Engineer

Serverless

Code ausführen, ohne die ausführenden Server zu verwalten

Was es ist

Serverless ist eine Art, Code auszuführen: Das Team schreibt die Anwendung und stellt sie bereit, während ein Anbieter die darunterliegenden Maschinen betreibt, skaliert und pflegt. Das Whitepaper der CNCF Serverless Working Group verwendet den längeren Namen Serverless Computing und beschreibt die Idee, Anwendungen zu bauen und auszuführen, ohne dass jemand im Team einen Server verwalten muss.

Der Name übertreibt, und das Whitepaper sagt das ausdrücklich: „Für den Betrieb einer Serverless-Plattform werden weiterhin Server benötigt." Die Server existieren weiterhin, ebenso die Betriebsarbeit. Was sich ändert, ist, wer diese Arbeit erledigt. Das Team wählt keine Maschinengrößen mehr, installiert kein Betriebssystem, spielt keine Patches dafür ein und entscheidet nicht, wie viele Kopien laufen sollen. Es übergibt Code oder ein Containerimage und eine Beschreibung, wann dieser Code laufen soll.

Das Whitepaper teilt das Feld in zwei Bereiche. Functions-as-a-Service, kurz FaaS, ist ereignisgesteuerte Datenverarbeitung: kleine Logikbausteine, die als Reaktion auf ein Ereignis laufen. Der Auslöser einer Funktion heißt Trigger. Meist ist er das Ergebnis eines Ereignisses wie einer eingehenden HTTP-Anfrage oder einer Datenbankänderung. Der andere Bereich ist Backend-as-a-Service, kurz BaaS: Über eine API erreichbare Dienste Dritter ersetzen ganze Teile der Anwendungsfunktionalität. Funktionen sollen zustandslos und kurzlebig sein; der Zustand liegt deshalb auf der BaaS-Seite.

Zwei Produkte zeigen die Spannweite. AWS Lambda beschreibt sich als Serverless-Rechendienst, der Code ohne Bereitstellen oder Verwalten von Servern ausführt. Seine ursprüngliche Form ist die Ereignisfunktion: ein Handler mit einem Trigger. Azure Container Apps bezeichnet sich als Dienst, der

Anwendungscontainer ausführt, während der Anbieter die Infrastruktur betreibt. Das erste Produkt nimmt eine Funktion entgegen, das zweite einen ganzen Container. Beide überlassen die Maschinen dem Anbieter.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Serverless?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Server werden nach dem tatsächlichen Bedarf dimensioniert und in die Cloud oder aus ihr heraus verlagert — die monatlichen Kosten stehen vorher fest.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 20 Docker-Apps auf Google Cloud und AWS betrieben — 20 bereitgestellte und überwachte Containeranwendungen

20 Docker-Apps auf Google Cloud und AWS betrieben Kontakt aufnehmen →

Warum es wichtig ist

Das Abrechnungsmodell ist der erste Grund, aus dem Teams sich dafür interessieren. Das Whitepaper beschreibt Serverless-Preise als nutzungsabhängig: Bezahlt wird nur die Zeit, in der Code läuft; läuft nichts, fällt keine Gebühr an. Lambda berechnet für eine Ereignisfunktion die Anfragen und die verbrauchte Ausführungszeit in Gigabyte-Sekunden. Ein Dienst, der täglich eine Stunde beschäftigt ist, bezahlt diese Stunde statt eines Servers, der die anderen dreiundzwanzig Stunden wartet.

Der zweite Grund ist die Skalierung in beide Richtungen. Lambda erzeugt und entfernt Ausführungsumgebungen mit steigendem und sinkendem Verkehr. Das Whitepaper beschreibt Funktionen, deren Instanzzahl ohne Arbeit auf null fällt. Microsoft weist darauf hin, dass Container Apps anhand von HTTP-Verkehr, Ereignissen oder CPU- und Arbeitsspeicherlast skaliert und dass die meisten Anwendungen auf null skalieren können.

Der dritte Grund ist die Arbeit, die das Team abgibt. AWS nennt Serverwartung, Kapazitätsbereitstellung, Skalierung und Patches als Aufgaben, die Lambda übernimmt. Bei einem kleinen Team können solche Pflichten eine Woche füllen, ohne dass etwas ausgeliefert wird. Mit einer Serverless-Plattform liegen sie in der Verantwortung des Anbieters.

Die Anwendungsfälle des Whitepapers zeigen, wo das am besten passt: auf Datenbankänderungen reagieren, Sensoreingaben von Geräten verarbeiten, Datenströme bearbeiten und kurze geplante Aufgaben ausführen, die sonst ein Cronjob wären. Microsoft ergänzt für Container Apps API-Endpunkte, Hintergrundjobs und Microservices.

Dasselbe Whitepaper warnt auch davor, dass Serverless zunächst als günstigste Option erscheinen kann, und fordert Teams auf, vor einer Festlegung alle Kosten abzuwägen. Nutzungsabhängige Abrechnung begünstigt Arbeit, die in Schüben anfällt und die meiste Zeit ruht. Bei Arbeit, die eine Maschine rund um die Uhr beschäftigt, sieht die Rechnung anders aus. Auch die Fehlersuche kostet; laut Whitepaper kann sie schwieriger sein als bei IaaS- oder PaaS-Angeboten.

