Serverless

Contenido

Serverless 1
¿Qué es Serverless? 2
Beneficios 3
Bajo coste 3
BaaS: costo de desarrollo reducido 4
FaaS: costos de escalado 4
La optimización es la raíz de algunos ahorros de costos 4
Gestión operativa más sencilla 5
Escalar los beneficios de FaaS más allá de los costos de infraestructura 5
Reducción de la complejidad de la implementación y el empaquetado 5
Tiempo de comercialización y experimentación continua 5
¿Computación "más verde"? 6
Inconvenientes 7
Inconvenientes inherentes 7
Control de proveedores 7
Problemas de tenencia múltiple 7
Dependencia de un proveedor 7
Preocupaciones de seguridad 7
Repetición de la lógica en las plataformas de los clientes 8
Pérdida de optimizaciones del servidor 8
Inconvenientes de implementación 8
Pruebas 8
Depuración 8
Seguimiento y observabilidad 8
Aplazamiento de operaciones 9
El futuro de Serverless 9
Mitigación de los inconvenientes 9
Herramientas 9
Mejoras de la plataforma 9
Educación 9
Comunidad 9
Referencias 10

¿Qué es Serverless?

Al igual que muchas tendencias en software, no hay una visión clara sobre lo que es Serverless.

Para empezar, abarca dos áreas diferentes pero superpuestas:

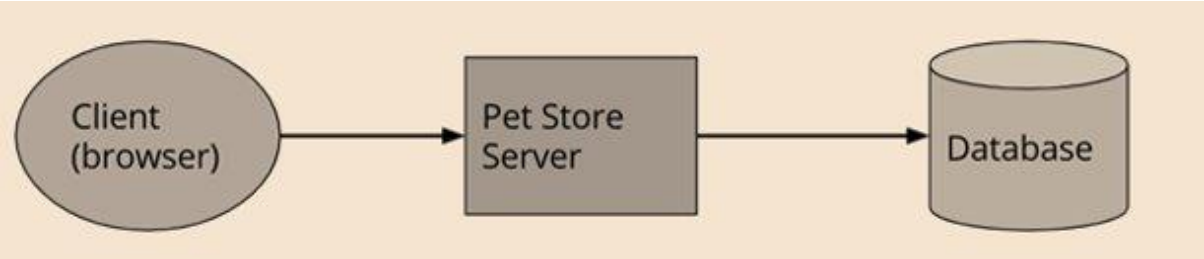
En primer lugar, Serverless se utilizó para describir aplicaciones que incorporan parcial o totalmente servicios alojados en la nube de terceros, para administrar la lógica y el estado del lado del servidor. Estos servicios se conoces como “(Mobile) Backend as a Service” or “BaaS”

Por otra parte, Serverless también puede tratarse de aplicaciones donde el desarrollador sigue escribiendo lógica del lado servidor, pero, a diferencia de las arquitecturas tradicionales, se ejecuta en contenedores de proceso sin estado que se desencadenan por eventos, son efímeros (es decir, solo pueden durar una invocación) y están completamente administrados por un tercero. Estos servicios se conocen como "Functions as a Service" o "FaaS.

A continuación, Nos vamos a centrar en un ejemplo de aplicación impulsada por la interfaz de usuario:

Pensemos en un sistema tradicional orientado al cliente de tres niveles con lógica del lado servidor.

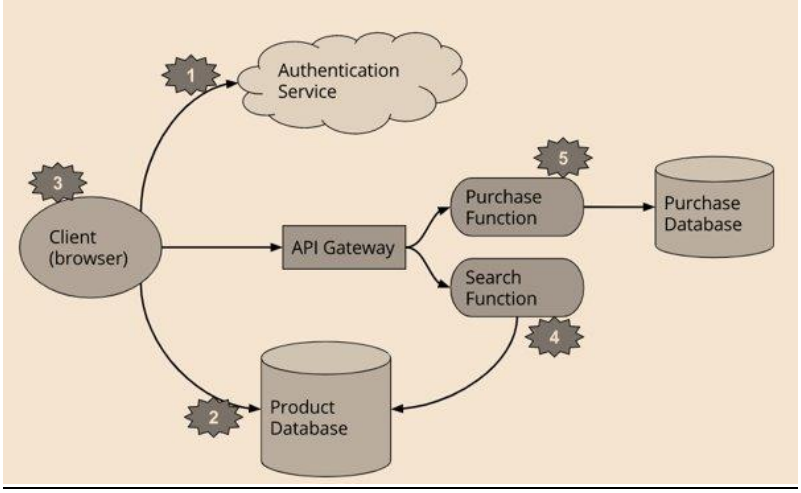
Tradicionalmente, la arquitectura se parecerá algo al siguiente diagrama.



