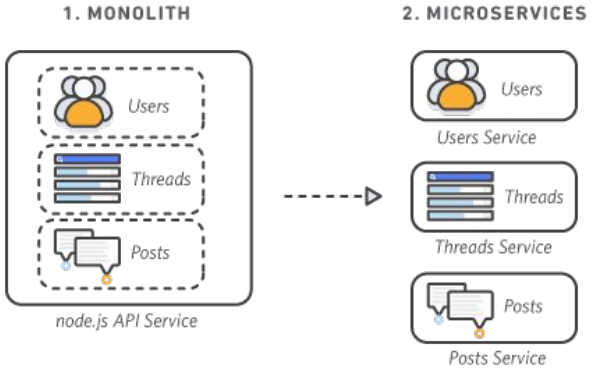
Microservicios

Grupo 3 | 2023 | UNLaM

¿Qué son los microservicios?

Los microservicios son un enfoque arquitectónico y organizativo para el desarrollo de software donde el software está compuesto por pequeños servicios independientes que se comunican a través de **API bien definidas**. Los propietarios de estos servicios son equipos pequeños independientes.

Las arquitecturas de microservicios hacen que las aplicaciones sean más fáciles de escalar y más rápidas de desarrollar. Esto permite la innovación y acelera el tiempo de comercialización de las nuevas características.



Arquitectura monolítica vs arquitectura de microservicios

Con las arquitecturas monolíticas, todos los procesos están estrechamente asociados y se ejecutan como un solo servicio. Esto significa que, si un proceso de una aplicación experimenta un pico de demanda, se debe escalar toda la arquitectura. Agregar o mejorar las características de una aplicación monolítica se vuelve más complejo a medida que crece la base de código. Esta complejidad limita la experimentación y dificulta la implementación de nuevas ideas. Las arquitecturas monolíticas aumentan el riesgo de la disponibilidad de la aplicación porque muchos procesos dependientes y estrechamente vinculados aumentan el impacto del error de un proceso.

Con una arquitectura de microservicios, una aplicación se crea con componentes independientes que ejecutan cada proceso de la aplicación como un servicio. Estos servicios se comunican a través de una interfaz bien definida mediante API ligeras. Los servicios se crean para las capacidades empresariales y cada servicio desempeña una sola función. Debido a que se ejecutan de forma independiente, cada servicio se puede actualizar, implementar y escalar para satisfacer la demanda de funciones específicas de una aplicación.

Características 1/2

Varios servicios de componentes

Los microservicios se componen de servicios de componentes individuales y poco vinculados que se pueden desarrollar, implementar, operar, cambiar y volver a implementar sin afectar al funcionamiento de otros servicios o a la integridad de una aplicación. Esto permite una implementación rápida y fácil de cada una de las funciones de una aplicación.

Muy fáciles de mantener y de probar

Los microservicios permiten a los equipos experimentar con nuevas funciones y revertirlas si no funcionan. Esto facilita la actualización del código y acelera el tiempo de salida al mercado de nuevas funciones. Además, simplifica el proceso de aislamiento y corrección de fallos y errores en los servicios individuales.

Pertenecen a equipos pequeños

Los equipos pequeños e independientes suelen crear un servicio dentro de microservicios, lo que los anima a adoptar prácticas de metodología ágil y de DevOps. Los equipos pueden trabajar de forma independiente y moverse rápidamente, lo que acorta el ciclo de desarrollo.

Características 2/2

Se organizan en torno a capacidades empresariales

Un enfoque de microservicios permite organizar los servicios en torno a las capacidades empresariales. Los equipos son multifuncionales, disponen de la gama completa de habilidades necesarias para el desarrollo y trabajan para crear una funcionalidad concreta.

Infraestructura automatizada

Los equipos que crean y mantienen microservicios suelen utilizar prácticas de automatización de infraestructuras, como la integración continua (CI), la entrega continua (CD) y la implementación continua (también CD). Esto permite a los equipos crear e implementar cada servicio de forma independiente sin afectar a los demás equipos, así como implementar una nueva versión de un servicio en paralelo con la versión anterior.

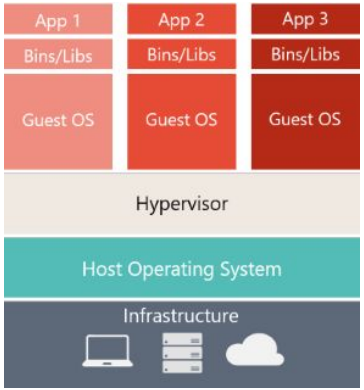
Microservicios y contenedores

En este contexto, **Docker** surge como una potente herramienta para trabajar con microservicios. Docker es una plataforma de código abierto que facilita el desarrollo, despliegue y gestión de aplicaciones a través de la contenedorización. Permite a los desarrolladores empaquetar aplicaciones y sus dependencias en contenedores ligeros y portátiles, garantizando que las aplicaciones se ejecuten de forma coherente en diferentes entornos y etapas de desarrollo. Al aprovechar Docker, los desarrolladores pueden agilizar eficazmente el proceso de creación, gestión y ampliación de microservicios.

Aunque Docker permite empaquetar y distribuir aplicaciones en contenedores de forma eficaz, es complicado ejecutar y gestionar contenedores a escala solo con esta herramienta.

