

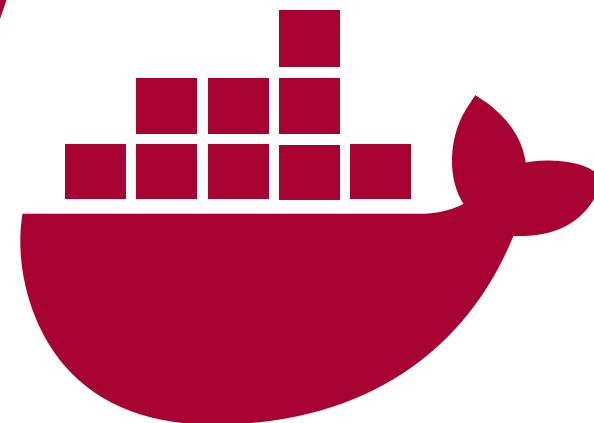
Grado en Ingeniería Informática – Ingeniería del Software


Evolución y Gestión de la Configuración




Escuela Técnica Superior de
Ingeniería Informática

Práctica 5 Contenedores y aislamiento



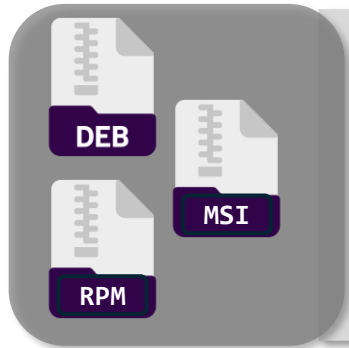
1. Introducción a Docker y DockerHub
 2. Imágenes
 3. Contenedores
 4. Volúmenes y redes
 5. Composición de servicios: Docker Compose
 6. Y yo, ¿qué puedo hacer en mi proyecto?
 7. Tutorial: *docker*
 8. Tutorial: *dockerizando una aplicación sencilla*
 9. Ejercicio práctico: *docker y uvlhub*
- 

1. **Introducción a Docker y DockerHub**
 2. **Imágenes**
 3. **Contenedores**
 4. **Volúmenes y redes**
 5. **Composición de servicios: Docker Compose**
 6. **Y yo, ¿qué puedo hacer en mi proyecto?**
 7. **Tutorial: *docker***
 8. **Tutorial: *dockerizando una aplicación sencilla***
 9. **Ejercicio práctico: *docker* y *uvlhub***
- 

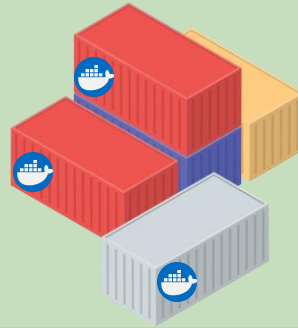
1. Introducción a Docker

Las máquinas virtuales solo permiten aislar la gestión de dependencias. No permiten meter una base de datos ahí.

Los contenedores en nube permiten a las distribuidoras aportar muchos más servicios por menos costes que con muchos entornos virtuales.



Distintas instalaciones de módulos y paquetes de python de forma simultánea



Aislar dependencias más allá de python



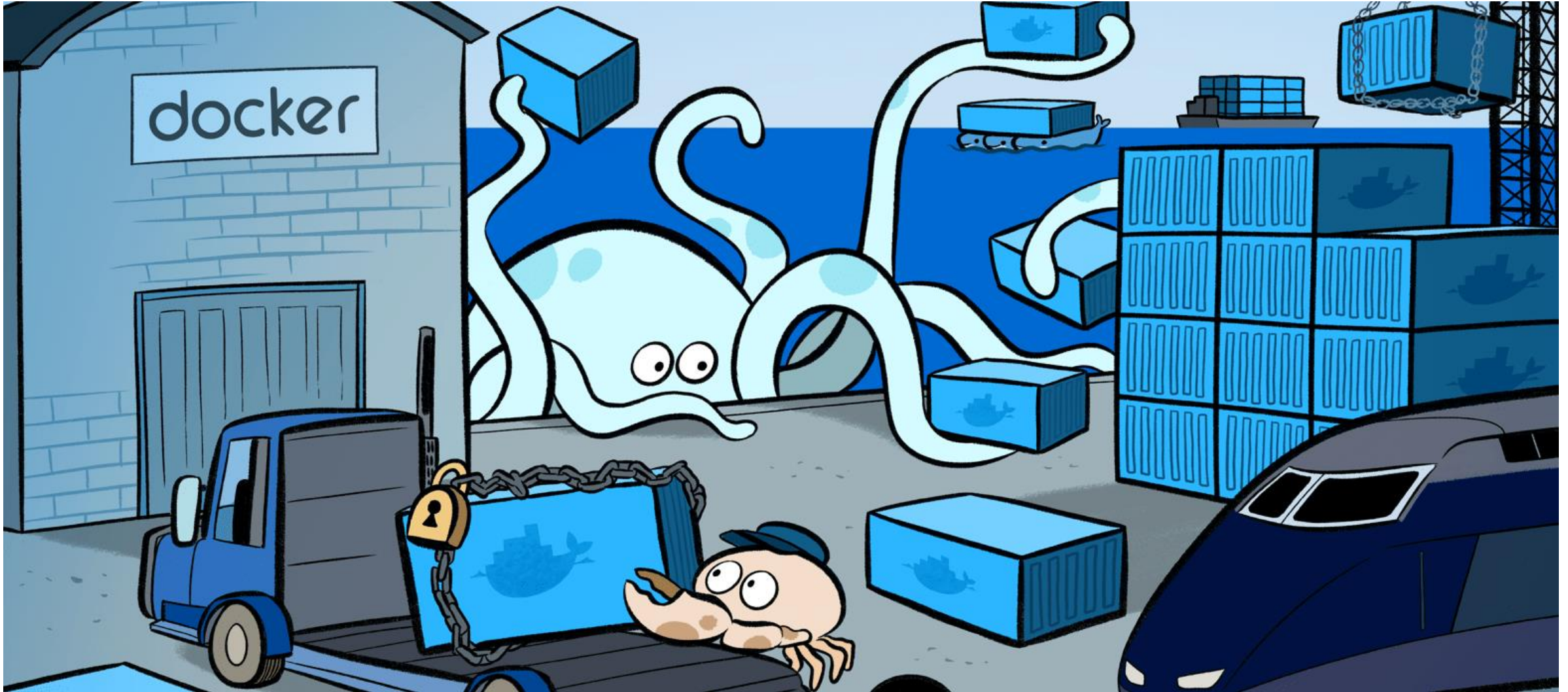
Aislar todas las dependencias del sistema

Overhead y aislamiento

Para aislar los sistemas más allá de lo local (venv), usamos máquinas virtuales o contenedores.

La máquina virtual es como una simulación del sistema completo.. El contenedor actúa directamente sobre el kernel, por lo que se ahorra de dedicar muchísimos recursos a simular el sistema completo. Cuanto más nos aislemos, mayor el coste.

