Universidad de Guadalajara
Sistema de Educación Media Superior
Centro Universitario de Ciencias Exactas e Ingenierías

A Netflix guide to Microservices

Materia: Computación Tolerante a Fallas

D06 2023 B

Alumno: Esquivel Barbosa Diego Humberto

Código: 211401635

Carrera: INCO

Fecha: 20/11/2023

Introducción

Los microservicios son una forma de diseñar y construir sistemas de software distribuidos, donde cada componente tiene una responsabilidad única y se comunica con los demás mediante interfaces bien definidas. Esta arquitectura permite escalar, desplegar y evolucionar los servicios de forma independiente, así como aprovechar la diversidad de lenguajes, plataformas y tecnologías disponibles.

Netflix es una de las empresas pioneras y líderes en el uso de microservicios, ya que ha logrado crear una plataforma global de streaming de vídeo que atiende a más de 200 millones de usuarios y ofrece miles de contenidos personalizados. Para lograr esto, Netflix ha tenido que superar numerosos desafíos técnicos y organizativos, así como adoptar una cultura de innovación, experimentación y resiliencia.

En esta presentación, Josh Evans, ex director de ingeniería de Netflix, nos guía por el caótico y vibrante mundo de los microservicios en Netflix, explicando los conceptos básicos, los retos, las soluciones y las lecciones aprendidas.

Desarrollo

Código monolítico. Todos contribuían a una única base de código que se liberaba una vez a la semana, cuando un cambio introducía un problema era difícil y lento de depurar.
Servicios monolíticos. Cada servicio tenía múltiples responsabilidades y dependencias, lo que dificultaba su escalabilidad, mantenibilidad y evolución.
Arquitectura monolítica. Todos los servicios seguían el mismo patrón y tecnología, lo que limitaba la innovación y la adaptación a las necesidades de cada caso.
Lo que los microservicios son:

Código modular. Cada servicio tiene su propio repositorio, ciclo de vida y equipo responsable, lo que facilita la integración continua, el despliegue continuo y la resolución de problemas.
Servicios modulares. Cada servicio tiene una única responsabilidad y una interfaz bien definida, lo que permite su escalabilidad, mantenibilidad y evolución.
Arquitectura modular. Cada servicio puede elegir el patrón y la tecnología más adecuados para su caso, lo que fomenta la innovación y la adaptación a las necesidades de cada caso.

Desafíos y soluciones

Dependencia

Uno de los principales desafíos de los microservicios es gestionar la dependencia entre ellos, es decir, cómo se comunican, cómo se coordinan y cómo se toleran los fallos. Algunas de las soluciones que Netflix ha aplicado son:

API REST. Los servicios se comunican mediante interfaces HTTP que siguen el estilo arquitectónico REST, que permite una comunicación simple, uniforme y sin estado entre los recursos.

Hystrix. Es una biblioteca de código abierto desarrollada por Netflix que implementa el patrón de circuit breaker, que permite cortar la comunicación con un servicio cuando detecta que está fallando, evitando así la propagación de errores y la degradación del rendimiento.

Ribbon. Es otra biblioteca de código abierto desarrollada por Netflix que implementa el patrón de cliente inteligente, que permite descubrir y balancear la carga entre las instancias de un servicio, así como aplicar políticas de reintentos y timeouts.

Zuul. Es un servicio de borde que actúa como proxy inverso y enrutador, que permite filtrar, monitorizar y transformar las peticiones que llegan desde los clientes a los servicios internos.

Chaos Monkey. Es una herramienta de código abierto desarrollada por Netflix que forma parte de la suite de Chaos Engineering, que consiste en inyectar fallos de forma controlada en los servicios para comprobar su resiliencia y capacidad de recuperación.

Escala

Otro de los principales desafíos de los microservicios es gestionar la escala, tanto en términos de volumen de datos, como de tráfico, como de complejidad. Algunas de las soluciones que Netflix ha aplicado son:

- Servicios sin estado. Los servicios que no necesitan almacenar información sobre el estado de las peticiones o las sesiones de los usuarios, pueden escalar horizontalmente de forma sencilla, añadiendo o quitando instancias según la demanda.

- Servicios con estado. Los servicios que sí necesitan almacenar información sobre el estado de las peticiones o las sesiones de los usuarios, pueden escalar mediante técnicas de particionamiento, replicación y caché, que permiten distribuir y sincronizar los datos entre las instancias.

