Wirtschaftliche Unternehmensoptimierung durch serverlose Architekturen

September 2017

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 verantwortlich für ihre eigene Interpretation der in diesem Dokument zur Verfügung gestellten Informationen und für die Nutzung der AWS-Produkte oder -Services. Diese werden alle ohne Mängelgewähr und ohne jegliche Garantie, weder ausdrücklich noch stillschweigend, bereitgestellt. Dieses Dokument gibt keine Garantien, Gewährleistungen, vertragliche 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 Bestandteil der Vereinbarungen von AWS mit seinen Kunden und ändert diese Vereinbarungen auch nicht.

Inhalt

Kurzbeschreibung

Dieses Whitepaper richtet sich an CIOs, CTOs und leitende IT-Architekten, um ihnen einen Einblick in serverlose Architekturen und deren Auswirkung auf die Markteinführungszeit von Produkten sowie die Teamagilität und die Wirtschaftlichkeit von IT-Organisationen zu bieten. Durch serverlose Architekturen lassen sich im Leerlauf befindliche bzw. nicht voll ausgelastete Server auf der Designebene eliminieren und Cloud-basierte Softwaredesigns drastisch vereinfachen, was derzeit zu einem rapiden Wandel in der IT-Landschaft führt.

In diesem Whitepaper werden die Grundlagen serverloser Ansätze sowie das AWS-Portfolio serverloser Lösungen erörtert. Verschiedene Fallstudien zeigen, wie Unternehmen, die bereits mit serverlosen Ansätzen arbeiten, ihre Agilität deutlich steigern und ihre wirtschaftlichen Vorteile verbessern konnten. Lesen Sie, wie Unternehmen jeglicher Größe mithilfe von serverlosen Architekturen reaktionsfähige, ereignisbasierte Systeme entwickeln und Cloud-Microservices schnell und zu einem Bruchteil der bisherigen Kosten bereitstellen.

Einführung

Viele Unternehmen führen ihre Anwendungen bereits in der Cloud aus und genießen deren Vorteile wie etwa Kosteneinsparungen durch ein nutzungsabhängiges Preismodell sowie erhöhte Agilität dank On-Demand-IT-Ressourcen. Mehrere Studien zu verschiedenen Anwendungstypen und Branchen haben ergeben, dass sich durch die Migration bestehender Anwendungsarchitekturen zur Cloud die Gesamtbetriebskosten senken und die Markteinführung beschleunigen lassen.[1]

Gegenüber lokalen und privaten Cloud-Lösungen können Unternehmen Serverflotten und die darauf ausgeführten Anwendungen in der öffentlichen Cloud erheblich leichter erstellen, bereitstellen und verwalten. Neben klassischen server- oder VM-basierten Architekturen haben Unternehmen heute jedoch weitere Möglichkeiten, um die Vorteile der öffentlichen Cloud auszuschöpfen. Durch die Nutzung der Cloud entfallen zwar der Kauf und die Verwaltung von Hardware, doch für die Skalierbarkeit und Zuverlässigkeit sind Unternehmen bei *allen* serverbasierten Architekturen weiterhin selbst zuständig. Auch das Erstellen von Patches für ihre sich weiterentwickelnden Anwendungen sowie die Installation der Patches auf den Serverflotten liegt in der Verantwortung der Unternehmen. Darüber hinaus müssen sie ihre Serverflotten während Spitzenzeiten hoch- und außerhalb davon herunterskalieren, um die Kosten möglichst gering zu halten. All dies soll weder die Erfahrung der Endbenutzer noch die Integrität interner Systeme beeinträchtigen. Im Leerlauf befindliche bzw. nicht voll ausgelastete Server vergeuden Geld. Laut Analystenschätzungen sind ganze 85 Prozent der eingesetzten Server nicht voll ausgelastete.[2]

Mit serverlosen Datenverarbeitungsservices wie AWS Lambda werden diese Herausforderungen auf andere Weise angegangen. Sie bieten Unternehmen einen neuen Ansatz des Anwendungsdesigns, der nachweislich zur Kostensenkung und zur Beschleunigung der Markteinführung beiträgt. *AWS Lambda vereinfacht die Serververwaltung auf allen Ebenen der Technologieplattform. Durch das nutzungsabhängige Abrechnungsmodell fallen zudem keinerlei Kosten mehr für ungenutzte Rechenkapazität an.* Darüber hinaus erleichtern Lambda-Funktionen Unternehmen die Nutzung von Microservices. Durch die Eliminierung der Infrastruktur und die Einführung eines Lambda-Modells profitieren Unternehmen wirtschaftlich gleich doppelt:

- Im Leerlauf befindliche Server und deren wirtschaftliche Folgen sind damit passé. Serverlose Datenverarbeitungsservices wie AWS Lambda sind stets effizient, da nur dann Gebühren (mit auf die Millisekunde genauer Abrechnung) anfallen, wenn sie tatsächlich Arbeit verrichten.
- Auch die Flottenverwaltung entfällt – einschließlich der Entwicklung und Installation von Sicherheits-Patches und der Serverüberwachung. Die Verwaltung der damit verbundenen Tools, Verfahren und Rotationen auf Abruf, die bislang für eine rund um die Uhr verfügbare Serverflotte erforderlich war, gehört damit der Vergangenheit an. Unternehmen, die ihre Microservices mit Lambda erstellen, profitieren von erhöhter Agilität. Durch den Wegfall der aufwändigen Serververwaltung können sie ihre knapp kalkulierten IT-Ressourcen gewinnbringend für ihr Geschäft einsetzen.

Unternehmen, die bereits mit serverlosen Ansätzen arbeiten, erhalten aufgrund der reduzierten Infrastrukturkosten, der gesteigerten Agilität und ihrer auf das Wesentliche fokussierten Teams einen wichtigen Wettbewerbsvorteil.

Einführung in serverlose Anwendungen

Die Vorteile des oben genannten serverlosen Ansatzes klingen interessant, doch was ist bei der praktischen Implementierung zu beachten? Worin liegt der Unterschied zwischen einer serverlosen Anwendung und ihrem konventionellen serverbasierten Pendant?

Bei serverlosen Anwendungen können sich Entwickler auf ihre Kernkompetenz konzentrieren – das Schreiben der tatsächlichen Geschäftslogik. Mit viele Bestandteilen der Anwendungen, wie Webservern und dem gesamten sonstigen Verwaltungsaufwand (etwa der Sicherstellung der Zuverlässigkeit und Skalierbarkeit) brauchen sich die Entwickler nicht mehr zu befassen. Übrig bleibt ein sauberer, funktionierender Ansatz mit einer bedarfsorientierten Auslösung der Geschäftslogik, etwa wenn ein mobiler Benutzer eine Nachricht sendet, ein Bild in die Cloud hochgeladen wird, Datensätze an einen Stream gesendet werden usw. Ein asynchroner, ereignisbasierter Ansatz ist beim serverlosen Anwendungsdesign äußerst gängig – obgleich nicht erforderlich. Es passt optimal zu dem Konzept von Code, der ausschließlich bei Bedarf ausgeführt wird (und auch nur dann Kosten verursacht).

