EDICIÓN #001

El secreto de AWS Lambda: Cómo Amazon gestiona billones de peticiones al mes sin que tú gestiones un solo servidor

12 de Enero de 2026

El secreto de AWS Lambda: Cómo Amazon gestiona billones de peticiones al mes sin que tú gestiones un solo servidor

Te explico de forma sencilla la arquitectura interna que hace posible el serverless y cómo resuelve los problemas de escalabilidad: Workers, MicroVMs, Cold Starts y más

17 September 2025

Hoy quiero tocar un tema que si te dedicas a desarrollar cualquier tipo de aplicación, te va a interesar mucho: ¿cómo escalar una app en la nube sin tener que preocuparte por la gestión de servidores? La respuesta corta es usar un servicio de computación **serverless** como AWS Lambda.

Seguro que has oído hablar de ello, pero quiero ir un paso más allá y contarte cómo funciona por dentro. Porque entender la **arquitectura interna** de estas herramientas nos ayuda a tomar mejores decisiones en nuestros propios proyectos. He usado Lambdas para varios proyectos personales y no me había parado a pensar en la tecnología que hay detrás hasta hace poco a si que te cuento todo lo que he aprendido.

El problema de siempre: gestionar servidores

Imagina que lanzas una app y, de repente, tiene un pico de tráfico enorme. Si tienes una infraestructura tradicional, empiezan los problemas:

- **Costes ineficientes:** Pagas por cada máquina virtual que tienes provisionada, aunque no la estés usando. Y si quieres escalar hacia abajo para ahorrar, necesitas configurar grupos de autoescalado, lo que añade complejidad.
- **Gestión constante:** Tienes que actualizar el sistema operativo, parchear vulnerabilidades, configurar el servidor... Es un trabajo que consume muchísimo tiempo que podrías dedicar a mejorar tu app.
- **Escalabilidad compleja:** A medida que la app crece, gestionar el tráfico de red y planificar la capacidad para que todo funcione bien se convierte en un auténtico dolor de cabeza.

Una solución podría ser **Kubernetes**, pero siendo sinceros, muchas veces se usa de forma desmesurada, sin aprovechar casi nada de su potencial y sobredimensionando la infraestructura.

La idea del *serverless* es precisamente liberarte de todos estos problemas para que puedas centrarte únicamente en desarrollar tu producto. Le das tu código a AWS Lambda (en un archivo zip o una imagen de contenedor) y se encarga de ejecutarlo cuando sea necesario, escalando automáticamente según el tráfico.

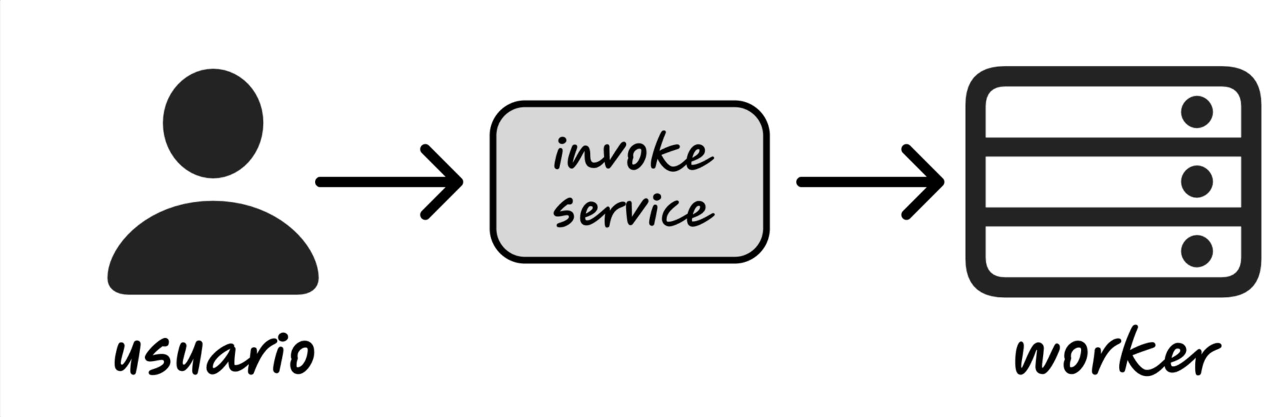
Pero claro, la magia no existe. Detrás de esa simplicidad hay una arquitectura brillante sobre la que quiero hablarte en este post.

