Docker Up and Running

Víctor Iranzo

27 de julio de 2018

Docker es una herramienta para la creación de artefactos que se pueden desplegar en cualquier entorno y se pueden distribuir y escalar bajo demanda. Es una herramienta que agiliza el proceso de desarrollo de software de las empresas.

La publicación a producción de software a la velocidad que se espera hoy en día es difícil de conseguir, sobretodo en empresas de gran tamaño donde los procesos de comunicación son más complejos. En estas empresas, el hecho de actualizar la versión de una librería puede suponer la coordinación de los equipos que la emplean. Docker construye límites que aíslan un producto de otro aunque se estén ejecutando sobre el mismo host.

La filosofía de Docker se centra en los contenedores desechables. Nada en el entorno donde se ejecuta la aplicación permanecerá ahí más allá de lo que dure la aplicación, por lo que no se delegará en entornos donde se encuentra casualmente un artefacto no especificado. Las dependencias del sistema o se incluyen directamente en el proceso de despliegue o son dependencias externas, como una base de datos. Las aplicaciones así se hacen más portables, escalables y confiables.

Si se integra dentro del proceso de desarrollo, cada cambio en el sistema de control de versiones puede iniciar una pipeline. En la pipeline se construye una nueva imagen Docker sobre la que se ejecutan pruebas de forma automatizada para después ser publicada a producción.

En cuanto a los recursos consumidos, un contenedor es un simple proceso que habla directamente con el kernel de Linux sobre el que se ejecuta. Esto no significa que la virtualización tradicional se pueda sustituir siempre por contenedores: las máquinas virtuales garantizan ... pero consumen muchos más recursos del sistema.

En el proceso de desarrollo de un sistema basado en contenedores, las tareas de construcción de la imagen, provisión de la configuración y despliegue

pueden ser realizadas por diferentes personas. De esta forma el equipo puede asegurarse de que el artefacto sobre el que se han ejecutado las pruebas es el mismo que se publica más tarde. El proceso de desarrollo también se simplifica: en aproximaciones como la de microservicios, un equipo encargado de un servicio solo necesita la imagen de un servicio para integrarlo con el suyo y no necesita conocer detalles internos de este.

Docker sigue una arquitectura cliente/servidor: los clientes son los responsables de indicar a los servidores qué hacer, mientras que los segundos se centran en la gestión de las aplicaciones contenerizadas. El servidor actúa como un puente entre los contenedores y los clientes y cada contenedor tiene su propia dirección IP.

Docker promueve las arquitecturas sin estado (stateless) o que externalizan este a otras infraestructuras como bases de datos. Sin embargo, se pueden crear contenedores que sí almacenan estado a través de bases de datos que pertenecen al contenedor, aunque estas serán desechadas al eliminar el contenedor. También se puede acceder desde un contenedor al sistema de archivos del host donde se ejecuta, aunque esto tampoco es recomendable porque introduce una relación de dependencia del contenedor hacia el host.

Es muy sencillo entender el resultado de crear una imagen porque cada sentencia del Dockerfile que la origina crea una nueva capa en la imagen. En la compilación solo se generará un artefacto: la imagen. Esta imagen puede ser más tarde desplegada sin apenas complejidad.