfe von Ressourcen-Richtlinien können Sie den Zugriff auf Ihre API von ausgewählten VPCs und VPC-Endpunkten aus ermöglichen oder verweigern, darunter auch über AWS-Konten. Jeder Endpunkt kann für den Zugriff auf mehrere APIs verwendet werden. Sie können außerdem AWS Direct Connect verwenden, um eine Verbindung von einem On-Premise-Netzwerk zu Amazon VPC herzustellen und über diese Verbindung auf Ihre private API zuzugreifen.

In allen Fällen verwendet der Datenverkehr zu Ihrer privaten API eine sichere Verbindung und verlässt nicht das Amazon-Netzwerk. Er ist vom öffentlichen Internet isoliert.

Firewall-Schutz mit AWS WAF

APIs, auf die über das Internet zugegriffen wird, sind anfällig für böswillige Angriffe. AWS WAF ist eine Webanwendungs-Firewall, die APIs vor solchen Angriffen schützt. Es schützt APIs vor gängigen Web-Exploits wie SQL-Injection- und Cross-Site-Scripting-Angriffen. Sie können AWS WAF mit API Gateway verwenden, um APIs zu schützen.

Datenschicht

Durch die Verwendung von AWS Lambda als Logikschicht werden die in Ihrer Datenschicht verfügbaren Datenspeicheroptionen nicht eingeschränkt. Lambda-Funktionen stellen eine Verbindung zu jeder Datenspeicheroption her, indem sie den entsprechenden Datenbanktreiber in das Lambda-Bereitstellungspaket aufnehmen, und verwenden den IAM-rollenbasierten Zugriff oder verschlüsselte Anmeldeinformationen (über AWS KMS oder AWS Secrets Manager).

Die Auswahl eines Datenspeichers für Ihre Anwendung hängt stark von Ihren Anwendungsanforderungen ab. AWS bietet eine Reihe von Serverless- und Nicht-Serverless-Datenspeichern, aus denen Sie die Datenschicht Ihrer Anwendung zusammenstellen können.

Serverless-Datenspeicheroptionen

Amazon S3 ist ein Objektspeicherdienst, der branchenführende Skalierbarkeit, Datenverfügbarkeit, Sicherheit und Leistung bietet.

Amazon Aurora ist eine mit MySQL und PostgreSQL kompatible relationale Datenbank für die Cloud, die die Leistung und Verfügbarkeit herkömmlicher Unternehmensdatenbanken mit der Einfachheit und Kosteneffizienz von Open-Source-Datenbanken kombiniert. Aurora bietet sowohl Serverless- als auch herkömmliche Nutzungsmodelle.

Amazon DynamoDB ist eine Schlüssel-Werte- und Dokumentdatenbank, die für beliebig große Datenmengen eine Leistung im einstelligen Millisekundenbereich bereitstellt. Es handelt sich um eine vollständig verwaltete, regionsübergreifende, beständige Serverless-Multi-Master-Datenbank mit integrierter Sicherheit, Backup und Wiederherstellung sowie In-Memory-Caching für internetfähige Anwendungen.

Amazon Timestream ist ein schneller, skalierbarer und vollständig verwalteter Zeitreihendatenbank-Service für IoT und Betriebsanwendungen, der die mühelose Speicherung und Analyse von Billionen von Ereignissen pro Tag ermöglicht – zu einem Zehntel der Kosten von relationalen Datenbanken. Durch den Siegeszug von IoT-Geräten, IT-Systemen und intelligenten Industriemaschinen gehören Zeitreihendaten – Daten, die messen, wie sich die Dinge im Laufe der Zeit ändern, – zu den am schnellsten wachsenden Datentypen.

Amazon Quantum Ledger Database (Amazon QLDB) ist eine vollständig verwaltete Ledger-Datenbank, die ein transparentes, unveränderliches und kryptografisch überprüfbares Transaktionsprotokoll im Besitz eines zentralen vertrauenswürdigen Ausstellers bereitstellt. Amazon

QLDB verfolgt und überprüft jede einzelne Änderung von Anwendungsdaten und pflegt die ganze Zeit über eine vollständige und überprüfbare Änderungshistorie.