Serverlose Anwendungen werden in der öffentlichen Cloud basierend auf einem Service wie AWS Lambda ausgeführt. Dieser instanziiert und führt den Code bei Empfang eines Ereignisses oder Client-Aufrufs. Dieses Modell bietet gegenüber herkömmlichen serverbasierten Anwendungsdesigns diverse Vorteile:

- Es müssen keine Server bereitgestellt, implementiert, aktualisiert, überwacht oder anderweitig verwaltet werden. Die gesamte Hardware und Software wird vom Cloud-Anbieter verwaltet.
- Die Anwendung wird automatisch entsprechend ihrer Nutzung skaliert. Herkömmliche Anwendungen hingegen erfordern eine Flotte von Empfängern sowie eine explizite Verwaltung der Kapazität zur Bewältigung von Spitzenlasten.
- Neben der Skalierbarkeit ist bei serverlosen Anwendungen auch die Verfügbarkeit und Fehlertoleranz integriert. Diese Funktionen stehen Unternehmen gänzlich ohne eigene Codierungs-, Konfigurations- oder Verwaltungsaufwand zur Verfügung.
- Nicht genutzte Kapazität wird nicht berechnet. Es muss vorab weder Kapazität noch Überkapazität bereitgestellt werden (dies ist in der Tat nicht möglich). Die Abrechnung erfolgt stattdessen nutzungsabhängig und nur für die Dauer, die Code ausgeführt wird.

Fallbeispiele für serverlose Anwendungen

Das serverlose Anwendungsmodell ist generisch und auf nahezu jedes Fallbeispiel anwendbar, angefangen von der Webanwendung eines Startups bis hin zur Aktienanalyseplattform eines Fortune 100-Unternehmens. Hier ein paar Beispiele:

- **Webanwendungen und Websites:** Durch die Eliminierung von Servern können Sie Webanwendungen erstellen, für die in inaktivem Zustand nahezu keinerlei Kosten anfallen. In Spitzenzeiten lassen sie sich hingegen jederzeit hochskalieren.
- **Mobile Back-Ends:** Serverlose mobile Back-Ends bieten ermöglichen Entwicklern, die sich auf die Client-Entwicklung spezialisiert haben, auf einfache Weise sichere, hochverfügbare und optimal skalierte Back-Ends zu erstellen, ohne dafür umfassende Designkenntnisse für verteilte Systeme zu benötigen.

- **Medien und Protokollverarbeitung:** Serverlose Ansätze vereinfachen die parallele Verarbeitung umfassender Workloads, ohne dass Unternehmen dafür Multithread-Systeme erstellen oder Serverflotten manuell skalieren müssen.
- **IT-Automatisierung:** Sie können serverlose Funktionen mit Alarmen und Überwachungsfunktionen verknüpfen und bei Bedarf anpassen. Cron-Jobs und andere IT-Infrastrukturanforderungen lassen sich erheblich einfacher implementieren, wenn Unternehmen dafür keine Server besitzen und verwalten müssen. Dies gilt insbesondere für seltene oder stetig variierende Jobs und Anforderungen.
- **IoT-Back-Ends:** Die Fähigkeit, beliebigen Code einschließlich nativer Bibliotheken nutzen zu können, vereinfacht das Erstellen von Cloud-basierten Systemen, mit denen sich gerätespezifische Algorithmen implementieren lassen.
- **Chatbots (einschließlich sprachaktivierte Assistenten) und andere Webhook-basierte Systeme:** Serverlose Ansätze eignen sich optimal für jegliche Webhook-basierten Systeme wie etwa Chatbots. Ihre Fähigkeit, Aktionen (z. B. das Ausführen von Code) nur bei Bedarf durchzuführen, etwa wenn ein Benutzer Informationen von einem Chatbot anfordert, macht sie zu einer einfachen und in der Regel kostengünstigen Lösung für diese Architekturen. Die meisten Alexa Skills für Amazon Echo werden beispielsweise mit AWS Lambda implementiert.
- **Clickstream und andere nahezu in Echtzeit ausgeführte Daten-Streaming-Prozesse**: Serverlose Lösungen bieten eine flexible Skalierung entsprechend dem Datenfluss. Sie passen sich entsprechend den Durchsatzanforderungen an, ohne dass für jede Anwendung ein skalierbares Datenverarbeitungssystem erstellt werden muss. In Verbindung mit einer Technologie wie Amazon Kinesis lassen sich mit AWS Lambda Datensätzen für Clickstream-Analysen in Hochgeschwindigkeit verarbeiten, NoSQL-Datentrigger auslösen, Informationen zum Aktienhandel abrufen usw.

Neben den bereits erwähnten weitverbreiteten Fallbeispielen nutzen Unternehmen serverlose Ansätze auch in folgenden Bereichen:

- Big Data (etwa zur Lösung von MapReduce-Herausforderungen), Hochgeschwindigkeits-Videotranscodierung, Aktienanalysen und rechenintensive Monte Carlo-Simulationen für Kreditanträge: Entwickler haben erkannt, dass sie sich die Arbeit durch die parallele Nutzung eines serverlosen Ansatzes erheblich erleichtern können.[3] Dies gilt speziell dann, wenn Aktionen durch Ereignisse ausgelöst werden. Sie wenden serverlose Techniken daher zunehmend für eine Vielzahl von Big Data-Herausforderungen an, ohne sich um die Infrastrukturverwaltung kümmern zum müssen.
- Geringe Latenz, benutzerdefinierte Verarbeitung in Webanwendungen und über Netzwerke für die Inhaltsbereitstellung gelieferte Assets: Entwickler verlagern die serverlose Ereignisverarbeitung zum Edge des Internets, um von der geringen Latenz und der einfachen Anpassungsfähigkeit von Inhaltsabrufen zu profitieren. Dies eröffnet ein neues Spektrum von Anwendungsfällen mit standortbasierter Latenzoptimierung.
- Verbundene Geräte, die die Ausführung serverloser Funktionen etwa von AWS Lambda auf gewerblichen, privaten und Handheld-IoT-Geräten ermöglichen: Serverlose Lösungen wie etwa Lambda-Funktionen bieten eine natürliche Abstraktion von der zugrundeliegenden physischen oder auch virtuellen Hardware. Sie vereinfachen dadurch die Verlagerung vom Rechenzentrum zum Edge sowie zwischen verschiedenen Hardwarearchitekturen, ohne das Programmiermodell zu beeinträchtigen.
- Benutzerdefinierte Logik und Datenverarbeitung in lokalen Anwendungen wie AWS Snowball Edge: Da die Geschäftslogik bei serverlosen Anwendungen nur grob mit der Ausführungsumgebung verbunden ist, können sie auf einfache Weise in einer Vielzahl von Umgebungen (z. B. auf einer Appliance) eingesetzt werden.

Serverlose Anwendungen basieren in der Regel auf einer *Microservice-Architektur*. Sie sind in unabhängige Komponenten unterteilt, die diskrete Jobs ausführen. Diese Komponenten setzen sich aus individuellen Lambda-Funktionen sowie APIs, Nachrichtenwarteschlangen, Datenbank- und weiteren Bestandteilen zusammen und können eigenständig bereitgestellt, getestet und skaliert werden. Serverlose Anwendungen eignen sich aufgrund ihres funktionsbasierten Modells tatsächlich optimal für Microservices. Indem Unternehmen monolithische Designs und Architekturen vermeiden, können sie ihre Agilität erhöhen. Entwickler haben dadurch die Möglichkeit, einzelne Komponenten wie etwa die Datenbankebene bei Bedarf inkrementell bereitzustellen und sie zu ersetzen oder zu aktualisieren.

Oft reicht es, die Geschäftslogik von einer Anwendung zu trennen, um sie in eine serverlose Anwendung zu verwandeln. Services wie AWS Lambda unterstützen gängige Programmiersprachen und ermöglichen die Verwendung benutzerdefinierter Bibliotheken. Lang andauernde Aufgaben werden zu Arbeitsabläufen mit individuellen Funktionen, die innerhalb angemessener Zeitraumen ausgeführt werden. Einzelne Recheneinheiten können dadurch bei Bedarf neu gestartet oder parallel ausgeführt werden.

