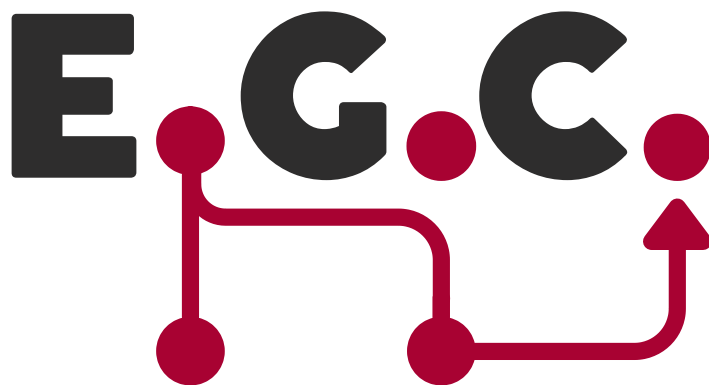




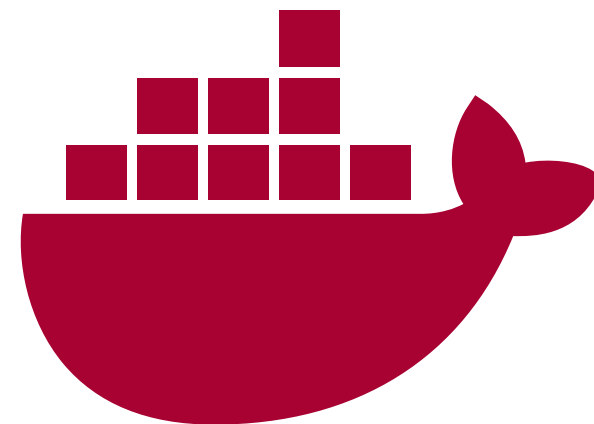
Grado en Ingeniería Informática – Ingeniería del Software

Evolución y Gestión de la Configuración



Escuela Técnica Superior de
Ingeniería Informática

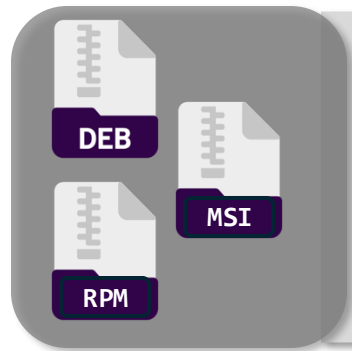
Práctica 5 Contenedores, dev-containers aislamiento



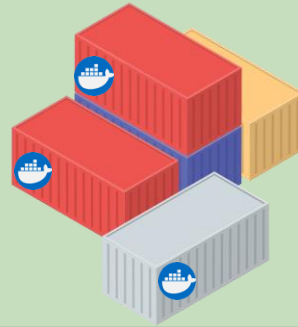
1. **Introducción a Docker y DockerHub**
 2. **Imágenes**
 3. **Contenedores**
 4. **Volúmenes y redes**
 5. **Composición de servicios: Docker Compose**
 6. **Dev-containers**
 7. **Y yo, ¿qué puedo hacer en mi proyecto?**
 8. **Tutorial: *dockerizando uvlhub***
- 