Amazon Keyspaces (für Apache Cassandra) ist ein skalierbarer, hochverfügbarer und verwalteter Apache Cassandra-kompatibler Datenbankservice. Mit Amazon Keyspaces können Sie Ihre Cassandra-Workloads in AWS mit demselben Anwendungscode und denselben Entwicklertools ausführen, die Sie heute verwenden. Sie müssen keine Server bereitstellen, patchen oder verwalten und sich nicht um die Installation, Fehlerbehebung oder Ausführung von Software kümmern. Amazon Keyspaces ist eine Serverless-Lösung, sodass Sie nur für die tatsächlich genutzten Ressourcen bezahlen. Tabellen können automatisch auf- und abwärts skaliert werden, abhängig vom Anwendungsdatenverkehr.

Das Amazon Elastic File System (Amazon EFS) bietet ein einfaches, elastisches Serverless-Set-and-Forget-Dateisystem, mit dem Sie Dateidaten gemeinsam nutzen können, ohne Speicher bereitzustellen oder zu verwalten. Es kann mit AWS-Cloud-Services und On-Premises-Ressourcen verwendet werden und ist so konzipiert, dass es bei Bedarf auf Petabyte skaliert werden kann, ohne die Anwendungen zu stören. Mit Amazon EFS können Sie Ihre Dateisysteme beim Hinzufügen und Entfernen von Dateien automatisch vergrößern und verkleinern, sodass Sie keine Kapazitäten mehr bereitstellen und verwalten müssen, um dem Wachstum Rechnung zu tragen. Amazon EFS kann mit Lambda-Funktionen bereitgestellt werden, was es zu einer praktikablen Dateispeicheroption für APIs macht.

Nicht-Serverless-Datenspeicheroptionen

Amazon Relational Database Service (Amazon RDS) ist ein verwalteter Webservice, der die Einrichtung, den Betrieb und die Skalierung einer relationalen Datenbank mithilfe einer der verfügbaren Engines (Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle und Microsoft SQL Server) erleichtert und auf mehreren verschiedene Datenbank-Instance-Typen ausgeführt werden kann, die für Arbeitsspeicher, Leistung oder I/O optimiert sind.

Amazon Redshift ist ein vollständig verwalteter Petabyte-Data-Warehouse-Service in der Cloud.

Amazon ElastiCache ist eine vollständig verwaltete Bereitstellung von Redis oder Memcached. Beliebte Open Source-kompatible In-Memory-Datenspeicher können nahtlos bereitgestellt, betrieben und skaliert werden.

Amazon Neptune ist ein schneller, zuverlässiger, vollständig verwalteter Diagrammdatenbank-Service, mit dem es ganz einfach ist, Anwendungen zu entwickeln und auszuführen, die mit

hochgradig verbundenen Datensätzen arbeiten. Neptune unterstützt gängige Diagrammmodelle – Eigenschaftsdiagramme und W3C Resource Description Framework (RDF) – und deren jeweilige Abfragesprachen, sodass Sie auf einfache Weise Abfragen erstellen können, die effizient durch hochverknüpfte Datensätze navigieren.

Amazon DocumentDB (mit MongoDB-Kompatibilität) ist ein schneller, skalierbarer, hochverfügbarer und vollständig verwalteter Service für Dokumentdatenbanken, der MongoDB-Workloads unterstützt.

Außerdem können Sie auch Datenspeicher verwenden, die unabhängig auf Amazon EC2 als Datenschicht einer mehrschichtigen Anwendung ausgeführt werden.

Präsentationsschicht

Die Präsentationsschicht interagiert über API-Gateway-REST-Endpunkte, die über das Internet bereitgestellt werden, mit der Logikschicht. Alle HTTPS-fähigen Clients oder Geräte können mit diesen Endpunkten kommunizieren, sodass Ihre Präsentationsschicht ganz unterschiedliche Formen haben kann (Desktop-Anwendungen, mobile Apps, Webseiten, IoT-Geräte usw.). Abhängig von Ihren Anforderungen kann Ihre Präsentationsschicht die folgenden Serverless-AWS-Services verwenden: Alle HTTPS-fähigen Clients oder Geräte können mit diesen Endpunkten kommunizieren, sodass Ihre Präsentationsschicht ganz unterschiedliche Formen haben kann (Desktop-Anwendungen, mobile Apps, Webseiten, IoT-Geräte usw.). Abhängig von Ihren Anforderungen kann Ihre Präsentationsschicht die folgenden Serverless-AWS-Services verwenden:

- Amazon Cognito – Ein Serverless-Service für Benutzeridentität und Datensynchronisierung, mit dem Sie Ihren Web- und mobilen Apps schnell und effizient Benutzerregistrierung und -anmeldung sowie Zugriffskontrolle hinzufügen können. Amazon Cognito kann für Millionen von Benutzern skaliert werden und unterstützt die Anmeldung über soziale Identitätsanbieter wie Facebook, Google und Amazon wie auch via SAML 2.0 über Unternehmens-Identitätsanbieter.
- Amazon S3 mit CloudFront – Ermöglicht Ihnen, statische Websites, z. B. Einzelseitenanwendungen, direkt aus einem S3-Bucket zu bedienen, ohne dass ein Webserver bereitgestellt werden muss. Sie können CloudFront als verwaltetes Content Delivery Network (CDN) verwenden, um die Leistung zu verbessern und SSL/TL mithilfe verwalteter oder benutzerdefinierter Zertifikate zu aktivieren.

AWS Amplify ist eine Reihe von Tools und Diensten, die zusammen oder einzeln verwendet werden können, um Frontend-Web- und Mobilentwickler beim Aufbau skalierbarer Full-Stack-Anwendungen Powered by AWS zu unterstützen. Amplify bietet einen vollständig verwalteten Service für die Bereitstellung und das Hosting statischer Webanwendungen auf globaler Ebene, der über das zuverlässige CDN von Amazon mit Hunderten von Points of Presence auf der ganzen Welt bereitgestellt wird und über integrierte CI/CD-Workflows verfügt, die den Veröffentlichungszyklus Ihrer Anwendungen beschleunigen. Amplify unterstützt beliebte Web-Frameworks wie JavaScript, React, Angular, Vue, Next.js und Mobilgeräte-Plattformen wie Android, iOS, React Native, Ionic und Flutter. Abhängig von Ihren Netzwerkkonfigurationen und Anwendungsanforderungen müssen Sie möglicherweise dafür sorgen, dass Ihre API-Gateway-APIs mit Cross-Origin Resource Sharing (CORS) kompatibel sind. Dank der CORS-Compliance können Webbrowser Ihre APIs direkt von statischen Webseiten aus aufrufen.

Wenn Sie eine Website mit CloudFront bereitstellen, erhalten Sie einen CloudFront-Domänennamen, um Ihre Anwendung zu erreichen (z. B. `d2d47p2vcczkh2.cloudfront.net`). Sie können Amazon Route 53 verwenden, um Domänennamen zu registrieren und sie an Ihre CloudFront-Verteilung weiterzuleiten, oder Sie können bereits in Ihrem Besitz befindliche Domänennamen an Ihre CloudFront-Verteilung weiterleiten. Auf diese Weise können Benutzer mit einem vertrauten Domänennamen auf Ihre Website zugreifen. Beachten Sie, dass sich Ihrer API-Gateway-Verteilung mithilfe von Route 53 auch ein benutzerdefinierter Domänennamen zuweisen lässt, sodass Benutzer APIs mit vertrauten Domänennamen aufrufen können.