1. Introducción a Docker



1. Introducción a Docker

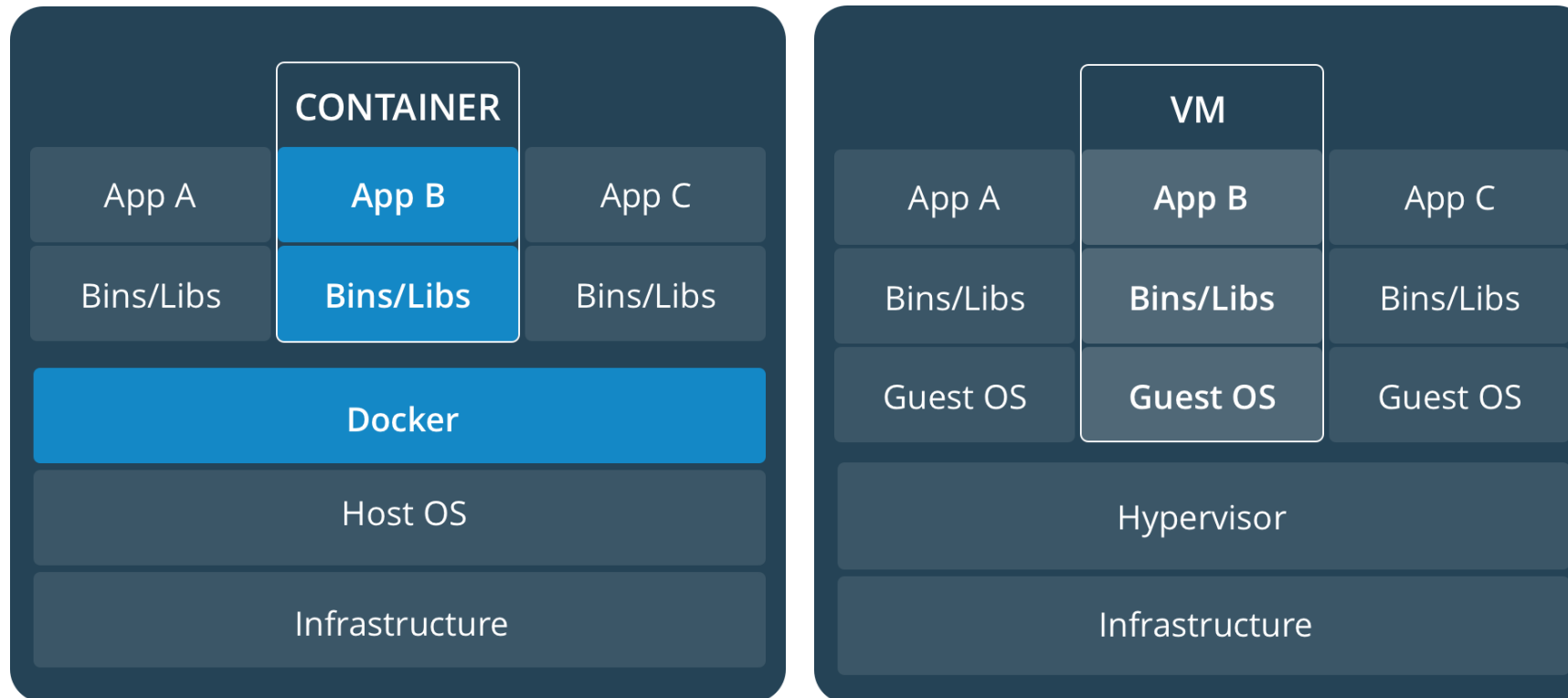


Docker parte de los contenedores de Linux.

Cuando lo usamos en Windows, usamos por detrás un hipervisor, que es prácticamente equivalente a usar una máquina virtual.

1. Introducción a Docker

¿Por qué usar contenedores si ya tenemos máquinas virtuales?



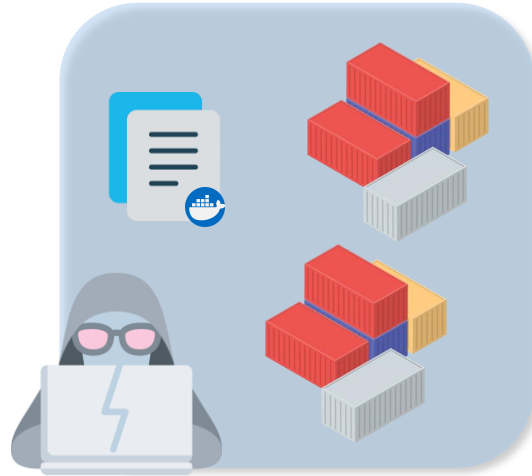
1. Introducción a Docker

¿Por qué usar contenedores si ya tenemos máquinas virtuales?




1. Introducción a DockerHub

Repositorios con ejemplos de imágenes.
Son usables, vamos. Si quiero hacer algo, primero miro
en DockerHub



<https://hub.docker.com/>



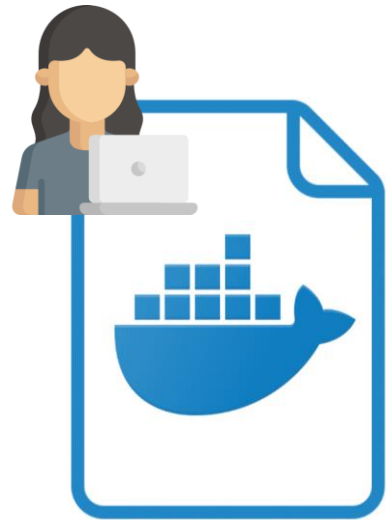
1. Introducción a Docker y DockerHub
 2. **Imágenes**
 3. Contenedores
 4. Volúmenes y redes
 5. Composición de servicios: Docker Compose
 6. Y yo, ¿qué puedo hacer en mi proyecto?
 7. Tutorial: *docker*
 8. Tutorial: *dockerizando una aplicación sencilla*
 9. Ejercicio práctico: *docker y uvlhub*
- 

2. Imágenes

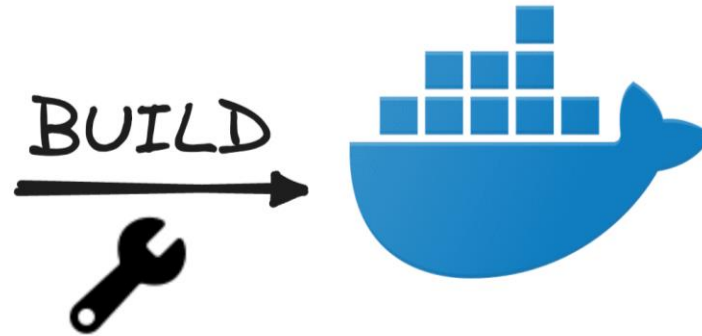
Para hacer un contenedor:
O escribimos un dockerfile o descargamos una imagen que ya existe y modificándola.

El dockerfile define una imagen.

Los contenedores son instancias ejecutables de una imagen.



