Engineer

serverless

🌩 запуск коду без керування серверами, на яких він працює

Що це таке

Serverless — це спосіб запускати код, за якого команда пише й розгортає застосунок, а провайдер експлуатує, масштабує й обслуговує машини під ним. Українською цей підхід часто називають безсерверними обчисленнями, а whitepaper робочої групи CNCF Serverless Working Group вживає довшу назву serverless computing і описує її як ідею створювати й запускати застосунки так, щоб нікому в команді не доводилося керувати сервером.

Назва перебільшує, і whitepaper прямо це визнає: щоб serverless-платформа працювала, сервери однаково потрібні. Сервери нікуди не зникають, як і робота з їх експлуатації. Змінюється лише те, хто її виконує. Команда більше не обирає розміри машин, не встановлює операційну систему, не латає її і не вирішує, скільки копій запускати. Вона передає код або образ контейнера разом з описом того, коли цей код має виконуватися.

Whitepaper поділяє цю галузь на дві частини. Functions-as-a-service, або FaaS, — це обчислення, керовані подіями: невеликі фрагменти логіки, які запускаються у відповідь на те, що відбулося. Те, що запускає функцію, називають тригером, і зазвичай це наслідок події, як-от вхідного HTTP-запиту чи зміни в базі даних. Backend-as-a-service, або BaaS, — друга частина: сторонні сервіси, до яких звертаються через API і які замінюють цілі частини функціональності застосунку. Від функцій очікують, що вони не зберігатимуть стану й житимуть недовго, тож стан живе саме на боці BaaS.

Два продукти показують діапазон. AWS Lambda описує себе як serverless-сервіс обчислень, що запускає код без потреби готувати чи адмініструвати сервери, і початкова форма цього сервісу — функція подій (event function): обробник плюс тригер. Azure Container Apps називає себе serverless-

платформою для контейнеризованих застосунків, де не треба керувати інфраструктурою під ними. Перший сервіс приймає функцію, другий — цілий контейнер, і обидва залишають машини провайдерові.

Застосуйте поняття до рішення

Застосуйте «Що таке serverless?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Сервери під те навантаження, яке у вас справді є, перенесені в хмару чи назад, із порахованим наперед місячним рахунком.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Вела 20 застосунків Docker у Google Cloud і AWS — 20 контейнеризованих застосунків розгорнуто й поставлено під моніторинг

Вела 20 застосунків Docker у Google Cloud і AWS Зв'язатися з нами →

Чому це важливо

Модель оплати — перша причина, чому команди звертають на це увагу. Whitepaper описує ціноутворення serverless як оплату за фактичне використання: платять лише за час, поки код працює, а в періоди бездіяльності не платять нічого. Lambda виставляє рахунок за функцію подій за кожен запит плюс за час виконання, який вимірюють у гігабайт-секундах. Сервіс, зайнятий годину на добу, платить за цю годину, а не за сервер, що чекає решту двадцять три години.

Друга причина — масштабування в обидва боки. Lambda створює й знищує середовища виконання, коли трафік зростає чи спадає, а whitepaper описує функції, кількість екземплярів яких падає до нуля, коли роботи немає. Microsoft зазначає, що Container Apps масштабується за HTTP-трафіком, подіями або навантаженням на процесор і пам’ять і що більшість застосунків може масштабуватися до нуля.

Третя причина — робота, яка йде з рук команди. AWS перелічує обслуговування серверів, виділення потужностей, масштабування й встановлення патчів серед завдань, які Lambda виконує за вас. Для невеликої команди саме ці клопоти здатні з’їсти цілий тиждень, за який так і не вийде жодного релізу, а на serverless-платформі вони стають відповідальністю провайдера.

Сценарії з whitepaper показують, де ця модель доречна найбільше: реагувати на зміни в базі даних, обробляти дані з датчиків пристроїв, працювати з потоками даних і виконувати короткі заплановані завдання, які інакше були б cron-задачами. Список Microsoft для Container Apps додає кінцеві точки API, фонові завдання й мікросервіси.

