



Centros de Datos

T5 - Virtualización en los Centros de Datos

David Ruano Ordás
Departamento de Informática

✉ drordas@uvigo.es

📍 Despacho 409

Contenidos

- Introducción
 - Conceptos
- Tipos de Virtualización
 - Virtualización completa
 - Paravirtualización
 - Virtualización asistida por hardware
 - Virtualización a nivel de sistema operativo
- Virtualización con Docker
 - Introducción
 - Componentes clave en Docker
 - Estructura y comandos básicos
 - Dockerfile
 - Docker Compose

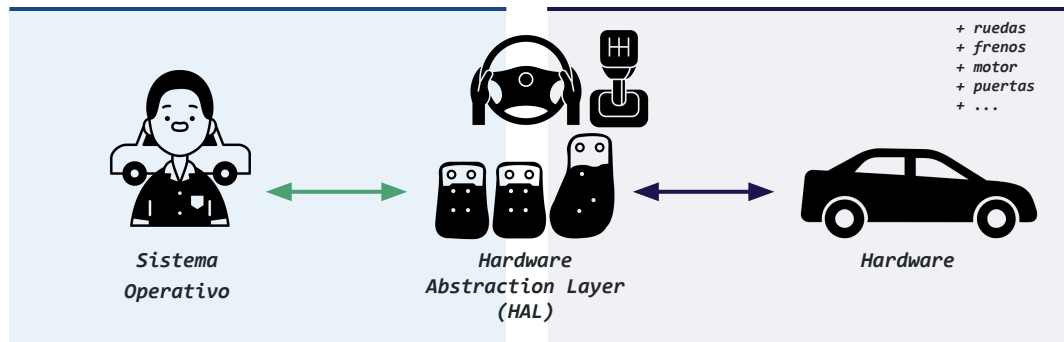
Introducción

- Conceptos
 - Abstracción

Simplificación: oculta la complejidad del hardware y presentar una vista simplificada a las aplicaciones
Permite que las aplicaciones se ejecuten sin tener que conocer los detalles específicos del hardware subyacente
Ejemplos:

- *Sistemas de ficheros*
- *Procesos*
- *Dispositivos de entrada/salida*

ANALOGÍA



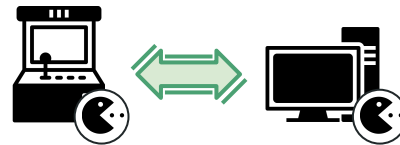
Introducción

- **Conceptos**

- **Emulador**

Permite *ejecutar programas de computadora o videojuegos en una plataforma (arquitectura hardware o sistema operativo) diferente de aquella para la cual fueron escritos originalmente*

✔ Traducción constante de *instrucciones del invitado a la plataforma base*



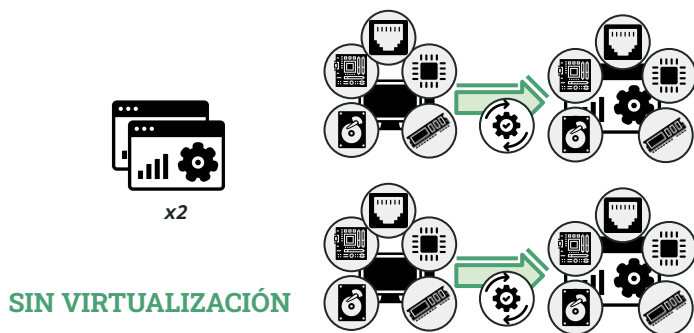
- **Simulador**

Se basa en *recrear/replicar el comportamiento exacto de un sistema específico con el fin de imitar una realidad.*



- **Virtualización**

Saca *partido de la abstracción para crear entornos independientes que permiten ejecutar múltiples sistemas operativos o aplicaciones en un mismo hardware.* Cada *entorno actúa como un sistema separado, utilizando los mismos recursos físicos de forma compartida y segura*
Permite *maximizar la eficiencia y el uso de recursos computacionales.*



Introducción

- **Conceptos**
 - **Hipervisor**

Software que permite crear y gestionar máquinas virtuales (VMs) en un solo equipo físico. Actúa como un intermediario entre el hardware y las máquinas virtuales, asignando y compartiendo recursos (CPU, memoria, almacenamiento y red) para que varias VMs puedan operar de manera independiente en el mismo hardware físico.



*Gestión de Máquinas Virtuales
(crear, iniciar, parar, eliminar)*

*Aislamiento
(separa las MVs entre ellas)*

*Asignación de Recursos
(distribuye recursos hardware entre las MVs acorde a las necesidades)*

*Migración en caliente
(Mover MVs de un hardware a otro)*



- **Host (anfitrión)**

Es el equipo físico (computadora, servidor) que proporciona los recursos de hardware (CPU, memoria, almacenamiento) necesarios para crear y ejecutar las máquinas virtuales.

- **Guest (invitado)**

Sistema operativo y las aplicaciones que se ejecutan sobre una máquina virtual. Se comporta como si tuviera su propio hardware físico, aunque en realidad está compartiendo los recursos del sistema anfitrión a través del hipervisor.

Tipos de Virtualización

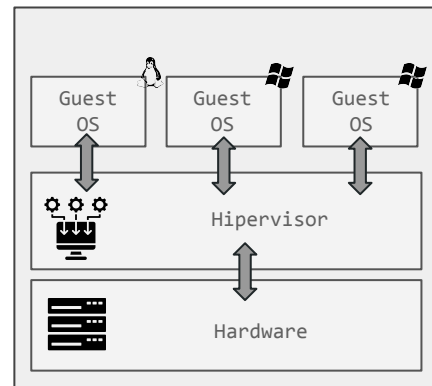
- Virtualización completa

Permite crear múltiples sistemas operativos y aplicaciones aislados, conocidos como máquinas virtuales, sobre un único servidor físico. Hipervisor, que actúa como un gestor de recursos, divide los recursos del hardware (CPU, memoria, almacenamiento, etc.) en porciones que se asignan a cada máquina virtual de forma independiente.