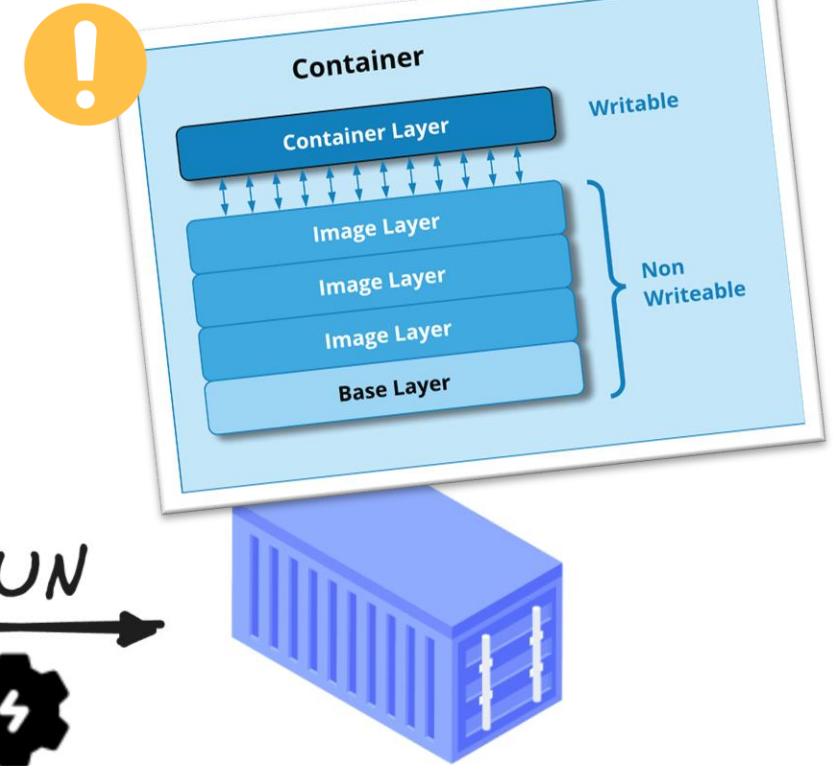
Dockerfile



Docker image



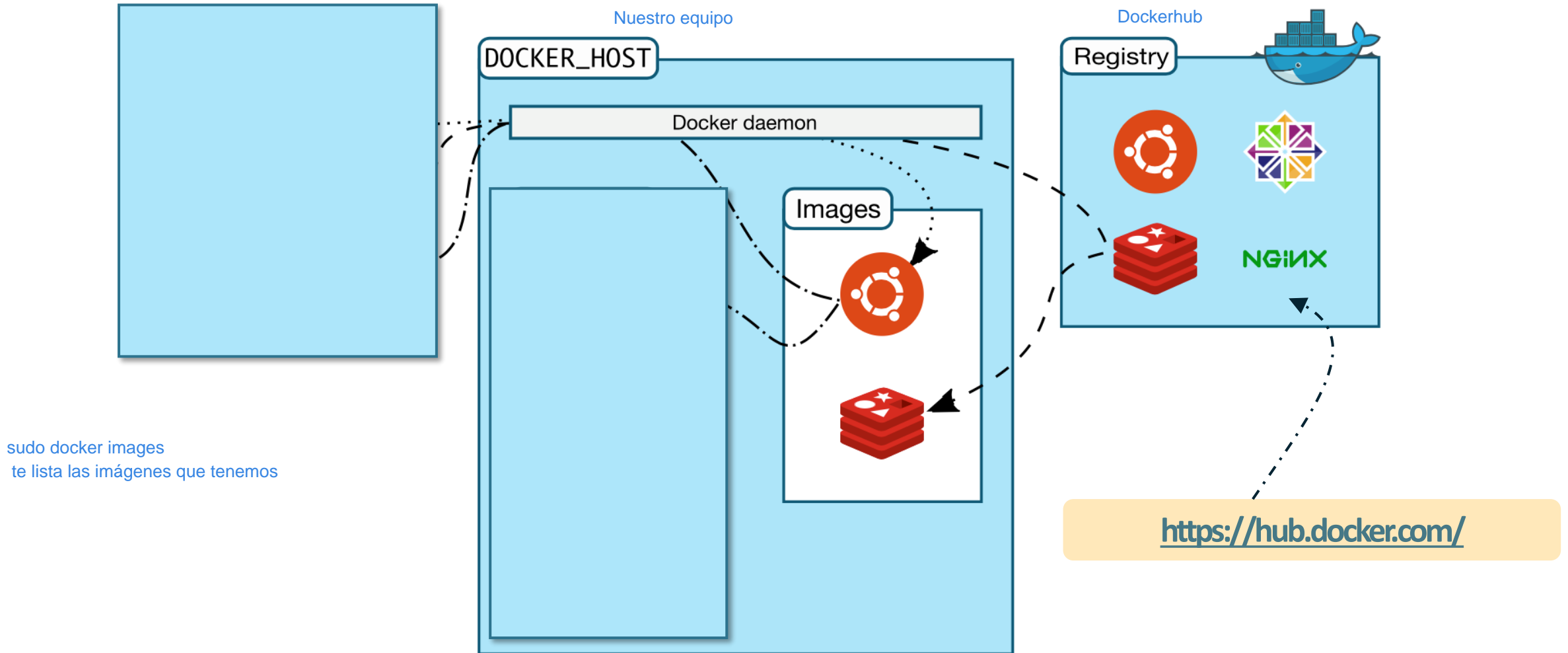
Docker container




Plantillas de solo lectura que contienen el código y las dependencias necesarias para ejecutar aplicaciones

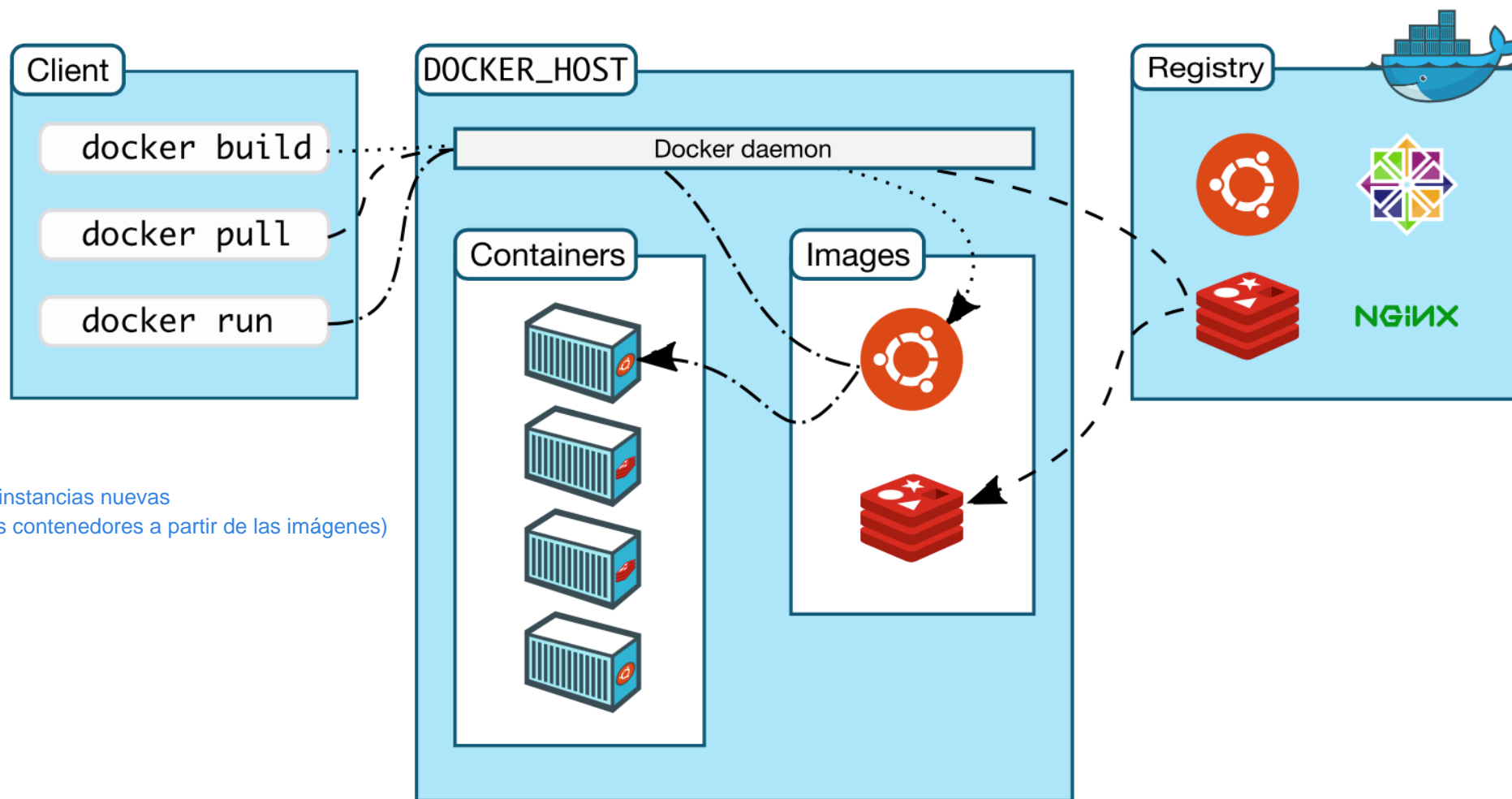
2. Imágenes

Y si no escribo un Dockerfile, ¿de dónde salen las imágenes?