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

1. Introducción a Docker



Distintas instalaciones de módulos y paquetes de python de forma simultánea



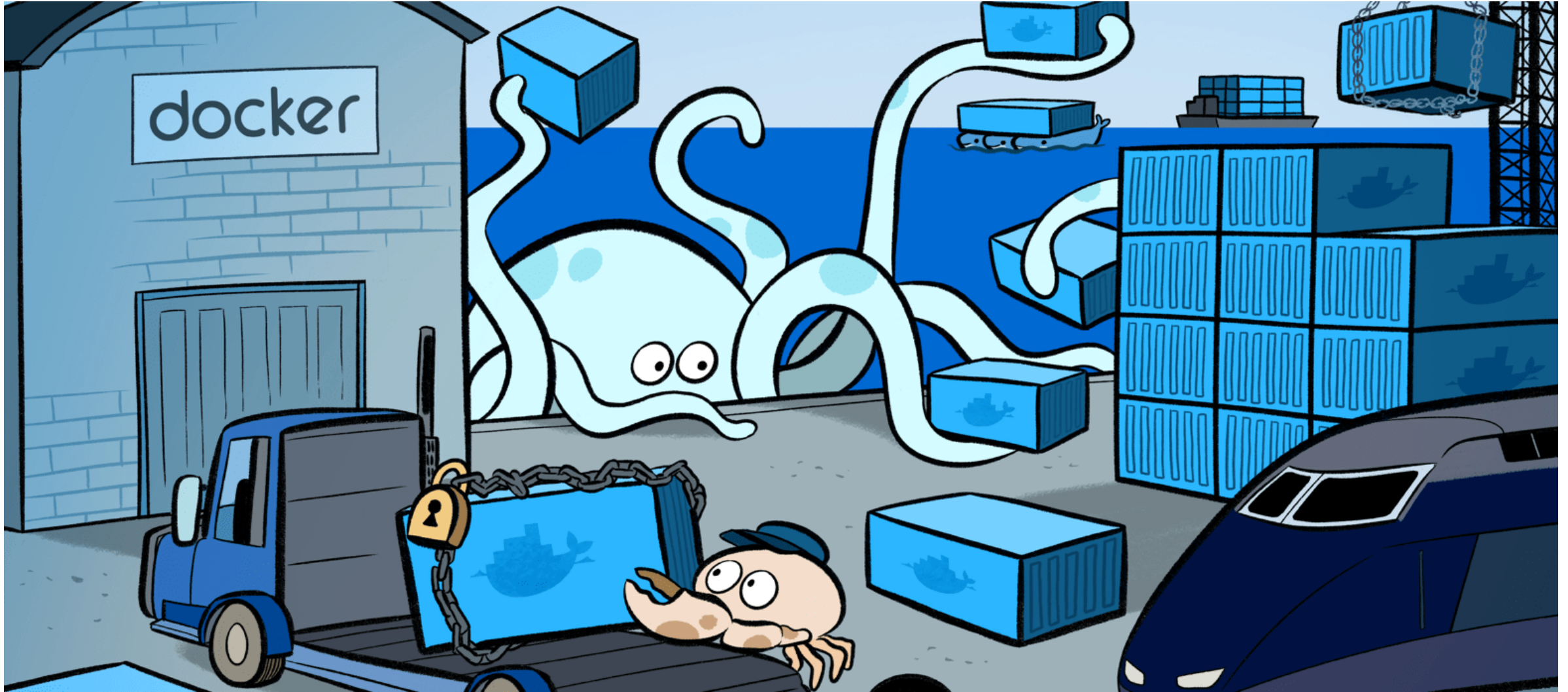
Aislar dependencias más allá de python



Aislar todas las dependencias del sistema