- Tipos:

- Tipo 1 (Bare -Metal):

- Hipervisor se instala sobre el hardware físico ↔ “reemplaza al SO”.
 - Dos tipos de hipervisores:
 - Micronúcleo:
 - Incluye funcionalidades básicas e inmutables (adm. de los recursos e interrupciones).
 - Funciones adicionales se cargan en módulos externos (gestión redes)
 - Monolíticos:
 - Incorpora las funcionalidades básicas
 - Incorpora las funcionalidades adicionales
 - Ejemplos:
 - VMware ESXi



Tipos de Virtualización

- **Virtualización completa**

Permite crear múltiples sistemas operativos y aplicaciones aislados, conocidos como máquinas virtuales, sobre un único servidor físico. Hipervisor, que actúa como un gestor de recursos, divide los recursos del hardware (CPU, memoria, almacenamiento, etc.) en porciones que se asignan a cada máquina virtual de forma independiente.

- **Tipos:**

- **Tipo 2 (hosted):**

- **Hipervisor se instala sobre el SO anfitrión**

- **Traducción binaria:**

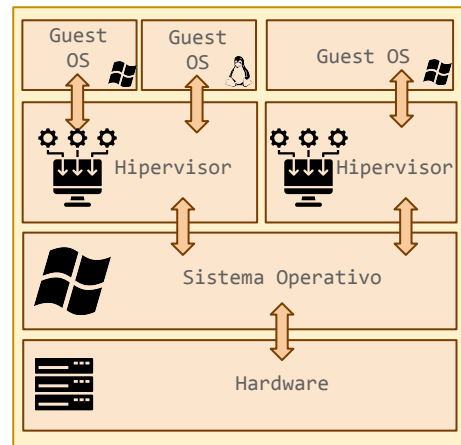
- Garantiza la ejecución segura de instrucciones críticas.
 - Instrucciones críticas se traducen a código máquina (entendido por el SO anfitrión).
 - Emplea una caché de código para almacenar las instrucciones traducidas y mejorar así el rendimiento.
 - Se realiza en tiempo real.

- **Ejecución directa:**

- Ejecuta directamente las instrucciones seguras del sistema operativo invitado en el anfitrión sin traducción.

- **Ejemplos:**

- VMWare Workstation
 - VirtualBox



Tipos de Virtualización

- **Paravirtualización:**

El sistema operativo invitado es consciente de que está corriendo en un entorno virtualizado → se modifica para utilizar una interfaz de programación de aplicaciones (API) proporcionada por el hipervisor.

API permite al sistema operativo invitado comunicarse directamente con el hipervisor y solicitar los recursos que necesita (como CPU, memoria, dispositivos de E/S).

	Virtualización Completa	Paravirtualización
Consciencia del SO invitado	No es consciente	Es consciente
Interfaz con el hipervisor	A través de la traducción binaria/llamadas a funciones	A través de una API específica
Rendimiento	Bajo (traducción binaria)	Alto (comunicación directa)
Complejidad de la implementación	Baja	Alta (importantes cambios en el kernel)

Tipos de Virtualización

- Virtualización asistida por hardware:

*El hardware (el procesador), incluye extensiones que facilitan la ejecución de máquinas virtuales.
Permite que el hipervisor acceda directamente a ciertos recursos físicos sin necesidad de emularlos/APIS → mejora el rendimiento y la eficiencia.*



Intel VT-x (Intel Virtualization Technology for x86)

- Virtualización a nivel de Procesador

Intel VT-d (Intel Virtualization Technology for Directed IO)

- Virtualización de Dispositivos de E/S



Soporte para ejecución de instrucciones con privilegios

Habilitan un nivel de privilegio especial (debajo del ring 0) para que el hipervisor gestione de manera eficiente el acceso a la CPU, la memoria y otros recursos de hardware sin interferencias, reduciendo la sobrecarga.



Funcionalidades requeridas por todos los hipervisores actuales

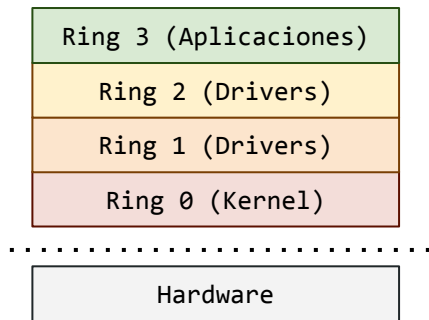
Tipos de Virtualización

- Virtualización asistida por hardware:

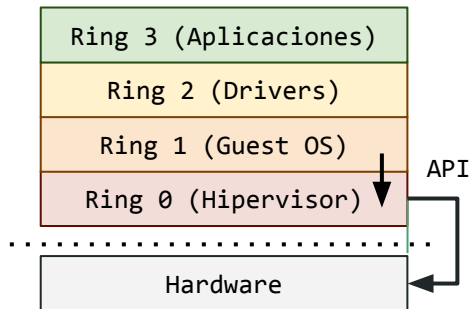
- Comparativa:

- privilegios

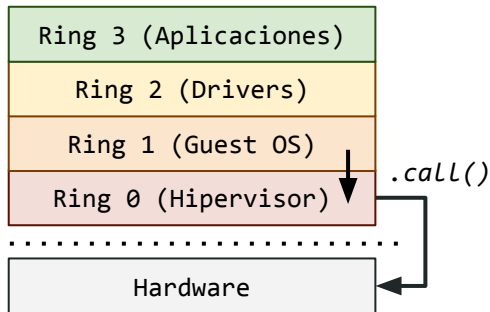
+ privilegios



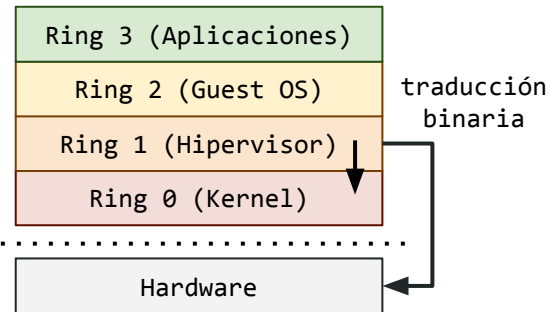
Paravirtualización



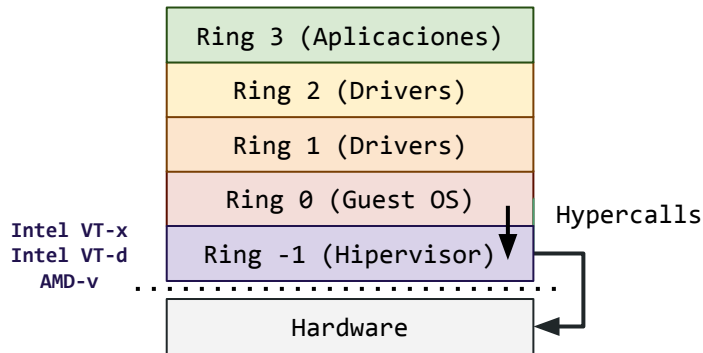
Virtualización Completa (Tipo 1)



Virtualización Completa (Tipo 2)



Virtualización Hardware



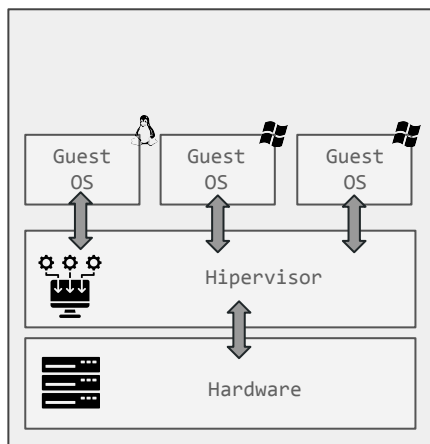
Tipos de Virtualización

- Virtualización a nivel de sistema operativo (basada en contenedores)

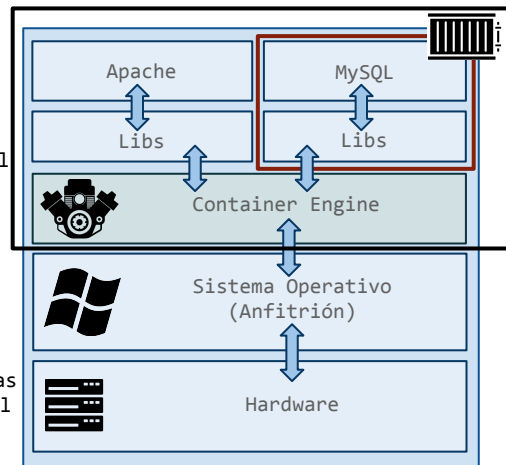
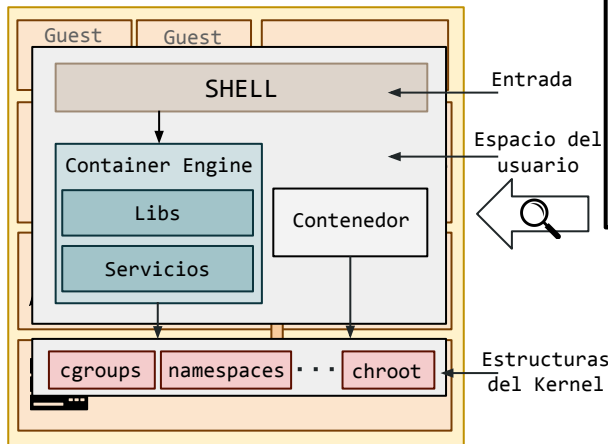
Permite **aislar aplicaciones** dentro de un mismo sistema operativo, **compartiendo el kernel** del sistema.

Componentes clave:

- Chroot:** que crea un entorno aislado al cambiar el directorio raíz (/) de un proceso y sus hijos
Procesos solo pueden ver y acceder a los archivos dentro del directorio específico, como si fuera su propio sistema raíz.
- Kernel:** núcleo del sistema operativo, compartido por todos los contenedores
- Espacio de nombres (namespaces):** aíslan contenedores, separando recursos y mejorando seguridad y portabilidad.
- Control de grupos (cgroups):** permite limitar los recursos (CPU, memoria, I/O) que un contenedor puede utilizar



(Virtualización Completa)



(Virtualización Basada en Contenedores)

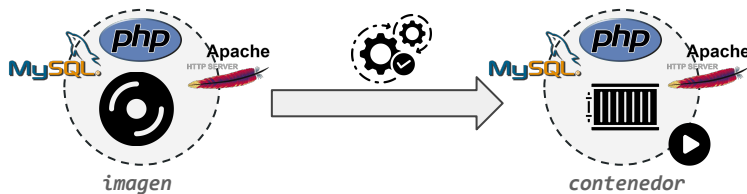
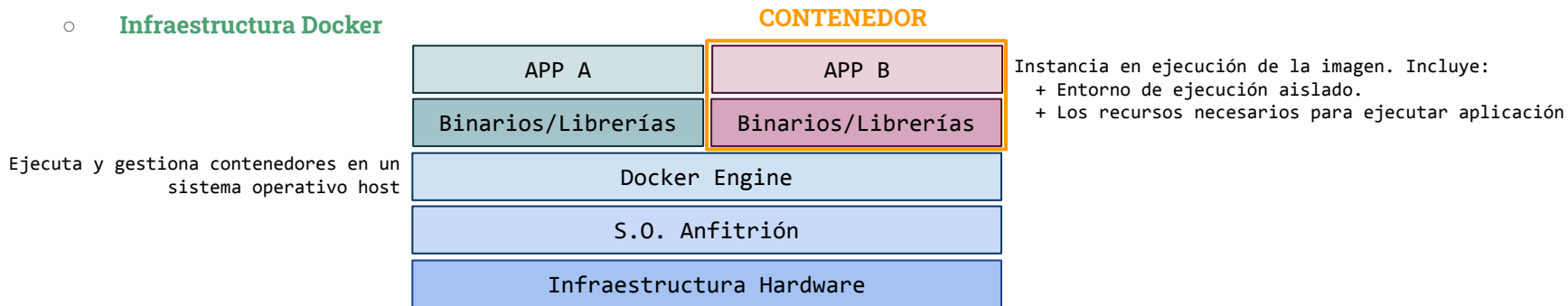
Tipos de Virtualización

- **Docker:**

- Proyecto de código abierto que automatiza la creación y despliegue de aplicaciones dentro de contenedores livianos de software.