1. Introducción a Docker y DockerHub
 2. Imágenes
 3. **Contenedores**
 4. Volúmenes y redes
 5. Composición de servicios: Docker Compose
 6. Y yo, ¿qué puedo hacer en mi proyecto?
 7. Tutorial: *docker*
 8. Tutorial: *dockerizando una aplicación sencilla*
 9. Ejercicio práctico: *docker y uvlhub*
- 

3. Contenedores



docker run crea instancias nuevas
(es decir, nuevos contenedores a partir de las imágenes)

3. Contenedores

Contenedores en la nube

Cada contenedor debería tener un único propósito, por lo que para una sola aplicación podemos desplegar varios contenedores.

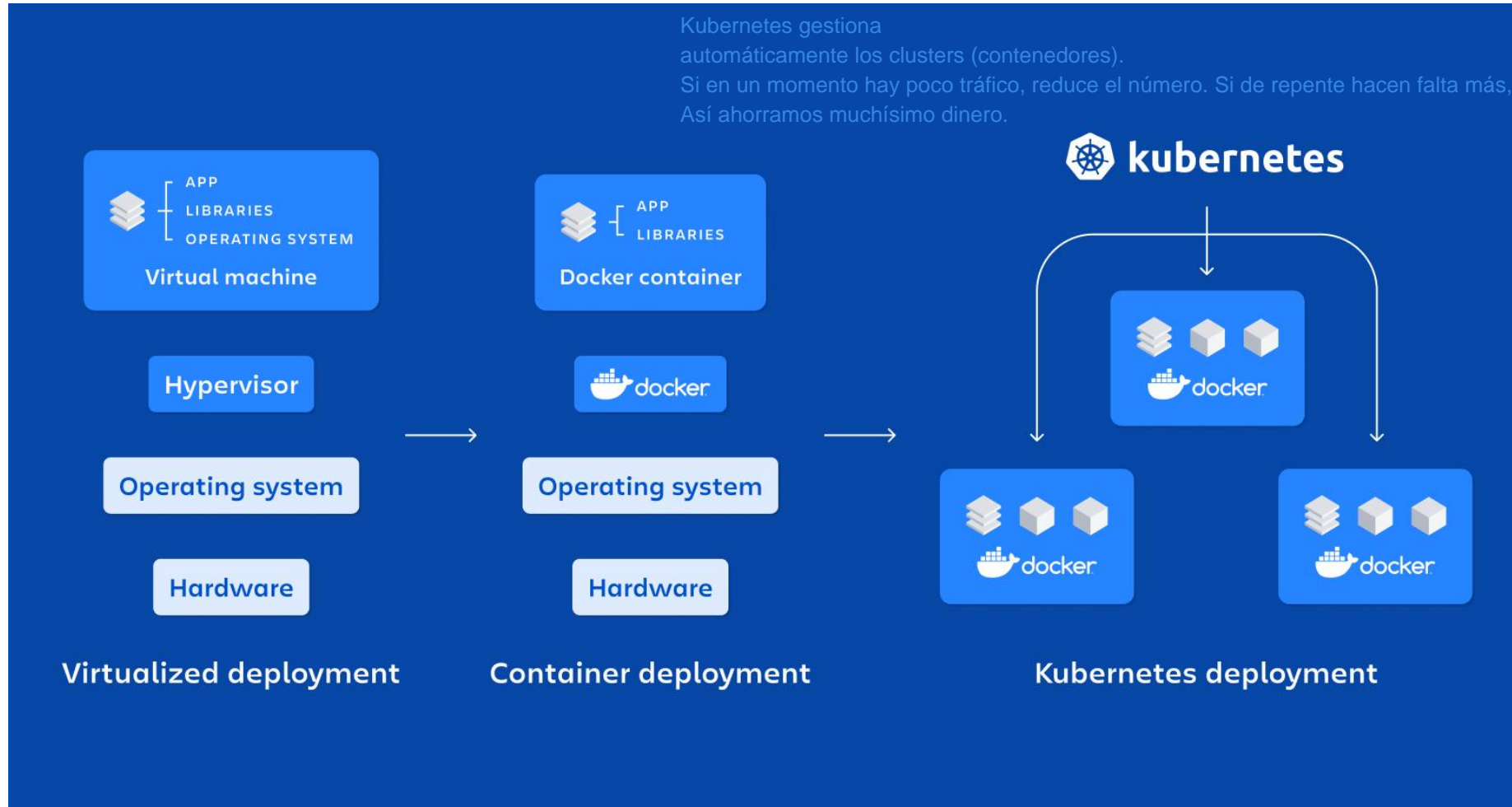
uvlhub usa al menos tres.

Normalmente, como mínimo, la base de datos hay que separarla de la aplicación en sí.

A lo mejor recibimos muchas peticiones HTTP por lo que necesitamos varias de la aplicación, pero hay muy poco tráfico con la BD, por lo que una sola vale.

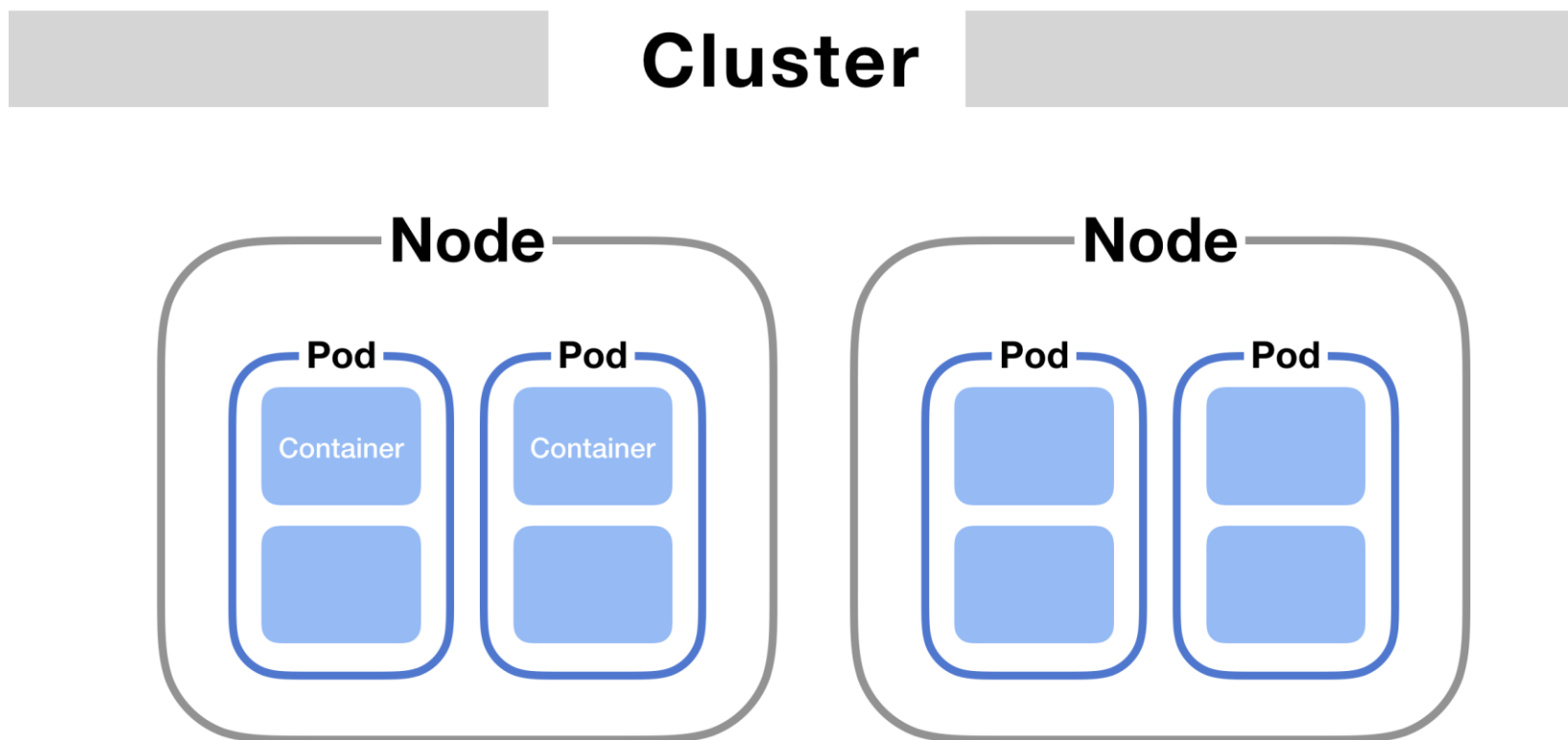
Kubernetes gestiona automáticamente los clusters (contenedores).