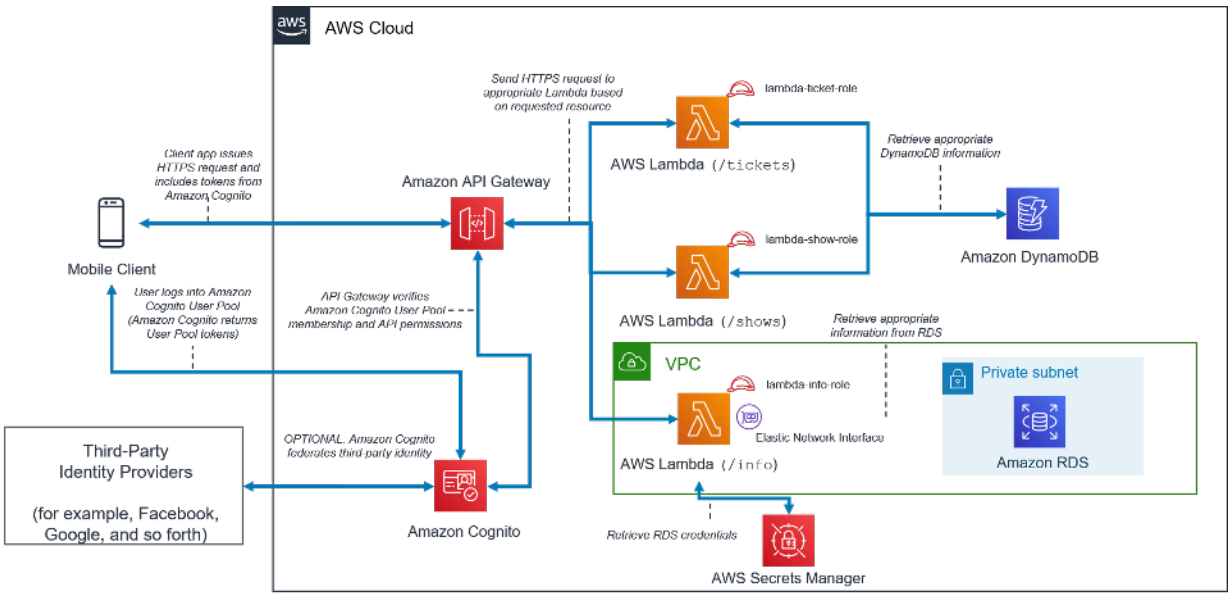
Beispielarchitekturmuster

Sie können gängige Architekturmuster mithilfe von API Gateway und AWS Lambda als Logikschicht implementieren. Dieses Whitepaper enthält die beliebtesten Architekturmuster, die AWS Lambda-basierte Logikschichten nutzen:

- Mobiles Backend – Eine mobile Anwendung kommuniziert mit API Gateway und Lambda, um auf Anwendungsdaten zuzugreifen. Dieses Muster kann auf generische HTTPS-Clients erweitert werden, die keine Serverless-AWS-Ressourcen zum Hosten von Ressourcen der Präsentationsschicht verwenden (z. B. Desktop-Clients, Webserver auf EC2 usw.).
- Einzelseitenanwendung – Eine in Amazon S3 und CloudFront gehostete Anwendung mit nur einer Seite kommuniziert mit API Gateway und AWS Lambda, um auf Anwendungsdaten zuzugreifen.
- Webanwendung – Die Webanwendung ist ein universelles, ereignisgesteuertes Webanwendungs-Backend, das AWS Lambda mit API Gateway für die Geschäftslogik verwendet. Außerdem werden Amazon DynamoDB als Datenbank und Amazon Cognito für die Benutzerverwaltung verwendet. Alle statischen Inhalte werden mit Amplify gehostet.

Zusätzlich zu diesen beiden Mustern wird in diesem Whitepaper die Anwendbarkeit von Lambda und API Gateway auf eine allgemeine Microservice-Architektur erörtert. Eine Microservice-Architektur ist ein beliebtes Muster, bei dem es sich zwar nicht um eine standardmäßige Drei-Schichten-Architektur handelt, aber Anwendungskomponenten entkoppelt und als zustandslose, einzelne Funktionseinheiten bereitgestellt werden, die miteinander kommunizieren.