Wie es in der Praxis aussieht

Die AWS-Dokumentation für Python-Handler enthält ein Beispiel, das eine Bestellung empfängt, einen Textbeleg schreibt und ihn in einem Amazon-S3-Bucket speichert. Eine gekürzte Fassung sieht so aus:

```
import os

import boto3

# Runs in the Init phase, once per execution environment.
s3_client = boto3.client("s3")


def lambda_handler(event, context):
    order_id = event["Order_id"]
    bucket_name = os.environ.get("RECEIPT_BUCKET")
    receipt_content = (
        f"OrderID: {order_id}\n"
        f"Amount: ${event['Amount']}\n"
        f"Item: {event['Item']}"
    )
    key = f"receipts/{order_id}.txt"
    s3_client.put_object(Bucket=bucket_name, Key=key, Body=receipt_content)
    return {"statusCode": 200, "message": "Receipt processed successfully"}
```

Der Handler nimmt genau zwei Argumente entgegen. Das Ereignis trägt die zu verarbeitenden Daten. Ein JSON-Ereignis kommt als Python-Dictionary an, weshalb der Handler im Beispiel folgende Eingabeform erwartet:

```
{"Order_id": "12345", "Amount": 199.99, "Item": "Wireless Headphones"}
```

Der Kontext enthält Angaben zum Aufruf selbst. Heißt die Datei `lambda_function.py`, lautet der Handlername `lambda_function.lambda_handler`. Das ist die Voreinstellung der Lambda-Konsole für eine neue Funktion. Der Bucketname stammt aus einer Umgebungsvariablen statt aus dem Code. So empfiehlt AWS die Übergabe von Betriebseinstellungen. Der Code enthält keine Zugangsdaten: Lambda stellt sie dem SDK über eigene Umgebungsvariablen bereit.

Es kommt darauf an, wo der SDK-Client initialisiert wird. Bei nach Bedarf gestarteten Ereignisfunktionen ohne SnapStart oder Managed Instances durchläuft eine Ausführungsumgebung in Lambda drei Phasen: Init, Invoke und Shutdown. Code außerhalb des Handlers läuft einmal während Init. AWS begrenzt den ersten Init-Versuch auf 10 Sekunden. Danach läuft der Handler für jedes Ereignis innerhalb des Zeitlimits der Funktion. Wird eine Umgebung für das nächste Ereignis wiederverwendet, was AWS einen Warmstart nennt, ist der während Init erzeugte Client noch vorhanden und wird nicht erneut aufgebaut.

Serverless muss nicht Funktionen bedeuten. Auf dieser Website nennt der Portfolioeintrag zur Bereitstellung einer Azure-Landschaft als Code zwei in Azure Container Apps bereitgestellte Dienste, `api` und `www`. Ihre Containerimages liegen in der projekteigenen Azure Container Registry. Microsoft nennt Container Apps eine Serverless-Plattform für Container. Diese beiden Dienste sind deshalb der Serverless-Teil jener Landschaft. Jede Umgebung wird durch einen Satz parametrisierter Bicep-Vorlagen beschrieben. Deshalb kann der Eintrag eine neue Umgebung als Pull Request behandeln.

Auch der umgekehrte Weg kann richtig sein. Der Eintrag zur Infrastrukturprüfung und Verlagerung von Arbeitslasten weg von AWS beginnt mit einer AWS-Umgebung, die weit größer dimensioniert war als der tatsächliche Bedarf und weiterhin ungenutzte Dienste enthielt. Die ungenutzten Dienste wurden entfernt, die verbleibende Kapazität angepasst und die Arbeitslasten von AWS weg verlagert. Serverless-Abrechnung ist ein Weg, nicht für ungenutzte Kapazität zu zahlen; das Prüfen und Entfernen ungenutzter Infrastruktur ist ein anderer. Um zu entscheiden, was weniger kostet, müsste man die tatsächliche Last und die Kosten vergleichen. Der Eintrag dokumentiert Änderungen, kein unter gleichen Bedingungen gemessenes Kostenverhältnis.

Wo es schiefgeht

Die meisten Serverless-Fehler entstehen dadurch, dass eine Funktion wie ein kleiner, ständig laufender Server behandelt wird. Das ist sie nicht.

Kaltstarts verzögern manche Anfragen. Ist keine Umgebung bereit, muss Lambda den Code herunterladen, eine Umgebung einrichten und den Init-Code ausführen, bevor der Handler das Ereignis sieht. Laut AWS betreffen Kaltstarts gewöhnlich weniger als 1 % der Aufrufe. Sie dauern von unter 100 Millisekunden bis über eine Sekunde, treten häufiger in Entwicklungs- und Testfunktionen auf, und die Init-Zeit wird berechnet. Halten Sie Init klein, importieren Sie nur das Nötige und laden Sie selten verwendete Objekte erst bei Bedarf. Wo die Verzögerung zählt, hält provisionierte Nebenläufigkeit eine festgelegte Anzahl vorab initialisierter Umgebungen bereit.