Si en un momento hay poco tráfico, reduce el número. Si de repente hacen falta más, aumenta el número. Así ahorramos muchísimo dinero.



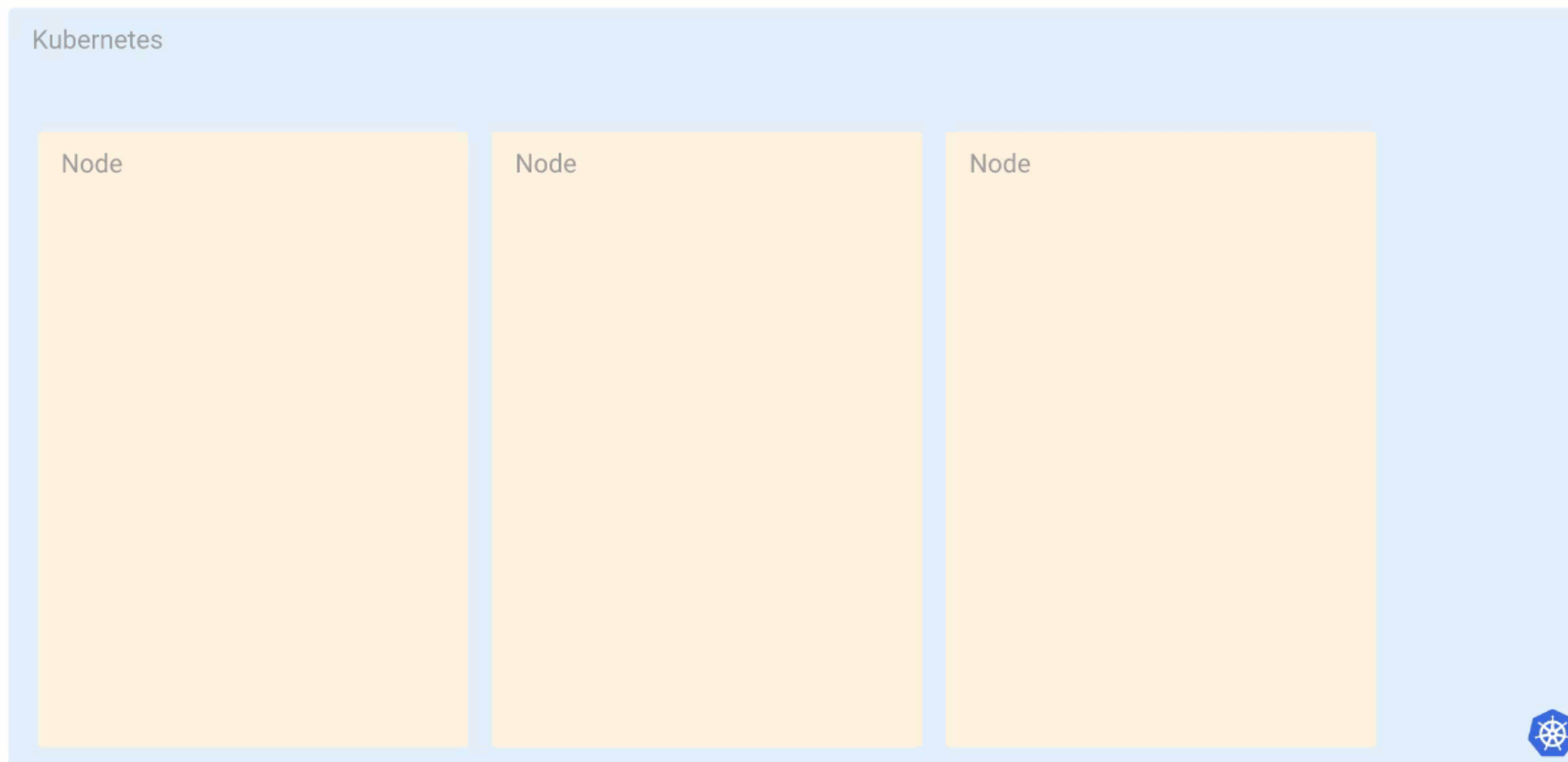
3. Contenedores


Contenedores en la nube



3. Contenedores

Contenedores en la nube

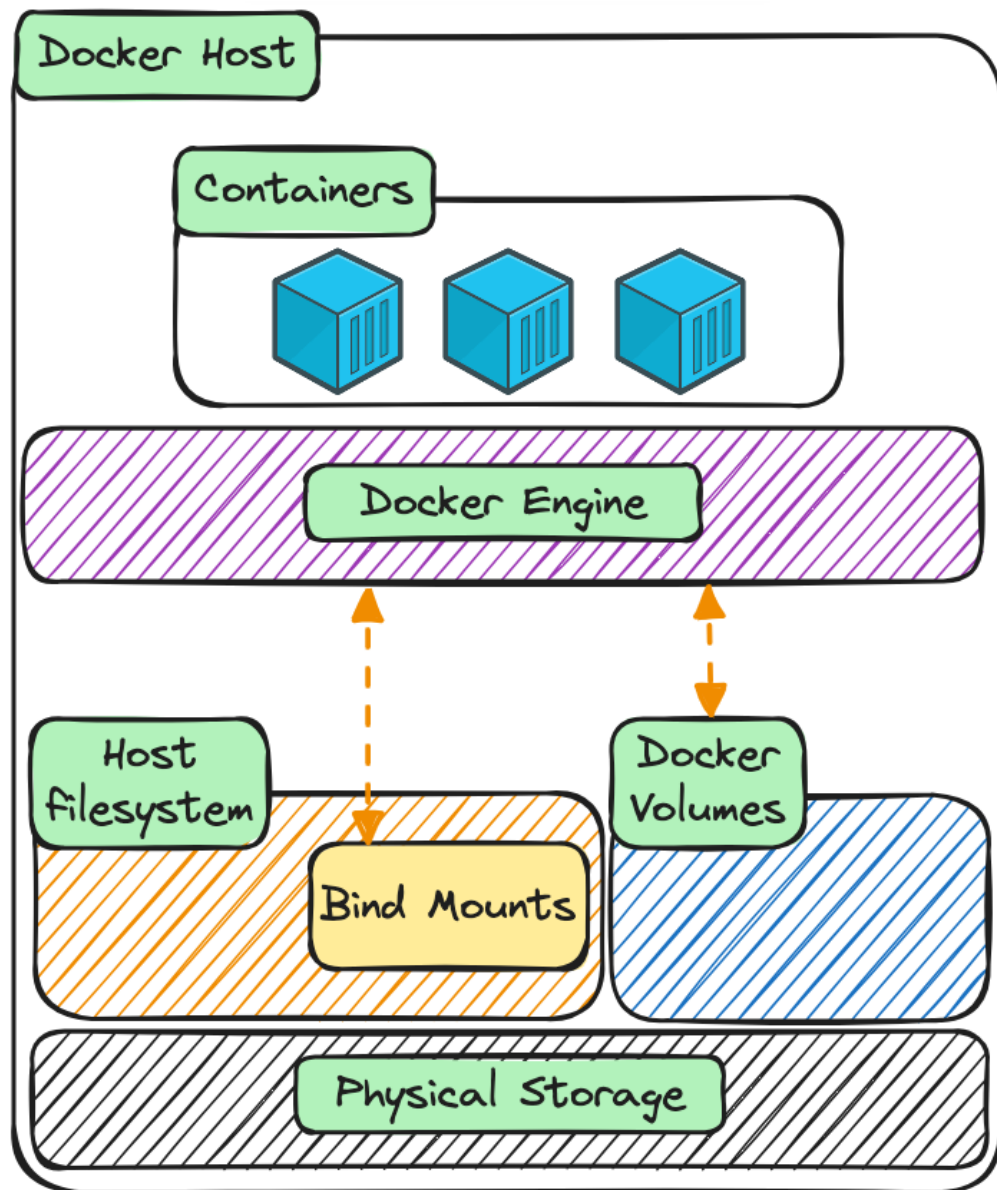


1. Introducción a Docker y DockerHub
 2. Imágenes
 3. Contenedores
 4. **Volúmenes y redes**
 5. Composición de servicios: Docker Compose
 6. Y yo, ¿qué puedo hacer en mi proyecto?
 7. Tutorial: *docker*
 8. Tutorial: *dockerizando una aplicación sencilla*