Eignet sich der serverlose Ansatz immer?

Nahezu alle modernen Anwendungen lassen sich für die Ausführung auf einer serverlosen Plattform anpassen. Sie werden dadurch meist auch wirtschaftlicher und besser skalierbar. Es gibt jedoch auch Fälle, in denen der serverlose Ansatz nicht die beste Wahl ist:

- Wenn Änderungen an einer Anwendung ausdrücklich vermieden werden sollen
- Wenn eine detaillierte Steuerung der Umgebung erforderlich ist, etwa beim Festlegen bestimmter Betriebssystem-Patches oder für den Zugriff auf Netzwerkvorgänge auf niedriger Ebene, damit der Code ordnungsgemäß ausgeführt wird
- Wenn eine lokale Anwendung nicht zur öffentlichen Cloud migriert wurde

Bewerten der serverlosen Plattformen von Cloud-Anbietern

Unternehmen müssen bei der Entwicklung einer serverlosen Anwendung mehr als nur die serverlosen Rechenfunktionen zum Ausführen des Anwendungscodes berücksichtigen. Gänzlich serverlose Anwendungen erfordern eine breite Palette von Services, Tools und Funktionen zum Zweck der Speicherung, Nachrichtenübermittlung, Diagnose und mehr. Ein unvollständiges oder fragmentiertes serverloses Portfolio eines Cloud-Anbieters kann für Entwickler serverloser Lösungen problematisch sein. Im schlechtesten Fall müssen sie zu serverbasierten Architekturen zurückkehren, wenn bei der Codierung keine konsistente Abstraktion möglich ist.

Eine optimale serverlose Plattform umfasst alle für die serverlose Anwendung erforderlichen Services etwa für die Datenverarbeitung und Speicherung. Darüber hinaus bietet sie die für die Entwicklung, Erstellung, Bereitstellung und Diagnose von serverlosen Anwendungen erforderlichen Tools. Für die Ausführung einer serverlosen Anwendung in einer Produktionsumgebung ist eine zuverlässige, flexible und vertrauenswürdige Plattform erforderlich. Diese sollte sowohl die Anforderungen kleiner Startups als auch weltweit tätiger Großkonzerne erfüllen. Die Plattform muss *alle* Elemente einer Anwendung skalieren können und durchgängige Zuverlässigkeit bieten. Die Herausforderungen beim Erstellen und Bereitstellen serverloser Lösungen sind für Entwickler ebenso vielfältig wie bei herkömmlichen Anwendungen. Damit eine serverlose Plattform den Anforderungen großer Unternehmen in unterschiedlichsten Branchen gerecht wird, sollte sie Folgendes bieten:

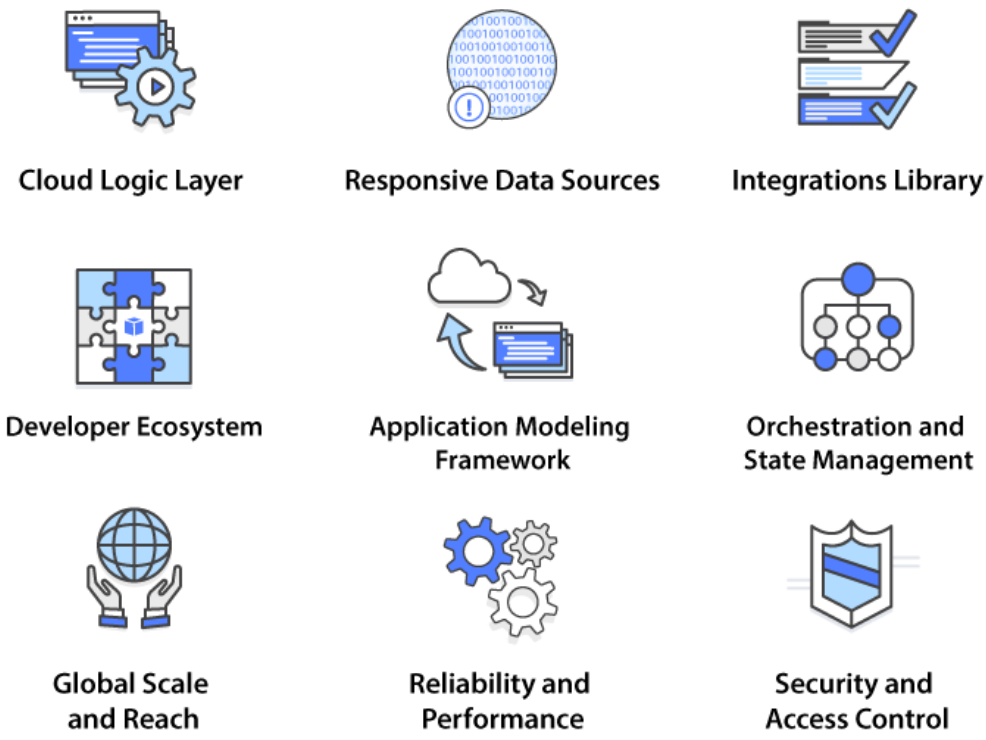


Abbildung 1: Merkmale einer serverlosen Plattform

- Eine leistungsfähige, skalierbare und zuverlässige *Cloud-Logikebene*
- Reaktionsfähige *Ereignis- und Datenquellen* des Erstanbieters sowie eine einfache *Konnektivität zu den Systemen von Drittanbietern*
- *Integrationsbibliotheken*, die Entwicklern einen leichten Einstieg ermöglichen und mit denen vorhandenen Lösungen schnell und sicher neue Muster hinzugefügt werden können
- Ein dynamisches *Entwicklernetzwerk*, in dem Entwickler Lösungen in unterschiedlichsten Bereichen erkunden und für eine Vielzahl von Drittanbietersystemen und Szenarien anwenden können
- Eine Sammlung zweckmäßiger *Frameworks für Anwendungsmodelle*
- Eine *Orchestrierung* mit Zustands- und Workflow-Verwaltung
- *Globale Reichweite* einschließlich Sicherheitsprogrammzertifizierung
- *Integrierte Zuverlässigkeit und skalierbare Leistung*, ohne dass eigene Kapazität bereitgestellt werden muss
- *Integrierte Sicherheit* und *flexible Zugriffskontrolle* für die Ressourcen und Services von Erst- und Drittanbietern

Im Kern jeder serverlosen Plattform ist die *Cloud-Logikebene* für die Ausführung der für die Geschäftslogik erforderlichen Funktionen zuständig. Da diese Funktionen häufig infolge von Ereignissen ausgelöst werden, ist die einfache *Integration in die Ereignisquellen von Erst- und Drittanbietern* wichtig, um einfache Lösungen zu erzielen, die automatisch entsprechend den variierenden Arbeitslasten skaliert werden. Es kann beispielsweise sein, dass serverlose Funktionen bei jeder Objekterstellung in einem Objektspeicher oder jedem Update einer serverlosen NoSQL-Datenbank ausgeführt werden müssen. Durch serverlose Architekturen verlagern sich die normalerweise für die Integration derartiger Systeme erforderliche Skalierung und Codeverwaltung auf den Cloud-Anbieter.

