Introducción a Docker

Antonio Espín Herranz

Introducción

- ¿Qué es Docker?
- Docker vs Máquinas virtuales
- Arquitectura de Docker
- Estándares de contenedores
- Versiones de Docker

¿Qué es Docker?

- Wikipedia
- Docker es un **proyecto de código abierto** que **automatiza el despliegue de aplicaciones dentro de contenedores de software.**
  - Proporciona una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

- Docker utiliza características de aislamiento de recursos del kernel Linux, tales como cgroups y espacios de nombres (namespaces) para permitir que "contenedores" independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales.
  - **1 instancia de Linux permite la ejecución de N Contenedores**

Docker

- Dentro del mundo de los contenedores Docker es la herramienta que más se utiliza.

- El proyecto lo inicio: **Salomón Hykes** dentro de la empresa **dotCloud**

- El proyecto se **liberó en marzo de 2013**.

¿Qué es un contenedor?

- En el mundo real y en el trasporte de mercancías cada medio de transporte utilizaba sus propios contenedores (barcos, trenes, camiones) → problemas de logística.

- La creación de un estándar en el mundo de los contenedores ahorro costes, tiempos, etc. El mismo contenedor puede pasar de un tres a un camión y del camión al barco sin tener que hacer cambios de las mercancías.

- Esta idea del mundo físico se aplicó a los contenedores sotfware.

Un contenedor en el Sotfware

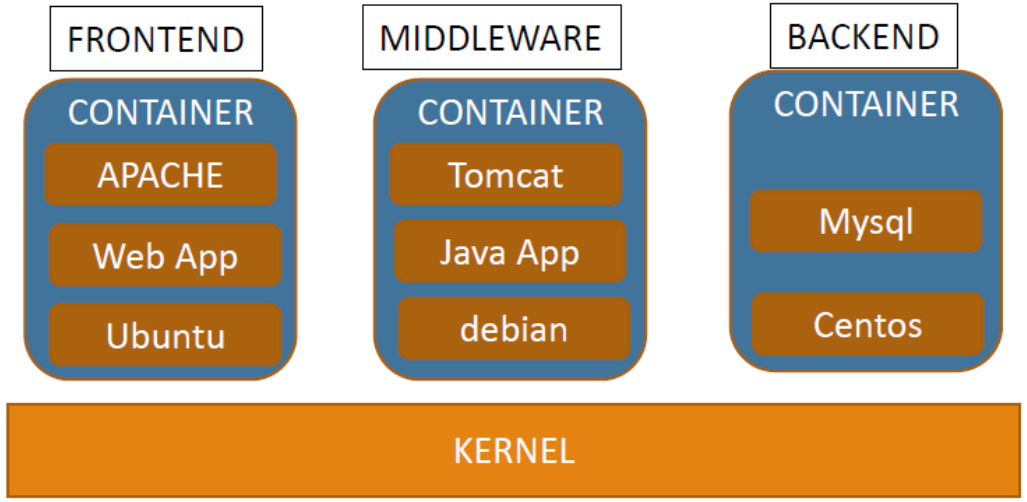
- El contenedor empaqueta de forma ligera todo lo necesario para que uno o mas procesos funcionen: código, herramientas del sistema, bibliotecas, dependencias, etc.

- Nos **proporcionan un servicio y todo lo necesario para que ese servicio funcione**.

- Completamente independiente y orientado a microservicios.
  - Están empaquetados de forma individual y son independientes.