 9. Ejercicio práctico: *docker y uvlhub*
- 

4. Volúmenes y redes

Montamos el proyecto en un Bind Mount si queremos hacer cambios en el host que van a tener su persistencia correspondiente en los contenedores.

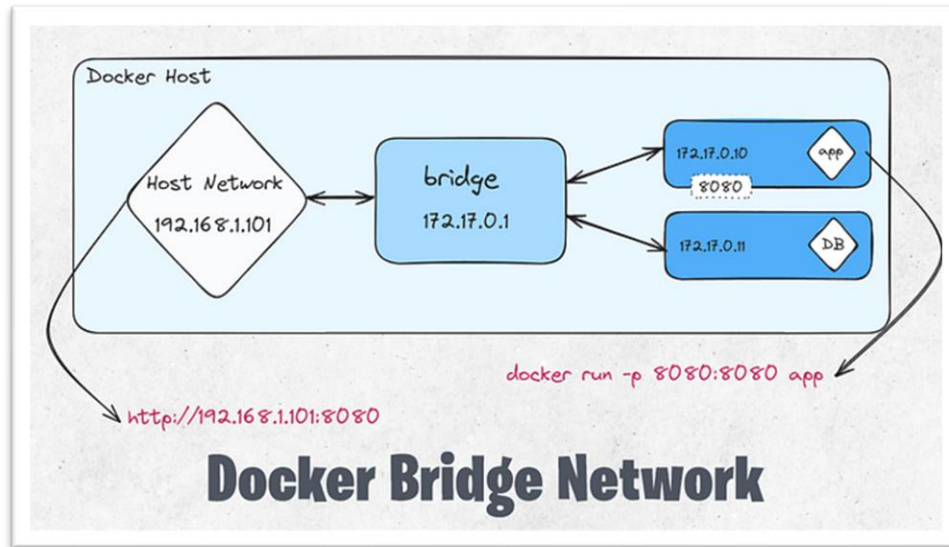
Si borramos la carpeta, adiós.



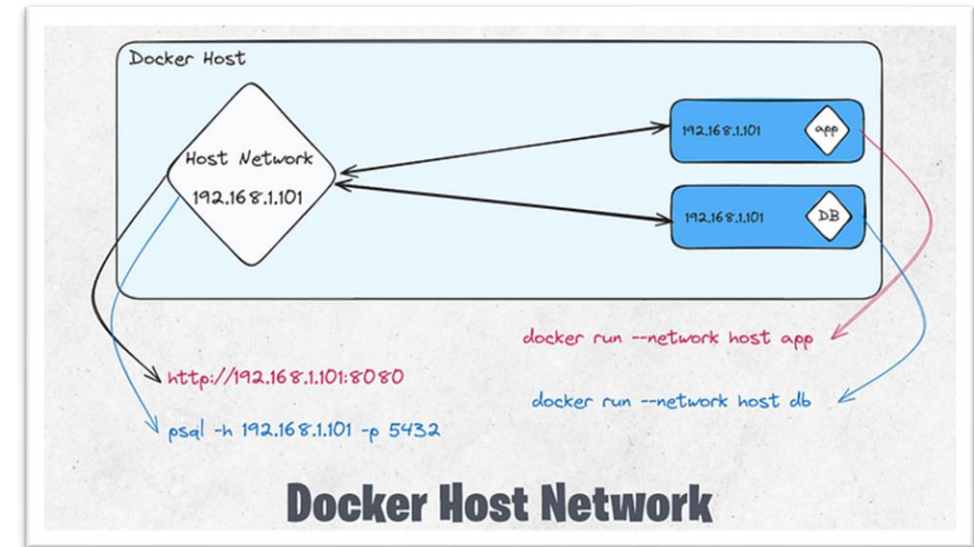
Los volúmenes permiten que los datos **se conserven incluso si el contenedor se elimina o reinicia**, manteniendo los archivos de manera persistente en el host.

4. Volúmenes y redes

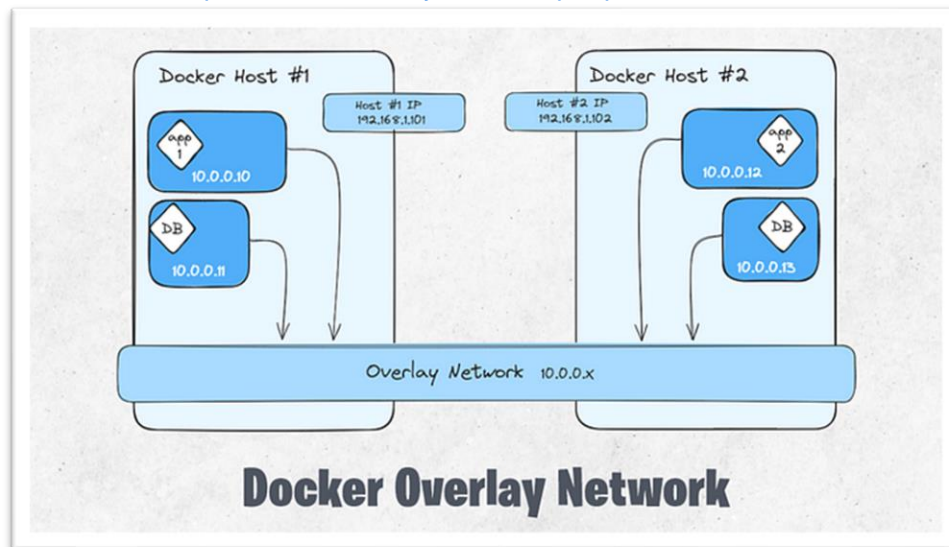
Bridge es la más normal.



