Engineer

serverless

Ejecutar código sin gestionar los servidores que lo ejecutan

Qué es

Serverless es una forma de ejecutar código en la que el equipo escribe y despliega la aplicación, y un proveedor opera, escala y mantiene las máquinas subyacentes. El libro blanco del grupo de trabajo Serverless de la CNCF, que utiliza el nombre más largo «computación serverless», lo describe como la idea de construir y ejecutar aplicaciones sin que nadie del equipo tenga que gestionar un servidor.

El nombre exagera, y el propio libro blanco lo dice con claridad: «Siguen haciendo falta servidores para ejecutar una plataforma serverless». Los servidores siguen existiendo, al igual que el trabajo de operaciones. Lo que cambia es quién lo hace. El equipo ya no elige el tamaño de las máquinas, ni instala el sistema operativo, ni le aplica parches, ni decide cuántas copias ejecutar. Entrega código o una imagen de contenedor y una descripción de cuándo debe ejecutarse ese código.

El libro blanco divide el ámbito en dos partes. Las funciones como servicio, o FaaS, son computación dirigida por eventos: pequeñas unidades de lógica que se ejecutan en respuesta a algo que sucede. Lo que inicia una función se llama disparador y suele ser el resultado de un evento, como una solicitud HTTP entrante o un cambio en una base de datos. El backend como servicio, o BaaS, es la otra parte: servicios de terceros accesibles mediante una API que sustituyen partes enteras de la funcionalidad de una aplicación. Se espera que las funciones carezcan de estado y duren poco, por lo que el estado reside en la parte BaaS.

Dos productos muestran la amplitud del concepto. AWS Lambda se describe como un servicio de computación serverless que ejecuta código sin aprovisionar ni gestionar servidores. Su forma original es la función de eventos: un manejador y un disparador. Azure Container Apps se presenta como un servicio que ejecuta

contenedores de aplicaciones mientras el proveedor opera la infraestructura. El primero recibe una función; el segundo, un contenedor completo. Ambos dejan las máquinas en manos del proveedor.

Utilice el concepto en una decisión

Aplique «¿Qué es serverless?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Servidores dimensionados para el uso real, trasladados a la nube o fuera de ella, con la factura mensual calculada de antemano.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 20 aplicaciones Docker en Google Cloud y AWS — 20 aplicaciones en contenedores desplegadas y monitorizadas

20 aplicaciones Docker en Google Cloud y AWS Póngase en contacto →

Por qué importa

El modelo de facturación es la primera razón por la que los equipos se interesan por él. El libro blanco describe los precios serverless como pago por uso: solo se cobra por el tiempo durante el que se ejecuta el código, sin cargo cuando no hay nada en ejecución. Lambda factura una función de eventos por solicitud y por el tiempo de ejecución que consume, medido en gigabytes-segundo. Un servicio ocupado una hora al día paga por esa hora, no por un servidor esperando durante las otras veintitrés.

La segunda razón es el escalado en ambas direcciones. Lambda crea y destruye entornos de ejecución a medida que el tráfico aumenta o disminuye, y el libro blanco describe funciones cuyas instancias se reducen a cero cuando no hay trabajo. Microsoft señala que Container Apps escala según el tráfico HTTP, los eventos o la carga de CPU y memoria, y que la mayoría de las aplicaciones pueden reducirse a cero.

La tercera razón es el trabajo que deja de asumir el equipo. AWS enumera el mantenimiento de servidores, el aprovisionamiento de capacidad, el escalado y la aplicación de parches entre las tareas de las que Lambda se encarga. Para un equipo pequeño, son las labores que llenan una semana sin entregar nada. Con una plataforma serverless pasan a ser responsabilidad del proveedor.

Los casos de uso del libro blanco muestran dónde encaja mejor: reaccionar a cambios en bases de datos, procesar lecturas de sensores de dispositivos, gestionar flujos de datos y ejecutar tareas programadas breves que de otro

modo serían trabajos de cron. La lista de Microsoft para Container Apps añade puntos de acceso de API, trabajos de procesamiento en segundo plano y microservicios.