Damit Entwickler Anwendungen erfolgreich auf einer serverlosen Plattform erstellen können, ist ein einfacher Einstieg ausschlaggebend. Dies beinhaltet auch den Zugriff auf gebrauchsfertige Vorlagen für gängige Anwendungsfälle, unabhängig davon, ob diese Services von Erst- oder Drittanbietern involvieren. Diese *Integrationsbibliotheken* sind für die Vermittlung erfolgreicher Muster – wie die Verarbeitung von Datensatz-Streams oder die Implementierung von Webhooks – wichtig. Dies gilt insbesondere dann, wenn Entwickler von serverbasierten zu serverlosen Architekturen migrieren. Eng damit verbunden

ist der Bedarf eines *umfassenden und vielfältigen Entwicklernetzwerks* um die Kernplattform. Ein großes, dynamisches Netzwerk ermöglicht es Entwicklern, Lösungen der Community auf einfache Weise zu erkunden und anzuwenden und selbst neue Ideen und Ansätze beizutragen. Angesichts der vielfältigen für die Verwaltung von Anwendungslebenszyklen verfügbaren Toolketten stellt ein gesundes Netzwerk auch sicher, dass jede Sprache, IDE und Enterprise-Build-Technologie über die erforderlichen Laufzeiten, Plug-ins und Open-Source-Lösungen verfügt, um die Erstellung und Bereitstellung serverloser Anwendungen in vorhandene Ansätze zu integrieren. Wichtig bei serverlosen Anwendungen ist auch, vorhandene Investitionen zu nutzen. Hierzu zählen auch die Kenntnisse der Entwickler bezüglich Frameworks wie Express und Flask sowie gängiger Programmiersprachen. Ein umfassendes Netzwerk sorgt für die erforderliche Beschleunigung branchenübergreifender Entwicklungen. Entwickler können vorhandene Code dadurch in einer serverlosen Architektur auch besser für neue Zwecke nutzen.

Frameworks für Anwendungsmodelle wie etwa das serverlose Anwendungsmodell von AWS (AWS SAM) mit offener Spezifikation ermöglichen Entwicklern, die Komponenten einer serverlosen Anwendung zu definieren. Sie können dadurch auch die zum Erstellen, Bereitstellen und Überwachen dieser Anwendung erforderlichen Tools und Arbeitsabläufe einsetzen. Ein weiteres für den Erfolg einer serverlosen Plattform wichtiges Framework besteht aus der *Orchestrierung und Zustandsverwaltung*. Da die serverlose Datenverarbeitung größtenteils zustandslos erfolgt, ist für lang andauernde Workflows ein ergänzender Mechanismus erforderlich. Durch Orchestrierungslösungen können Entwickler die zahlreichen Komponenten einer serverlosen Anwendung koordinieren. Gleichzeitig ermöglichen sie die Zusammensetzung dieser Anwendungen aus kleinen und nur kurz dauernden Funktionen. Orchestrierungsservices vereinfachen auch die Fehlerverarbeitung und ermöglichen die Integration in ältere Systeme und Arbeitsabläufe. Dies gilt auch für Arbeitsabläufe, deren Ausführung länger dauert, als serverlose Funktionen diese normalerweise zulassen.

Damit eine serverlose Plattform Kunden weltweit unterstützen kann – einschließlich multinationaler Unternehmen mit globaler Reichweite – muss sie eine *globale Skalierung* ermöglichen, die auch weltweit stationierte Rechenzentren und Edge-Standorte einbezieht. Edge-Standorte sind für eine serverlose Datenverarbeitung mit geringer Latenz in der Nähe von Endbenutzern wichtig. Da für die Skalierbarkeit und Hochverfügbarkeit serverloser Anwendungen die Plattform und nicht der Anwendungsentwickler zuständig ist, ist die inhärente

Zuverlässigkeit der Plattform ausschlaggebend. Funktionen wie integrierte Wiederholversuche und Warteschlangen für nicht verarbeitete Ereignisse erleichtern Entwicklern die Erstellung stabiler serverloser Systeme mit durchgängiger Zuverlässigkeit. Ebenso wichtig ist die Leistung – insbesondere die geringe Latenz –, da die Laufzeitumgebung und der Kundencode in serverlosen Anwendungen nach Bedarf instanziiert werden.

Schließlich muss die Plattform umfassende *Sicherheits- und Zugriffskontrollen* besitzen. Hierzu zählen die Unterstützung virtueller privater Netzwerke, rollen- und zugriffsbasierte Berechtigungen, die zuverlässige Integration in API-basierte Authentifizierungs- und Zugriffskontrollmechanismen (einschließlich Systeme von Erst- und Drittanbietern) sowie die Verschlüsselung von Anwendungselementen wie die Einstellungen von Umgebungsvariablen. Serverlose Systeme bieten von Haus aus ein erhöhtes Maß an Sicherheit und Kontrolle. Dies hat folgende Gründe:

- **Erstklassige Flottenverwaltung einschließlich Sicherheits-Patches:** In einem System wie AWS Lambda werden die Anfragen ausführenden Server kontinuierlich überwacht, rotiert und auf Sicherheitslücken überprüft. Patches können innerhalb weniger Stunden nach der Ausgabe wichtiger Sicherheitsaktualisierungen installiert werden. Zahlreiche Enterprise-Serverflotten sind demgegenüber mit erheblich lockereren SLAs für Patches und Updates ausgestattet.
- **Begrenzte Serverlebensdauer:** Jede Maschine, auf der in AWS Lambda Kundencode ausgeführt wird, wird mehrmals täglich rotiert. Dies reduziert das Angriffsrisiko und stellt sicher, dass sich das Betriebssystem und die Sicherheits-Patches stets auf dem aktuellen Stand befinden.
- **Authentifizierung, Zugriffskontrolle und Prüfung pro Anforderung: Jede Datenverarbeitungsanforderungen in AWS Lambda wird ungeachtet der Quelle individuell authentifiziert, für den Zugriff auf bestimmte Ressourcen autorisiert und vollständig überprüft.** Bei Anforderungen außerhalb von AWS-Rechenzentren über Amazon API Gateway greifen zusätzliche mit dem Internet verbundene Abwehrsysteme einschließlich Schutzmaßnahmen vor DoS-Angriffen. Unternehmen, die zu serverlosen Architekturen migrieren, erhalten mit AWS CloudTrail einen detaillierten Einblick, welche Benutzer mit welchen Berechtigungen auf die verschiedenen Systeme zugreifen. Die Prüfdatensätze können sie mit AWS Lambda programmatisch verarbeiten.

Serverlose AWS-Plattform

AWS hat seit der Einführung von Lambda 2014 ein vollständige serverlose Plattform erstellt. Sie umfasst eine breite Palette vollständig verwalteter Services, mit denen Unternehmen serverlose Anwendungen erstellen können, die sich nahtlos in andere Services von AWS und Drittanbietern integrieren lassen. Abbildung zeigt einen Teil dieser Komponenten auf der serverlosen AWS-Plattform und deren Beziehung zueinander.