Tres tipos:



Kubernetes usa por defecto Overlay Network, porque "están en diferentes redes" (?)




Los contenedores tienen que poder conectarse y hablarse. Montamos una red de contenedores.

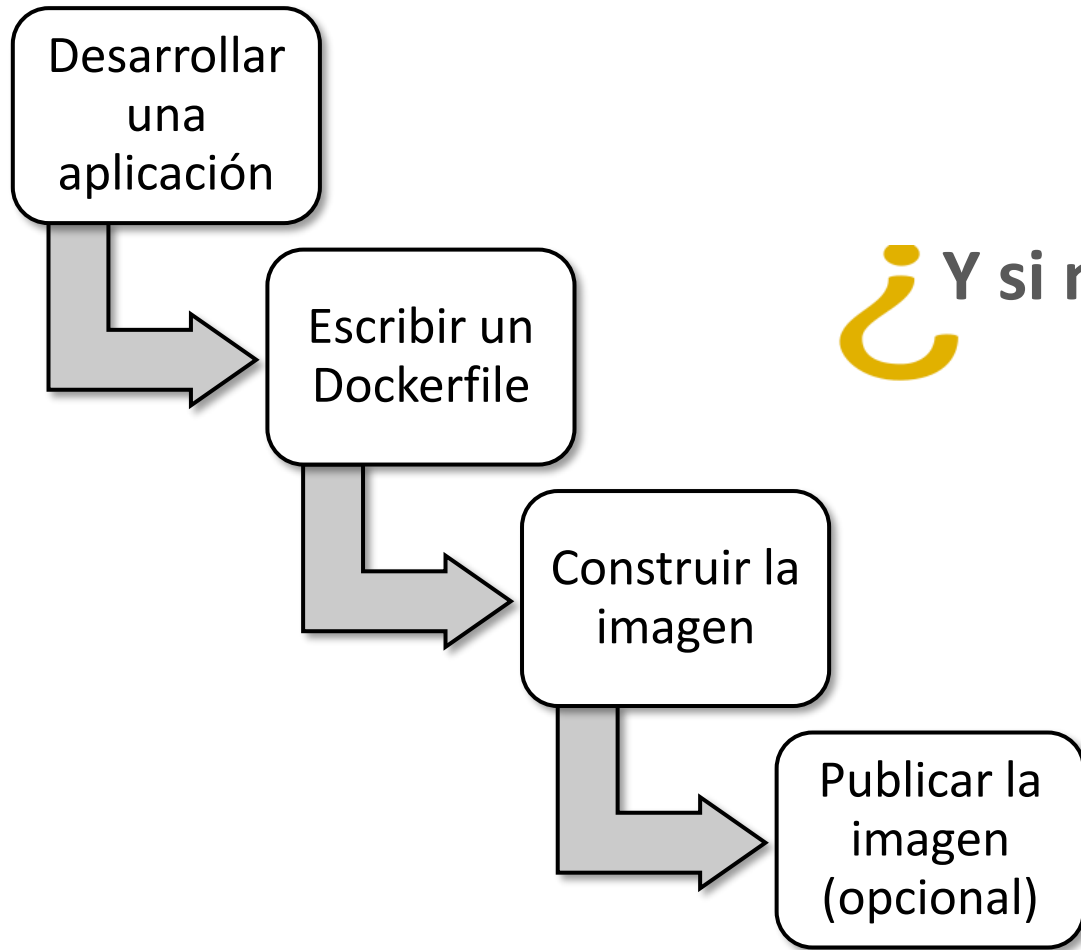
De normal, una BD y una aplicación se conectan mediante su puerto.

Ahora hay que hacer algo similar, pero con una red virtual.

Las redes permiten que los contenedores **se comuniquen entre sí, con el host y con redes externas**. Docker proporciona **varios tipos de redes** (bridge, host, overlay, etc.) que permiten controlar el aislamiento, la seguridad y la visibilidad entre contenedores, facilitando la configuración de infraestructuras distribuidas o microservicios.

1. Introducción a Docker y DockerHub
 2. Imágenes
 3. Contenedores
 4. Volúmenes y redes
 5. **Composición de servicios: Docker Compose**
 6. Y yo, ¿qué puedo hacer en mi proyecto?
 7. Tutorial: *docker*
 8. Tutorial: *dockerizando una aplicación sencilla*
 9. Ejercicio práctico: *docker y uvlhub*
- 

5. Composición de servicios: Docker compose

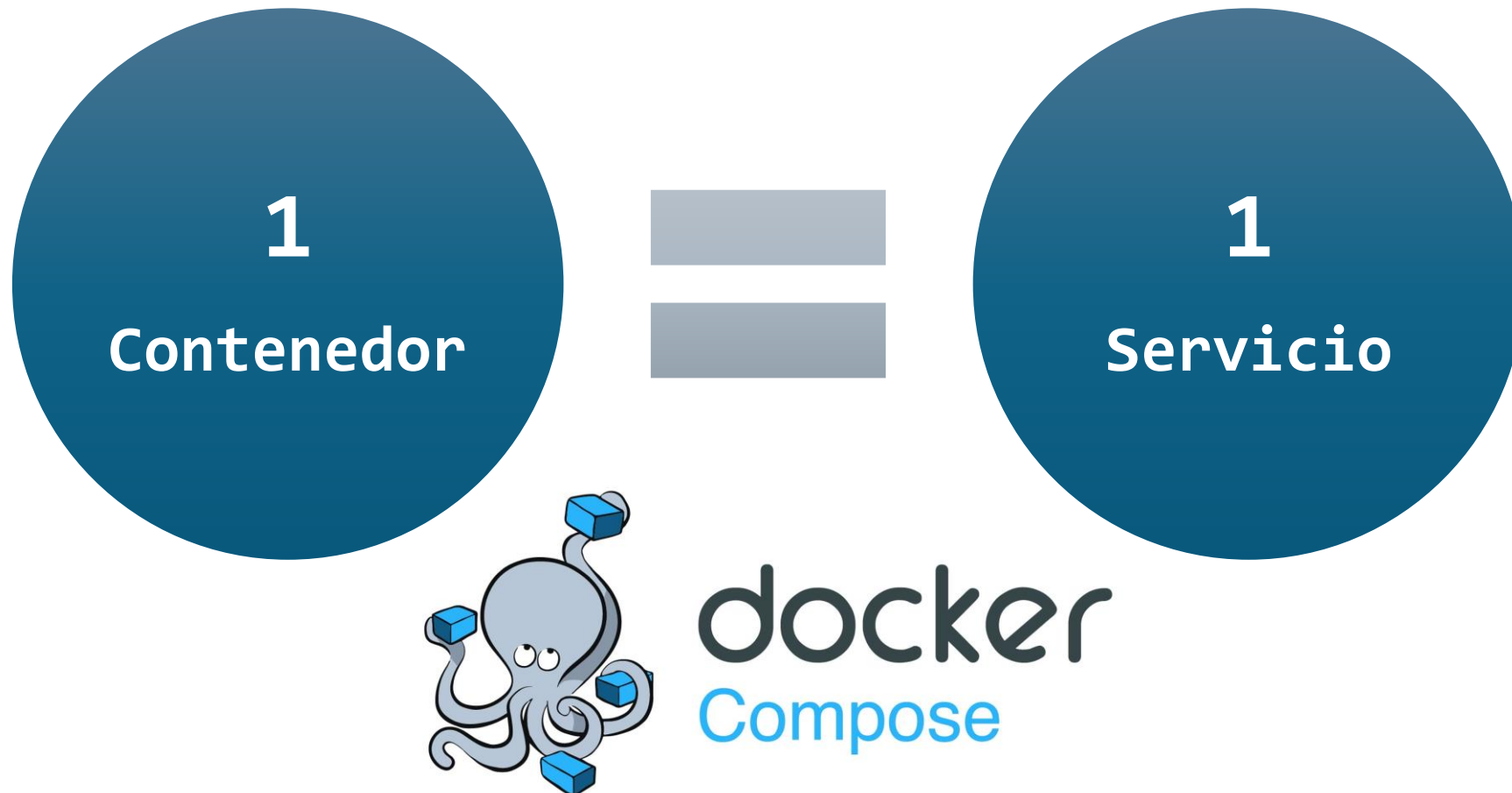


¿Y si necesito varias imágenes y varios contenedores?