¿Cómo funciona AWS Lambda por dentro?

Para abstraer toda la gestión de servidores, los ingenieros de Amazon tuvieron que resolver varios retos de escalabilidad, rendimiento y latencia. Y lo hicieron con ideas sorprendentemente simples.

1. Escalabilidad: La arquitectura de microservicios

Para empezar, el servicio de Lambda se basa en una arquitectura de microservicios. Cuando se invoca una función, la petición llega a un servicio llamado **Invoke Service**, que actúa como un router y la reenvía a un servidor disponible, conocido como **Worker**.



Cómo se ejecuta una función Lambda

Para que todo esto sea escalable y tenga alta disponibilidad, hay miles de estos Workers repartidos en diferentes zonas de disponibilidad por todo el mundo.

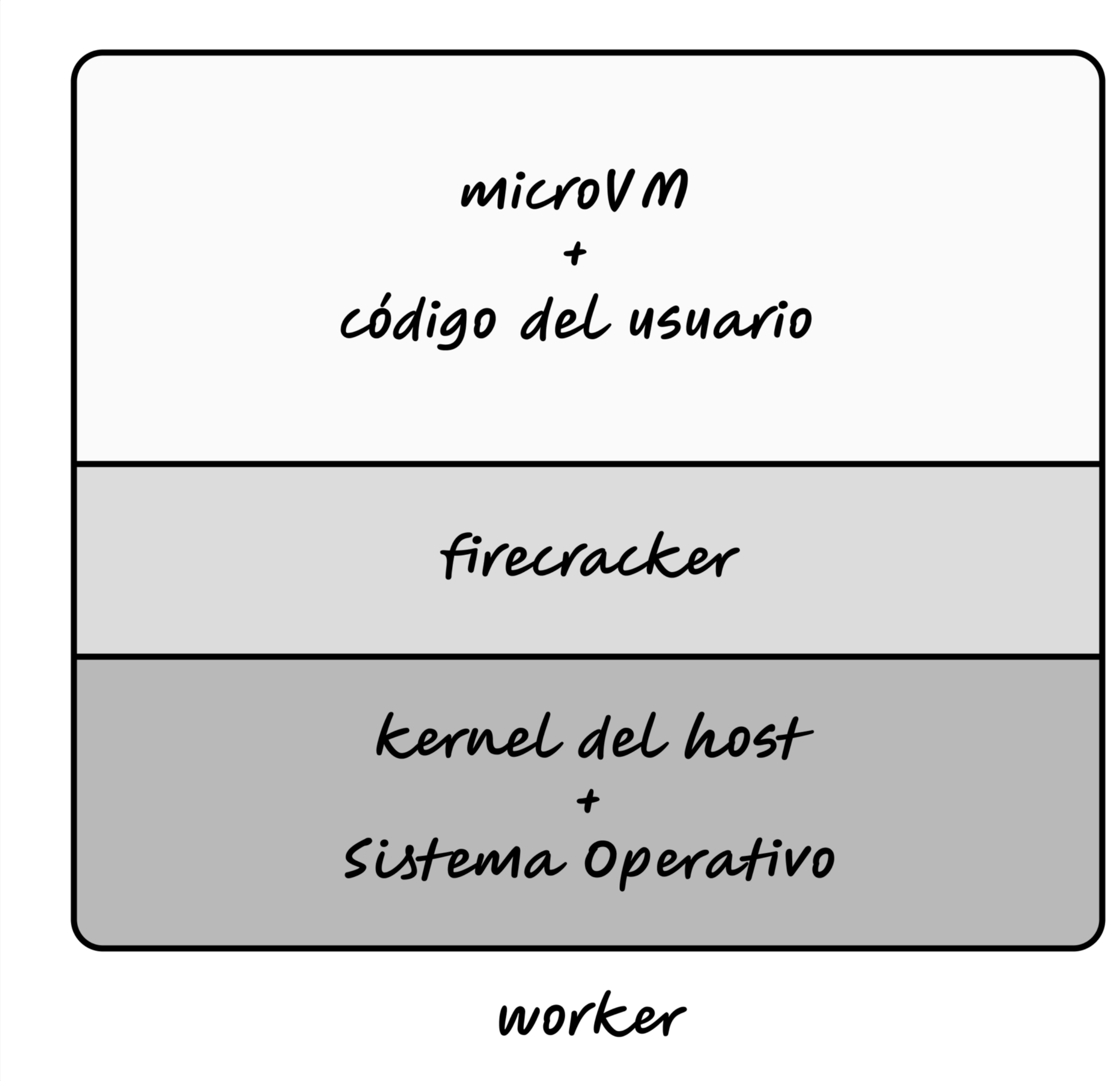
Pero, ¿cómo sabe el sistema qué Worker está libre o cuál está ejecutando una función concreta? Aquí entra en juego el **Assignment Service** que es un servicio encargado de:

- Rastrear qué Workers están ejecutando cada función.
- Saber si un Worker está disponible para recibir una nueva invocación.
- Marcar un Worker como "no disponible" si falla.

Para que este sistema sea tolerante a fallos, toda la información sobre los Workers se guarda en un *journal log* externo. Así, si el Assignment Service falla, otro puede tomar el relevo rápidamente sin perder el estado de los Workers.

2. Rendimiento y aislamiento: Las MicroVMs

Los Workers son, en esencia, instancias de AWS EC2. Pero, lógicamente, no pueden dedicar un Worker entero a cada cliente porque sería un desperdicio de recursos brutal. La solución a la que llegaron los ingenieros de Amazon fue usar máquinas virtuales ultraligeras llamadas **MicroVMs**, gestionadas por una tecnología llamada **Firecracker**.



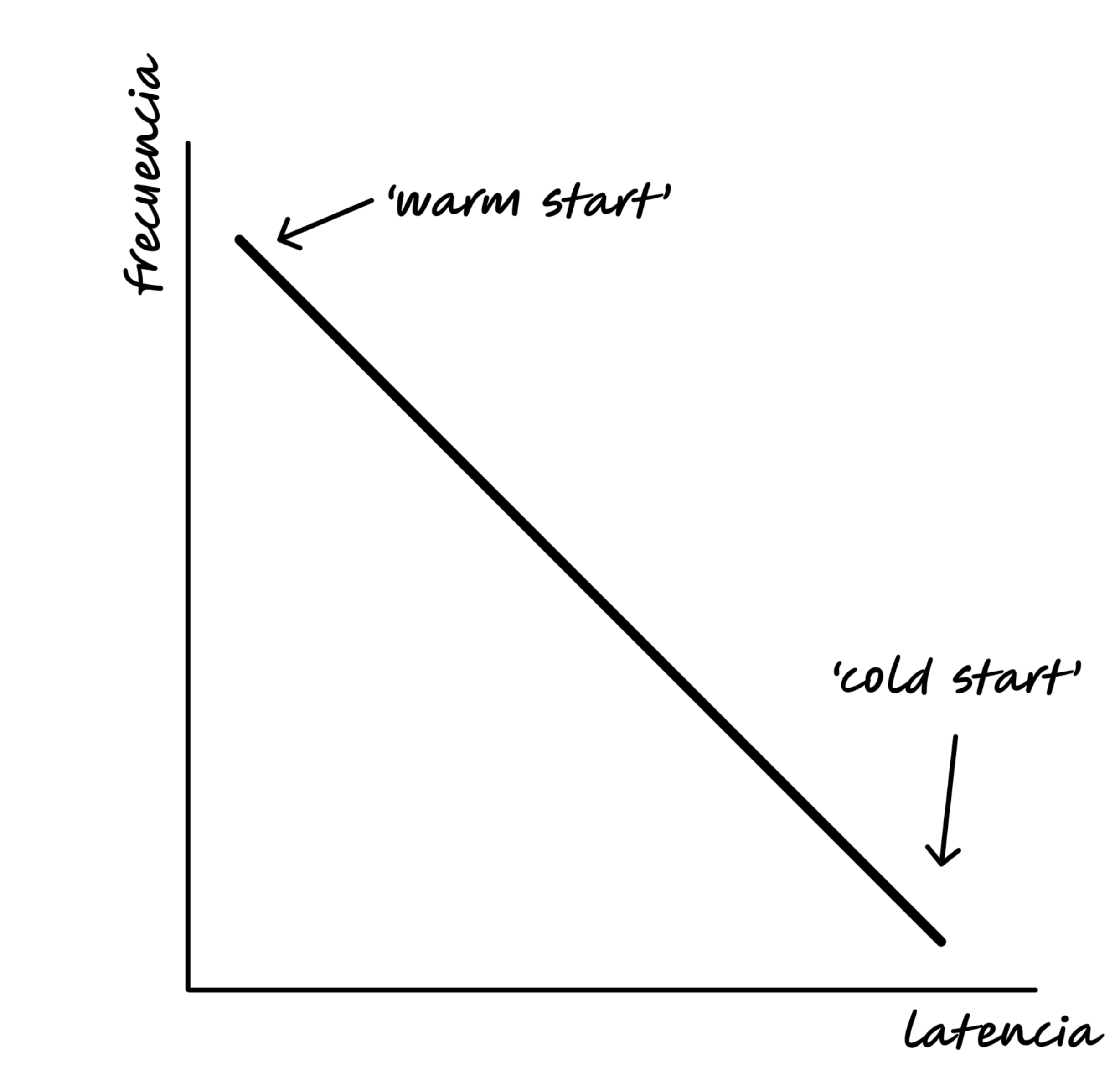
Un microVM ejecutándose en un worker

Esto permite que en un solo Worker se ejecuten muchísimas MicroVMs, cada una completamente aislada de las demás. De esta forma, consiguen un aislamiento a nivel de cliente (*tenant-level isolation*) sin sacrificar la eficiencia.

Cuando se necesita un nuevo Worker, un servicio llamado **Placement Service** se encarga de aprovisionarlo, monitorizar su salud y escalar la flota según el tráfico.

3. Latencia: La lucha contra el "Cold Start"

Aquí viene uno de los conceptos más importantes del mundo serverless. El 99% de las veces, tu función Lambda se ejecutará en una MicroVM que ya está lista y en caliente. Esto se conoce como **Warm Start**, y es súper rápido.