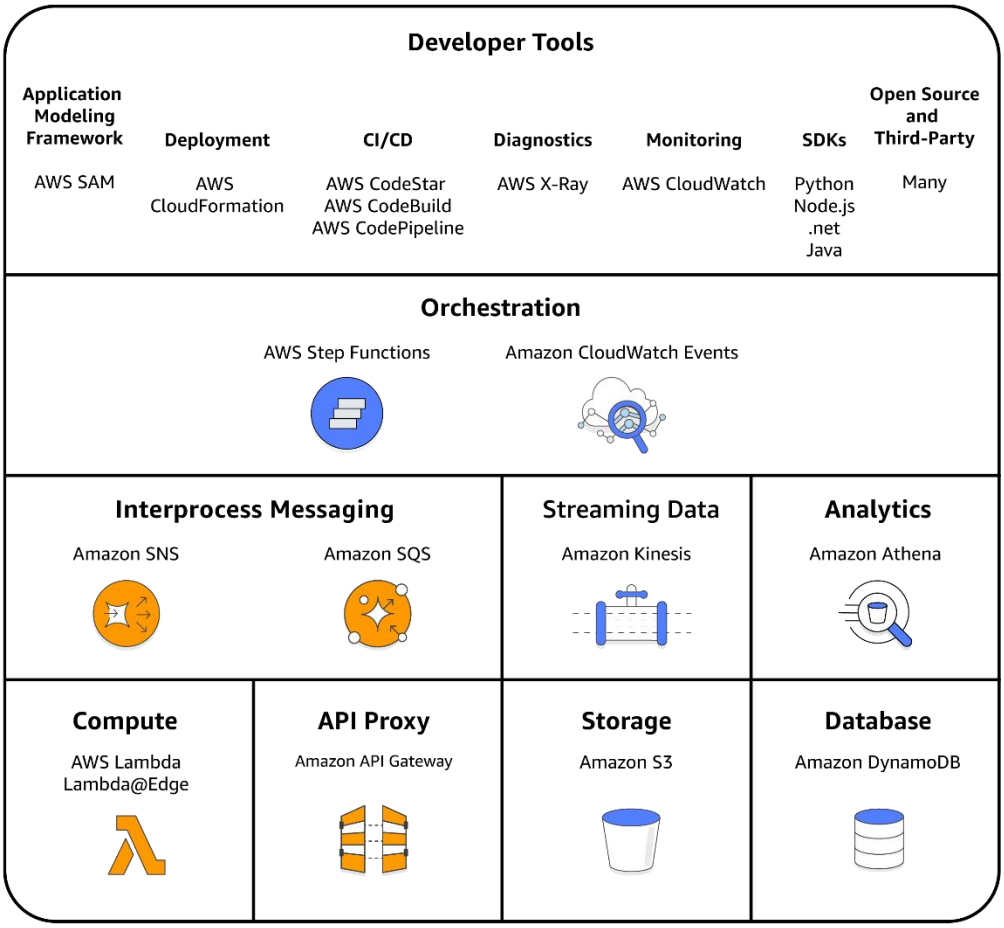
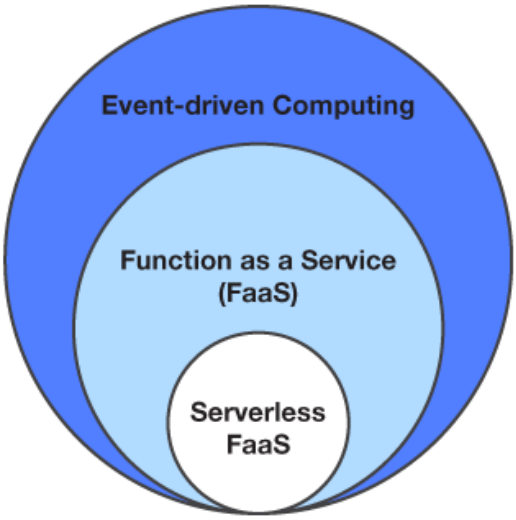


Abbildung 2: Komponenten der serverlosen AWS-Plattform

Funktionen der serverlosen AWS-Plattform

AWS bietet alle im vorangegangenen Abschnitt erläuterten Kernfunktionen, die für eine vollständig serverlose Plattform erforderlich sind. Die Bereitstellung der Cloud-Logikebene erfolgt durch AWS Lambda, einen hochgradig skalierbaren, bereitstellungsfreien serverlosen Datenverarbeitungsservice, der auf Funktionen basiert. AWS Lambda wird durch AWS Lambda@Edge ergänzt. Dieser Service bietet ähnliche Unterstützung für die Ausführung von Lambda-Funktionen mit extrem geringer Latenz mittels Edge-optimiertem Routing. Als weitere Ergänzung ermöglicht AWS Greengrass die Ausführung von Lambda-Funktionen auf verbundenen Geräten. Dies gilt auch für Appliances wie AWS Snowball.

Lambda-Funktionen lassen sich leicht durch verschiedene Ereignisse von Erst- und Drittanbietern auslösen. Entwickler können dadurch reaktionsfähige, ereignisgesteuerte Systeme (siehe Abbildung 3) erstellen, ohne dafür eine Infrastruktur einrichten oder verwalten zu müssen. Bei mehreren simultanen Ereignissen führt Lambda einfach parallel für jeden Trigger eine Kopie der Funktion aus. Lambda-Funktionen werden exakt entsprechend der Arbeitslast skaliert, bis hin zu jeder einzelnen Anforderung. Daher gibt es weder Server noch Container im Leerlauf. In Architekturen, in denen Lambda-Funktionen genutzt werden, wurde das Problem vergeudeter Infrastrukturausgaben *gezielt* eliminiert.



FaaS bzw. *Function as a Service* ist ein Ansatz zur Erstellung ereignisgesteuerter Datenverarbeitungssysteme, deren Bereitstellung und Ausführung auf Funktionen basiert. *Serverless FaaS* ist eine Art der FaaS, bei der das Programmiermodell keine virtuellen Maschinen oder Container enthält und der Anbieter bereitstellungsfreie Skalierbarkeit und integrierte Zuverlässigkeit bietet.

Abbildung 3: Beziehung zwischen ereignisgesteuerter Datenverarbeitung, FaaS und Serverless

Die von Lambda gebotenen serverlosen AWS-Datenverarbeitungsfunktionen sind ein wichtiger Bestandteil der folgenden verwalteten AWS-Services. Sie alle lassen sich nahtlos gegenseitig integrieren:

- Amazon API Gateway: HTTP-Endpunkte für Lambda-Funktionen einschließlich einer vollständigen Reihe von API-Proxy- und API-Verwaltungsfunktionen
- Amazon S3: Ermöglicht beim Erstellen, Kopieren oder Löschen von Objekten die Verwendung von Lambda-Funktionen als automatische Ereignistrigger
- Amazon DynamoDB: Ermöglicht die Verwendung von Lambda-Funktionen zur Verarbeitung jeglicher an einer Datenbanktabelle vorgenommenen Änderungen
- Amazon SNS: Ermöglicht die Weiterleitung von Nachrichten zur Verarbeitung an Lambda-Funktionen sowie die dynamische Reaktion auf veröffentlichte Inhalte
- Amazon SQS: Ermöglicht die einfache Verarbeitung von Nachrichten in Warteschlangen durch Lambda-Funktionen
- Amazon Kinesis Streams: Ermöglicht die sequenzielle Verarbeitung von Streaming-Daten mit Lambda-Funktionen und erleichtert dadurch das Erstellen von nahezu in Echtzeit ausgeführten Analysemodulen
- Amazon Kinesis Firehose: Ermöglicht das automatische Anwenden von Lambda-Funktionen auf die von einem Firehose-Service aufgenommenen Datensätze, um einen Daten-Stream auf einfache Weise umzuwandeln, zu filtern und zu analysieren.
- Amazon Athena: Ermöglicht das automatische Auslösen von Lambda-Funktionen für jedes Objekt im Ergebnissatz einer Abfrage
- AWS Step Functions: Ermöglicht die Orchestrierung mehrere Lambda-Funktionen, um lang andauernde Workflows für manuell orientierte und automatisierte Verfahren zu erstellen
- Amazon CloudWatch Events: Ermöglicht die automatische Reaktion auf Ereignisse – auch von Dritten – mit Lambda-Funktionen
- Amazon Aurora: Ermöglicht das Schreiben von Datenbanktriggern in Form von Lambda-Funktionen

Lambda bietet eine Integrationsbibliothek mit Entwürfen für eine Vielzahl von Drittanbieterservices wie Slack, Algorithmia, Twilio, Loggly, Splunk, SumoLogic, Box usw. Entwickler können dadurch mit nur wenigen Codezeilen reaktionsfähige Anwendungen mit erweiterten Algorithmen, Analyse- und Kommunikationsfunktionen und mehr erstellen. Eine Vielzahl von Webanwendungs-Frameworks wie etwa Express (für NodeJS-Anwendungen) und Flask (für Python-Anwendungen) wurden für die Verwendung von Lambda-Funktionen optimiert. Open-Source-Webanwendungsprojekte umfassen das Serverless Framework, Sparta, Chalice und viele weitere.

