Arquitecturas Serverless

¿Qué es Serverless?

Serverless hace referencia a las aplicaciones que utilizan servicios en la nube para manejar la lógica de negocio y su estado, también se puede definir como las aplicaciones en las que la lógica está escrita por los desarrolladores, pero la ejecución o el despliegue está llevado a cabo por una nube, que responderá a diferentes peticiones sin estar consumiendo recursos de forma continua, es decir, estará respondiendo bajo demanda.

Las aplicaciones Serverless combinan las funciones o peticiones sin estado y por demanda, las cuales se llama Functions-as-a-Service o FaaS, y un gestor completamente previsto de servicios tales como mensajería, almacenamiento de ficheros, bases de datos, streaming o autenticación al cual se le conoce como Backend-as-a-Service o BaaS. Esto último es una de las razones principales por las que los desarrolladores optan por Serverless, al poder centrarse en la lógica de negocio y poder despreocuparse de todo el despliegue, de la escalabilidad de la aplicación o del guardado de datos.

¿Por qué Serverless?

Existen varias razones por las que los desarrolladores deciden que su aplicación sea Serverless:

- Reducción de costes (Al utilizar el modelo de respuestas bajo demanda, hace que una aplicación Serverless pueda utilizar un modelo de pay-per-use, es decir, de pagar solo por los recursos utilizados y así no pagar por recursos extra o por menos recursos de los necesarios).
- Desarrolladores centrados en la lógica de negocio (Al utilizar Serverless, los desarrolladores se ahorran tener que pensar en cosas como el despliegue, la escalabilidad de la aplicación, la monitorización... Y gracias a eso se pueden centrar en el desarrollo de nuevas características para la aplicación)
- Escalabilidad de las aplicaciones Serverless (Las aplicaciones Serverless ofrecen una escalabilidad casi infinita y sin necesidad de un gran esfuerzo de ingeniería, esto ocurre gracias al Functions-as-a-Service antes comentado)
- Existen otras razones como el mejor rendimiento o la mayor velocidad para lanzar el producto al mercado.

La razón por la que más gente desarrolla aplicaciones Serverless es por la reducción de los costes de la aplicación, aunque realmente en orden de ventajas que más aporta Serverless esta:

- Arquitectura basada en eventos
- Reducción de costes
- Velocidad del desarrollo de la aplicación
- Flexibilidad para escalar la aplicación
- Rendimiento de la aplicación

¿Cuándo utilizar Serverless?

Cuando se habla de Serverless se suele pensar que solo se debería de utilizar en aplicaciones funcionales y sin latencias.

Sin embargo, un estudio de mercado revela que un 39% de las aplicaciones Serverless tienen un alto tráfico de peticiones y un 47% de las aplicaciones tienen un bajo tráfico de peticiones dado que son aplicaciones bajo demanda.

Un argumento en contra de las aplicaciones Serverless es que tienen inicios lentos dado que el servidor puede suspendido cuando un cliente requiera de sus servicios, y esto implica que el servidor debe preparar el entorno de ejecución de la aplicación Serverless solicitada y lanzarla. Lo cual son varios segundos que el cliente debe esperar añadidos a la latencia que ya tendría la aplicación de por sí.

Sin embargo, un estudio de mercado revela que existen aplicaciones Serverless en producción en las que los tiempos de latencia son críticos para el funcionamiento de estas, aún con los inicios lentos de éstas.

Concretamente, 38% de las aplicaciones Serverless estudiadas no tienen latencias en sus funcionalidades, mientras que el 32% de ellas tienen latencias en todas sus funcionalidades y un 28% tiene latencias parciales en algunas de sus funcionalidades.

Otro razonamiento en contra de las aplicaciones Serverless es que pueden ser inadecuadas para tareas que lleven mucho tiempo y manejan una gran cantidad de datos.

El estudio de mercado apoya esta hipótesis ya que el 69% de las aplicaciones Serverless estudiadas utilizan un volumen de datos inferior a los 10MB.

En resumen, las aplicaciones Serverless se suelen usar para tareas cortas que no manejen una cantidad muy grande de datos, aunque también se utilizan para tareas esenciales donde mantener las latencias en valores lo más bajos posibles es muy importante.

Al final existen todo tipo de aplicaciones Serverless.

¿Cómo se utiliza Serverless?

Cuando iniciamos un proyecto usando Serverless, ya desde el primer momento nos surgen dudas, ¿Cuál proveedor a utilizar? ¿Qué lenguaje utilizar? ¿javascript, python? Granularidad de las funciones… Más importantes que de costumbre estas decisiones, pues, por ejemplo, dependiendo del lenguaje de programación podemos encontrarnos con que un arranque en frio de la función tarde demasiado para él cliente. O proveedores como AWS que optimizan las funciones guardándolas en “prewarm”, es decir, guardando las funciones en una cache para ejecuciones posteriores, son cosas para tener en cuenta dependiendo del tipo de aplicación que estemos desarrollando.

Pensando en el Backend as a Service, tendremos que decidir qué hacer con los datos, normalmente usaremos bases de datos descentralizadas y ya administradas por el proveedor del servicio en la nube, como DynamoDB, que está gestionada por Amazon.

No es oro todo lo que reluce

Rendimiento

El código Serverless utilizado con poca frecuencia puede sufrir una mayor latencia.

Límite de recursos

La computación Serverless no es adecuada para algunas cargas de trabajo de computación, como la computación de alto rendimiento, sería más barato aprovisionar la cantidad de servidores que se cree que se requieren en un determinado momento.

Monitoreo y depuración

Depurar código, o monitorizar el uso de recursos es más complicado, normalmente el programador no tiene a su disponibilidad la consola, o esta, esta truncada a la salida que quiere mostrar el proveedor.

Seguridad

Serverless, a veces se considera erróneamente que es más seguro que las arquitecturas tradicionales. Esto es cierto hasta cierto punto porque el proveedor de la nube se ocupa de las vulnerabilidades del sistema operativo, pero la superficie total de ataque es significativamente mayor ya que hay muchos más componentes en la aplicación en comparación con las arquitecturas, otras desventajas son que los clientes no pueden controlar e instalar nada en el punto final ni en el nivel de la red, como un sistema de detección / prevención de intrusiones.

Intimidad

La privacidad, que harán los proveedores con los datos que procesan, si los guardan, si los venden. No lo sabemos nunca al 100%.

Dependencia de un proveedor

Muchas herramientas de los proveedores solo funcionan en su entorno, así por ejemplo podemos encontrarnos con que la base de datos no sea compatible con otro proveedor, por lo que migrar a otro proveedor se hace inviable en algunos casos.

Referencias

https://squadex.com/insights/what-is-serverless/

https://martinfowler.com/articles/serverless.html

https://en.wikipedia.org/wiki/Serverless_computing