Mobiles Backend



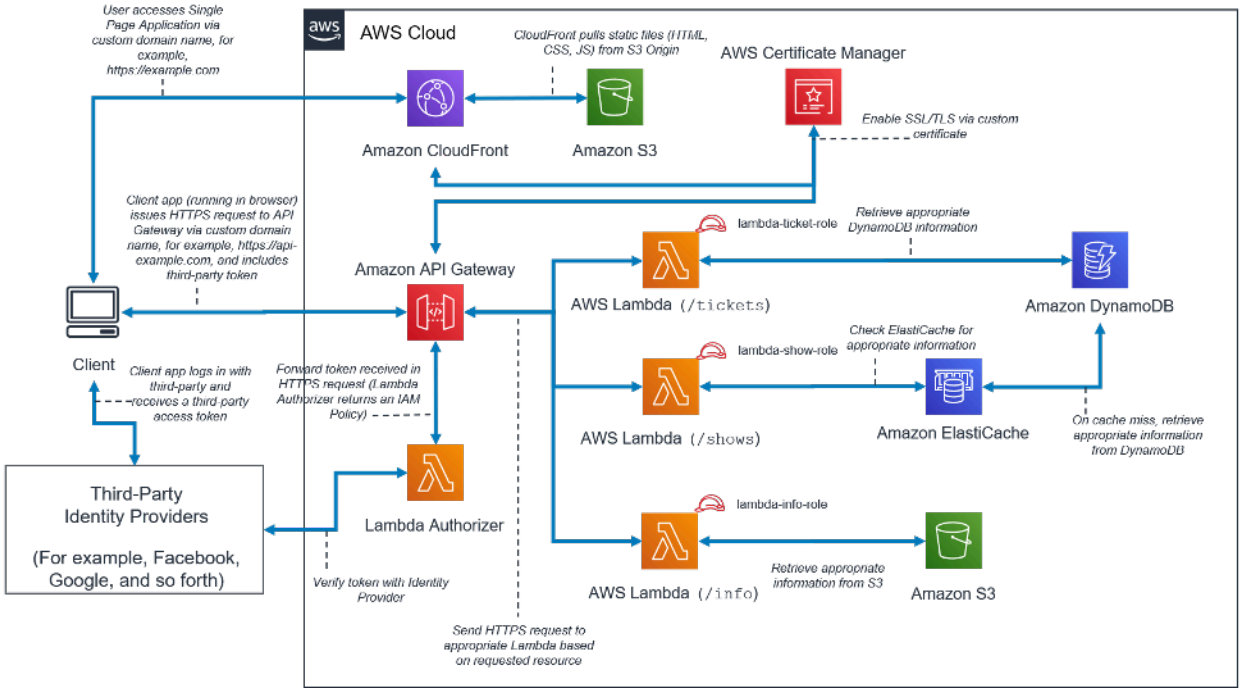
Architekturmuster für ein mobiles Serverless-Backend

Tabelle 1: Komponenten der mobilen Backend-Schicht

Schicht	Komponenten
Präsentation	Mobile Anwendung, die auf einem Benutzergerät ausgeführt wird.
Logik	Amazon API Gateway mit AWS Lambda. Diese Architektur zeigt drei exponierte Dienste (`/tickets`, `/shows` und `/info`). API-Gateway-Endpunkte werden durch Amazon Cognito-Benutzerpools geschützt. Bei dieser Methode melden sich Benutzer bei Amazon Cognito-Benutzerpools an (gegebenenfalls über einen verbundenen Drittanbieter) und erhalten Zugriffs- und ID-Token, die zur Autorisierung von API-Gateway-Aufrufen verwendet werden. Jeder Lambda-Funktion wird eine eigene Identity and Access Management (IAM)-Rolle

Schicht	Komponenten
	zugewiesen, um den Zugriff auf die entsprechende Datenquelle zu ermöglichen.
Daten	DynamoDB wird für die `/tickets`- und `/shows`-Services verwendet. Amazon RDS wird für den `/info`-Service verwendet. Diese Lambda-Funktion ruft Amazon RDS-Anmeldeinformationen von AWS Secrets Manager ab und verwendet eine Elastic-Network-Schnittstelle, um auf das private Subnetz zuzugreifen.