Serverlose Anwendungen umfassen in der Regel mehrere Komponenten: eine oder mehrere Funktionen, eine serverlose Datenbank wie etwa Amazon DynamoDB und entweder eine von Clients aufrufbare API oder eine Ereignisquelle für die Anwendungsauslösung. Zum Koordinieren dieser Komponenten verwendet AWS das serverlose Anwendungsmodell SAM, das sich durch seine offene Spezifikation auszeichnet. Mit SAM können Entwickler die Funktionen, APIs, Ereignisquellen, Datenbanktabellen und sonstigen Bestandteilte einer serverlosen Anwendung auf einfache Weise beschreiben. SAM erleichtert Entwicklern auch die Verwaltung aller Schritte des Softwareentwicklungszyklus. AWS bietet ergänzend dazu eine Reihe von Tools und Services. Die Schritte umfassen die native Unterstützung lokaler Tests und Debugging-Vorgänge in einer beliebigen IDE (oder über die Befehlszeile) mithilfe von SAM Local, das Bereitstellen von SAM-Anwendungen mit AWS CloudFormation, das Erstellen von SAM-Anwendungen in AWS CodeBuild sowie die GitHub-basierte CI/CD für SAM-Anwendungen in der AWS CodePipeline. Neben der Unterstützung durch den Erstanbieter unterstützen auch eine Reihe von Open-Source-Frameworks, CI/CD-Anbietern und Leistungsverwaltungsanbietern SAM- und Lambda-Funktionen wie Serverless Framework, Claudia, CloudBees, Datadog und viele mehr. Weitere Beispiele erhalten Sie unter Entwickler-Tools für serverlose Anwendungen.[4]

Nach der Erstellung einer Lambda-Funktion – bzw. im Fall einer SAM-Anwendung mehrerer zusammenwirkender Lambda-Funktionen – können Entwickler diese bequem mithilfe von automatisch erstellten Metriken und Protokollen in Amazon CloudWatch und CloudWatch Logs überwachen. Mit AWS X-Ray bietet AWS darüber hinaus eine serviceübergreifende Lösung für die Anforderungsverfolgung und Leistungsanalyse. Entwickler können damit die Ausführung und das Verhalten einzelner Funktionen sowie die verarbeiteten Ereignisse verfolgen.

Die serverlosen Plattformprodukte von AWS zeichnen sich durch ihre globale Reichweite aus. Sie unterstützen AWS Lambda und Amazon API Gateway in praktisch allen AWS-Regionen weltweit. Lambda@Edge ist an allen Edge-Standorten verfügbar.[5] Lambda bietet eine Reihe von Funktionen, mit denen Kunden die Zuverlässigkeit ihrer Anwendungen erhöhen können. Hierzu zählen Wiederholversuche bei asynchronen und geordneten Ereignissen sowie Warteschlangen für die Erfassung von Ereignissen, die von der Anwendung nicht verarbeitet werden konnten. Durch die umfassende Integration in Amazon Virtual Private Cloud (Amazon VPC) und die flexible Reihe der Authentifizierungsfunktionen und Zugriffskontrollen von AWS Lambda können Unternehmen sichere Anwendungen nach bewährten Methoden wie etwa dem Grundsatz der geringsten Berechtigungen erstellen. Auch die Sicherheit und Verwaltung von Endbenutzern ist einfach. Die von Amazon Cognito gebotenen Autorisierungs- und Authentifizierungsfunktionen lassen sich bequem mit Amazon API Gateway und AWS Lambda kombinieren. Dies ermöglicht die serverlose Benutzerregistrierung und -anmeldung, unter anderem bei Social Media-Anbietern wie Facebook sowie Unternehmensverzeichnissen.

Fallstudien

Unternehmen nutzen serverlose Architekturen für vielfältige Anwendungsfälle, angefangen von Aktienanalysen über die Erstellung von E-Commerce-Websites bis hin zur Verarbeitung natürlicher Sprache. Mit AWS Lambda und dem restlichen AWS-Portfolio serverloser Lösungen können Sie flexibel eine Vielzahl von Anwendungen erstellen, auch solche, die mit den Sicherheitsvorgaben von PCI oder HIPAA kompatibel sein müssen. In den folgenden Abschnitten werden ein paar der gängigsten Anwendungsfälle aufgeführt. Die Liste ist jedoch nicht erschöpfend. Eine vollständige Auflistung der Kundenreferenzen und Anwendungsfällen erhalten Sie unter Serverlose Datenverarbeitung und serverlose Anwendungen.[6]

Serverlose Websites, Webanwendungen und mobile Back-Ends

Serverlose Ansätze eignen sich optimal für Anwendungen, deren Workload dynamisch variieren kann. Durch den serverlosen Ansatz fallen nur bei durch Endbenutzer generiertem Datenverkehr Kosten für die Datenverarbeitung an. In Spitzenzeiten, etwa infolge eines Flash Sales auf einer E-Commerce-Website

oder eines Social Media-Kommentars, der zu einem plötzlichen Anstieg des Datenverkehrs führt, ist dennoch eine sofortige Skalierung möglich. Im Verhältnis zu herkömmlichen Infrastrukturansätzen lassen sich webbasierte oder mobile Back-Ends, die auf einem serverlosen Ansatz basieren auch erheblich kostengünstiger entwickeln, bereitstellen und betreiben.

AWS stellt Entwicklern die erforderlichen Services für eine schnelle Anwendungserstellung zur Verfügung:

- Amazon S3 bietet eine einfache Hosting-Lösung für statische Inhalte.
- AWS Lambda unterstützt in Verbindung mit Amazon API Gateway dynamische API-Anforderungen mithilfe von Funktionen.
- Amazon DynamoDB bietet eine einfache Speicherlösung für benutzerbezogene Sitzungen.
- Amazon Cognito vereinfacht die Registrierung und Authentifizierung von Endbenutzern sowie die Zugriffskontrolle auf Ressourcen.
- AWS SAM kann von Entwicklern genutzt werden, um die verschiedenen Elemente einer Anwendung zu beschreiben.
- AWS CodeStar ermöglicht das Einrichten einer CI/CD-Toolkette mit nur wenigen Klicks.

Weitere Informationen erhalten Sie im Whitepaper zu mehrschichtigen serverlosen AWS-Architekturen. Darin werden eingehend Muster zum Erstellen serverloser Webanwendungen erläutert.[7] Eine vollständige Referenzarchitektur finden Sie in GitHub unter Serverless Reference Architecture for creating a Web Application[8] und Serverless Reference Architecture for creating a Mobile Backend[9].

