Engineer

serverless

at køre kode uden selv at drive serverne under den

Hvad det er

Serverless er en måde at køre kode på, hvor teamet skriver og udruller applikationen, mens en udbyder driver, skalerer og vedligeholder maskinerne under den. CNCF's Serverless Working Group bruger i sin whitepaper den længere betegnelse serverless computing og beskriver det som at bygge og køre applikationer, uden at nogen på teamet skal administrere en server.

Navnet er en overdrivelse, og det siger whitepaperen selv ligeud: Der skal stadig servere til at køre en serverless platform. Serverne findes stadig, og det gør driftsarbejdet også. Forskellen er, hvem der udfører det. Teamet vælger ikke længere maskinstørrelser, installerer ikke styresystemet, opdaterer det ikke og beslutter ikke, hvor mange kopier der skal køre. Det afleverer kode eller et container-image og en beskrivelse af, hvornår koden skal køre.

Whitepaperen deler feltet i to. Functions-as-a-service, forkortet FaaS, er hændelsesdrevet databehandling: små stykker logik, der kører, når noget sker. Det, der starter en funktion, kaldes en trigger, og den er som regel resultatet af en hændelse, for eksempel en indkommende HTTP-forespørgsel eller en ændring i en database. Backend-as-a-service, forkortet BaaS, er den anden del: tredjepartstjenester, som man når gennem et API, og som erstatter hele dele af en applikations funktionalitet. Funktionerne forventes at være tilstandsløse og kortlivede, så det er på BaaS-siden, tilstanden bor.

To produkter viser spændvidden. AWS Lambda beskriver sig selv som en serverless computetjeneste, der kører kode, uden at man skal klargøre eller administrere servere, og dens oprindelige form er event-funktionen: en handler plus en trigger. Azure Container Apps kalder sig en serverless platform til at køre

containeriserede applikationer uden at skulle administrere den underliggende infrastruktur. Den første tager imod en funktion, den anden en hel container, og begge overlader maskinerne til udbyderen.

Brug begrebet i en beslutning

Brug »Hvad er serverless?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Servere dimensioneret efter det, I rent faktisk bruger, flyttet op i skyen eller hjem igen, med den månedlige regning regnet ud først.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har drevet 20 Docker-apps på Google Cloud og AWS — 20 containeriserede applikationer udrullet og overvåget

Har drevet 20 Docker-apps på Google Cloud og AWS Kontakt os →

Hvorfor det betyder noget

Afregningsmodellen er den første grund til, at teams kigger på det. Whitepaperen beskriver prissætningen som pay-as-you-go, hvor man kun betaler for den tid, koden kører, og ikke betaler noget, mens den står stille. Lambda afregner en event-funktion pr. forespørgsel plus den kørselstid, den bruger, målt i gigabytesekunder. En tjeneste, der har travlt en time om dagen, betaler for den time og ikke for en server, der venter de øvrige 23.

Den anden grund er skalering i begge retninger. Lambda opretter og nedlægger kørselsmiljøer, efterhånden som trafikken stiger og falder, og whitepaperen beskriver funktioner, hvis antal instanser falder til nul, når der ikke er arbejde. Microsoft skriver, at Container Apps skalerer på HTTP-trafik, hændelser eller CPU- og hukommelsesbelastning, og at de fleste applikationer kan skalere helt ned til nul.

Den tredje grund er det arbejde, der forsvinder fra teamets bord. AWS nævner servervedligeholdelse, klargøring af kapacitet, skalering og patching blandt de opgaver, Lambda klarer for en. For et lille team er det netop de gøremål, der kan fylde en hel uge, uden at noget bliver leveret, og på en serverless platform bliver de udbyderens ansvar.

Whitepaperens anvendelseseksempler viser, hvor modellen hører hjemme: at reagere på ændringer i en database, behandle sensordata fra enheder, håndtere datastrømme og korte planlagte opgaver af den slags, der ellers ville være et cronjob. Microsofts liste for Container Apps tilføjer API-endepunkter, baggrundsjob og mikrotjenester.

