Introducción a docker

Sistemas de Gestión Empresarial

IES Clara del Rey

Índice

Conceptos previos

Virtualización

La virtualización es un conjunto de tecnologías de hardware y software que permiten la abstracción de hardware, creando así la "ilusión" de administrar recursos virtuales como si fueran recursos reales, de forma transparente para los usuarios.

Máquina virtual

Una máquina virtual permite simular una máquina (con su sistema operativo) y ejecutar programas como si estuvieran utilizando una máquina real e independiente.

Conceptos previos

Tecnologías

- Máquinas virtuales de proceso.
- Emuladores.
- Hipervisores.
- Contenedores. **(Docker se engloba en esta categoría)**

Máquina virtual de proceso

Máquinas virtuales que permiten ejecutar un programa diseñado para un sistema operativo/arquitectura concreta (distinta de la máquina actual), como un proceso más de nuestra máquina actual.

Ejemplos de este tipo de virtualización son:

- **Máquina virtual de Java (JVM)**: ejecuta los bytecodes de Java en cualquier sistema y arquitectura que la tenga implementada.
- **Plataforma .NET**: análoga a la máquina virtual de Java con productos Microsoft.

Emulador

Un emulador es un software encargado de emular un hardware completo muy específico. Como ejemplos tenemos

- emuladores de videoconsolas antiguas
- una API concreta como **Wine**, que permite ejecutar aplicaciones Windows sobre Linux

Hipervisor

Un hipervisor, es una máquina virtual que simula total o parcialmente un hardware de una máquina, permitiendo la instalación de distintos sistemas operativos.

Como ejemplos típicos tenemos **VMWare** o **VirtualBox.**

Contenedores

Los contenedores son una tecnología de virtualización que utiliza el sistema base de la máquina anfitrión y actúa realmente como un "entorno privado" que comparte recursos con el sistema anfitrión, sin virtualizar el hardware completo.

En concreto, los contenedores suelen tener entornos privados aislados a nivel de procesos, memoria, sistema de ficheros y red.

Un hipervisor es el encargado de virtualizar el hardware y cada máquina virtual tiene su propio sistema operativo.

En un sistema de contenedores no existe esa virtualización del hardware y cada contenedor es un entorno privado.

Contenedores

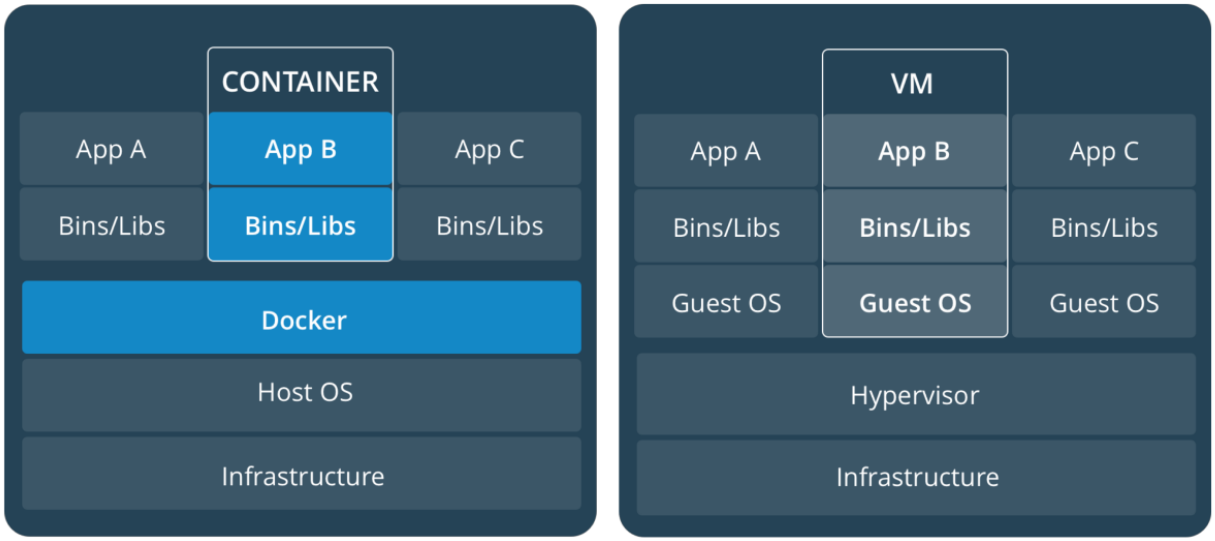


Figure 1: Diferencias entre contenedores y máquinas virtuales

Uso para despliegue de aplicaciones

- **Compilación**: tenemos el entorno de compilación/depuración montado con las versiones que necesitamos
- **Testeo**: permite la creación de distintos entornos de prueba con diferentes configuraciones
- **Compatibilidad de versiones**: evitan problemas de compatibilidad al desplegar las aplicaciones, teniendo siempre las versiones adecuadas para ejecutar nuestro software.

Uso para despliegue de servicios

- Permiten unificar configuraciones de servidores en local, incluso involucrando a distintos servicios en distintos contenedores.
- Facilitan el **escalado horizontal** de servicios, especialmente si se apoyan de herramientas llamadas orquestadores.