Kundenbeispiel: Bustle.com

Bustle.com ist eine speziell auf weibliche Leser ausgerichtete Website mit Neuigkeiten, Unterhaltung, Lifestyle und Mode. Durch den Wechsel zu einer serverlosen, auf AWS Lambda und Amazon API Gateway basierenden Architektur konnte Bustle etwa 84 Prozent seiner Kosten einsparen. Die Techniker von Bustle gewannen an Agilität und konnten sich auf die Erstellung neuer Produkt fokussieren, anstatt sich um die Infrastrukturverwaltung und die Skalierung zu kümmern. Das Bustle-Team ist effizienter geworden und benötigt für die Erstellung und den Betrieb von Websites gegenüber ähnlichen Unternehmen nur halb so viele Mitarbeiter. Das serverlose Back-End von Bustle unterstützt außerdem die iOS-Apps für zwei seiner Websites (Bustle und Romper). Weitere Informationen erhalten Sie in der Fallstudie zu Bustle.[10]

IoT-Back-Ends

Die Vorteile einer serverlosen Architektur für Webanwendungen und mobile Apps erleichtern auch das Erstellen von IoT-Back-Ends und gerätebasierten Analysesystemen, die nahtlos entsprechend der Gerätezahl skalierbar sind. Ein Beispiel einer Referenzarchitektur finden Sie in GitHub unter Serverless Reference Architecture for creating an IoT Backend.[11]

Kundenbeispiel: iRobot

iRobot stellt Roboter wie den Reinigungsroboter Roomba. Für die Erstellung eines serverlosen Back-Ends für seine IoT-Plattform verwendet der Hersteller AWS Lambda in Verbindung mit dem AWS IoT-Service. Die serverlosen Architektur entbindet das Technikerteam von iRobot von der Infrastrukturverwaltung sowie dem manuellen Schreiben von Code, um die Verfügbarkeit und Skalierbarkeit sicherzustellen. Das Unternehmen kann dadurch schneller Innovationen entwickeln und sich auf seine Kunden fokussieren. Weitere Informationen finden Sie auf den Folien der im Rahmen von AWS re:Invent 2016 abgehaltenen Präsentation Serverless IoT Back Ends (IOT401)[12]. Hier können Sie das Video dazu ansehen.[13]

Datenverarbeitung

Große serverlose Anwendungen verarbeiten riesige Datenmengen, oft in Echtzeit. In typischen serverlosen Datenverarbeitungsarchitekturen werden Streaming-Daten mit einer Kombination aus Amazon Kinesis und AWS Lambda verarbeitet. Zum Auslösen der Datenverarbeitung infolge von Objekterstellungen oder Aktualisierungsereignissen wird Amazon S3 zusammen mit AWS Lambda genutzt. Wenn Arbeitslasten anstatt eines einfachen Triggers eine komplexerer Orchestrierung erfordern, können Entwickler mit AWS Step Functions zustandsbehaftete oder lang andauernde Workflows generieren, die im Verlauf eine oder mehrere Lambda-Funktionen auslösen. Weitere Informationen zu serverlosen Datenverarbeitungsarchitekturen erhalten Sie in GitHub unter folgenden Themen:

- Serverless Reference Architecture for Real-time Stream Processing[14]
- Serverless Reference Architecture for Real-time File Processing[15]
- Image Recognition and Processing Backend reference architecture[16]

Kundenbeispiel: FINRA

Die Financial Industry Regulatory Authority (FINRA) nutzte AWS Lambda zum Erstellen einer serverlosen Datenverarbeitungslösung, mit der die Behörde täglich eine halbe Billion Datenvalidierungen zu 37 Milliarden Börsenereignissen durchführen kann. In seinem Vortrag bei der AWS re:Invent 2016 mit dem Titel The State of Serverless Computing (SVR311)[17] erklärte Tim Griesbach, Senior Director bei FINRA: „Wir haben festgestellt, dass Lambda für uns die beste Lösung für diesen serverlosen Cloud-Ansatz ist. Durch Lambda konnten wir das System beschleunigen und besser skalieren und gleichzeitig die Kosten senken. Unter dem Strich haben wir unsere Kosten um mehr als 50 Prozent reduziert … und können diese täglich oder auch stündlich verfolgen."

Kundenbeispiel: Thomas Reuters

Der Medienkonzern Thomson Reuters erstellte eine serverlose Geschäftsanalyselösung, mit der seine Produktteams auf einfache Weise Produktnutzungsdaten analysieren können. Die Lösung vereint AWS Lambda, Amazon Kinesis Streams und Amazon Kinesis Firehose, um Streaming-Daten zu Ereignissen zu erfassen und zu analysieren. Das Ergebnis mit dem Namen Product Insight konnte zwei Monate früher als geplant eingeführt werden und hat die technischen Erwartungen überstiegen.

Anders Fritz, Senior Manager für Produktinnovation bei Thomson Reuters, erklärte „Ursprünglich wollten wir pro Sekunde 2 000 Ereignisse verarbeiten. Unsere Tests haben ergeben, dass Product Insight auf AWS pro Sekunde bis zu 4 000 Ereignisse verarbeiten kann. Geplant ist, diese Zahl in einem Jahr auf über 10 000 Ereignisse pro Sekunde zu steigern." Diese Zahl bedeutet über 25 Milliarden Ereignisse pro Monat. Trotz des hohen Durchsatzes sind seit der Einführung des Systems keine Daten verloren gegangen. „Aufgrund der stabilen Failover-Architektur und der technischen Funktionen von AWS haben wir seit Beginn unserer Datenerfassung kein einziges Ereignis verloren", erklärt Fritz. Weitere Informationen entnehmen Sie der Fallstudie zu Thomson Reuters[18], oder sehen sie sich die AWS re:Invent 2016-Präsentation Real-time Data Processing Using AWS Lambda (SVR301) an.[19]

Big Data

AWS Lambda eignet sich optimal für die parallele Verarbeitung umfangreicher Arbeitslasten. Ein Beispiel einer Referenzarchitektur unter Verwendung von MapReduce erhalten Sie in GitHub unter Reference architecture for running serverless MapReduce jobs.[20]

Kundenbeispiel: Fannie Mae

Fannie Mae, eine führende Hypothekenbank, nutzt AWS Lambda zur Ausführung einer „unsäglich parallelen“ Arbeitslast für sein Finanzmodell. Fannie Mae nutzt Monte Carlo-Simulationsverfahren, um zukünftige Cashflows von Hypotheken hochzurechnen und dadurch sein Hypothekenrisiko möglichst gering zu halten. Das Unternehmen hatte festgestellt, dass seine bestehenden HPC-Raster seine wachsenden Geschäftsanforderungen nicht mehr erfüllen. Fannie Mae baute seine neue Plattform auf Lambda auf und erzielte während eines Testlaufs eine Skalierung auf 15 000 parallel ausgeführte Funktionen. Dabei wurde eine Simulation mit 20 Millionen Hypotheken innerhalb von 2 Stunden durchgeführt. Dies entspricht einem Drittel des Zeitaufwands des alten Systems. Dank seiner neuen serverlosen Architektur kann Fannie Mae umfassende Monte Carlo-Simulationen jetzt kosteneffizient ausführen, da keine Gebühren mehr für im Leerlauf befindliche Datenverarbeitungsressourcen anfallen. Durch die parallele Ausführung mehrere Lambda-Funktionen beschleunigt die Hypothekenbank außerdem die Datenverarbeitung. Auch die durchschnittliche Markteinführungszeit von Fannie Mae hat sich verkürzt. Das Unternehmen konnte die Serververwaltung und -überwachung sowie den Großteil des komplexen Codes eliminieren, der bislang für die Skalierung und Zuverlässigkeit der Anwendung erforderlich war. Weitere Informationen erhalten Sie in der Fannie Mae AWS Summit 2017-Präsentation SMC303: Real-time Data Processing Using AWS Lambda.[21]

IT-Automatisierung

Durch serverlose Ansätze entfällt der mit der Serververwaltung verbundene Aufwand. Darüber hinaus vereinfacht sich die Erstellung und Verwaltung der meisten Infrastrukturaufgaben wie die Bereitstellung, Konfiguration und Verwaltung sowie Alarm- und Überwachungsfunktionen. Auch zeitgesteuerte Cron-Jobs lassen sich erheblich leichter erstellen und verwalten.

