Ashan Fernando Seguir
AWS Certified Solutions Architect Professional. Para obtener más detalles, encuéntreme en Linkedin https://www.linkedin.com/in/ashan256/
18 de enero · 2 minutos de lectura

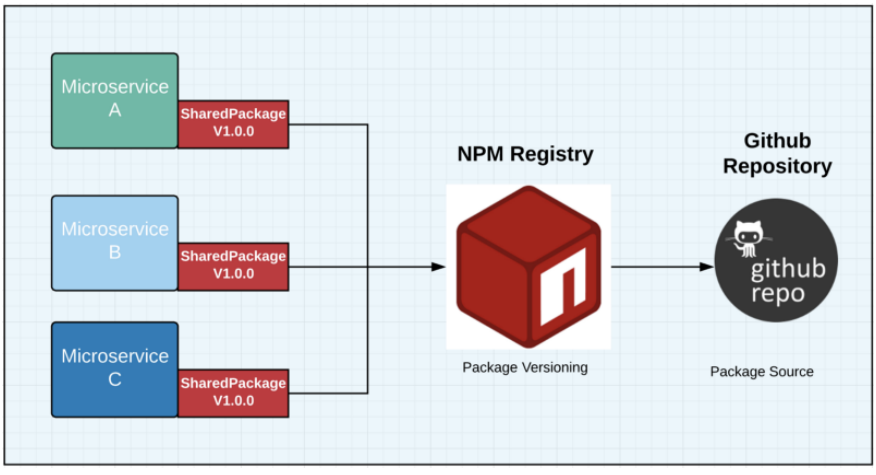
¿Podemos reutilizar el código entre microservicios?

Esta es una de las preguntas fundamentales que me han hecho muchos, que comienza a construir proyectos con Microservicios. De hecho, también causa mucha confusión ya que Microservicios debe ser autónomo y funciona como "Reutilizar", "Compartir" puede causar desafíos fundamentales al concepto.

> *Algunos de ustedes se preguntarán, ¿por qué compartir el código de una manera tradicional es un problema para Microservicios? Esto se debe principalmente a que nuevamente terminará en un caos de dependencia, a la larga, convirtiendo el conjunto de Microservicios en un monolito. Por lo tanto, es importante mantener la naturaleza autónoma de Microservicios, donde cada servicio puede ser desarrollado de forma independiente por un equipo y desplegado sin afectar a los demás al usar diferentes versiones de código compartido.*

Este artículo lo guiará a través de algunos enfoques para construir microservicios robustos y autónomos sin duplicación de código.

En primer lugar, es realmente importante no duplicar el código entre Microservicios, especialmente si se está copiando y pegando el mismo código. Puede causar más daño que tener dependencias.



> *El enfoque que recomiendo es usar un repositorio o registro externo, para mantener el código compartido (por ejemplo, Utils) y usar una versión específica del mismo dentro de un Microservicio.*

De esta manera puede mantener los Microservicios autónomos al tener su propio ciclo de vida e incluso el código compartido se cambia a una nueva versión, los servicios que tienen la necesidad de usar la versión más nueva pueden usarlo, sin afectar a los demás.

Unidades de código reutilizables como módulos

También es importante entender cómo estructuramos cada módulo de código reutilizable. A menudo uso un repositorio de Github con su propio ciclo de vida y versiones semánticas donde cada módulo se construirá, se probará independientemente de los demás para construir componentes robustos (unidades de código) para compartir entre microservicios.

Usando un administrador de paquetes

Una de las herramientas que recomiendo encarecidamente es usar un administrador de paquetes. Por ejemplo, si está desarrollando aplicaciones con NodeJS, prefiero usar el Administrador de paquetes de NPM. Requerirá un repositorio privado para proyectos comerciales y, a menudo, recomiendo crear un NPM Orgs que admita espacios de nombres para módulos. Esto requerirá una versión paga pero vale la pena.

Usando Source Control

Cuando crea módulos compartibles y los almacena en un repositorio de Git, puede usar las etiquetas de Git para versionar cada versión y usarla en sus microservicios. Si estás usando NPM. Puede definirlo en el paquete.json directamente refiriendo el hash de confirmación con la URL.

```
{
"Nombre": "microservicio-a",

"Versión": "1.0.0",
```

```
"Dependencias": {

"Serverless-dynamodb-local": "git + https:
//github.com/99xt/serverless-dynamodb-
local#67ef2998624e10f5ef734fbcc6b70b471b057ed0"

}
}
```

Para obtener más detalles, consulte las *URL de Github como* sección de *dependencias* en la documentación de NPM.