Aplicación + Herramientas (librerías) que necesita la aplicación

- **Infraestructura Docker**

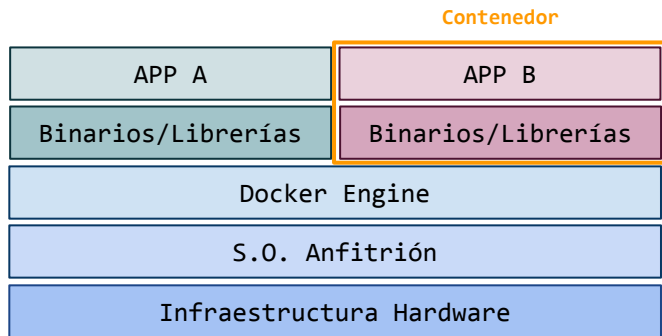


Virtualización con Docker

- **Virtualización vs Docker:**

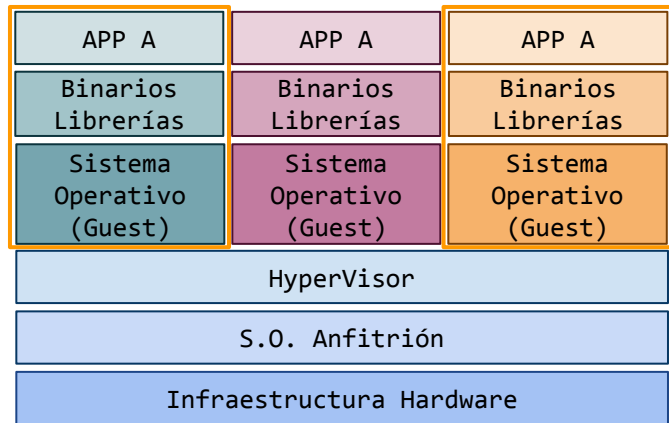
- **Virtualización:**

- Abstrae (virtualiza) el hardware del SO.
 - Uso del hardware por varios SO al mismo tiempo.
 - Requiere incluir SO en cada MV.
 - Gran uso de recursos



MV

MV



- **Docker:**

- Abstrae el SO de las aplicaciones.
 - Portabilidad
 - Velocidad de ejecución
 - Contenedor no incluye SO.
 - Mejor uso de recursos
 - Ejecución rápida

Virtualización con Docker

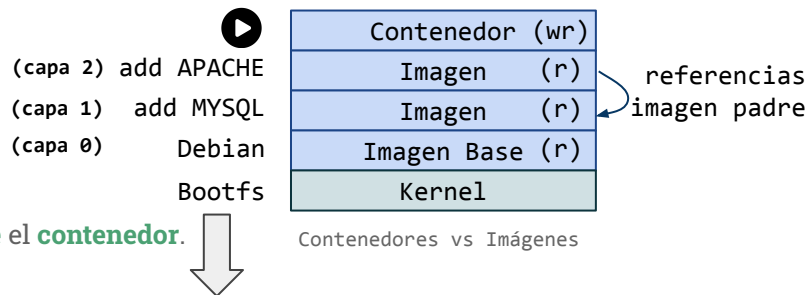
- **Componentes clave de Docker (I):**

- **Imagen:**

- **Plantilla** de **sólo lectura** (ro) usada para la **creación** de **contenedores**.
 - Imagen para un servidor web.
 - Imagen para una base de datos.
 - **Jerarquía** de **capas**: **capas superiores indican** las **modificaciones sobre** las **capas inferiores**.
 - **Contiene** el **código** y **dependencias** y **librerías** necesarias para **ejecutar** las(s) **aplicación(es)**.
 - **Creadas** por los **usuarios** o por la **comunidad** (Docker Hub).
 - Se **crean** a partir de un **Dockerfile**

- **Contenedor:**

- Son **instancias** en **ejecución** de una **imagen**.
 - **Basado** en **una** o más **imágenes**.
 - **Varios contenedores** a **partir** de una **imagen**.
 - **Aislamiento** de los **recursos** a los que se tiene **acceso desde** el **contenedor**.



Sistema de archivos base de una imagen (S.O mínimo + Librerías).
Punto de partida para la construcción de imágenes de contenedores.

+ **Imagen** → "paquete" que **contiene** todo lo **necesario** para **ejecutar** una **aplicación**.

+ **Contenedor** → **instancia** en **ejecución** de esa **imagen**, con su **propio espacio** de **procesos** y **sistema** de **archivos**, pero **compartiendo** el **kernel** del **sistema operativo host** con otros **contenedores**.