Einzelseitenanwendung

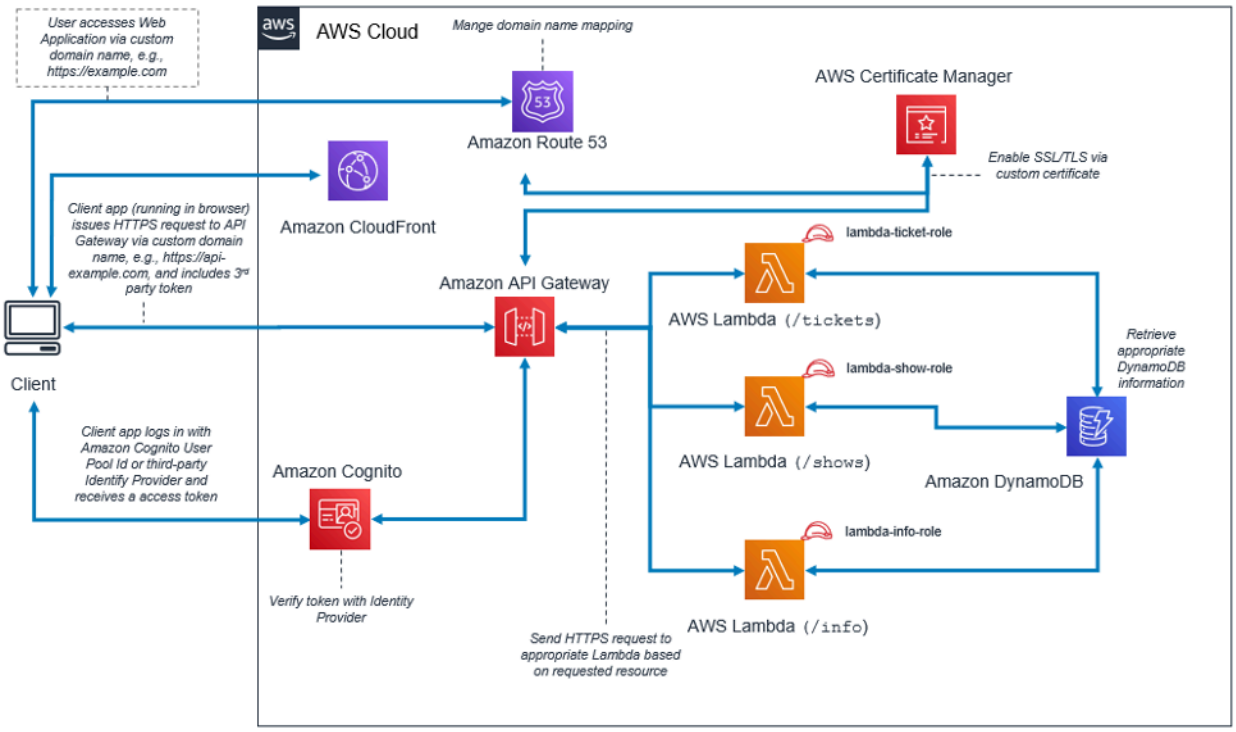


Architekturmuster einer Serverless-Einzelseitenanwendung

Tabelle 2: Komponenten einer Einzelseitenanwendung

Schicht	Komponenten
Präsentation	In Amazon S3 gehosteter statischer Website-Inhalt, der von CloudFront verteilt wird. AWS Certificate Manager ermöglicht die Verwendung eines benutzerdefinierten SSL-/TLS-Zertifikats.
Logik	API Gateway mit AWS Lambda. Diese Architektur zeigt drei exponierte Dienste (`/tickets`, `/shows` und `/info`). API-Gateway-Endpunkte werden durch einen Lambda-Genehmiger gesichert. Bei dieser Methode melden sich Benutzer über einen Identitätsanbieter (Drittanbieter) an und erhalten Zugriffs- und ID-Token. Diese Token werden in API-Gateway-Aufrufe eingeschlossen. Der Lambda-Genehmiger validiert diese Token und generiert eine IAM-Richtlinie mit API-Initierungsberechtigungen. Jeder Lambda-Funktion wird eine eigene IAM-Rolle zugewiesen, um den Zugriff auf die entsprechende Datenquelle zu ermöglichen.
Daten	Amazon DynamoDB wird für die `/tickets`- und `/shows`-Services verwendet. Amazon ElastiCache wird vom `/shows`-Service verwendet, um die Datenbankleistung zu verbessern. Cache-Fehler werden an DynamoDB gesendet. Amazon S3 wird verwendet, um statischen Inhalt zu hosten, der vom `/info service` verwendet wird.

