Máster Universitario en Ingeniería Informática

CLOUD COMPUTING

ARQUITECTURA DE MICROSERVICIOS

UNIVERSIDAD DE GRANADA

Carlos Morales Aguilera

Índice

1. ¿Qué es un microservicio?

Los microservicios o arquitectura de microservicios es un sistema de desarrollo software en el que los componentes pueden emplear diferentes tecnologías, donde resuelven diversos problemas, para poder colaborar y ofrecer una funcionalidad global completa. Estos servicios poseen ciertas características como que pueden ser desplegados, van evolucionando y se actualizan de forma independiente, pero trabajan de manera conjunta.

Cada componente o microservicio encapsula la lógica de una función completa y ofrece un servicio en particular el cual se despliega y puede cuando se necesita. Partiendo de este concepto surge la idea de un servicio de tamaño o funcionalidad reducida (de ahí microservicio), para conformar de forma general una arquitectura cuya estructura consiste principalmente en la comunicación de estos servicios.

Bajo esta idea cabría destacar una principal característica que claramente aporta una gran ventaja frente a un sistema monolítico. ¿Entonces cual es la principal novedad que aporta este nuevo modelo de arquitectura? Modularidad, ya que en una arquitectura monolítica, una simple modificación del sistema requiere un parón y un redespliegue del sistema completo, mientras que la caída de un servicio en este nuevo modelo no implica este tipo de acción, ya que el sistema sin un servicio sigue trabajando (a excepción de este), y la funcionalidad se ve reducida, no detenida.

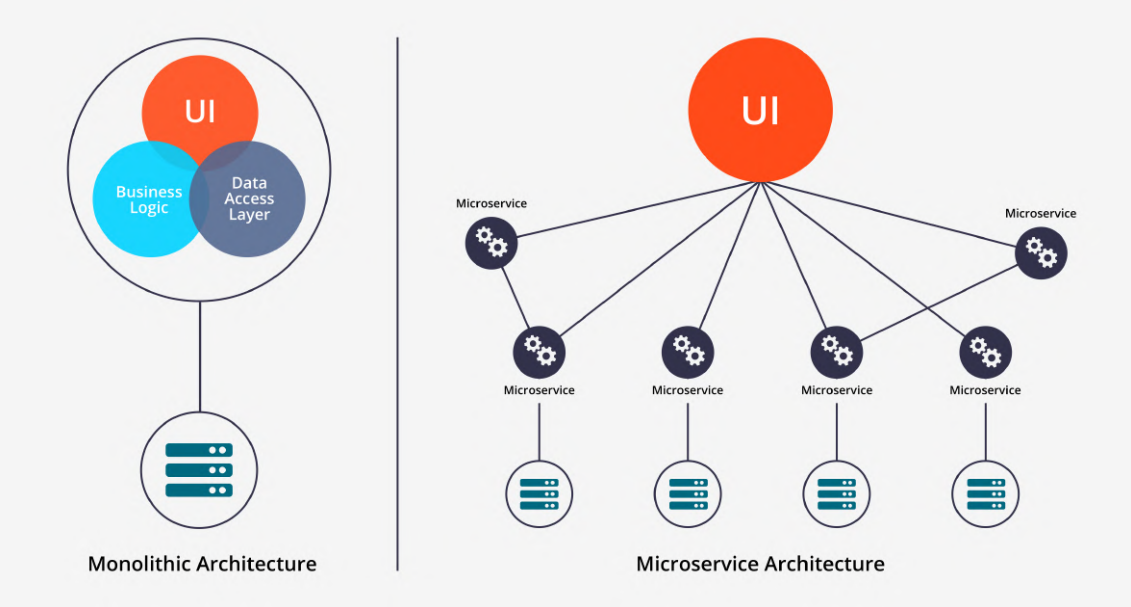


Imagen 1: Modelo monolítico VS Modelo de microservicios

Ante esta nueva idea, surgiría una duda: ¿los microservicios tienen que ser pequeños? La división en todo caso tiene que ver más con una funcionalidad que con el tamaño. Cada microservicio pretender ser una unidad funcional, con sentido en la lógica global del negocio o servicio final a ofrecer. La independencia de cada servicio finalmente es la clave de este nuevo modelo.

