Kubernetes (k8s)

> ⚠ **DOCUMENTO EN DESARROLLO** ⚠

Orchestrating Systems with Kubernetes

Hace años, la mayoría de las aplicaciones de software eran grandes monolitos, que se ejecutaban como un solo proceso o como una pequeña cantidad de procesos distribuidos en un puñado de servidores. Estos sistemas heredados todavía están muy difundidos en la actualidad. Tienen ciclos de liberación lentos y se actualizan con poca frecuencia. Al final de cada ciclo de lanzamiento, los desarrolladores empaquetan todo el sistema y se lo entregan al equipo de operaciones, quien luego lo implementa y lo supervisa. En caso de fallas de hardware, el equipo de operaciones lo migra manualmente a los servidores en buen estado restantes.

Hoy en día, estas grandes aplicaciones heredadas monolíticas se están dividiendo lentamente en componentes más pequeños, que se ejecutan de forma independiente, llamados microservicios. Debido a que los microservicios están desacoplados entre sí, se pueden desarrollar, implementar, actualizar y escalar individualmente. Esto le permite cambiar componentes rápidamente y con la frecuencia que sea necesaria para mantenerse al día con los cambiantes requisitos comerciales de hoy.

Pero con un mayor número de componentes desplegables y centros de datos cada vez más grandes, cada vez es más difícil configurar, administrar y mantener todo el sistema funcionando sin problemas. Es mucho más difícil averiguar dónde colocar cada uno de esos componentes para lograr una alta utilización de los recursos y, por lo tanto, mantener bajos los costos de hardware. Hacer todo esto manualmente es un trabajo duro. Necesitamos automatización, que incluye la programación automática de esos componentes en nuestros servidores, la configuración automática, la supervisión y el manejo de fallas. Aquí es donde entra **Kubernetes**.

Kubernetes permite a los desarrolladores implementar sus aplicaciones por sí mismos y con la frecuencia que deseen, sin necesidad de asistencia del equipo de operaciones (ops). Pero Kubernetes no solo beneficia a los desarrolladores. También ayuda al equipo de operaciones al monitorizar y reprogramar automáticamente esas aplicaciones en caso de una falla de hardware. El enfoque para los administradores de sistemas (*sysadmins*) pasa de supervisar aplicaciones individuales a supervisar y administrar principalmente a Kubernetes y al resto de la infraestructura, mientras que Kubernetes se encarga de las aplicaciones.

Kubernetes abstrae la infraestructura de hardware y expone todo su centro de datos como un enorme recurso computacional. Le permite implementar y ejecutar sus componentes de software sin tener que conocer los servidores reales que se encuentran debajo. Al implementar una aplicación de múltiples componentes a través de Kubernetes, selecciona un servidor para cada componente, lo implementa y le permite encontrar y comunicarse fácilmente con todos los demás componentes de su aplicación.

Monolithic apps versus microservices

Las aplicaciones monolíticas consisten en componentes que están todos estrechamente acoplados y que deben desarrollarse, implementarse y gestionarse como una entidad, porque todos se ejecutan como un solo proceso de sistema operativo. Los cambios en una parte de la aplicación requieren una redistribución de toda la aplicación y, con el tiempo, la falta de límites entre las partes da como resultado un aumento de la complejidad y un deterioro consecuente de la calidad de todo el sistema debido al crecimiento ilimitado de las dependencias entre estas partes.

Estos y otros problemas han obligado a comenzar a dividir aplicaciones monolíticas complejas en componentes desplegables independientes más pequeños llamados **microservicios**. Cada microservicio se ejecuta como un proceso independiente y se comunica con otros microservicios a través de interfaces simples y bien definidas (API).

Los microservicios se comunican a través de protocolos síncronos como HTTP, a través de los cuales generalmente exponen las API RESTful (REpresentational State Transfer), o a través de protocolos asíncronos como AMQP (Advanced Message Queue Protocol).

Cada microservicio se puede escribir en el lenguaje más apropiado para implementar ese microservicio específico.

Debido a que cada microservicio es un proceso independiente con una API externa relativamente estática, es posible desarrollar e implementar cada microservicio por separado. Un cambio en uno de ellos no requiere cambios ni redistribución de ningún otro servicio, siempre que la API no cambie o cambie solo de una manera compatible con versiones anteriores.

Uno de los inconvenientes de los microservicios es que su número puede ser elevado y por tanto se complica gestionar su número. Además, cada microservicio puede necesitar diferentes versiones de librerías.

Para solucionar estos problemas tenemos la tecnología de contenederos, con Docker, y una forma de gestionar u orquestarlos, con Kubernetes. Gracias a Docker, podemos crear microservicios aislados y autocontenidos con sus propias dependencias, librerías y componentes, de forma que no interfieran entre sí y con un bajo consumo de recursos, a diferencia de las Máquinas Virtuales (VM). Esto permite tener varios contenedores en una misma máquina anfitrión.

Enlaces

Kubernetes

- ◆ Kubernetes | Production-Grade Container Orchestration
- Kubernetes for workstations and appliances

Kubernetes - Learning

- Learning-K8S - Keep it simple
- https://cheatsheets.zip/kubernetes
- https://roadmap.sh/kubernetes
- https://github.com/collabnix/dockerlabs/blob/master/kubernetes/cheatsheets/kubectl.md

Licencia