Virtualización con Docker

- **Componentes clave de Docker (II):**

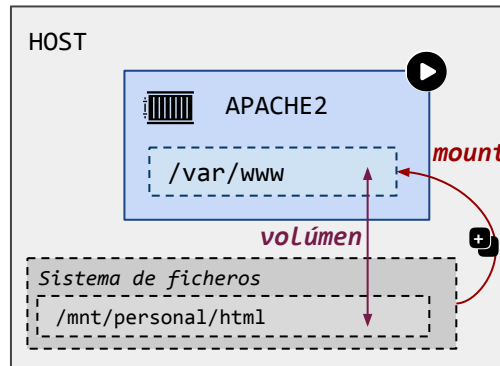
- **Volúmenes:**

- Permiten **compartir** y **persistir** datos entre contenedores y el **host**.
 - **Punto de montaje** para **compartir** datos.
 - **Datos almacenados** en un **volumen** no se **pierden** cuando se **detiene** o se **elimina** un **contenedor**.
 - **Características:**
 - **Independencia** del **sistema** de **archivos**:
 - No están **vinculados** directamente al **sistema** de **archivos** del **contenedor**
 - Pueden **existir** fuera del **espacio** del **contenedor**.
 - **Permiten** ser **compartidos** entre **varios** **contenedores**.
 - **Compartición** de **datos** entre **contenedores**.
 - Compartir volúmenes → datos se pueden compartir
 - **Tipos de Volúmenes:**
 - **Volúmenes nombrados**
 - **Volúmenes anónimos**

- **Bind mount:**

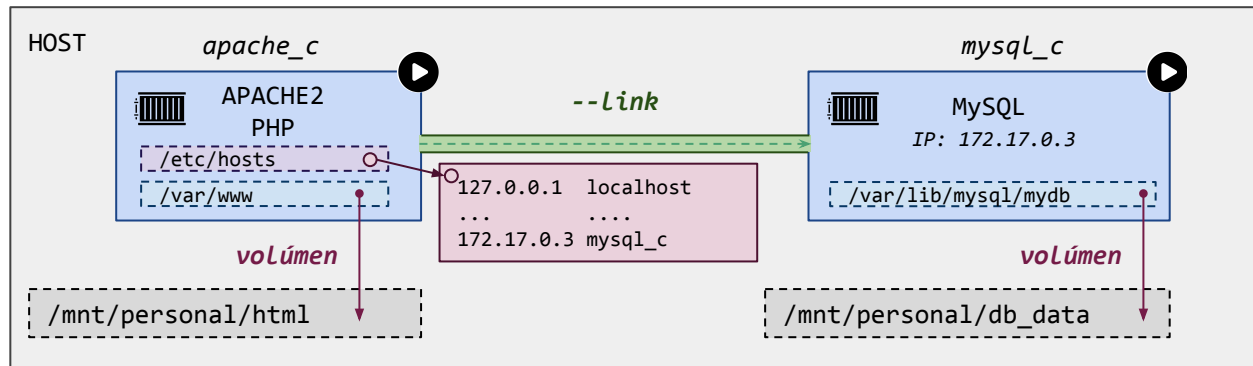
- Permite **montar** **directorios** o **ficheros** del **anfitrión** en el **contenedor**.
 - **Compartir** **datos** con el **anfitrión**.
 - **Mismo funcionamiento** que el comando **mount**

Datos creados dentro del contenedor son efímeros (no se mantienen entre ejecuciones)



Virtualización con Docker

- **Componentes clave de Docker (III):**
 - **Enlazamiento (Linking):**
 - **“Túnel”** que permite la **comunicación** y la **conexión** entre **contenedores**
 - **Contenedor** pueda **acceder** a **servicios/recursos/información** específicos **de otro contenedor**
 - **Añade** una **entrada** en el **/etc/hosts** con la **información** del **contenedor enlazado**
 - Permite **resolver internamente** la **IP** del **contenedor** a **partir del alias**.
 - **Evita especificar manualmente** la **información** del **contenedor** a **enlazar**.



Virtualización con Docker

- **Componentes clave de Docker (IV):**

- **Redes:**

- **Docker establece reglas** de filtrado que **impiden tráfico** entre **contenedores** en **redes distintas**

- Existen **tres tipos** de **configuraciones**:

- **Bridge:**

- **Contenedores tienen:**

- **Comunicación interna** entre sí por **nombre** o **IP**.
 - **Conectividad externa indirecta** (uso NAT).

- **Permite comunicación intra-hosts.**

- **Host:**

- **Contenedores tienen:**

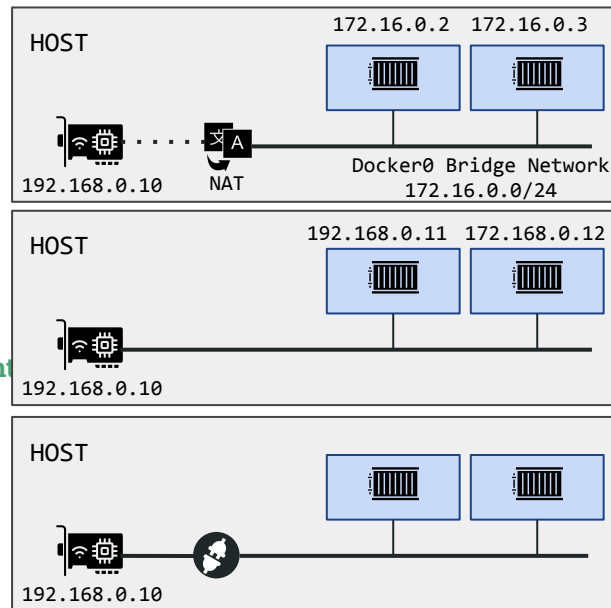
- **Comunicación interna** entre sí por **nombre** o **IP**.
 - **Conectividad externa directa** (sin NAT).