- Servicios híbridos. Los servicios que combinan características de los servicios sin estado y con estado, pueden escalar mediante técnicas de enrutamiento, segmentación y agregación, que permiten dirigir y procesar las peticiones de forma óptima.

Varianza dentro de la arquitectura

Otro de los principales desafíos de los microservicios es gestionar la varianza dentro de la arquitectura, es decir, cómo se maneja la diversidad de patrones, tecnologías y prácticas que existen entre los servicios. Algunas de las soluciones que Netflix ha aplicado son:

- Deriva operacional, que ocurre con el tiempo. Los servicios pueden divergir en sus configuraciones, versiones y dependencias, lo que puede provocar inconsistencias e incompatibilidades. Para evitar esto, Netflix ha establecido una lista de verificación de producción, que incluye los requisitos mínimos que debe cumplir un servicio para estar listo para el despliegue, como por ejemplo: tener métricas, alertas, logs, documentación, pruebas, etc.

- Políglota (nuevos lenguajes) y contenedores. Los servicios pueden elegir el lenguaje y la plataforma más adecuados para su caso, lo que puede aportar ventajas de rendimiento, productividad y expresividad. Para facilitar esto, Netflix ha adoptado el uso de contenedores, que permiten empaquetar y aislar los servicios con todas sus dependencias, así como desplegarlos y ejecutarlos de forma homogénea en cualquier entorno.

- Coste de la varianza. La varianza tiene un coste asociado, tanto en términos de recursos, como de tiempo, como de conocimiento. Para minimizar este coste, Netflix ha desarrollado una serie de plataformas y herramientas comunes que ofrecen funcionalidades transversales a los servicios, como por ejemplo: seguridad, monitorización, despliegue, experimentación, etc.

Organización y arquitectura

La organización y la arquitectura de los microservicios están estrechamente relacionadas, como lo expresa la ley de Conway, que dice que las organizaciones tienden a diseñar sistemas que reflejan su estructura de comunicación. Algunos de los aspectos que Netflix ha tenido en cuenta son:

Ley de Conway. Netflix ha organizado sus equipos en torno a los servicios, de forma que cada equipo tiene la autonomía y la responsabilidad de diseñar, desarrollar, desplegar y operar su servicio, siguiendo el principio de "you build it, you run it". Esto permite una mayor agilidad, innovación y alineación con los objetivos de negocio.

Resultados y lecciones. Netflix ha obtenido una serie de resultados y lecciones de su experiencia con los microservicios, entre los que se destacan:

- Los microservicios permiten escalar el sistema y la organización de forma eficiente y efectiva, pero también introducen una mayor complejidad y varianza que hay que gestionar.

- Los microservicios requieren una cultura de confianza, colaboración y aprendizaje, donde los equipos tienen libertad para tomar decisiones, pero también asumen la responsabilidad de sus acciones y resultados.

- Los microservicios exigen una práctica continua de ingeniería del caos, donde se simulan y se anticipan los posibles fallos y se diseñan soluciones robustas y resilientes.

Conclusión

Mastering Chaos - A Netflix Guide to Microservices es una presentación que nos ofrece una visión global y detallada de cómo Netflix ha adoptado y evolucionado su arquitectura de microservicios, así como los beneficios y los desafíos que ha obtenido y superado. Se trata de un recurso muy valioso y didáctico para aprender sobre los microservicios y cómo aplicarlos en la práctica.

Bibliografía

Docker Desktop. (n.d.). Docker. https://www.docker.com/products/docker-desktop/

Alpine Linux. (n.d.). Alpine Linux. https://alpinelinux.org/

Node.js. (n.d.). Node.js. https://nodejs.org/

Express - Node.js web application framework. (n.d.). Express.js. https://expressjs.com/

Nginx. (n.d.). Nginx. https://www.nginx.com/

Merkle, S. (2018). Docker Containers: Build and Deploy with Kubernetes, Flannel, Cockpit, and Atomic. Apress.

Lebkov, R., & Prikshet, P. (2018). Docker: Up and Running: Shipping Reliable Containers in Production. O'Reilly Media.

Boyd, C., & Johnston, J. (2016). Docker in Action. Manning Publications.