El sistema está Implementado en Java o Javascript en el lado del servidor, con un componente HTML + Javascript como cliente.

Con esta arquitectura, gran parte de la lógica del sistema (autenticación, navegación de página, búsqueda, transacciones) está implementada por la aplicación del servidor.

Con una arquitectura Serverless, el sistema terminaría teniendo un aspecto más parecido al siguiente:



Podemos identificar una serie de cambios significativos:

1. Hemos eliminado la lógica de autenticación en la aplicación original y la hemos reemplazado por un servicio Backend as a Service de terceros (por ejemplo, Auth0).
2. Hemos permitido al cliente acceso directo a un subconjunto de nuestra base de datos (para listados de productos), que a su vez está completamente alojado por un tercero (por ejemplo, Google Firebase).
3. Estos dos puntos anteriores implican un tercer cambio muy importante: cierta lógica que estaba en el servidor está ahora dentro del cliente, por ejemplo, para realizar un seguimiento de sesión de usuario o leer desde una base de datos.
4. En lugar de tener un servidor siempre en ejecución, como existía en la arquitectura original, podemos implementar una función Functions as a Service que responda a las solicitudes HTTP a través de una puerta de enlace API. Podemos utilizar AWS Lambda, una plataforma informática sin servidor, como nuestra plataforma Functions as a Service.
5. Por último, podemos reemplazar nuestra funcionalidad de negocio por otra función FaaS independiente, manteniéndola en el lado del servidor por razones de seguridad.

Este ejemplo muestra otro punto muy importante de las arquitecturas sin servidor. En la versión original, la aplicación del servidor central administró todo el flujo, el control y la seguridad. En la versión serverless vemos una preferencia **por la coreografía sobre la orquestación,** primando las relaciones entre los diferentes servicios, una idea común en un enfoque de microservicios.

Beneficios

Hasta ahora, hemos tratado principalmente definir y explicar lo que han llegado a significar las arquitecturas serverless Ahora vamos a hablar de los beneficios y desventajas de esta forma de diseñar e implementar aplicaciones.

Para empezar, vamos a ver los beneficios:

Bajo coste

Serverless es, en su forma más simple, una solución de subcontratación. Permite pagarle a alguien para que administre servidores, bases de datos e incluso la lógica de la aplicación. Dado que está utilizando un servicio predefinido que muchas otras personas también usarán, vemos un efecto de Economía de escala: paga menos por su base de datos administrada porque un proveedor está ejecutando miles de bases de datos muy similares.

Los costos reducidos le aparecen como el total de dos aspectos. El primero son las ganancias en los costos de infraestructura que provienen exclusivamente de compartir la infraestructura (por ejemplo, hardware, redes) con otras personas. El segundo son las ganancias en el costo de mano de obra: se puede dedicar menos tiempo a un sistema sin servidor subcontratado que a un equivalente desarrollado y alojado por nosotros mismos.

BaaS: costo de desarrollo reducido

IaaS y PaaS se basan en la premisa de que la gestión del servidor y del sistema operativo puede comercializarse. El backend sin servidor como servicio, por otro lado, es el resultado de la mercantilización de componentes completos de la aplicación.

La autenticación es un buen ejemplo. Muchas aplicaciones codifican su propia funcionalidad de autenticación, que a menudo incluye características como registro, inicio de sesión, administración de contraseñas e integración con otros proveedores de autenticación. En general, esta lógica es muy similar en la mayoría de las aplicaciones, y servicios como Auth0 se han creado para permitirnos integrar la funcionalidad de autenticación lista para usar en nuestra aplicación sin que tengamos que desarrollarla nosotros mismos.

En el mismo hilo están las bases de datos BaaS, como el servicio de base de datos de Firebase. Algunos equipos de aplicaciones móviles han descubierto que tiene sentido que el cliente se comunique directamente con una base de datos del lado del servidor. Una base de datos BaaS elimina gran parte de la sobrecarga de administración de la base de datos y, por lo general, proporciona mecanismos para realizar la autorización adecuada para diferentes tipos de usuarios, en los patrones que se esperan de una aplicación sin servidor.