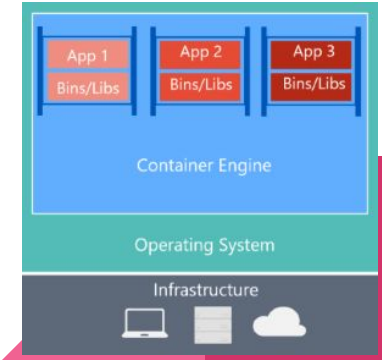
Kubernetes es una popular plataforma de código abierto que orquesta sistemas de tiempo de ejecución de contenedores en un clúster de recursos en red. Se puede usar con o sin Docker. Docker es un tiempo de ejecución de contenedores, mientras que Kubernetes es una plataforma para ejecutar y gestionar contenedores a partir de muchos tiempos de ejecución de contenedores.

Máquinas Virtuales vs Docker



Las máquinas virtuales incluyen la aplicación, las bibliotecas o los archivos binarios necesarios y **un sistema operativo invitado completo**. La virtualización completa requiere más recursos que la inclusión en contenedores.

Los contenedores incluyen la aplicación y todas sus dependencias. Sin embargo, comparten el kernel del sistema operativo con otros contenedores, que se ejecutan como procesos aislados en el espacio de usuario en el sistema operativo host. (Excepto en los contenedores de Hyper-V, en que cada contenedor se ejecuta dentro de una máquina virtual especial por contenedor).



OpenAPI y Swagger

OpenAPI es un estándar para la descripción de las interfaces de programación, o application programming interfaces (API). La especificación OpenAPI define un formato de descripción abierto e independiente de los fabricantes para los servicios de API. En particular, OpenAPI puede utilizarse para describir, desarrollar, probar y documentar las API compatibles con REST.

La actual especificación OpenAPI surgió del proyecto predecesor **Swagger**. La empresa de desarrollo SmartBear sometió la especificación existente de Swagger a una licencia abierta y dejó el mantenimiento y desarrollo posterior en manos de la iniciativa OpenAPI. Además de SmartBear, entre los miembros de la iniciativa OpenAPI se encuentran gigantes de la industria como Google, IBM y Microsoft. La Fundación Linux también apoya este proyecto.

Desafíos de los Microservicios 1/2

Expansión descontrolada

Pasar de una arquitectura monolítica a los microservicios conlleva una mayor complejidad. Hay más servicios en más lugares y creados por varios equipos. Esto hace que sea difícil determinar cómo se relacionan los diferentes componentes entre sí, quién es el propietario de un componente en particular y cómo evitar causar un impacto negativo en los componentes dependientes. Si no se gestiona la expansión, se reduce la velocidad de desarrollo y el rendimiento operativo. A medida que un sistema crece, requiere un equipo de operaciones experimentado para gestionar las reimplementaciones constantes y los cambios frecuentes en la arquitectura.

Falta de claridad en cuanto a la propiedad

Con una arquitectura de microservicios, es más difícil saber quién es el propietario de cada componente. Un equipo de DevOps puede ejecutar una combinación de API, bibliotecas de componentes, herramientas de monitorización e imágenes de Docker para que los usuarios implementen una aplicación. Es importante tener clara cierta información sobre los componentes, como sus propietarios, recursos y relaciones cambiantes con otros componentes. Es necesario tener una comunicación y coordinación precisas entre numerosos equipos para que todos los involucrados puedan encontrar fácilmente los conocimientos necesarios para entender un producto.

Desafíos de los Microservicios 2/2

Costes de infraestructura exponenciales

Cada nuevo microservicio que se añade a la implementación de producción viene con sus propios costes asociados al conjunto de pruebas, manuales de estrategias de implementación, infraestructura de alojamiento, herramientas de monitorización, etc.

Sobrecarga organizativa añadida

Se necesita un nivel adicional de comunicación y colaboración para coordinar las actualizaciones e interfaces entre los equipos de arquitectura de microservicios.

Depuración

Puede resultar difícil depurar una aplicación que contiene varios microservicios, cada uno con su propio conjunto de registros. Un solo proceso empresarial puede ejecutarse en varias máquinas en distintos momentos, lo que complica aún más la depuración.

Conclusión

En resumen, la arquitectura de microservicios representa un enfoque de diseño de software que busca dividir una aplicación en componentes independientes y altamente especializados llamados microservicios. Estos microservicios se comunican entre sí a través de APIs y funcionan de manera autónoma, lo que ofrece una serie de ventajas, como escalabilidad, agilidad y mantenibilidad.

Es importante recordar que la elección de la arquitectura de microservicios no es adecuada para todos los proyectos. Debe considerarse en situaciones donde la modularidad, la escalabilidad y la independencia de implementación son cruciales. Algunos casos de uso ideales incluyen aplicaciones web y móviles grandes y complejas, proyectos que requieren actualizaciones y despliegues frecuentes, y equipos de desarrollo distribuidos.

En última instancia, la adopción de una arquitectura de microservicios debe basarse en las necesidades específicas del proyecto y el entorno de desarrollo. Si se implementa de manera adecuada, esta arquitectura puede ofrecer beneficios significativos en términos de flexibilidad y eficiencia, lo que la convierte en una opción poderosa para aplicaciones modernas y escalables.

¡Gracias!