Contenedores y apliaciones

Alberto Molina Coballes
José Domingo Muñoz Rodríguez

IES Gonzalo Nazareno

25 de enero de 2021

Contenedores

- Contenedores
- ¿Para qué sirve un sistema operativo?¿Qué es un proceso?
- Compartir o no compartir, ésa es la cuestión: .so., dependencias
- ¿Qué es un contenedor y para qué se utiliza?
- Precedentes en linux:
  - chroot
  - OpenVZ
  - Linux vservers
- Precedentes en otros sistemas operativos: FreeBSD Jails, Solaris Zones, etc.

Contenedores

- El gran hito: inclusión de cgroups y namespaces en el kérnel linux (a partir de 2007)
- cgroups (límite de memoria, cpu, I/O o red para un proceso y sus hijos) https://wiki.archlinux.org/index.php/Cgroups
- cgroupsv2 (rootless containers) https://medium.com/nttlabs/cgroup-v2-596d035be4d7
- namespaces: proporcionan un punto de vista diferente a un proceso (interfaces de red, procesos, usuarios, etc.) http://laurel.datsi.fi.upm.es/~ssoo/SOA/namespaces.html
- Todo esto unido a la expansión de linux en el centro de datos ha provocado la explosión en el uso de contenedores de los últimos años

Despliegue de aplicaciones

Aplicación monolítica

- Todos los componentes en el mismo nodo
- Escalado vertical
- Arquitectura muy sencilla
- Suele utilizarse un solo lenguaje (o un conjunto pequeño de ellos)
- No pueden utilizarse diferentes versiones de un lenguaje a la vez
- Interferencias entre componentes en producción
- Complejidad en las actualizaciones. Puede ocasionar paradas en producción
- Normalmente usan infraestructura estática y fija por años
- Típicamente la aplicación no es tolerante a fallos

Aplicación distribuida

- Idealmente un componente por nodo
- Escalado horizontal
- Arquitectura más compleja
- Menos interferencias entre componentes
- Mayor simplicidad en las actualizaciones
- Diferentes enfoques no excluyentes: SOA, cloud native, microservicios, …

SOA

- SOA: Service Oriented Architecture
- Servicios independientes
- Múltiples tecnologías, lenguajes y/o versiones interactuando
- Comunicación vía SOAP
- Uso de XML, XSD y WSDL
- Colas de mensajes
- Se relaciona con aplicaciones corporativas
- Se le achaca mucha complejidad y no ha terminado de extenderse

Cloud Native Application

- Énfasis en la adaptación de la infraestructura a la demanda
- Uso extensivo de la elasticidad: Infraestructura dinámica
- Aplicaciones resilientes
- Elasticidad horizontal
- Automatización
- Puede ser complejo de implementar en una aplicación

Aplicación con o sin estado

Aplicaciones con estado

- La operación se realiza en un contexto de un estado anterior
- Típicamente requerirá que se interactúe con el mismo servidor
- Suelen hacer uso de algún tipo de almacenamiento permanente
- Para escalar se pueden utilizar clusters de alta disponibilidad

Ejemplo: Webmail

Aplicaciones sin estado

- Cada operación independiente de las anteriores o posteriores
- Puede interactuar cada vez con un servidor diferente
- No necesita almacenamiento permanente o solo de lectura
- Para escalar puede utilizar clusters de balanceo de carga

Ejemplo: Buscador de Internet

Microservicios

- Deriva del esquema SOA
- No existe una definición formal, ni WSDL, XSD, etc. Solución más pragmática
- Servicios llevados a la mínima expresión (idealmente un proceso por nodo): Microservicios
- Comunicación vía HTTP REST y colas de mensajes, ¿gRPC?
- Relacionado con procesos ágiles de desarrollo, facilita enormemente la actualizaciones de versiones, llegando incluso a la entrega continua o despliegue continuo.
- Suele implementarse sobre contenedores
- Muy adecuada para aplicaciones sin estado
- Aumento de la latencia entre componentes
- Puede utilizar características de “cloud native application”

Microservicios. Ejemplo

OpenStack

- Cada componente es un microservicio que puede ejecutarse en un nodo independiente
- kolla-ansible: Despliegue de OpenStack con ansible en múltiples contenedores docker

https://elatov.github.io/2018/01/openstack-ansible-and-kolla-on-ubuntu-1604/