El mismo libro blanco también advierte de que serverless puede parecer al principio la opción más económica y pide evaluar todos los costes antes de comprometerse. La facturación por uso favorece las cargas de trabajo que llegan en ráfagas y permanecen inactivas la mayor parte del tiempo. El trabajo que mantiene una máquina ocupada a todas horas exige otro cálculo. La depuración, que según el libro blanco puede resultar más difícil que en ofertas IaaS o PaaS, también tiene un coste.

Cómo se aplica en la práctica

La documentación de AWS sobre manejadores de Python incluye un ejemplo que recibe un pedido, escribe un recibo de texto y lo almacena en un bucket de Amazon S3. Esta es una versión abreviada:

```
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"}
```

El manejador recibe exactamente dos argumentos. El evento contiene los datos que se deben procesar, y un evento JSON llega como un diccionario de Python. Por tanto, el manejador del ejemplo espera una entrada de la siguiente forma:

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

El contexto contiene información sobre la propia invocación. Si el archivo se llama `lambda_function.py`, el manejador se denomina `lambda_function.lambda_handler`, el nombre predeterminado que la consola de Lambda asigna a una función nueva. El nombre del bucket procede de una variable de entorno, no del código, que es la forma en que AWS recomienda pasar los parámetros de operación. El código no contiene credenciales: Lambda las proporciona al SDK mediante sus propias variables de entorno.

Importa dónde se inicializa el cliente del SDK. Para funciones de eventos bajo demanda sin SnapStart ni Managed Instances, Lambda hace pasar un entorno de ejecución por tres fases: Init, Invoke y Shutdown. El código situado fuera del manejador se ejecuta una vez, durante Init, y AWS limita el intento inicial de Init a 10 segundos. Después, el manejador se ejecuta para cada evento, dentro del límite de tiempo de la función. Cuando un entorno se reutiliza para el siguiente evento, lo que AWS llama un arranque en caliente, el cliente creado durante Init sigue ahí y no se vuelve a construir.

Serverless no tiene por qué significar funciones. En este sitio, la entrada del portfolio sobre aprovisionar una infraestructura Azure como código enumera dos servicios, `api` y `www`, desplegados en Azure Container Apps, con sus imágenes de contenedor guardadas en el Azure Container Registry del propio proyecto. Microsoft llama a Container Apps plataforma serverless para contenedores: esos dos servicios son, por tanto, la parte serverless de esa infraestructura. Cada entorno se describe mediante un mismo conjunto de plantillas Bicep parametrizadas, lo que permite tratar un entorno nuevo como una solicitud de incorporación de cambios.

Recorrer el camino contrario también puede ser lo adecuado. La entrada sobre revisar la infraestructura y sacar cargas de trabajo de AWS parte de una instalación AWS dimensionada muy por encima del uso real del negocio y que seguía manteniendo servicios que nadie necesitaba. Se eliminaron los servicios sin uso, se ajustó la capacidad restante y se trasladaron las cargas fuera de AWS. La facturación serverless es una forma de dejar de pagar por capacidad inactiva; revisar y eliminar infraestructura no utilizada es otra. Haría falta comparar la carga y los costes reales para determinar qué opción cuesta menos. La entrada documenta los cambios, no una proporción de costes entre situaciones equivalentes.

Dónde surgen los problemas

La mayoría de los fallos serverless nacen de tratar una función como un pequeño servidor siempre encendido, cuando no lo es.

Los arranques en frío añaden demora a algunas solicitudes. Cuando no hay un entorno preparado, Lambda debe descargar el código, configurar un entorno y ejecutar el código de Init antes de que el manejador vea el evento. AWS dice que los arranques en frío suelen afectar a menos del 1 % de las invocaciones y duran desde menos de 100 milisegundos hasta más de un segundo, que son más frecuentes en funciones de desarrollo y pruebas, y que el tiempo de Init se factura. La solución es mantener ligera la fase Init, importar solo lo que necesita la función y cargar de forma diferida los objetos que se usan poco. Cuando la demora importa, la concurrencia aprovisionada mantiene de antemano un número elegido de entornos inicializados.

