# COMENTARIOS FINALES

Este archivo será una breve discusión sobre los puntos débiles de la arquitectura en cuanto a fiabilidad y escalabilidad, mencionando alguna solución a los problemas detectados en los diferentes bloques.

El primer bloque se basa en una MV pesada.
Las aplicaciones monolíticas, es decir, las aplicaciones que se ejecutan en una sola máquina virtual tienen varios puntos débiles en cuanto a fiabilidad y escalabilidad.
Una de las principales desventajas es que son más propensas a fallos. Si un componente de la aplicación falla, toda la aplicación puede dejar de funcionar.
Aunque en el desarrollo no nos hayamos topado con ningún fallo.

En el segundo bloque usamos Docker.
Los contenedores Docker pueden proporcionar una serie de ventajas en términos de fiabilidad y escalabilidad, en comparación con las aplicaciones monolíticas desplegadas en máquinas virtuales.
Los contenedores son aislantes, lo cual nos ayuda a solucionar el problema detectado en el anterior bloque.
Además, los contenedores son ligeros. Esto significa que requieren menos recursos que las máquinas virtuales lo que significa mayor rendimiento. Esto ayuda a reducir el impacto de un fallo en un contenedor en la aplicación general.
Por último, los contenedores son escalables. Se pueden crear nuevos contenedores para agregar capacidad a una aplicación.

En el tercer bloque se añaden más servicios y usamos Docker Compose: Docker Compose permite definir aplicaciones de forma declarativa. Esto significa que se pueden definir las dependencias de la aplicación y la configuración de los contenedores en un archivo de YAML. Esto facilita la gestión de la aplicación y nos permite añadir los dos servicios de forma muy fácil.
Además, Docker Compose nos permite definir réplicas de los contenedores. Esto permite escalar la aplicación agregando más contenedores de un tipo de servicio determinado.

En el último bloque lo hemos implementado en Kubernetes:
Para este, tuvimos que crear otra instancia de Google Cloud en la que hicimos todo el desarrollo, que, aunque de primeras llevó más tiempo al tener que crear de cero el entorno, el desarrollo se hizo mucho más sencillo gracias a K8s.
Kubernetes permite orquestar contenedores, lo que significa que puede administrar automáticamente la creación, el escalado y la destrucción de contenedores. Esto puede ayudar a mejorar la fiabilidad de las aplicaciones al garantizar que siempre haya suficientes contenedores disponibles para satisfacer la demanda.
Además de que usa el mismo sistema réplicas que nos solucionaba los problemas del bloque 1 y bloque 2.

Por último, decidimos hacer la parte opcional con Helm Charts.
Si K8s hacía más fácil y mejor el desarrollo de la aplicación, Helm lo simplifica aún más. Como se puede ver en los pasos de la implementación con 4 comandos usando helm ya tenemos la aplicación corriendo con las mismas ventajas de fiabilidad y escalabilidad que K8s. Nos ha soprendido esta última tecnología.

Eric, Luis y Hernán