Overhead y aislamiento

1. Introducción a Docker

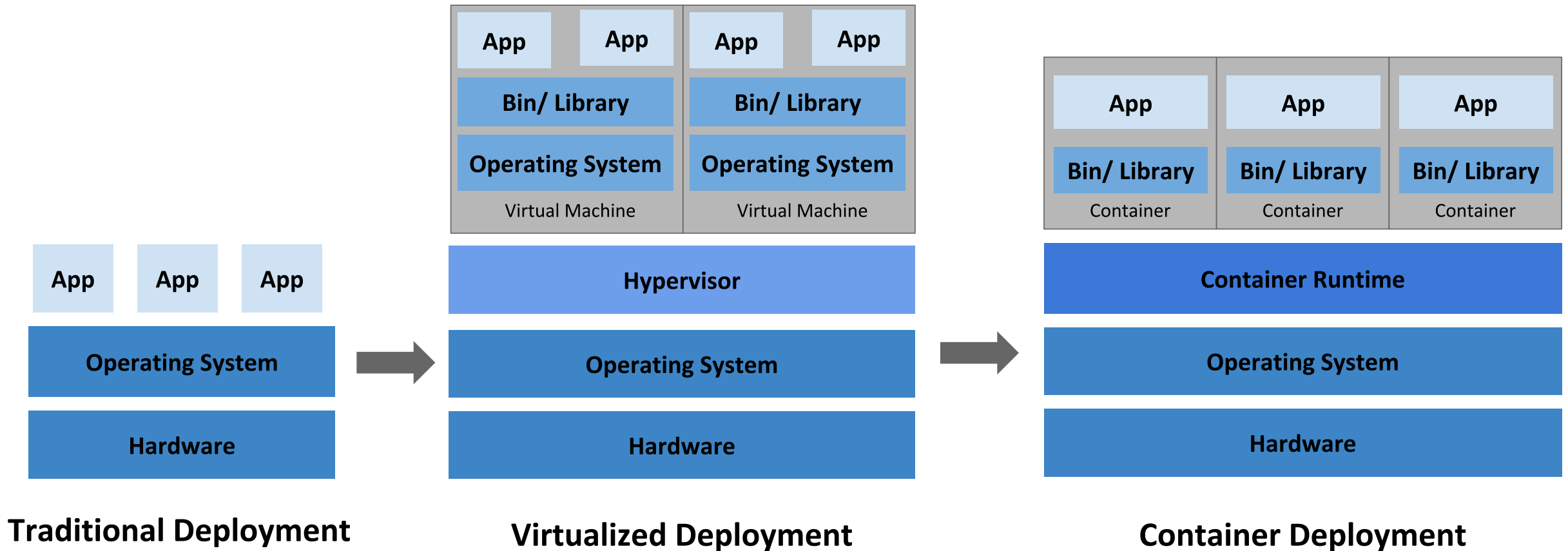


1. Introducción a Docker

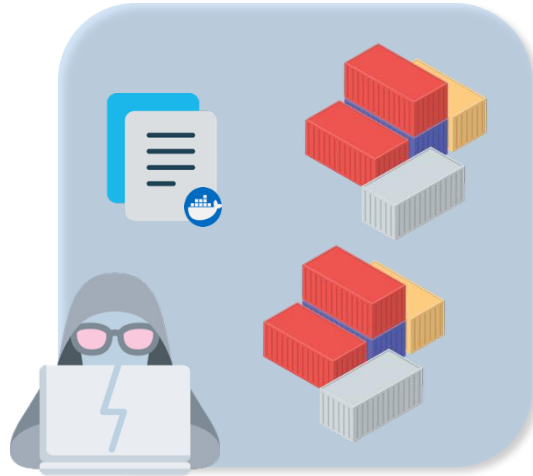
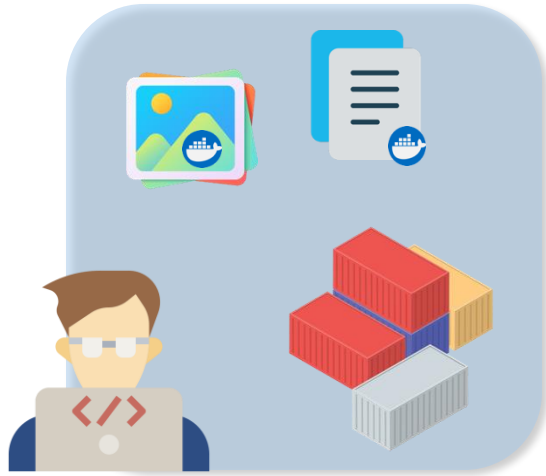


1. Introducción a Docker

¿Por qué usar contenedores si ya tenemos máquinas virtuales?