Webanwendung



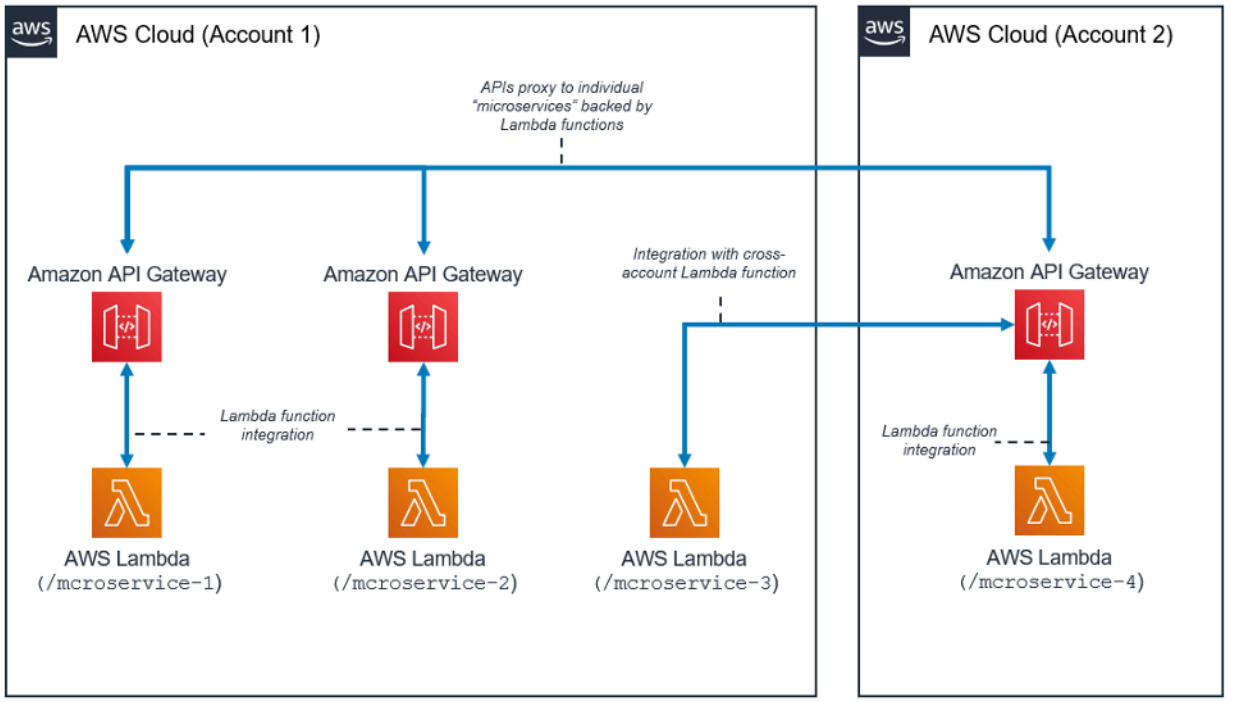
Architekturmuster einer Webanwendung

Tabelle 3: Komponenten einer Webanwendung

Schicht	Komponenten
Präsentation	Die Frontend-Anwendung besteht ausschließlich aus statischen Inhalten (HTML, CSS, JavaScript und Bilder), die von React-Diensprogrammen wie create-react-app generiert werden. Diese Objekte werden alle von Amazon CloudFront gehostet. Wenn die Webanwendung verwendet wird, lädt sie alle Ressourcen in den Browser herunter und wird von dort aus ausgeführt. Die Webanwendung stellt eine Verbindung zum Backend her, das die APIs aufruft.

Schicht	Komponenten
Logik	Die Logikschicht wird mit Lambda-Funktionen erstellt, die hinter API-Gateway-REST-APIs ausgeführt werden. Diese Architektur zeigt mehrere exponiert e Services. Es gibt mehrere verschiedene Lambda-Funktionen, die jeweils für einen anderen Aspekt der Anwendung zuständig sind. Die Lambda-Funktionen befinden sich hinter API Gateway und sind über API-URL-P fade zugänglich. Die Benutzerauthentifizierung erfolgt mithilfe von Amazon Cognito-Benutzerpools oder Verbundbenutzeranbietern. API Gateway nutzt die sofort einsatzbereite Integration mit Amazon Cognito. Erst nachdem ein Benutzer authentif iziert wurde, erhält der Client ein JSON-Web-Token (JWT), das er dann bei den API-Aufrufen verwenden sollte. Jeder Lambda-Funktion wird eine eigene IAM-Rolle zugewiesen, um den Zugriff auf die entsprechende Datenquelle zu ermöglichen.
Daten	In diesem speziellen Beispiel wird DynamoDB als Datenspeicher verwendet, aber je nach Anwendungsfall und Anwendungsszenario können auch andere zweckgebundene Amazon-Datenbank- oder -Speicherdienste verwendet werden.

Microservices mit Lambda



Architekturmuster für Microservices mit Lambda

Das Microservice-Architekturmuster ist nicht an die typische Drei-Schichten-Architektur gebunden. Dieses beliebte Muster kann jedoch erhebliche Vorteile durch die Verwendung von Serverless-Ressourcen erzielen.