- **Máxima velocidad y eficiencia** de red pero **sin aislamiento**

- **None:**

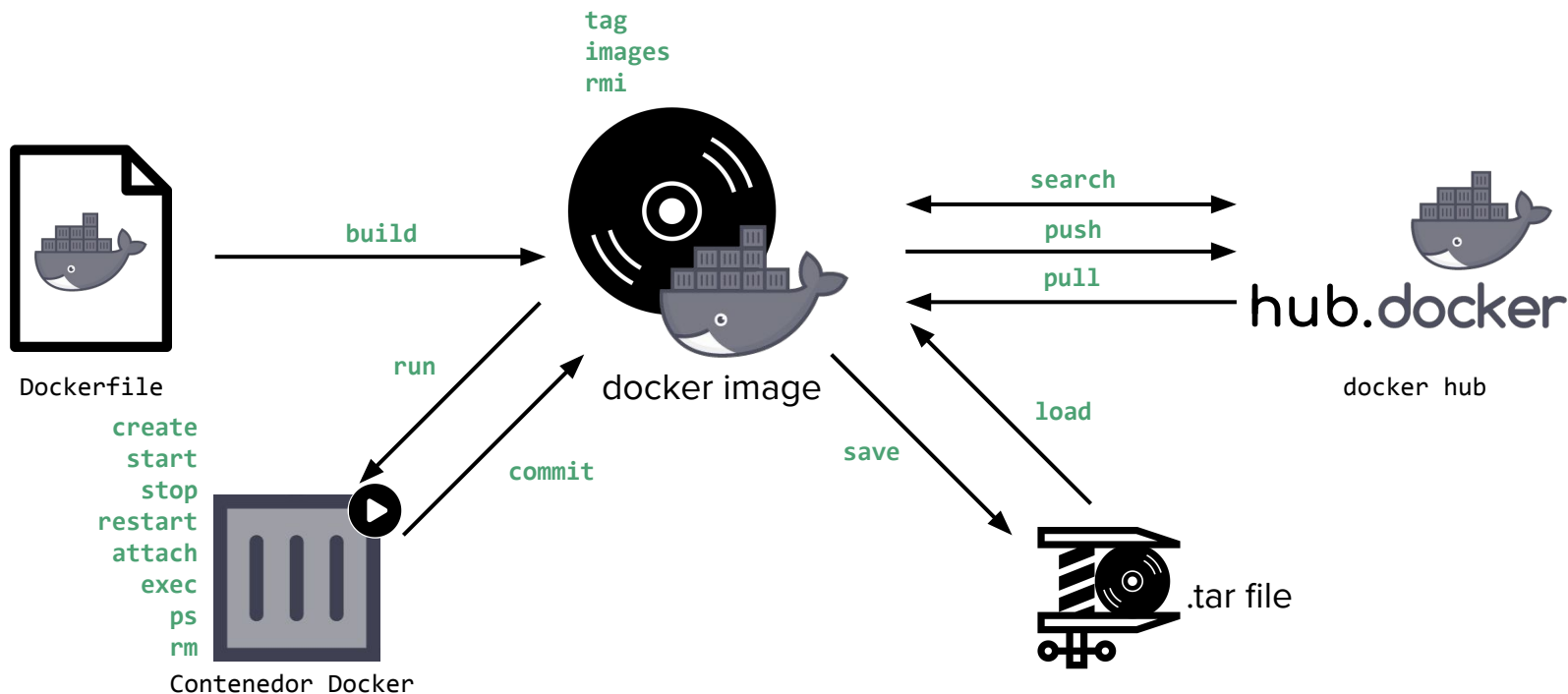
- **Sin interfaz de red asignada** al **contenedor**.

- **Aislamiento total.**



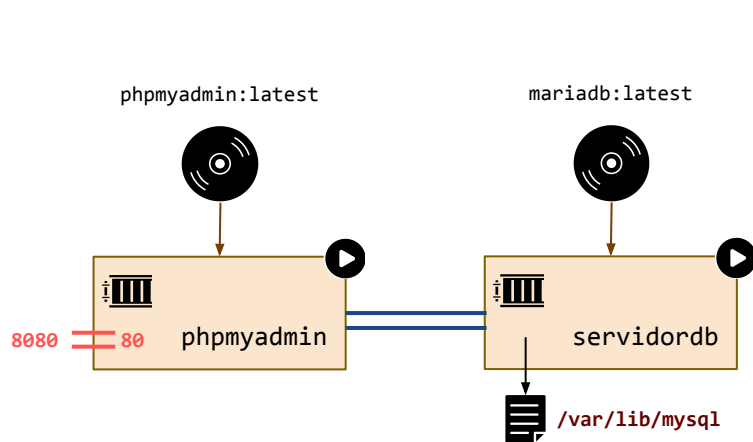
Virtualización con Docker

- Estructura y comandos básicos en Docker:
 - Esquema general:



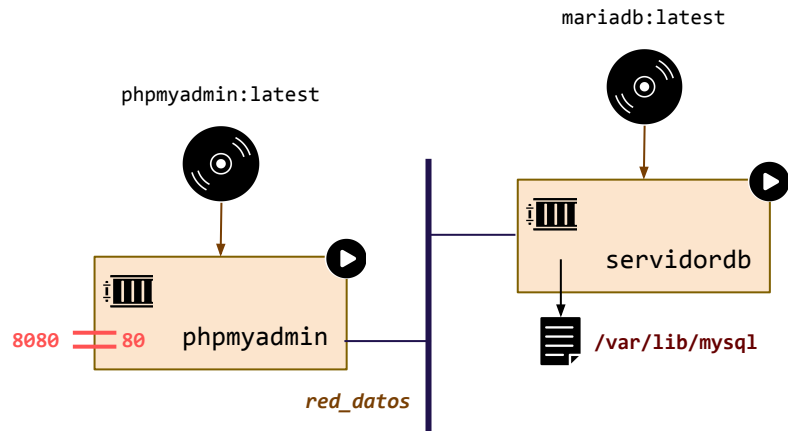
Virtualización con Docker

- Estructura y comandos básicos en Docker:
 - Ejemplos:



```
docker run -d --name servidordb -e MYSQL_ROOT_PASSWORD=«pass» \  
  --volume $PWD/db_folder:/var/lib/mysql mariadb \  
docker run -d --name phpmyadmin --link mysql-container:db -p 8080:80 phpmyadmin
```