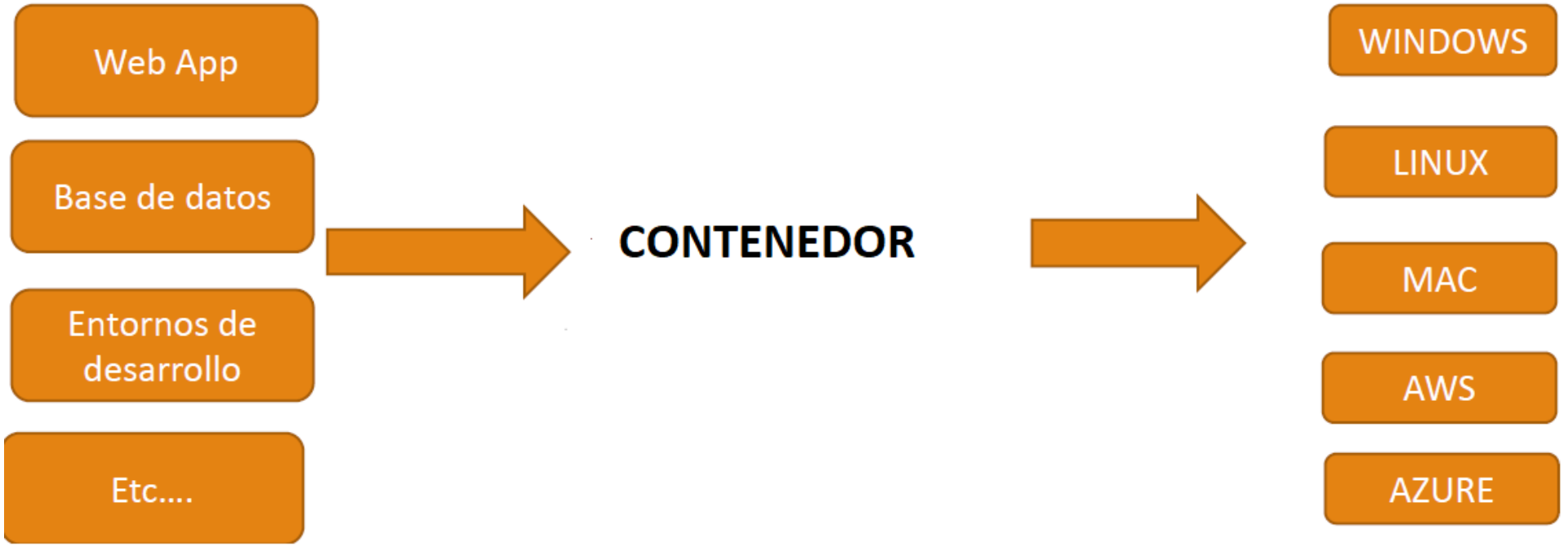
Un contenedor en el Sotfware

- La idea es poder llevarnos el contenedor a otro sistema sin tener que hacer ningún tipo de cambio → el contenedor es autosuficiente.
  - **Solo necesita un Runtime de contenedores**.
  - Distintos contenedores proporcionando cada uno distintos servicios.



Empaquetado de App

- Dentro del contenedor se pueden desplegar varios servicios y este contenedor se ejecuta cualquier S.O. que disponga de un Runtime de contendores.