Kundenbeispiel: Autodesk

Autodesk, ein Hersteller von 3D-Design- und Engineering-Software, hat mit AWS Lambda die Erstellung und Verwaltung seiner AWS-Konten innerhalb seiner Engineering-Abteilung automatisiert. Laut Schätzung von Autodesk belaufen sich die Kosteneinsparungen auf 98 Prozent (unter Berücksichtigung der geschätzt eingesparten Arbeitsstunden für die Kontoeinrichtung). Der Softwarehersteller kann Konten jetzt in unter 10 Minuten anstelle von 10 Stunden bereitstellen, die dies mit dem vorherigen infrastrukturbasierten Verfahren dauerte. Die serverlose Lösung ermöglicht Autodesk, Konten bereitzustellen, Standards zu konfigurieren und durchzusetzen und Prüfungen verstärkt automatisch und mit weniger manuellen Berührungspunkten zu bewerkstelligen. Weitere Informationen erhalten Sie in der Fannie Mae AWS Summit 2017-Präsentation SMC301: The State of Serverless Computing.[22] Besuchen Sie GitHub, um mehr zum Tailor-Service von Autodesk zu erfahren.

Weitere Fallstudien

Die im vorangegangenen Abschnitt beschriebenen Anwendungsfälle veranschaulichen nur einen Bruchteil der Möglichkeiten, die Lambda und die anderen serverlosen Lösungen von AWS bieten. Zu den weiteren Anwendungsfällen zählen die leistungsfähige menschliche Spracherkennung durch Chatbots unter Verwendung von Amazon Lex und AWS Lambda, das globale Edge-Computing mit von Lambda@Edge und Amazon CloudFront sowie die leistungsstarke lokale Dateiverarbeitung mit in eine AWS Snowball-Appliance integrierten Lambda-Funktionen. Dies sind nur ein paar der interessanten Möglichkeiten dieses vielseitigen Ansatzes. Weitere Informationen erhalte Sie unter AWS Lambda.[23]

Fazit

Serverlose Ansätze dienen zur Lösung von zwei klassischen Herausforderungen der IT-Verwaltung: im Leerlauf befindliche Server, die ein Unternehmen Geld kosten, ohne einen Mehrwert zu liefern, und die Kosten für das Erstellen und Betreiben von Serverflotten und Serversoftware. Beides lenkt von der Erzielung einer wettbewerbsfähigen Wertschöpfung für Kunden ab. AWS Lambda und die anderen serverlosen Produkte von AWS lösen diese seit längerem bestehenden Herausforderungen durch Eliminieren der Server, Container, Festplatten und sonstigen Infrastrukturressourcen aus dem Programmier- und

Abrechnungsmodell. Entwickler können folglich mit einem sauberen Anwendungsmodell arbeiten und dadurch Bereitstellungen beschleunigen. Unternehmen bezahlen zudem nur für wertschöpfende Arbeit. Am einfachsten und schnellsten erstellen Sie reaktive, ereignisbasierte Systeme und Cloud-native Microservices mithilfe einer serverlosen Architektur. Weitere Informationen und Whitepaper zu verwandten Themen finden Sie unter Serverlose Datenverarbeitung und serverlose Anwendungen.[24]

Mitwirkende

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

- Tim Wagner, General Manager für AWS Serverless Applications, Amazon Web Services

Weitere Informationen

Zusätzliche Informationen finden Sie in den folgenden Ressourcen:

- Serverless Reference Architectures with AWS Lambda von Werner Vogels, CTO bei Amazon.com
- AWS re:Invent 2016: The State of Serverless Computing [Präsentation] von Tim Wagner, General Manager bei AWS Serverless Applications
- The economics of serverless cloud computing von Owen Rogers, wissenschaftlicher Leiter bei at 451 Research

Referenzarchitekturen

- Webanwendungen
- Mobile Back-Ends
- IoT-Back-Ends
- Datenverarbeitung
- Stream-Verarbeitung
- Verarbeitung der Bilderkennung
- MapReduce

Am Dokument vorgenommene Änderungen

Datum	Beschreibung
September 2017	Erstveröffentlichung

Notes

[1] https://www.forbes.com/sites/moorinsights/2016/04/11/tco-analysis-demonstrates-how-moving-to-the-cloud-can-save-your-company-money/#537e2bd07c4e
http://www.cloudstrategymag.com/articles/86033-understanding-tco-cloud-economics

[2] In einem 2012 von Gartner veröffentlichten Artikel wird die Auslastung von Rechenzentren auf 7 bis 12 % geschätzt (http://www.nytimes.com/2012/09/23/technology/data-centers-waste-vast-amounts-of-energy-belying-industry-image.html). McKinsey schreibt in einer 2008 herausgegebenen Studie von 6 % (https://www.sallan.org/pdf-docs/McKinsey_Data_Center_Efficiency.pdf). In einem Whitepaper von Accenture, in dem eine Reihe EC2-basierter Anwendungen analysiert wurden, wird die Auslastung mit etwa 7 % beziffert (http://ieeexplore.ieee.org/document/6118751/). Eine 2014 von NRDC und Anthesis durchgeführte Studie ergab, dass 2013 über 30 % der Server vollständig „komatös“ waren – sie waren zwar eingeschaltet, lieferten aber keinerlei Wert (http://anthesisgroup.com/wp-content/uploads/2015/06/Case-Study_DataSupports30PercentComatoseEstimate-FINAL_06032015.pdf).

[3] Occupy the Cloud: *Distributed Computing for the 99%*, Eric Jonas et al., https://arxiv.org/abs/1702.04024.

[4] https://aws.amazon.com/serverless/developer-tools

[5] https://aws.amazon.com/lambda/edge/

[6] https://aws.amazon.com/serverless/

[7] https://d0.awsstatic.com/whitepapers/AWS_Serverless_Multi-Tier_Architectures.pdf

[8] https://github.com/awslabs/lambda-refarch-webapp

[9] https://github.com/awslabs/lambda-refarch-mobilebackend

[10] https://aws.amazon.com/solutions/case-studies/bustle/

[11] https://github.com/awslabs/lambda-refarch-iotbackend

[12] https://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-serverless-iot-back-ends-iot401

[13] https://www.youtube.com/watch?v=gKMaf5E-z7Q

[14] https://github.com/awslabs/lambda-refarch-streamprocessing

[15] https://github.com/awslabs/lambda-refarch-fileprocessing

[16] https://github.com/awslabs/lambda-refarch-imagerecognition

[17] https://www.youtube.com/watch?v=AcGv3qUrRC4&feature=youtu.be&t=1153

[18] https://aws.amazon.com/solutions/case-studies/thomson-reuters/

[19] https://www.youtube.com/watch?v=VFLKOy4GKXQ&feature=youtu.be&t=1449

[20] https://github.com/awslabs/lambda-refarch-mapreduce

[21] https://www.slideshare.net/AmazonWebServices/smc303-realtime-data-processing-using-aws-lambda/28

[22] https://www.slideshare.net/AmazonWebServices/smc301-the-state-of-serverless-computing-75290821/22

[23] https://aws.amazon.com/lambda/

[24] https://aws.amazon.com/serverless/