docker linking



```
docker network create red_datos \  
docker run -d --name servidordb --network red_datos -e MYSQL_ROOT_PASSWORD=«pass» \  
  --volume $PWD/db_folder:/var/lib/mysql mariadb \  
docker run -d --name phpmyadmin --network red_datos -e PMA_HOST=servidordb -p 8080:80 phpmyadmin
```

indica a phpmyadmin el nombre del contenedor de la red al que se debe conectar

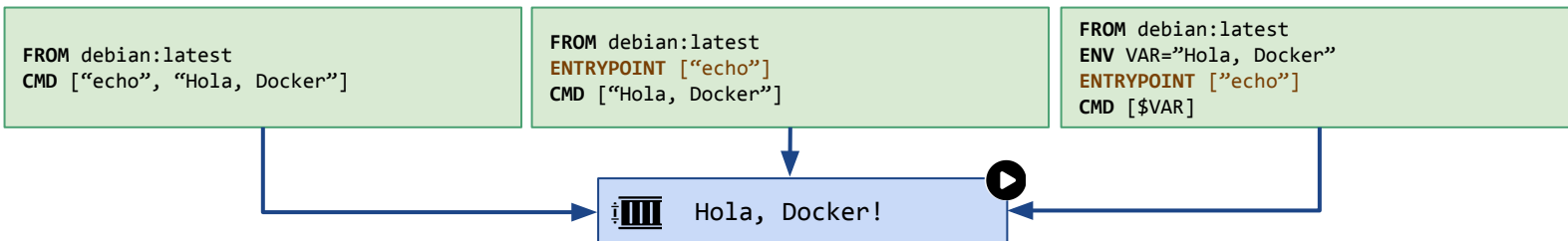
docker networks

Virtualización con Docker

- **Dockerfile:**
 - **Fichero** con las **instrucciones** necesarias para **automatizar** la **creación** de una **Imagen Docker**.
 - Tiene una **sintaxis** y **estructura específica**.

FROM Definir la imagen base sobre la que crear la nueva imagen.
LABEL Hace referencia al creador de la imagen
ENV Define variables de entorno (HOME).
WORKDIR Establece el directorio donde se ejecutará **RUN/CMD/ENTRYPOINT**.
RUN Permite ejecutar comandos durante la construcción de la imagen.
ADD/COPY Permite agregar/copiar archivos desde el equipo local a la imagen
ENTRYPOINT Define el comando a ejecutar cuando se inicie el contenedor (defecto: /bin/sh -c)
VOLUME Establece los volúmenes a montar al ejecutar el contenedor.
EXPOSE Indica los puertos (TCP/IP) en los que escuchará el contenedor
CMD Define los valores predeterminados para un comando.

- **Ejemplos (I):**



Virtualización con Docker

- **Dockerfile:**
 - **Ejemplos (II):**

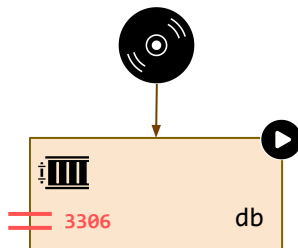
```
FROM debian:10
ENV DEBIAN_FRONTEND noninteractive
RUN apt-get update \
    && apt-get install -y mariadb-server \
    && apt-get clean && rm -rf /var/lib/apt/lists/*
RUN echo "mariadb-server mariadb-server/root_password password root" | debconf-set-selections
RUN echo "mariadb-server mariadb-server/root_password again password root" | debconf-set-selections
RUN sed -i 's/127\.0\.0\.1/0.0.0.0/g' /etc/mysql/mariadb.conf.d/50-server.cnf
EXPOSE 3306
CMD ["service mysql start"]
```

docker build -t mi-mysql .

mi_mysql

Permitir acceso remoto
(desde cualquier IP)

Definir root password para la BD
(root en este caso)



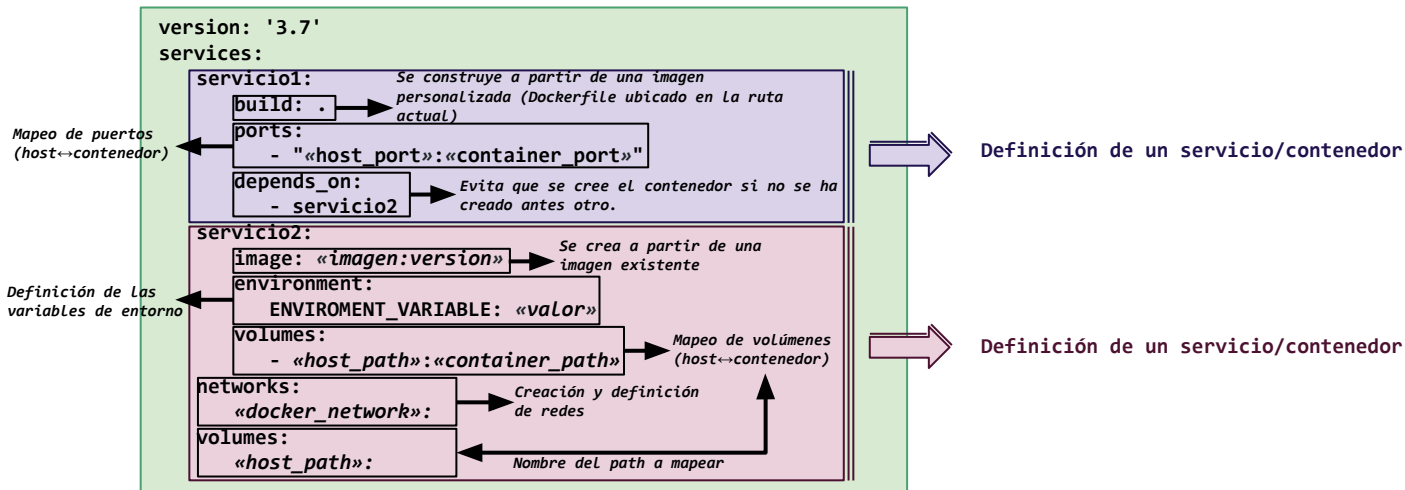
Virtualización con Docker

- Docker Compose

Es una herramienta de Docker que **permite definir y gestionar aplicaciones multi-contenedor de manera sencilla**.

