Microservices

Ing. Ronaldo Tovar Reyes **10-05-2024**

Introducción.

En un principio la forma estándar de desarrollar aplicaciones era con una arquitectura monolítica, significa que; todos los componentes de la aplicación y el código completo básicamente es parte de una sola unidad, es decir, todo se desarrolla, se implementa y se escala como una unidad, lo que implica que la aplicación debe estar escrita en un solo lenguaje, como una pila de tecnología con un único tiempo de ejecución y si se tienen diferentes equipos trabajando en diferentes partes de la aplicación, deberá coordinarse para asegurar que no afecten el trabajo de los demás, ejemplificando una aplicación de tienda online, en donde los componentes de carrito de compra, catálogo de productos, autenticación de usuarios, ventas, todo englobado en un solo componente. Esto repercute en que, si los desarrolladores quieren modificar el código para funcionalidades de pago, necesitaría crear toda la aplicación e implementarlo como un paquete, no permitiría simplemente actualizar e implementar solo los cambios de la funcionalidad de pago por separado, a medida que estas aplicaciones crecieron en tamaño y complejidad. Esto impacta en costos de infraestructura y menos flexibilidad para escalar la aplicación, ya que en términos específicos si se requería de una actualización o cualquier característica por qué es necesario probar y crear toda la aplicación completa. Es por esta razón que surgen los micro servicios, para dar una arquitectura más flexible para las aplicaciones. Con el crecimiento en tamaño y complejidad de las aplicaciones, las limitaciones de la arquitectura monolítica impulsaron la evolución hacia modelos más ágiles y eficientes.

Micro servicios.

Los micro servicios llegan como solución de descomponer una gran aplicación en aplicaciones más pequeñas, es decir, los micro servicios definen como dividir la aplicación en código y como se van a comunicar entre sí todas las pequeñas aplicaciones, por lo tanto, la mejor práctica será dividir la aplicación en componentes basados en funcionalidades comerciales y no en funcionalidades técnicas. Ejemplo en la aplicación antes mencionada de tienda en línea estas componentes serán carrito de compras, productos, cuentas de usuario, pago, etc. Es decir, cada uno de estos componentes comerciales debe ser unos solo aislado, esto es una característica importante, ya que significa que cada servicio será escalable, se podrá desarrollar e implementarse por separado, sin dependencias estrictas de ningún otro servicio, aun a pesar de formar parte de la misma aplicación, a esto se le conoce como perdida de acoplamiento.

Volviendo al ejemplo, si se utiliza este enfoque y requiere hacer una actualización en el servicio de pago, solo este servicio se verá afectado, ya que es completamente independiente y solo se creará e implementará este componente.

Microservices

Ing. Ronaldo Tovar Reyes **10-05-2024**

Comunicación De Micro Servicios.

Al ser los micro servicios componentes aislados, cómo se conectan entre sí, de modo que, el servicio de pago debe tener información de la cuenta del usuario y del carrito de compras, en este caso la forma más común para la comunicación de micro servicios es utilizar las API, por lo tanto, cada uno tendrá un punto final en el que acepta solicitudes de otros, de modo que los servicios pueden comunicarse entre sí enviándose solicitudes HTTP que es una comunicación sincrónica.

La comunicación asincrónica es otra forma común y utiliza un intermediario de mensajes como Rabbitmq y luego el corredor de mensajes reenviará ese mensaje al servicio respectivo.

La tercera forma de comunicación dentro del campo de Kubernetes, que utiliza una malla de servicios con otra malla de servicios, que tiene una especie de servicio auxiliar que se hace cargo de la lógica de comunicación, por lo que no se tiene que codificar esta lógica en los micro servicios y se tiene delegada esta lógica a esté servicio externo.

Al trabajar con componentes aislados la principal ventaja es que cada servicio puede trabajarse con un lenguaje de programación diferente.

Otra ventaja principal es tener equipos dedicados para cada servicio y pueden elegir su propia pila de tecnología y trabajar en su servicio sin afectar a otros equipos de servicio.

Sin embargo, estos beneficios repercuten en que, si un micro servicio está inactivo o en mal estado y otros micro servicios comienzan a enviar solicitudes API y esperan una respuesta cumplida, se obtienen resultados inesperados. También con los servicios aislados se puede resultar difícil mantener una visión general y descubrir cuándo un servicio no funciona o que servicio realmente no funciona cuando algo en la aplicación no funciona correctamente.

Por este problema surge la herramienta de Kubernetes, que es una plataforma perfecta para ejecutar grandes aplicaciones de micro servicios.

Existe un elemento importante al implementar microservicios y es una peeline CI/CD que sirven para implementar cientos de micro servicios miles de veces al día, en este caso se necesitaría saber cómo configurar el proceso de lanzamiento con una canalización de CI/CD.

Para administrar el código de nuestro micro servicio en un proyecto podemos hacerlo con un repositorio de git y es simple, ya que, solo se tiene una aplicación y su propio repositorio, es aquí, donde tenemos dos formas de administrar el código el primero es **Monorepo;** que significa repositorio único y **Polirepo;** repositorio múltiple. La principal diferencia es que, a medida que se escala la aplicación resulta

más difícil administrar el código en un Monorepo, ya que, se vuelve lenta la búsqueda, clonación y el envío. Otra desventaja es que al trabajarse en una rama principal del repositorio si algún desarrollador rompe la rama otros servicios se verán afectados.

Polirepo utiliza un repositorio para cada servicio para que el código este totalmente aisaldo, también puede clonarlos y trabajar en ellos por separado, para facilitar los repositorios se pueden agrupar por aplicación, esto ayuda a mantener una visión general de quien conforma cada proyecto. Sin embargo, la principal desventaja es que puede ser tedioso trabajar en dos servicios a la vez porque una característica o corrección de error afecta a múltiples servicios, además de buscar algo en varios proyectos desde el editor de código puede ser difícil o imposible, incluso no se pueden compartir archivos en el proyecto como Kubernetes, ya que se tendrían que duplicar en los repositorios de cada proyecto.

Conclusión

En resumen, los micro servicios emergen como una respuesta a los desafíos de las arquitecturas monolíticas al permitir la descomposición de aplicaciones en unidades autónomas y escalables. Este enfoque moderno no solo ofrece flexibilidad y eficiencia, sino que también impulsa una mayor capacidad de adaptación a las demandas del mercado y mejora la eficacia del proceso de desarrollo de software en su conjunto.