Dependiendo de su experiencia, estas ideas pueden hacer que se retuerza (probablemente por razones que veremos en la sección de inconvenientes), pero no se puede negar la cantidad de empresas exitosas que han podido producir productos atractivos con apenas nada de su propio servidor.

FaaS: costos de escalado

Una de las ventajas de Serverless FaaS es que, "el escalado horizontal es completamente automático, elástico y administrado por el proveedor". Esto tiene varios beneficios, pero en el lado de la infraestructura básica, el mayor beneficio es que solo paga por la computación que necesita, hasta un límite de 100 ms. Dependiendo de la escala y la forma de su tráfico, esto puede ser una gran ventaja económica.

Si el tráfico es uniforme y haría un buen uso constante de un servidor en ejecución, es posible que no vea este beneficio de costo y, de hecho, puede gastar más usando FaaS. Se deben hacer algunos cálculos y comparar los costos actuales del proveedor con los equivalentes de ejecutar servidores a tiempo completo para ver si los costos son aceptables.

La optimización es la raíz de algunos ahorros de costos

Hay un aspecto más interesante que mencionar sobre los costos de FaaS: cualquier optimización de rendimiento que se realice en el código no solo aumentará la velocidad de la aplicación, sino que tendrá un vínculo directo e inmediato para la reducción de los costos operativos, sujeto a la granularidad del esquema de cobro del proveedor. Por ejemplo, supongamos que una aplicación inicialmente tarda un segundo en procesar un evento. Si, a través de la optimización del código, esto

se reduce a 200 ms, (en AWS Lambda) verá inmediatamente un ahorro del 80 por ciento en los costos de computación sin realizar ningún cambio de infraestructura.

Gestión operativa más sencilla

·En el lado de Serverless BaaS es bastante obvio por qué la administración operativa es más simple que otras arquitecturas: admitir menos componentes equivale a menos trabajo.

· En el lado de FaaS hay varios aspectos en los que vamos a profundizar:

Escalar los beneficios de FaaS más allá de los costos de infraestructura

Vale la pena señalar que la funcionalidad de escalado de FaaS no solo reduce el costo de computación, sino que también reduce la administración operativa porque el escalado es automático.

En el mejor de los casos, si su proceso de escalado fue manual, por ejemplo, se necesita agregar y eliminar explícitamente instancias a una serie de servidores, con FaaS se puede u olvidarse de eso y dejar que el proveedor de FaaS escale la aplicación.

De manera similar, dado que el proveedor realiza el escalado en cada solicitud / evento, ya no hace falta pensar en la cuestión de cuántas solicitudes simultáneas se pueden manejar antes de quedarse sin memoria o ver un impacto excesivo en el rendimiento, al menos no dentro de sus componentes alojados en FaaS. Las bases de datos posteriores y los componentes que no son FaaS deberán reconsiderarse a la luz de un posible aumento significativo en su carga.

Reducción de la complejidad de la implementación y el empaquetado

Empaquetar e implementar una función FaaS es simple en comparación con implementar un servidor completo. Todo lo que está haciendo es empaquetar todo su código en un archivo zip y cargarlo. Cuando se está comenzando, ni siquiera se necesita empaquetar nada; es posible que pueda escribir su código en la propia consola del proveedor.

Este proceso no lleva mucho tiempo para describirlo, pero para algunos equipos este beneficio puede ser absolutamente enorme: una solución completamente sin servidor requiere cero administraciones del sistema.

Tiempo de comercialización y experimentación continua

Una gestión operativa más sencilla es un beneficio que nosotros, como ingenieros, comprendemos, pero ¿qué significa eso para nuestras empresas?

La razón obvia es el costo: menos tiempo dedicado a las operaciones equivale a menos personas necesarias para las operaciones. Pero una razón mucho más importante es el tiempo de comercialización. A medida que nuestros equipos y productos se orientan cada vez más hacia procesos ágiles y ajustados, queremos probar continuamente cosas nuevas y actualizar rápidamente nuestros sistemas existentes. Tener una buena capacidad de implementación inicial de idea nueva nos permite probar nuevos experimentos con baja fricción y costo mínimo.

Si bien los beneficios de costos son las mejoras que se expresan más fácilmente con Serverless, esta reducción en el tiempo de entrega es también muy importante. Puede permitir una mentalidad de desarrollo de productos de experimentación continua, y eso es una verdadera revolución en la forma en que entregamos software en las empresas.