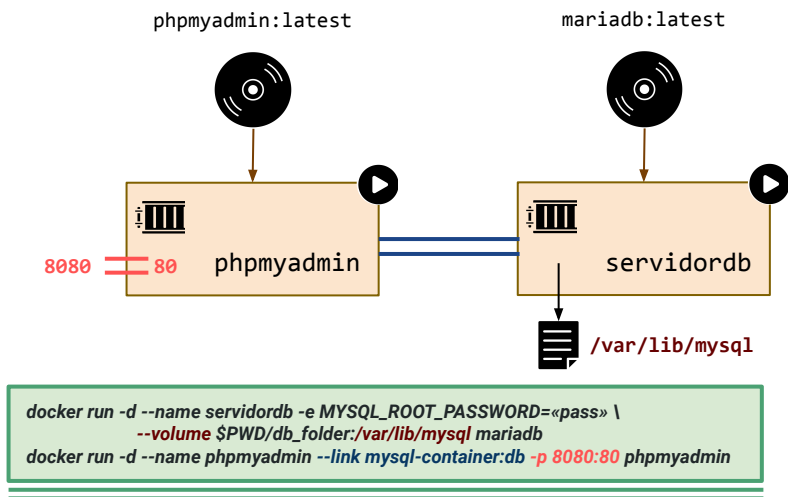
Fichero en **formato YAML** que permite:

- **Definición de múltiples servicios:** especificar **múltiples contenedores (o servicios) y sus configuraciones**
 - Ejemplo: Aplicación web (Apache + MySQL + PHPMyAdmin).
- **Automatizar el despliegue:** **construye las imágenes (si es necesario) y levanta todos los servicios especificados en el archivo YAML**
- **Manejo de redes y volúmenes:** **configura automáticamente redes y volúmenes entre los contenedores, facilitando la comunicación y el acceso a los datos persistentes**



Virtualización con Docker

- Docker Compose
 - Ejemplos (I):

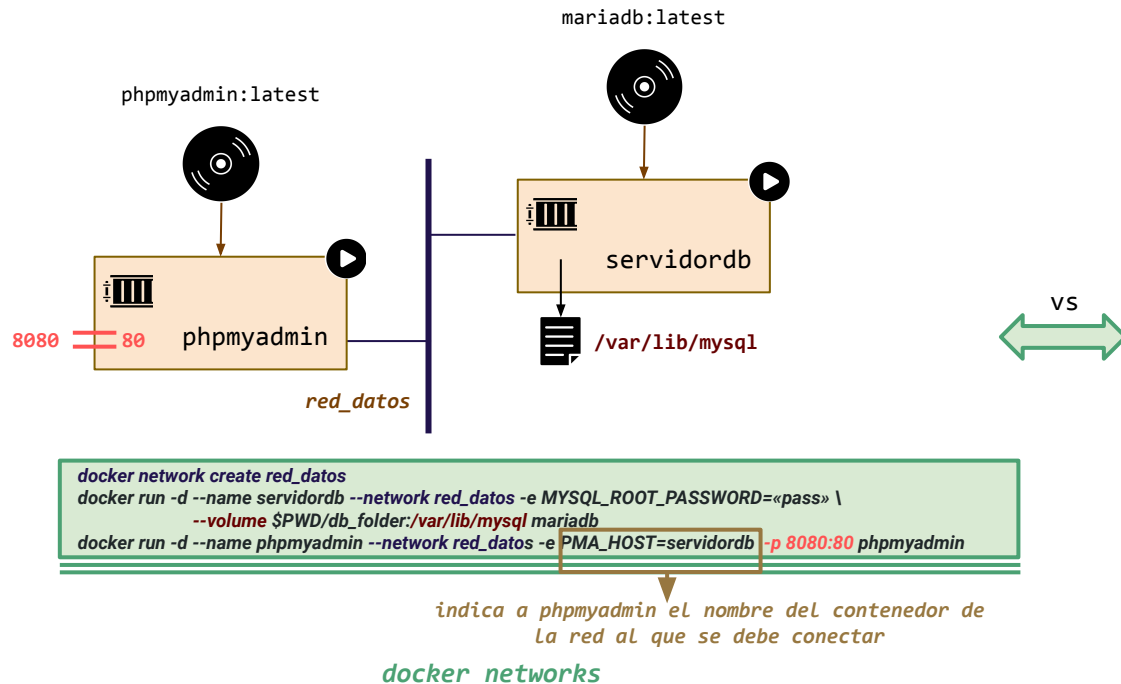


```
version: '3.7'
services:
  db:
    image: mariadb:latest
    container_name: servidordb
    environment:
      MYSQL_ROOT_PASSWORD: «pass»
    volumes:
      db_folder: /var/lib/mysql
  phpmyadmin:
    image: phpmyadmin:latest
    container_name: phpmyadmin
    ports:
      - "8080:80"
    links:
      - db:db
    depends_on:
      - db
  volumes:
    db_folder:
```

docker linking

Virtualización con Docker

- Docker Compose
 - Ejemplos (I):



```
version: '3'
services:
  db:
    image: mariadb:latest
    container_name: servidordb
    environment:
      MYSQL_ROOT_PASSWORD: «pass»
    networks:
      - red_datos
    volumes:
      db_folder:/var/lib/mysql
  phpmyadmin:
    image: phpmyadmin:latest
    container_name: phpmyadmin
    ports:
      - "8080:80"
    environment:
      PMA_HOST: servidordb
    networks:
      - red_datos
      depends_on:
        - db
networks:
  red_datos:
    driver: bridge
volumes:
  db_folder:
```



Centros de Datos

T5 - Virtualización en los Centros de Datos

David Ruano Ordás
Departamento de Informática

✉ drordas@uvigo.es

📍 Despacho 409