5. Composición de servicios: Docker compose

Principio de responsabilidad única

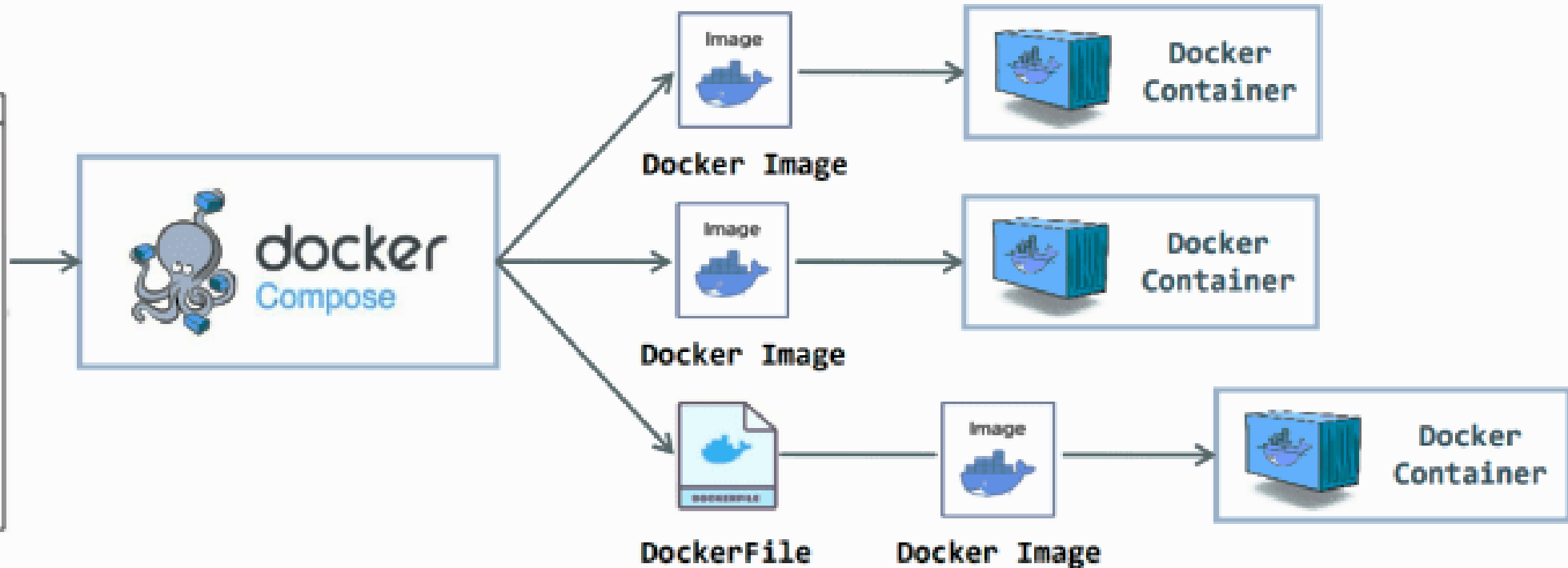



Nos permite automatizar la creación, configuración y despliegue de los contenedores.

5. Composición de servicios: Docker compose

docker-compose.yml

```
*** docker-compose.yml
version: "3.7"
services:
  db:
    image: mysql:5.0.19
    restart: always
    environment:
      - MYSQL_DATABASE=example
      - MYSQL_ROOT_PASSWORD=password
  app:
    build: app
    restart: always
  web:
    build: web
    restart: always
    ports:
      - 80:80
```



1. Introducción a Docker y DockerHub
 2. Imágenes
 3. Contenedores
 4. Volúmenes y redes
 5. Composición de servicios: Docker Compose
 6. Y yo, ¿qué puedo hacer en mi proyecto?
 7. Tutorial: *docker*
 8. Tutorial: *dockerizando una aplicación sencilla*
 9. Ejercicio práctico: *docker y uvlhub*
- 


6. Y yo, ¿qué puedo hacer en mi proyecto?



**¡Diseña tu propio
despliegue en Docker!**

Modifica y/o crea
Dockerfiles propios


Añade nuevos
contenedores,
volúmenes y redes a tus
docker-compose.yml

1. Introducción a Docker y DockerHub
 2. Imágenes
 3. Contenedores
 4. Volúmenes y redes
 5. Composición de servicios: Docker Compose
 6. Y yo, ¿qué puedo hacer en mi proyecto?
 - 7. Tutorial: *docker***
 8. Tutorial: *dockerizando una aplicación sencilla*
 9. Ejercicio práctico: *docker y uvlhub*
- 

7. Tutorial: *docker*




https://1984.lsi.us.es/wiki-egc/index.php/Tutorial_Campo_de_entrenamiento_de_Docker

1. Introducción a Docker y DockerHub
 2. Imágenes
 3. Contenedores
 4. Volúmenes y redes
 5. Composición de servicios: Docker Compose
 6. Y yo, ¿qué puedo hacer en mi proyecto?
 7. Tutorial: *docker*
 8. Tutorial: *dockerizando una aplicación sencilla*
 9. Ejercicio práctico: *docker* y *uvlhub*
- 

8. Tutorial: *dockerizando* una aplicación sencilla



https://1984.lsi.us.es/wiki-egc/index.php/Tutorial_Dockerizando_una_aplicación

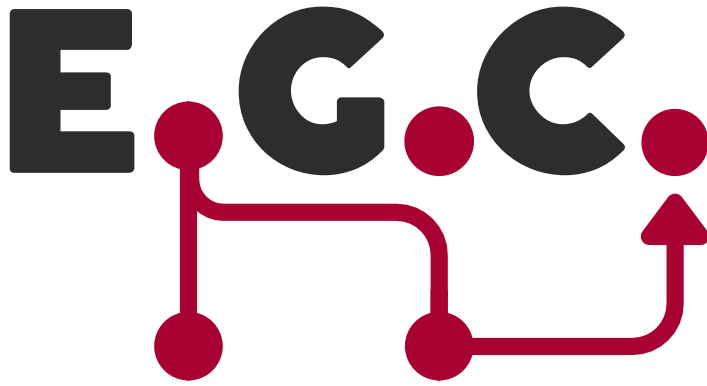
1. Introducción a Docker y DockerHub
 2. Imágenes
 3. Contenedores
 4. Volúmenes y redes
 5. Composición de servicios: Docker Compose
 6. Y yo, ¿qué puedo hacer en mi proyecto?
 7. Tutorial: *docker*
 8. Tutorial: *dockerizando una aplicación sencilla*
 9. Ejercicio práctico: *docker y uvlhub*
- 

9. Ejercicio práctico: docker y uvlhub



docs.uvlhub.io/installation/installation_with_docker





Grado en Ingeniería Informática – Ingeniería del Software

Evolución y Gestión de la Configuración



Escuela Técnica Superior de
Ingeniería Informática

¡Gracias!

*“Dado un número
suficientemente elevado de
ojos, todos los errores se
vuelven obvios.”*

- Eric S. Raymond (ley de Linus)