El estado guardado dentro del entorno se filtra o desaparece. Cada invocación se ejecuta de forma independiente, y un entorno puede reutilizarse o no. AWS también sustituye los entornos cada pocas horas para actualizar el runtime, incluso con tráfico constante, y un reinicio tras un fallo o un tiempo de espera agotado no vacía el directorio `/tmp` . Los datos de un evento pueden, por tanto, aparecer durante otro o desaparecer sin aviso. La solución es no conservar en el entorno datos de usuario ni eventos y guardar todo lo que deba persistir en una base de datos o un servicio de almacenamiento.

El mismo evento se procesa dos veces. Una función puede recibir eventos duplicados. La lista de buenas prácticas de AWS pide código idempotente que valide cada evento y trate los duplicados igual que el original. En el ejemplo anterior, la clave del recibo se deriva del identificador del pedido: una segunda ejecución sobrescribe el mismo objeto en lugar de crear un segundo recibo. Cada efecto secundario que no pueda repetirse con seguridad, como enviar un correo o cobrar una tarjeta, necesita su propio registro de lo ya realizado.

Los límites estrictos interrumpen el trabajo a medias. Para las funciones de eventos que no usan Lambda Managed Instances, una invocación puede durar como máximo 900 segundos, o 15 minutos. Una solicitud síncrona y una respuesta síncrona con búfer tienen un límite de 6 MB cada una; una solicitud asíncrona, de 1 MB; y las variables de entorno, de 4 KB en total. Superar un límite provoca un error, no una ralentización. La solución es diseñar desde el principio

teniendo en cuenta esos límites: dividir los trabajos largos en pasos, mantener los datos grandes en almacenamiento y pasar una referencia a ellos, y trasladar el trabajo que no encaje a una plataforma de contenedores o una máquina virtual.

La concurrencia se agota antes que el tráfico. La cuota predeterminada de Lambda es de 1.000 ejecuciones simultáneas por región, mientras que API Gateway admite por defecto 10.000 solicitudes por segundo. La entrada puede aceptar así más solicitudes de las que pueden ejecutar las funciones que hay detrás. Las cuentas nuevas empiezan con cuotas reducidas de concurrencia y memoria, que AWS aumenta automáticamente según el uso. La solución es solicitar un aumento antes del lanzamiento y probar con la carga máxima prevista.

Una función que se dispara a sí misma dispara la factura. Una función que se invoca a sí misma o inicia un proceso que vuelve a invocarla puede producir una avalancha de invocaciones, y se cobra cada una. AWS aconseja evitar las invocaciones recursivas. Si aparece un volumen inesperado, recomienda fijar inmediatamente la concurrencia reservada de la función en `0`, lo que impide nuevas ejecuciones hasta que se corrija el código.

El modelo de programación crea dependencia de una sola plataforma. El libro blanco señala el modelo de programación de la plataforma, su interfaz de eventos y mensajería y sus ofertas BaaS como causas de dependencia del proveedor. La solución es mantener la lógica del negocio en funciones normales que el manejador se limite a invocar y, cuando cambiar de plataforma más adelante sea una posibilidad real, preferir opciones como Container Apps, que ejecutan una imagen de contenedor estándar de cualquier registro.

Lecturas adicionales

- Libro blanco de Serverless de la CNCF, versión 1.0 — las definiciones de serverless, FaaS y BaaS del grupo de trabajo, y sus inconvenientes.
- ¿Qué es AWS Lambda? — qué gestiona Lambda por usted, sus tipos de función y su modelo de facturación.
- Definir un manejador de función de eventos de Lambda en Python — el ejemplo completo del recibo, el nombre del manejador y las buenas prácticas de código.
- Comprender el ciclo de vida del entorno de ejecución de Lambda — las fases Init, Invoke y Shutdown, los arranques en frío y la reutilización del entorno.

- Cuotas de Lambda — límites de concurrencia, memoria, tiempo de espera y tamaño de los datos, y cómo aumentarlos.
- Información general de Azure Container Apps — la plataforma serverless de Microsoft para contenedores y cómo se reduce a cero.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es serverless?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

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

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/serverless/