2. ¿Cómo se gestionan los microservicios?

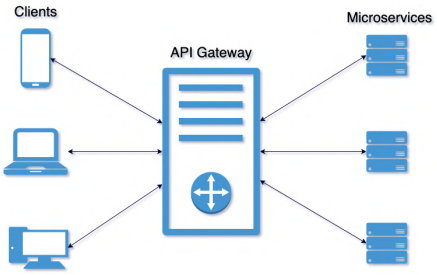
Para empezar a comprender la gestión, cabe destacar dos conceptos principales dentro del ámbito de los microservicios, como son orquestación y coreografía.

Orquestación: Se trata de un componente que coordina llamadas a los diferentes servicios de forma secuencial con peticiones y respuestas. La principal característica es que este componente se encarga de gestionar los diferentes errores que se pueden producir en esta arquitectura. Esta opción tiene un bajo nivel de acoplamiento, pero la gestión de errores y seguimiento del flujo requiere una monitorización más precisa.

Coreografía: En esta opción, existe un sistema central por eventos, donde funciona como un canal al que los diferentes servicios estan suscritos, y son notificados de cuando un servicio termina una tarea. Aunque esta opción encaja más con esta arquitectura, es más compleja su gestión.

Por otro lado, existen una serie de elementos, que si bien no son obligatorios, suelen ser bastante comunes en este tipo de arquitectura:

- Registro: Es un sistema que básicamente mantiene un registro activo de que sistemas se encuentran activos y sus estados, y por lo tanto se debe mantener constanemente actualizado.
- Balanceador: Entra en acción con el registro previamente descrito, comprobando los servicios activos y se encarga de repartir la carga de peticiones entre estos servicios.
- Proxy inverso: Es una puerta de entrada única que recibe todas las peticiones y las redirige a los puntos adecuados. Al ser el punto de entrada, puede cumplir muchas funciones como seguridad, autenticación, API gateway, etc.
- Logs: Es un sistema encargado de recoger los logs de todos los servicios, procesarlos y presentar una interfaz para su explotación masiva.
- Servidores de configuración: Es un componente que monitoriza las comunicaciones entre servicios. Está pendiente de diferentes tipos de errores para evitar que se propaguen entre los diferentes servicios. Para ello corta una llamada y así no alcanza a un servicio nuevo.



3. Pros y Contras

Entre las principales ventajas de los microservicios se pueden encontrar:

- Alta escalabilidad y con posibilidad de integración de nuevos elementos en cualquier momento.
- Modularización, ya que la caída de un servicio no implica la caída del sistema global.
- Posibilidad de utilizar diferentes metodologías en cada uno de los microservicios.
- Formación de equipos de trabajo colaborativos con grupos reducidos.
- Compatibilidad con diferentes plataformas, ya que se pueden utilizar diferentes entornos además de las tecnologías empleadas.
- Agilidad, ya que el desarrollo de servicios aislados es más rápido que el desarrollo de un sistema global.
- Mantenimiento sencillo y barato, ya que es modular y no total del sistema.

Por otro lado, si hablamos de contras frente a este nuevo modelo, se encontrarían:

- La información, al estar distribuida puede dar lugar a una gestión de un gran número de bloques.
- Consumo de memoria, ya que cada sistema posee sus sitemas de gestión y/o bases de datos.
- Solución de errores, ya que no existen problemas en un único sistema, en diferentes sistemas se pueden producir errores de latencia, balanceo de carga, etc.
- Tests de integración, si bien los tests de servicio no conllevan una mayor dificultad, a la hora de realizar pruebas de integración, se ha de controlar tanto los servicios como sus comunicaciones.
- Coste alto de implantación, ya que pese a su mantenibilidad sencilla, conlleva un alto coste de planificación, de diseño, computacional, de esfuerzo, etc.
- Distribución en todos los sentidos, ya que un sistema monolítico funciona como un bloque, mientras que la gestión, errores, costes y demás posibles aspectos son distribuidos en los diferentes sistemas.