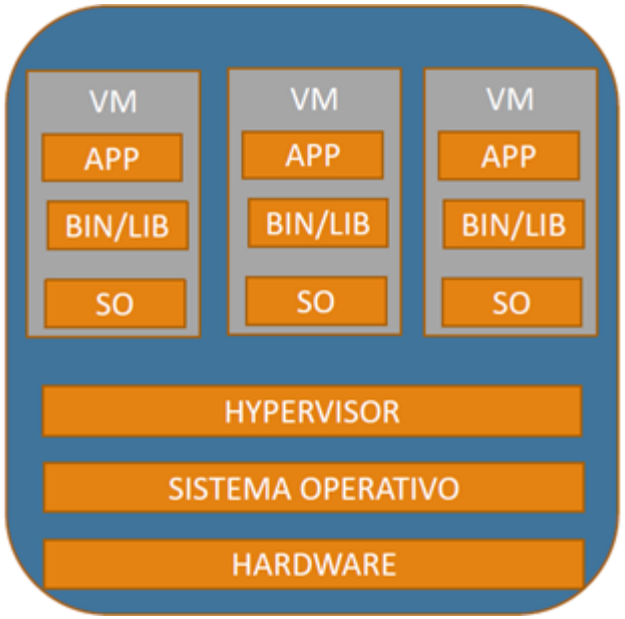


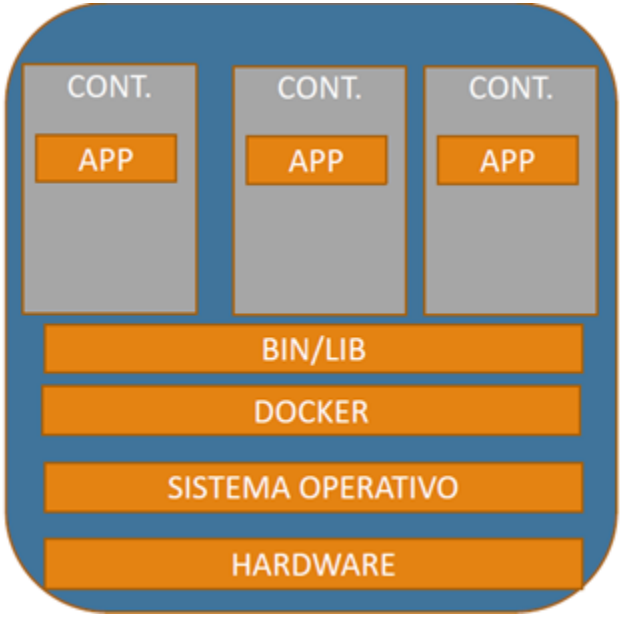
Resumen

- Un contenedor es:
  - Un **objeto ligero** donde,
  - Se **empaqueta** todo lo necesario
  - Para **ofrecer un servicio**
  - A un tercero.

Docker vs Máquinas virtuales

- Docker **no requiere un S.O. independiente**, en cambio, en una máquina virtual si es necesario.





- Cada M.Virtual es independiente:
- Tienen un S.O. por cada máquina
- Librerías y se ejecutan las apps.

Componentes monolíticos totalmente Independientes unos de otros.

En Docker sería un proceso o Daemon que se ejecuta en el S.O. sería el equivalente al Hypervisor, pero los contenedores Utilizan los recursos de S.O. para poder compartir: lib, bin ...

NO implementan su propio S.O., ni LIBs

LOS CONTENEDORES SON INDEPENDIENTES, PERO COMPARTEN RECURSOS!!

Ventajas

- Aplicación en las distintas problemáticas de la empresa.
  - Modernizar aplicaciones tradicionales.
  - Se pueden integrar en la nube.
  - Solución ideal para microservicios.

  - AHORRAN recursos tanto HW como SW

Arquitectura de Docker (3 partes)

- **Docker HOST**
  - **Dockerd** → Daemon de Docker
  - **Imágenes** → Plantillas de contenedores.
    - **Ejemplo**: una imagen para los desarrolladores de la empresa: Windows, Python y Git
    - 1 plantilla → Múltiples contenedores
    - Almacenan todo lo necesario para crear un contenedor con unas determinadas características. **SON DE SOLO LECTURA**
  - **Contenedores:** Se crean a partir de las imágenes. Las máquinas que ya funcionan y están basadas en esas plantillas.
    - **Un contenedor tiene un ciclo de vida: parar, arrancar, crearlo, etc…**
- **Docker CLIENT**
  - Se conecta al **dockerd (Docker Daemon)**
  - **Comandos** para interactuar con el Docker Host: **pull, push, run …**
- **Registro de imágenes**:
  - **Repositorio** de imágenes creadas:
    - https://hub.docker.com/
  - Repositorios **privados** (para una empresa)

Estándares de contenedores

- **OCI (Open Container Initiative)**
  - Estándares de contenedores (desde 2015), unificar el mundo de los contenedores.
    - Un poco parecido al transporte de las mercancías en el mundo de la logística.
  - Los contenedores pueden ejecutarse en cualquier HW y sistema y componentes relacionados.

  - Hay dos estándares:
    - **Runtime Specification**: Como se deben ejecutar los contendores.
    - **Image Specification**: como deben ser las imágenes de los contenedores.
    - Esto se recopila en **OCI Runtime FileSystem Bundle** (en un paquete de ejecución dentro del sistema de archivos)

OCI Image Spec

- Implementa varios componentes
  - **Sistema de archivos**: implementa el contenido en forma de capas o pilas que se van agregando para construir la imagen final
  - **Archivo de configuración**: guardar la información necesaria para conocer la configuración de esta imagen: variables de entorno, parámetros, etc.
  - **Archivo de tipo manifest**
  - **Archivo de índice**

OCI Runtime Spec

- Especifica como funciona un Container Runtime y sus Estados:
  - Creación →
  - Creado →
  - En ejecución →
  - Parado →
  - Borrado

OCI runC

- Es un Runtime Container universal ligero que cumple con la especificación OCI.
- Donado por Docker para implementar un estándar.
- **runc** es utilizado por **containerd** para generar y ejecutar contenedores de acuerdo con las especificaciones de OCI.

runc

- **Docker Engine → Containerd (Daemon)** **runc**

runc

- **Runc** → Las ejecuciones de los contenedores.