Той самий whitepaper застерігає, що спершу serverless може видаватися найвигіднішим варіантом, і радить зважити всі витрати, перш ніж на нього переходити. Оплата за використання вигідна для навантаження, яке приходить сплесками й більшу частину часу простоює. Робота, що тримає машину зайнятою цілодобово, — зовсім інший розрахунок, а налагодження, яке, за словами whitepaper, буває складнішим, ніж на IaaS чи PaaS, теж коштує грошей.

Як це виглядає на практиці

У документації AWS про обробники на Python є приклад, який отримує замовлення, формує текстову квитанцію й зберігає її в бакеті Amazon S3. Скорочена версія виглядає так:

```
import os

import boto3

# Виконується у фазі Init, один раз на середовище виконання.
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"}
```

Обробник приймає рівно два аргументи. Перший, подія, містить дані для обробки, а подія у форматі JSON надходить як словник Python, тож ця функція очікує вхідні дані такого вигляду:

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

Другий аргумент, об'єкт контексту, містить відомості про сам виклик. Якщо файл називається `lambda_function.py`, обробник отримує ім'я `lambda_function.lambda_handler` — саме таке ім'я консоль Lambda за замовчуванням дає новій функції. Назва бакета береться зі змінної середовища, а не з коду, як AWS і радить передавати робочі налаштування, а облікових даних у коді немає зовсім: Lambda передає їх SDK через власні змінні середовища.

Важливо, де створено клієнт. Для функцій обробки подій, що запускаються на вимогу без SnapStart або Managed Instances, Lambda проводить середовище виконання через три фази: Init, Invoke і Shutdown. Код поза обробником виконується один раз, під час Init, і AWS обмежує тривалість першої спроби Init 10 секундами. Далі обробник виконується для кожної події в межах тайм-ауту функції. Коли середовище використовують повторно для наступної події (AWS називає це теплим стартом), клієнт, створений під час Init, уже на місці, і створювати його повторно не треба.

Serverless не обов'язково означає функції. Кейс із портфоліо про інфраструктуру Azure, розгорнуту як код називає два сервіси, `api` і `www`, розміщені в Azure Container Apps; їхні образи контейнерів зберігаються у власному Azure Container Registry проєкту. Microsoft називає Container Apps serverless-платформою для контейнерів, тож саме ці два сервіси й є serverless-частиною тієї інфраструктури. Кожне середовище описує один набір параметризованих шаблонів Bicep, тому нове середовище там додають через pull request.

Протилежне рішення теж буває слушним. Запис про перевірку інфраструктури та перенесення навантажень із AWS починається з конфігурації AWS, розрахованої на значно більше, ніж використовував бізнес, і сервісів, які вже нікому не були потрібні. Невикористані сервіси прибрали, потужність решти скоригували, а навантаження перенесли з AWS. Serverless-модель оплати — один зі способів не платити за потужності, що простоюють; перевірка й видалення невикористаної інфраструктури — інший. Щоб визначити, який спосіб дешевший, потрібне порівняння реального навантаження та витрат; запис фіксує зміни, а не співвідношення витрат за однакових умов.

Де виникають проблеми

Більшість невдач із serverless походить з того, що функцію сприймають як маленький сервер, який завжди ввімкнений, а вона ним не є.

Холодні старти затримують частину запитів. Коли готового середовища немає, Lambda мусить завантажити код, підготувати середовище й виконати код Init, перш ніж обробник побачить подію. За даними AWS, холодні старти зазвичай стосуються менш ніж 1 % викликів і тривають від менш ніж 100 мілісекунд до понад секунди; у функціях для розробки й тестування вони трапляються частіше, а час Init оплачується. Рішення — тримати Init невеликим: імпортувати лише те, що потрібно функції, а рідко вживані об'єкти створювати тоді, коли вони знадобляться. Там, де затримка критична, provisioned concurrency заздалегідь тримає ініціалізованою обрану кількість середовищ.