Den samme whitepaper advarer også om, at serverless kan se ud som den billigste løsning i starten, og beder teams om at se på hele omkostningsbilledet, før de lægger sig fast. Forbrugsafregning belønner arbejde, der kommer i ryk og står stille det meste af tiden. Arbejde, der holder en maskine beskæftiget døgnet rundt, er et andet regnestykke, og fejlfinding, som ifølge whitepaperen kan være sværere end på IaaS og PaaS, er også en omkostning.

Sådan ser det ud i praksis

AWS' dokumentation for handlere i Python har et eksempel, der modtager en ordre, skriver en kvittering som tekst og gemmer den i en Amazon S3-bucket. En forkortet udgave ser sådan ud:

```
import os

import boto3

# Kører i Init-fasen, én gang pr. kørselsmiljø.
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"}
```

Handleren tager præcis to argumenter. Det første indeholder de data, der skal behandles, og en hændelse i JSON ankommer som en Python-dictionary, så denne funktion forventer input med følgende form:

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

Det andet argument, kontekstobjektet, indeholder oplysninger om selve kaldet. Hedder filen `lambda_function.py`, får handleren navnet `lambda_function.lambda_handler`, som er det standardnavn, Lambda-konsollen giver

en ny funktion. Bucketnavnet kommer fra en miljøvariabel og ikke fra koden, sådan som AWS anbefaler at sende driftsindstillinger ind, og koden indeholder ingen legitimationsoplysninger: Lambda giver dem til SDK'et gennem sine egne miljøvariabler.

Placeringen af klienten er vigtig. For hændelsesfunktioner med on-demand-kørsel uden SnapStart eller Managed Instances fører Lambda et kørselsmiljø gennem tre faser: Init, Invoke og Shutdown. Kode uden for handleren kører én gang under Init, og AWS begrænser det første Init-forsøg til 10 sekunder. Derefter kører handleren for hver hændelse inden for funktionens tidsgrænse. Når et miljø genbruges til den næste hændelse, det AWS kalder en warm start, findes klienten fra Init stadig og bygges ikke igen.

Serverless behøver ikke at betyde funktioner. Porteføljeindlægget om Azure-infrastruktur provisioneret som kode nævner to tjenester, `api` og `www`, som kører på Azure Container Apps, og hvis containerimages ligger i projektets eget Azure Container Registry. Microsoft kalder Container Apps en serverless platform til containere, så det er de to tjenester, der udgør den serverless del af opsætningen. Hvert miljø beskrives af ét sæt Bicep-templates med parametre, og derfor kan indlægget behandle et nyt miljø som en pull request.

Den modsatte vej kan også være den rigtige. Indlægget om at gennemgå infrastrukturen og flytte workloads væk fra AWS begynder med en AWS-opsætning, der var langt større end forretningens behov, og som stadig bar rundt på tjenester, ingen brugte. De ubrugte tjenester blev fjernet, den resterende kapacitet blev tilpasset, og workloads blev flyttet væk fra AWS. Serverless-afregning er én måde at slippe for at betale for ledig kapacitet; at gennemgå og fjerne ubrugt infrastruktur er en anden. En sammenligning af den reelle belastning og omkostningerne ville være nødvendig for at afgøre, hvad der koster mindst; indlægget dokumenterer ændringer frem for et omkostningsforhold opgjort på sammenligneligt grundlag.

Hvor det går galt

De fleste fejl med serverless kommer af at behandle en funktion som en lille server, der altid er tændt, hvad den ikke er.

Koldstarter forsinker nogle forespørgsler. Når der ikke står et miljø klar, skal Lambda hente koden, sætte et miljø op og køre Init-koden, før handleren ser hændelsen. Ifølge AWS rammer koldstarter typisk under 1 % af kaldene og varer

fra under 100 millisekunder til over et sekund; de er hyppigere i udviklings- og testfunktioner, og Init-tiden afregnes. Løsningen er at holde Init lille, kun importere det, funktionen har brug for, og først indlæse sjældent brugte objekter, når de skal bruges. Hvor forsinkelsen betyder noget, holder provisioned concurrency et valgt antal miljøer initialiseret på forhånd.