¿Computación "más verde"?

Durante las últimas dos décadas, ha habido un aumento masivo en el número y tamaño de los centros de datos en el mundo. Además de los recursos físicos necesarios para construir estos centros, los requisitos energéticos asociados son muy grandes.

Los servidores inactivos, pero encendidos, consumen una cantidad inapropiada de esta energía, y son una gran parte de la razón por la que necesitamos tantos centros de datos más grandes:

Eso es extraordinariamente ineficiente y crea un enorme impacto ambiental.

Por un lado, es probable que la infraestructura en la nube ya haya ayudado a reducir este impacto, ya que las empresas pueden "comprar" más servidores a pedido, solo cuando los necesitan absolutamente, en lugar de aprovisionar todos los servidores posiblemente necesarios con mucho tiempo de anticipación. Sin embargo, también se podría argumentar que la facilidad de aprovisionamiento de servidores puede haber empeorado la situación si muchos de esos servidores se quedan sin una gestión de capacidad adecuada.

Con un enfoque serverless, que el proveedor proporcione solo la capacidad de cómputo suficiente para nuestras necesidades en tiempo real. Luego, el proveedor puede tomar sus propias decisiones de capacidad en conjunto entre sus clientes.

Esta diferencia debería conducir a un uso mucho más eficiente de los recursos en los centros de datos y, por lo tanto, a reducciones en el impacto ambiental en comparación con los enfoques tradicionales de gestión de la capacidad.

Inconvenientes

. Algunas de estos inconvenientes son inherentes a los conceptos; el progreso no puede solucionarlos por completo y siempre será necesario tenerlos en cuenta. Otros están vinculados a implementaciones actuales; con el tiempo podemos esperar verlos resueltos.

Inconvenientes inherentes

Control de proveedores

Con cualquier estrategia de subcontratación, cede el control de parte de su sistema a un proveedor externo. Tal falta de control puede manifestarse como tiempo de inactividad del sistema, límites inesperados, cambios de costos, pérdida de funcionalidad, actualizaciones forzadas de API y más

Problemas de tenencia múltiple

La tenencia múltiple se refiere a la situación en la que se ejecutan varias instancias de software para varios clientes (o inquilinos) diferentes en la misma máquina y posiblemente dentro de la misma aplicación de alojamiento. Es una estrategia para lograr los beneficios de la economía de escala que mencionamos anteriormente. Ninguna solución perfecta y, a veces, múltiples inquilinos pueden tener problemas de seguridad (un cliente puede ver los datos de otro), solidez (un error en el software de un cliente que causa una falla en el software de un cliente diferente) y rendimiento (un cliente de alta carga). causando que otro se ralentice).

Dependencia de un proveedor

Es muy probable que cualquier característica serverless que esté utilizando de un proveedor sea implementada de manera diferente por otro proveedor. Si se desea cambiar de proveedor, es casi seguro que se necesitará actualizar las herramientas operativas (implementación, monitoreo, etc.), probablemente necesitará cambiar su código (por ejemplo, para satisfacer una interfaz FaaS diferente), e incluso puede necesitar cambiar su diseño o arquitectura si hay diferencias en el comportamiento de las implementaciones de los proveedores de la competencia.

Debido a esto, algunas personas adoptan un enfoque de "múltiples nubes", desarrollando y operando aplicaciones de una manera que es independiente del proveedor de la nube real que se está utilizando. A menudo, esto es incluso más costoso que un enfoque de nube única.

Preocupaciones de seguridad

Adoptar un enfoque serverless abre a una gran cantidad de preguntas de seguridad. A continuación, presentamos algunos aspectos muy breves a considerar

Cada proveedor serverless que se utiliza aumenta la cantidad de implementaciones de seguridad diferentes que adopta su ecosistema, lo que aumenta la probabilidad de un ataque exitoso.

Si se utiliza una base de datos BaaS directamente desde plataformas móviles, está perdiendo la barrera protectora que proporciona una aplicación del lado del servidor en una aplicación tradicional. Si bien esto no es un factor decisivo, requiere un cuidado significativo al diseñar y desarrollar una aplicación.