4. Tecnologías

Aunque previamente se ha comentado que una de las ventajas es la posibilidad de ofrecer una arquitectura que puede utilizar diferentes herramientas y tecnologías para diseñar los diferentes microservicios que la conforman, en este apartado se pretende ofrecer una idea general de algunas tecnologías que se pueden utilizar para las diferentes gestiones de la arquitectura, más que de los microservicios (que como previamente se ha mencionado son independientes en muchos sentidos como lógica o tecnología).

4.1. Orquestación

Kubernetes es de hecho el motor de orquestación de contenedores más popular que existe en el mercado. Es un proyecto original de Google y en la actualidad es utilizado por miles de equipos para desplegar contenedores en producción. Es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios. Facilita la automatización y la configuración declarativa. El soporte, las herramientas y los servicios para Kubernetes están ampliamente disponibles.

Otras opciones a destacar son: Docker Swarm, Mesosphere Datacenter (DC/OS), Google Container Engine o HashiCorp Nomad.

4.2. Balanceador de carga y proxy

HAProxy es un software gratuito de código abierto que proporciona un equilibrador de carga de alta disponibilidad y un servidor proxy para aplicaciones basadas en TCP y HTTP que distribuye las solicitudes en varios servidores. Está escrito en C y tiene la reputación de ser rápido y eficiente.

Zevenet es una solución de balanceo de carga con licencia GNU LGPL, que ofrece alta disponibilidad y escalabilidad a los sistemas informáticos. El pasado Martes de introdujeron los conceptos básicos de Zevenet y su funcionamiento y se intentó convertir una

debian base en un balanceador zevenet para hacer algunas configuraciones de casos reales creando alta disponiblidad y escalabilidad para servicios web.

Nginx es un servidor web/proxy inverso ligero de alto rendimiento y un proxy para protocolos de correo electrónico. Es software libre y de código abierto, licenciado bajo la Licencia BSD simplificada; también existe una versión comercial distribuida bajo el nombre de Nginx Plus.

Traefik es un enrutador de borde de código abierto que hace que la publicación de sus servicios sea una experiencia divertida y fácil. Recibe solicitudes en nombre de su sistema y descubre qué componentes son responsables de manejarlas.

4.3. Herramientas de logs

Logstash es una herramienta para la administración de logs. Esta herramienta se puede utilizar para recolectar, analizar y guardar los logs para futuras búsquedas. La aplicación se encuentra basada en jRuby y requiere de Java Virtual Machine para ejecutarse.

Fluentd es un proyecto de software de recolección de datos de código abierto multi-plataforma desarrollado originalmente en Treasure Data. Está escrito principalmente en el lenguaje de programación Ruby.

Como se ha comentado al inicio, estas son solo algunas de las herramientas principales que se han encontrado al hacer una búsqueda sobre herramientas utilizadas en arquitecturas, de microservicios, aunque como se ha repetido continuamente, es una arquitectura tan flexible que realmente las limitaciones dependerán más de la lógica del negocio que de cualquier otro aspecto.

5. Casos de éxito

5.1. Walmart

El problema inicial es que no podían manejar 6 millones de páginas vistas por minuto y no podían mantener una experiencia de usuario positiva. Antes de comenzar a usar microservicios, Walmart adoptó una arquitectura de Internet de 2005, que fue diseñada para computadoras de escritorio, laptops y dispositivos en general.