- Enlace: https://opencontainers.org/

Otra iniciativa: CNCF

- **Cloud Native Computing Foundation**
  - Es una organización encargada de aglutinar y ordenar la gran cantidad de proyectos e iniciativas de código abierto que tratan de crear herramientas que apoyen a los desarrolladores de software a correr sus aplicaciones en la nube.
  - Entidad neutral que crea un marco de crecimiento de las herramientas que desean llegar a ser parte del ecosistema Cloud.
  - Pertenece a Linux Foundation
  - Gestiona proyectos tecnológicos como:
    - Kubernetes
    - Prometheus
    - Open Tracing
  - Enlace: https://www.cncf.io/
  - Los proyectos que hace el seguimiento: https://landscape.cncf.io/

Otras opciones del mercado

- **Alternativas a Docker:**
  - **Podman**:
    - https://podman.io/
  - **Cri-o**:
    - https://cri-o.io/
  - **Katacontainers**:
    - https://katacontainers.io/
  - **Rkt**:
    - https://www.redhat.com/es/topics/containers/what-is-rkt
  - **Buildah**:
    - https://buildah.io/
  - **Skopeo**:
    - https://www.redhat.com/es/topics/containers/what-is-skopeo

Versiones de Docker

- **Docker Engine (server) para Linux**
  - Para entornos empresariales
  - 3 Partes:
    - Servidor Docker (Daemon **dockerd**)
    - APIs de acceso al servidor
    - Un cliente para trabajar en modo comando (**docker**)
    - Orientado a entornos de **producción**

- **Docker Desktop for Windows, Linux y Mac**
  - Versión de Docker preparada para trabajar en estos sistemas operativos
  - Dispone de un **entorno gráfico** que permite simplificar el trabajo con Docker.
  - Orientado a entornos de **desarrollo**.
    - Servidor docker
    - APIs de acceso al servidor, docker compose, kubernetes

Sistemas operativos

Docker Engine
CentOS
Fedora
Ubuntu
Debian
Red Hat en IBM Z
Suse on IBM Z
.....

Docker Desktop for Windows y Mac
Windows 11 64-bit: Home, Pro, Enterprise, Education
Windows 10 64-bit: Home, Pro 2004 (build 19041), Enterprise o Education 1909 (build 18363)
macOS version 10.15 o superiores
.....

Plataformas Cloud
Amazon AWS
Microsoft Azure
IBM Cloud
Otros

docker.com

- http://docker.com

- **A nivel personal** docker es gratuito, permite utilizar:
  - **Docker desktop**
  - **Docker Hub** (repositorio de imágenes público).

  - **Hay que crear una cuenta para luego poder acceder a docker Hub, o descargar docker desktop**
    - **Ir a Sing In y crear la cuenta**

- A nivel empresa hay suscripciones.

Opciones para trabajar

- **Probar dos opciones:**
  - 1) Crear una máquina virtual por ejemplo con **Ubuntu** e instalar **Docker Engine** → trabajaremos en modo comando.
  - Si estamos en una máquina Linux (instalar directamente).

  - 2) **Docker desktop** para **Windows** o **Mac**
    - **Incluyen docker engine**

- **Docker desktop for Linux** (**no se aconseja**), está montado sobre una máquina virtual. Es mejor la opción Docker Engine.
  - Como no está soportada la virtualización anidada está opción solo se puede montar en una máquina Linux (real).
  - No se puede probar dentro de una máquina virtual y que se estuviera ejecutando dentro de Windows.

- **Ver**:
  - https://www.docker.com/blog/how-to-check-docker-version/#:~:text=Users%20can%20leverage%20the%20Docker,Docker%20CLI%20to%20run%20commands.

Componentes de docker

- **Contenedores**
  - Se crea a partir de una imagen. Se puede definir como un proceso que ha sido aislado de todos los demás procesos de la máquina donde se ejecuta (el host de Docker).
  - ***Buenas prácticas: 1 sólo proceso en 1 contenedor. 1→1***

- **Imágenes**
  - Las imágenes contienen el sistema de archivos que utilizarán los contenedores Docker. Para crear un contenedor, es obligatorio utilizar una imagen.
  - Plantilla. 1 imagen → N contenedores

- **Volúmenes**
  - Mecanismo de persistencia para los contenedores. Si un contenedor no tiene un volumen asociado al terminar y eliminar el contenedor los datos se pierden.

- **Redes**
  - Docker permite crear diferentes tipos de redes para que los contenedores se puedan comunicar entre ellos. La gestión de redes en Docker se gestiona con **libnetwork**