Стан усередині середовища витікає або зникає. Кожен виклик виконується незалежно, а середовище можуть використати повторно, а можуть і ні. Крім того, AWS кожні кілька годин замінює середовища, щоб оновити runtime, навіть за сталого трафіку, а скидання після збою чи тайм-ауту не очищує каталог `/tmp` . Тож дані однієї події можуть випливти під час іншої або безслідно зникнути. Рішення — взагалі не тримати дані користувачів і подій у середовищі, а все, що має зберегтися, класти в базу даних або сервіс зберігання.

Одна й та сама подія обробляється двічі. Функція може отримувати дублікати подій, і в переліку найкращих практик AWS просить писати ідемпотентний код, який перевіряє кожну подію й обробляє дублікат так само, як оригінал. У прикладі вище ключ квитанції утворено з номера замовлення, тож повторний запуск перезапише той самий об'єкт, а не створить другу квитанцію. Будь-який побічний ефект, який не можна безпечно повторити, як-от надсилання листа чи списання коштів із картки, потребує власного запису про те, що вже зроблено.

Жорсткі обмеження обривають роботу на півдорозі. Для функцій обробки подій поза Lambda Managed Instances виклик може тривати щонайбільше 900 секунд, тобто 15 хвилин. Синхронний запит і буферизована синхронна відповідь мають обмеження по 6 МБ, асинхронний запит — 1 МБ, а змінні середовища — 4 КБ загалом. Вихід за межу — це помилка, а не уповільнення. Рішення — від самого початку проєктувати з урахуванням

обмежень: ділити довгі завдання на кроки, зберігати великі обсяги даних у сховищі й передавати посилання на них, а роботу, яка не вкладається в обмеження, переносити на контейнерну платформу або у віртуальну машину.

Ліміт одночасних виконань вичерпується раніше, ніж спадає трафік. За замовчуванням квота Lambda становить 1 000 одночасних виконань на регіон, тоді як API Gateway за замовчуванням пропускає 10 000 запитів на секунду, тож вхідні двері можуть прийняти більше запитів, ніж функції за ними здатні виконати. Нові облікові записи починають зі зменшених квот на паралельність і пам'ять, які AWS підвищує автоматично залежно від використання. Рішення — попросити підвищення квоти до запуску й провести навантажувальне тестування на очікуваному піку.

Функція, що запускає сама себе, роздуває рахунок. Функція, яка викликає сама себе або запускає процес, що знову її викликає, може породити лавину викликів, і кожен із них оплачується. AWS радить уникати рекурсивних викликів, а якщо раптом з'явиться неочікуваний обсяг, негайно встановити для функції reserved concurrency `0`, що зупиняє всі виклики, доки код не виправлено.

Модель програмування прив'язує до однієї платформи. Whitepaper називає модель програмування платформи, її інтерфейс подій і повідомлень та її BaaS-пропозиції джерелами прив'язки до постачальника. Рішення — тримати бізнес-логіку у звичайних функціях, які обробник лише викликає, і, якщо переїзд згодом цілком можливий, віддавати перевагу платформам на кшталт Container Apps, які запускають стандартний образ контейнера з будь-якого реєстру.

Що почитати далі

- CNCF Serverless Whitepaper v1.0 — визначення serverless, FaaS і BaaS від робочої групи та недоліки підходу.
- What is AWS Lambda? — що Lambda бере на себе, які в неї типи функцій і як вона тарифікується.
- Define Lambda Event Function handler in Python — повний приклад із квитанцією, іменування обробників і найкращі практики.
- Understanding the Lambda execution environment lifecycle — фази Init, Invoke і Shutdown, холодні старти й повторне використання.

- Lambda quotas — ліміти паралельності, пам’яті, тайм-ауту й розміру даних і їх підвищення.
- Azure Container Apps overview — serverless-платформа Microsoft для контейнерів і її масштабування до нуля.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке serverless?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/serverless/