Tilstand i miljøet lækker eller forsvinder. Hvert kald kører uafhængigt, og et miljø bliver måske genbrugt, måske ikke. AWS udskifter desuden miljøer med få timers mellemrum for at opdatere runtime, selv under konstant trafik, og en nulstilling efter et nedbrud eller en timeout rydder ikke mappen `/tmp`. Data fra én hændelse kan derfor dukke op under en anden eller forsvinde uden varsel. Løsningen er at holde brugerdata og hændelser helt ude af miljøet og lægge alt, der skal bevares, i en database eller en lagringstjeneste.

Den samme hændelse behandles to gange. En funktion kan modtage dublerede hændelser, og AWS' liste over god praksis beder om idempotent kode, der validerer hver hændelse og behandler en dublet på samme måde som originalen. I eksemplet ovenfor er kvitteringens nøgle afledt af ordrenummeret, så en ny kørsel overskriver det samme objekt i stedet for at lave endnu en kvittering. Enhver sideeffekt, der ikke tåler at blive gentaget, som at sende en e-mail eller trække penge på et kort, har brug for sin egen registrering af, hvad der allerede er gjort.

Hårde grænser stopper arbejdet midtvejs. For hændelsesfunktioner uden Lambda Managed Instances kan et kald højst køre i 900 sekunder, altså 15 minutter. En synkron forespørgsel og et bufferet synkront svar er hver begrænset til 6 MB, en asynkron forespørgsel til 1 MB og miljøvariabler til 4 KB i alt. At overskride en grænse giver en fejl, ikke en opbremsning. Løsningen er at designe efter grænserne fra begyndelsen: dele lange job op i trin, gemme store datamængder i et lager og sende en reference videre, og flytte arbejde, som grænserne ikke rummer, til en containerplatform eller en virtuel maskine.

Samtidigheden slipper op før trafikken. Lambdas standardkvote er 1.000 samtidige kørsler pr. region, mens API Gateway som standard tillader 10.000 forespørgsler i sekundet, så indgangen kan tage imod flere forespørgsler, end funktionerne bag den kan køre. Nye konti starter med nedsatte kvoter for samtidighed og hukommelse, som AWS hæver automatisk ud fra forbruget. Løsningen er at bede om en forhøjelse før lanceringen og at belastningsteste ved den forventede spidsbelastning.

En funktion, der starter sig selv, får regningen til at løbe løbsk. En funktion, der kalder sig selv eller sætter en proces i gang, som kalder den igen, kan udløse en strøm af kald, og hvert eneste afregnes. AWS råder til at undgå rekursive kald og, hvis der pludselig kommer en uventet mængde, straks at sætte funktionens reserved concurrency til `0`, hvilket stopper alle kald, indtil koden er rettet.

Programmeringsmodellen binder en til én platform. Whitepaperen peger på platformens programmeringsmodel, dens grænseflade til hændelser og beskeder og dens BaaS-tilbud som kilder til fastlåsning. Løsningen er at holde forretningslogikken i almindelige funktioner, som handleren blot kalder, og at foretrække platforme som Container Apps, der kører et standard container-image fra et vilkårligt registry, når en senere flytning er en reel mulighed.

Læs mere

- CNCF Serverless Whitepaper v1.0 — arbejdsgruppens definitioner af serverless, FaaS og BaaS samt ulemperne.
- What is AWS Lambda? — hvad Lambda tager sig af, dens funktionstyper og dens afregningsmodel.
- Define Lambda Event Function handler in Python — hele kvitteringseksemplet, navngivning af handlere og god praksis for koden.
- Understanding the Lambda execution environment lifecycle — faserne Init, Invoke og Shutdown, koldstarter og genbrug af miljøer.
- Lambda quotas — grænser for samtidighed, hukommelse, timeout og payload, og hvordan de øges.
- Azure Container Apps overview — Microsofts serverless platform til containere, der kan skalere til nul.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er serverless?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/serverless/