Repetición de la lógica en las plataformas de los clientes

Con una arquitectura BaaS "completa", no se escribe ninguna lógica personalizada en el lado del servidor; todo está en el cliente. Esto puede estar bien para la primera plataforma de cliente, pero tan pronto como se necesite la próxima plataforma, se necesitará repetir la implementación de un subconjunto de esa lógica, y no se habría necesitado esta repetición en una arquitectura más tradicional.

Pérdida de optimizaciones del servidor

Con una arquitectura BaaS completa, no hay oportunidad de optimizar el diseño de su servidor para el rendimiento del cliente

Tanto este como el inconveniente anterior existen para las arquitecturas BaaS completas donde toda la lógica personalizada está en el cliente y los únicos servicios de backend son proporcionados por el proveedor. Una mitigación de ambos es adoptar FaaS, o algún otro tipo de patrón ligero del lado del servidor, para mover cierta lógica al servidor.

Inconvenientes de implementación

Es probable que los inconvenientes descritos anteriormente siempre existan con Serverless. Veremos mejoras en las soluciones de mitigación, pero siempre estarán ahí.

Sin embargo, los demás inconvenientes se reducen puramente al estado actual de la técnica.

Pruebas

Parte de la razón por la que considerar las pruebas de integración es un gran problema es que nuestras unidades de integración con Serverless FaaS (es decir, cada función) son mucho más pequeñas que con otras arquitecturas, por lo que confiamos en las pruebas de integración mucho más que con otras Estilos arquitectonicos.

Depuración

La depuración con FaaS es un área interesante. Ha habido avances aquí, principalmente relacionados con la ejecución de funciones FaaS localmente. Microsoft proporciona un excelente soporte de depuración para funciones que se ejecutan localmente, pero que se activan mediante eventos remotos.

Las funciones de depuración que se ejecutan realmente en un entorno de nube de producción es una historia diferente. Lambda al menos no tiene soporte para eso todavía.

Seguimiento y observabilidad

El monitoreo es un área complicada para FaaS debido a la naturaleza efímera de los contenedores. La mayoría de los proveedores de la nube le brindan cierta cantidad de soporte de monitoreo, y también hemos visto muchos trabajos de terceros de parte de los proveedores de monitoreo tradicionales. Aun así, lo que sea que ellos puedan hacer en última instancia, depende de los datos fundamentales que le proporcione el proveedor. Esto puede estar bien en algunos casos, pero para AWS Lambda, al menos, es muy básico. Lo que realmente necesitamos en esta área son API abiertas y la capacidad de los servicios de terceros para ayudar más.

Aplazamiento de operaciones

Todavía hay mucho que hacer desde el punto de vista de la supervisión, el escalado arquitectónico, la seguridad y las redes. El peligro aquí es dejarse llevar por una falsa sensación de seguridad.

La solución aquí es la educación. Los equipos que utilizan sistemas serverless deben considerar las actividades operativas desde el principio, y los proveedores y la comunidad deben proporcionar la enseñanza que les ayude a comprender lo que esto significa. Áreas como las pruebas de carga preventivas también ayudarán a los equipos a aprender por sí mismos.

El futuro de Serverless

Para cerrar vamos a discutir algunas áreas donde creemos que el mundo Serverless puede desarrollarse en los próximos meses y años.

Mitigación de los inconvenientes

Serverless sigue siendo un mundo bastante nuevo. Los desarrollos más importantes de Serverless deben ser capaces de mitigar los inconvenientes inherentes y eliminar los inconvenientes de implementación.

Herramientas

Las herramientas siguen siendo una preocupación para Serverless, y eso se debe a que muchas de las tecnologías y técnicas son nuevas.

La monitorización distribuida es probablemente el área que necesita la mejora más significativa.

Mejoras de la plataforma

Ciertos inconvenientes de Serverless en este momento se reducen a la forma en que se implementan las plataformas. La duración de la ejecución, la latencia de inicio y los límites entre funciones son tres obvios.

Es probable que estas soluciones se arreglen con nuevas soluciones o que se den soluciones alternativas.

Educación

Muchos inconvenientes se están mitigando a través de la educación. Por lo que se debe seguir incidiendo en este campo.

Comunidad

Ya hay una comunidad sin servidor bastante amplia con múltiples conferencias, reuniones en muchas ciudades y varios grupos en línea. Esperamos que siga creciendo en la misma línea que comunidades como Docker y Spring.

Referencias

https://martinfowler.com/articles/serverless.html#unpacking-faas