Zustand in der Umgebung wird offengelegt oder verschwindet. Jeder Aufruf läuft unabhängig, und eine Umgebung wird möglicherweise wiederverwendet, möglicherweise auch nicht. Für Laufzeitupdates ersetzt AWS Umgebungen außerdem alle paar Stunden, selbst bei gleichbleibendem Verkehr. Ein Zurücksetzen nach einem Absturz oder Timeout leert das Verzeichnis `/tmp` nicht. Daten eines Ereignisses können deshalb bei einem anderen wieder auftauchen oder ohne Vorwarnung verschwinden. Halten Sie Benutzerdaten und Ereignisse vollständig aus der Umgebung heraus und legen Sie alles, was bestehen bleiben muss, in einer Datenbank oder einem Speicherdienst ab.

Dasselbe Ereignis wird zweimal verarbeitet. Eine Funktion kann Ereignisse doppelt empfangen. Die AWS-Liste bewährter Verfahren verlangt idempotenten Code, der jedes Ereignis validiert und ein Duplikat wie das Original behandelt. Im obigen Beispiel wird der Belegschlüssel aus der Bestell-ID abgeleitet. Ein zweiter Lauf überschreibt daher dasselbe Objekt, statt einen zweiten Beleg anzulegen. Jeder nicht sicher wiederholbare Seiteneffekt, etwa der Versand einer E-Mail oder eine Kartenbelastung, braucht einen eigenen Nachweis des bereits Erledigten.

Harte Grenzen beenden die Arbeit mittendrin. Bei Ereignisfunktionen außerhalb von Lambda Managed Instances darf ein Aufruf höchstens 900 Sekunden, also 15 Minuten, laufen. Eine synchrone Anfrage und eine gepufferte synchrone Antwort sind jeweils auf 6 MB begrenzt, eine asynchrone Anfrage auf 1 MB und Umgebungsvariablen insgesamt auf 4 KB. Eine Überschreitung führt zu einem Fehler statt zu einer Verlangsamung. Berücksichtigen Sie diese Grenzen

von Anfang an im Entwurf: Teilen Sie lange Jobs in Schritte auf, legen Sie große Daten in einem Datenspeicher ab und übergeben Sie einen Verweis darauf. Arbeit, die nicht hineinpasst, gehört auf eine Containerplattform oder eine virtuelle Maschine.

Mehr Anfragen kommen an, als gleichzeitig laufen dürfen. Lambdas Standardkontingent beträgt 1.000 gleichzeitige Ausführungen pro Region. API Gateway erlaubt standardmäßig 10.000 Anfragen pro Sekunde. Der Eingang kann also mehr Anfragen annehmen, als die dahinterliegenden Funktionen ausführen können. Neue Konten beginnen mit verringerten Kontingenten für Nebenläufigkeit und Arbeitsspeicher, die AWS anhand der Nutzung automatisch anhebt. Beantragen Sie eine Erhöhung vor dem Start und führen Sie einen Lasttest mit der erwarteten Spitzenlast durch.

Eine sich selbst auslösende Funktion treibt die Rechnung hoch. Eine Funktion, die sich selbst aufruft oder einen Prozess startet, der sie erneut aufruft, kann eine Flut von Aufrufen erzeugen. Jeder wird berechnet. AWS rät, rekursive Aufrufe zu vermeiden und bei unerwartetem Volumen die reservierte Nebenläufigkeit der Funktion sofort auf `0` zu setzen. Das unterbindet weitere Ausführungen, bis der Code behoben ist.

Das Programmiermodell bindet an eine Plattform. Das Whitepaper nennt das Programmiermodell der Plattform, ihre Ereignis- und Nachrichtenschnittstelle und ihre BaaS-Angebote als Ursachen einer Anbieterbindung. Halten Sie die Geschäftslogik in gewöhnlichen Funktionen, die der Handler lediglich aufruft. Ist ein späterer Wechsel eine reale Möglichkeit, bevorzugen Sie Plattformen wie Container Apps, die ein Standard-Containerimage aus jeder Registry ausführen.

Weiterführende Lektüre

- CNCF Serverless Whitepaper v1.0 — Definitionen der Arbeitsgruppe für Serverless, FaaS und BaaS sowie Nachteile.
- Was ist AWS Lambda? — Von Lambda übernommene Aufgaben, Funktionstypen und Abrechnungsmodell.
- Lambda-Ereignisfunktionshandler in Python definieren — Vollständiges Belegbeispiel, Handlernamen und bewährte Verfahren für den Code.
- Lebenszyklus der Lambda-Ausführungsumgebung — Die Phasen Init, Invoke und Shutdown, Kaltstarts und Wiederverwendung der Umgebung.
- Lambda-Kontingente — Grenzen für Nebenläufigkeit, Arbeitsspeicher, Laufzeit und Nutzdaten sowie ihre Erhöhung.

- Überblick zu Azure Container Apps — Microsofts Serverless-Plattform für Container und ihre Skalierung auf null.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Serverless?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/serverless/