Ventajas

- Los contenedores ocupan menos espacio ya que utilizan el sistema de la máquina anfitrión.
- La ejecución del software de los contenedores es mucho más rápida, con velocidades cercanas a las nativas.
- Multitud de empresas de software (Microsoft, Apache, Nginx, MySQL, Oracle, Wordpress, Moodle, y un largo etc.) apoyan estas tecnologías y dan soporte incorporando sistemas de contenedores a sus sistemas operativos, como ofreciendo imágenes oficiales de sus productos.

Inconvenientes

- Siguen teniendo un rendimiento peor que una ejecución sobre un sistema real, ya que el aislamiento consume recursos.
- La persistencia y el acceso/modificación a datos persistentes entre contenedores es más tedioso que sobre una máquina real.
- Los contenedores están pensados generalmente para el uso vía línea de comandos.

Cuándo usar contenedores

- Como **usuarios**: para probar algo rápido y sin complicarnos mucho en la configuración
  - Podemos utilizar servicios de distribución de imágenes de contenedores públicas como **Docker Hub** https://hub.docker.com/
- Como **desarrolladores**: para desarrollar una aplicación que se pueda distribuir en local o desplegar en la nube sin problemas de configuración
- Para **testear nuestra aplicación** con distintas configuraciones, límites de recursos, juegos de prueba, etc.
- Para realizar un **escalado horizontal** de servicios, es decir ejecutar múltiples copias de una misma aplicación/conjunto de aplicaciones que funcionan como un cluster.

Contenedores Docker

Docker es un sistema de contenedores Linux que utiliza las características del núcleo de Linux para permitir el desarrollo y despliegue de aplicaciones.

Su web oficial es ***https://www.docker.com***

Docker es un proyecto de código abierto. Dispone de varias versiones:

- **Docker CE (Community Edition)**: el motor de Docker, de código abierto.
- **Docker EE (Enterprise Edition)**: lo mismo que la versión CE, que además incluye certificación de funcionamiento en algunos sistemas concretos y soporte con la empresa Docker Inc.

El sistema de contenedores de Docker es integrable con otros servicios populares en la nube, tales como **Google Cloud**, **Amazon AWS**, **Microsoft Azure**, **Digital Ocean**