In dieser Architektur wird jede der Anwendungskomponenten entkoppelt und unabhängig bereitgestellt und betrieben. Eine API, die mit Amazon API Gateway erstellt wurde, und Funktionen, die anschließend von AWS Lambda gestartet werden, sind alles, was Sie zum Entwickeln eines Microservice benötigen. Ihr Team kann diese Dienste nutzen, um Ihre Umgebung bis zur gewünschten Detailstufe zu entkoppeln und zu fragmentieren.

Im Allgemeinen kann eine Microservices-Umgebung die folgenden Schwierigkeiten mit sich bringen: wiederholter Aufwand für die Erstellung jedes neuen Microservice, Probleme bei der Optimierung der Serverdichte und -auslastung, Komplexität der gleichzeitigen Ausführung mehrerer Versionen verschiedener Microservices und vermehrte clientseitige Codeanforderungen für die Integration in viele separate Dienste.

Wenn Sie Microservices mit Serverless-Ressourcen erstellen, sind diese Probleme weniger schwierig zu lösen und in einigen Fällen einfach nicht mehr vorhanden. Das Architekturmuster mit Serverless-Microservices erleichtert die Erstellung jedes nachfolgenden Microservice (API Gateway ermöglicht

sogar das Klonen vorhandener APIs und die Verwendung von Lambda-Funktionen in anderen Konten). Die Optimierung der Serverauslastung ist bei diesem Muster nicht mehr relevant. Außerdem bietet Amazon API Gateway programmatisch generierte Client-SDKs in einer Reihe gängiger Sprachen, um den Integrationsaufwand zu reduzieren.

Fazit

Das mehrschichtige Architekturmuster unterstützt bewährte Methoden zum Erstellen von Anwendungskomponenten, die einfach zu warten, zu entkoppeln und zu skalieren sind. Wenn Sie eine Logikschicht erstellen, in der die Integration durch API Gateway und die Berechnung innerhalb von AWS Lambda erfolgt, erreichen Sie diese Ziele mit weniger Aufwand. Zusammen bieten diese Dienste ein HTTPS-API-Frontend für Ihre Kunden und eine sichere Umgebung, um Ihre Geschäftslogik anzuwenden und gleichzeitig die Verwaltungslast einer typischen serverbasierten Infrastruktur zu vermeiden.

Mitwirkende

An diesem Dokument haben folgende Personen mitgewirkt:

- Andrew Baird, AWS Solutions Architect
- Bryant Bost, AWS ProServe Consultant
- Stefano Buliani, Senior Product Manager, Tech, AWS Mobile
- Vyom Nagrani, Senior Product Manager, AWS Mobile
- Ajay Nair, Senior Product Manager, AWS Mobile
- Rahul Popat, Global Solutions Architect
- Brajendra Singh, Senior Solutions Architect

Weitere Informationen

Weitere Informationen finden Sie unter:

- AWS-Whitepaper und -Leitfäden

Dokumentversionen

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

Update-Historie-Änderung	Update-Historie-Beschreibung	Update-Historie-Datum
Whitepaper aktualisiert	Aktualisiert für neue Servicefunktionen und -muster.	20. Oktober 2021
Whitepaper aktualisiert	Aktualisiert für neue Servicefunktionen und -muster.	1. Juni 2021
Whitepaper aktualisiert	Aktualisiert für neue Servicefunktionen.	25. September 2019
Erste Veröffentlichung	Whitepaper veröffentlicht.	1. November 2015

Hinweise

Kunden sind eigenverantwortlich für die unabhängige Bewertung der Informationen in diesem Dokument zuständig. Dieses Dokument: (a) dient rein zu Informationszwecken, (b) spiegelt die aktuellen Produktangebote und Verfahren von AWS wider, die sich ohne vorherige Mitteilung ändern können, und (c) impliziert keinerlei Verpflichtungen oder Zusicherungen seitens AWS und dessen Tochtergesellschaften, Lieferanten oder Lizenzgebern. AWS-Produkte oder -Services werden im vorliegenden Zustand und ohne ausdrückliche oder stillschweigende Gewährleistungen, Zusicherungen oder Bedingungen bereitgestellt. Die Verantwortung und Haftung von AWS gegenüber seinen Kunden wird durch AWS-Vereinbarungen geregelt. Dieses Dokument ist weder ganz noch teilweise Teil der Vereinbarungen zwischen AWS und seinen Kunden und ändert diese Vereinbarungen auch nicht.