Cold Start VS Warm Start en AWS Lambda

Sin embargo, a veces no hay MicroVMs disponibles y hay que crear una desde cero para atender una petición. Este proceso se llama **Cold Start**. Un *cold start* es más lento porque implica:

- Lanzar la MicroVM.
- Descargar el código de tu función.
- Inicializar el código.

Para reducir esta latencia, que puede ser crítica en algunas aplicaciones, en AWS implementaron dos optimizaciones geniales:

- **Snapshots de MicroVMs:** En lugar de crear la MicroVM desde cero, crean una "foto" (snapshot) de una MicroVM ya inicializada y la restauran cuando ocurre un *cold start*. Esto es como entregar una máquina ya configurada y lista para funcionar, lo que reduce la latencia del *cold start* hasta en un 90%.
- **Carga perezosa de contenedores (Lazy Loading):** Cuando usas una imagen de contenedor para tu función, no es necesario descargar la imagen completa para empezar a procesar peticiones. Lambda es capaz de dividir la imagen en pequeños trozos y descargar únicamente los que son estrictamente necesarios para arrancar. El resto se va descargando bajo demanda.

Al final, como ves, los fundamentos no han cambiado tanto. Se trata de una arquitectura distribuida muy bien pensada que nos permite a los desarrolladores despreocuparnos de la infraestructura.

Espero que esta pequeña inmersión en las tripas de AWS Lambda te haya parecido interesante. A veces, entender cómo funcionan por dentro las herramientas que usamos nos da una perspectiva mucho más completa.