Arquitectura de Docker

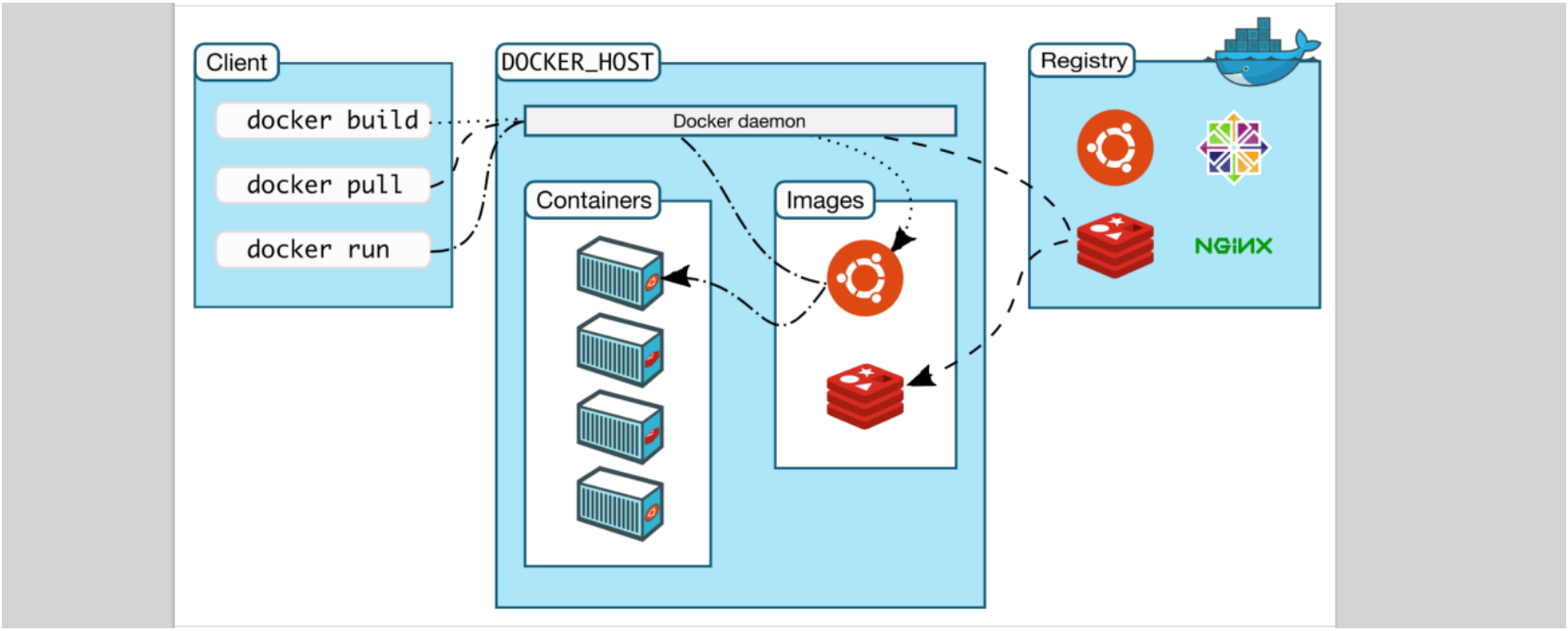


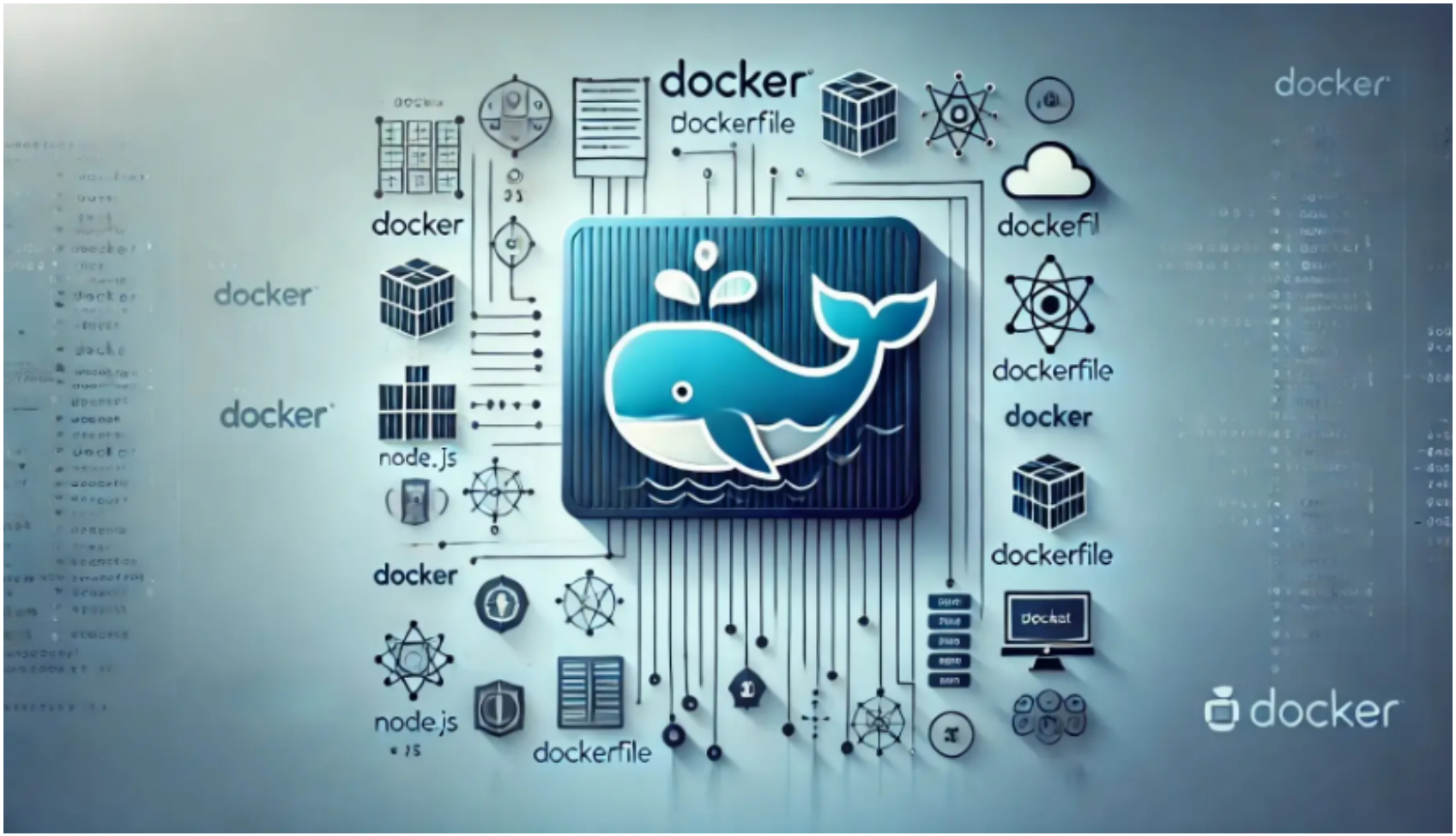
Figure 2: La arquitectura básica de Docker

Arquitectura de Docker

- **Cliente**: es el software encargado de comunicarse con el servidor Docker.
- **Servidor (Docker Host)**: servicio Docker, donde se atienden a las peticiones de los clientes y se gestionan los contenedores e imágenes.
- **Registro (Registry)**: lugar donde se almacenan imágenes Docker (públicas o privadas). Incluso, de una misma imagen, se almacenan las distintas versiones.

El registro más popular y configurado por defecto en Docker es **"Docker Hub" https://hub.docker.com/**

Empecemos!!


docker
Dockerfile
docker
docker
node.js
docker
node.js
dockerfile
dockerfil
dockerfile
docker
dockerfile
docker
docker