La compañía decidió reformatear su antiguo sistema en 2012 porque no podía escalar a 6 millones de páginas vistas por minuto y estaba inactivo durante la mayor parte de las horas pico. Esperan prepararse para el mundo en 2020, cuando 4 mil millones de personas estarán conectadas y más de 25 millones de aplicaciones estarán disponibles. Por lo tanto, Walmart ha remodelado la arquitectura de microservicios, con el objetivo de lograr una disponibilidad cercana al 100 % a un costo razonable.

5.2. Spotify

Spotify ofrece servicios a bastante más de 75 millones de usuarios activos por mes, con una duración promedio de sesión de 23 min, a medida que realiza papeles empresariales increíblemente complicados entre bastidores. Y Spotify construyó una arquitectura de microservicio con grupos autónomos de pila completa a cargo para eludir el infierno de sincronización en la organización.

Lo cual a Spotify realmente le fascina de los microservicios es que no poseen gigantes fallos; los enormes servicios fallan, los servicios pequeños fallan en menor implicación. La obra de una arquitectura de microservicios posibilita a Spotify tener una enorme proporción de servicios simultáneamente sin que los usuarios lo noten. Han construido su sistema suponiendo que los servicios tienen la posibilidad de fracasar constantemente, por lo cual los servicios particulares que podrían estar fracasando no poseen demasiada implicación total, por lo cual no tienen la posibilidad de arruinar la vivencia de utilizar Spotify.

5.3. Amazon

En Amazon tenían una gigantesca cantidad de desarrolladores trabajando en un gran website monolítico, y aunque todos estos desarrolladores solo trabajaba en una parte bastante pequeña de esa aplicación, aún necesitaban ocuparse de coordinar sus cambios con todos los demás que también trabajaban en el mismo proyecto. Si querían hacer una solución instantánea para enviarla inmediatamente a sus clientes, no podían hacerlo en su propio calendario, tenían que coordinar eso con todos los demás desarrolladores que habían procesado los cambios al mismo tiempo.

Después de todos estos cambios, Amazon mejoró dramáticamente su lapso de vida de desarrollo front-end.

5.4. Netflix

La plataforma tiene una arquitectura común, que ha estado en uso durante dos años y ha comenzado a utilizar microservicios para ejecutar sus productos. Cada día recibe una media de mil millones de llamadas a sus diferentes servicios (se dice que es responsable del 30 % del tráfico de Internet), y es capaz de pasar su API de transmisión de vídeo, enviará cinco solicitudes a diferentes servidores para no perder nunca la continuidad de la transmisión.

5.5. eBay

En 2011, una vez que eBay estaba introduciendo microservicios, la compañía poseía 97 millones de usuarios activos y 62 mil millones de volumen bruto de mercancía. En un día regular, los sistemas de TI de eBay tenían que lidiar con un tráfico masivo, como 75 mil millones de denominadas a bases de datos, 4 mil millones de visitas a páginas y 250 mil millones de consultas de averiguación. El cambio a la arquitectura de microservicios no ha sido el primer cambio fundamental de tecnología en eBay; la organización confrontó transiciones semejantes en 1999 y 2005.

6. Bibliografía

[1] Arquitectura de microservicios: qué es, ventajas y desventajas - Decidesoluciones. Enlace a artículo.

[2] Qué son Microservicios y ejemplos reales de uso - OpenWebinars - Esaú Abril Nuñez Enlace a artículo.

[3] Microservicios. Todo lo que querías saber. - KeepCoding. Enlace a artículo.

[4] 10 balanceador de carga de código abierto para alta disponibilidad y rendimiento mejorado - GeekFlare - Chandan Kumar. Enlace a artículo.

[5] ¿Qué son los microservicios? - Deloitte. Enlace a artículo.

[6] Las 10 herramientas más importantes para orquestación de contenedores Docker - CampusMVP. Enlace a artículo.

[7] Beneficios y ejemplos de la implementación de los microservicios - ApiumHub - Ekaterina Novoseltseva. Enlace a artículo.