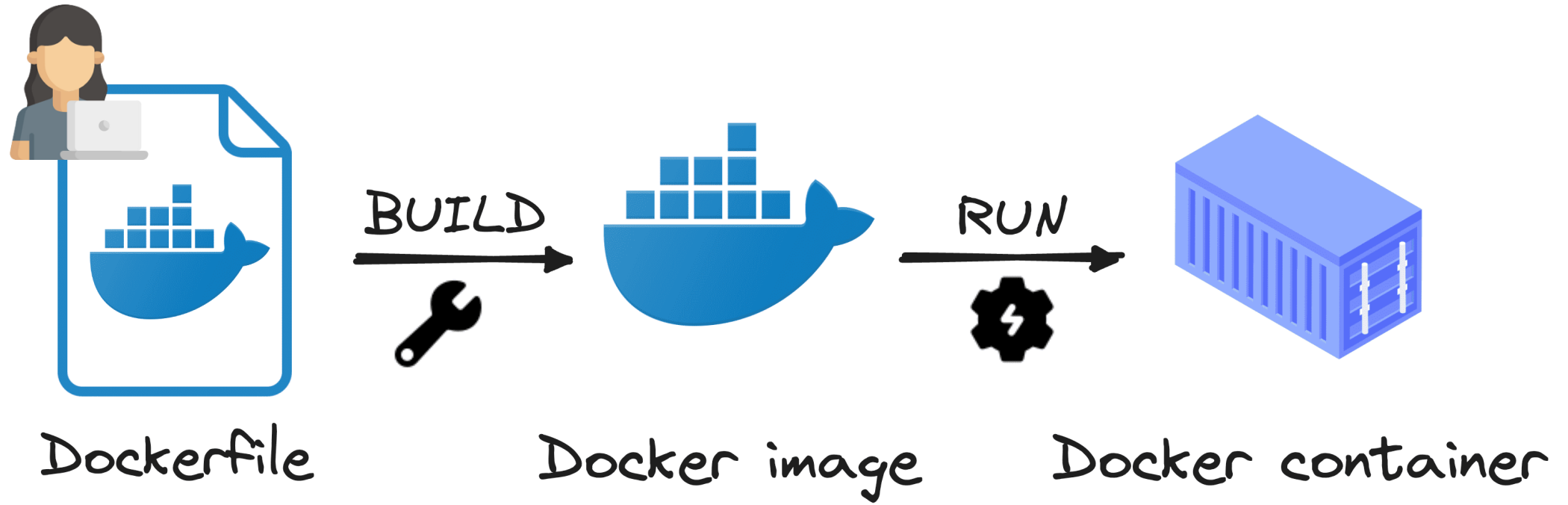


1. Introducción a DockerHub

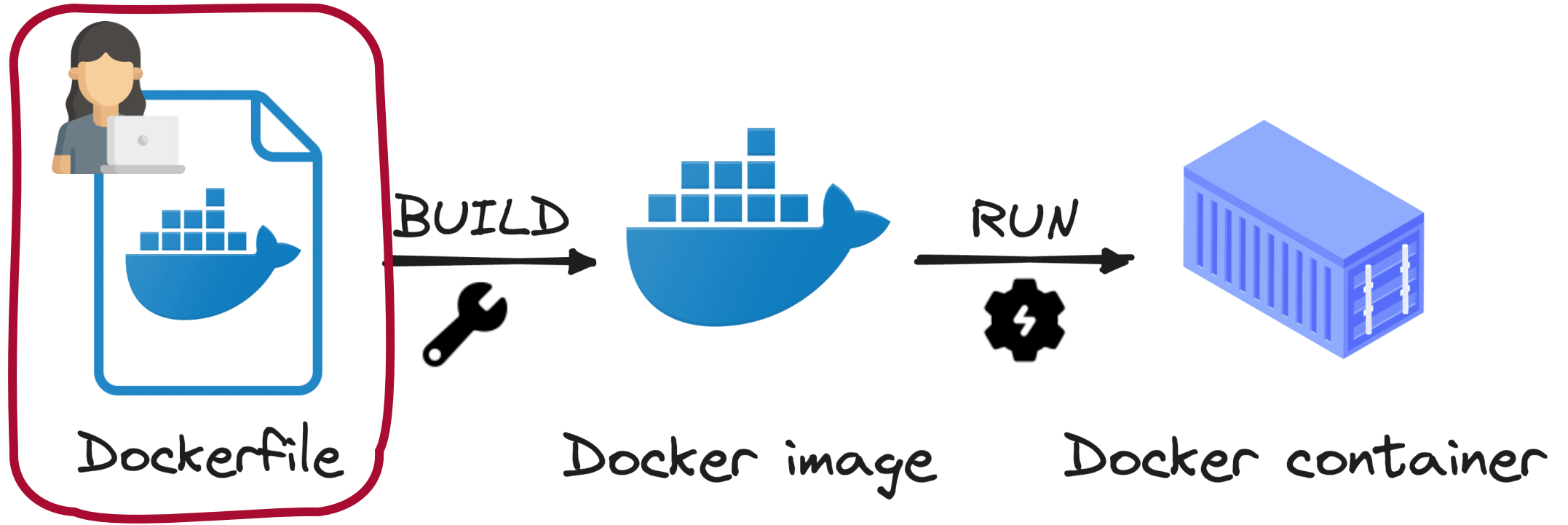


1. Introducción a Docker y DockerHub
 2. **Imágenes**
 3. Contenedores
 4. Volúmenes y redes
 5. Composición de servicios: Docker Compose
 6. Dev-containers
 7. Y yo, ¿qué puedo hacer en mi proyecto?
 8. Tutorial: *dockerizando uvlhub*
- 

2. Imágenes

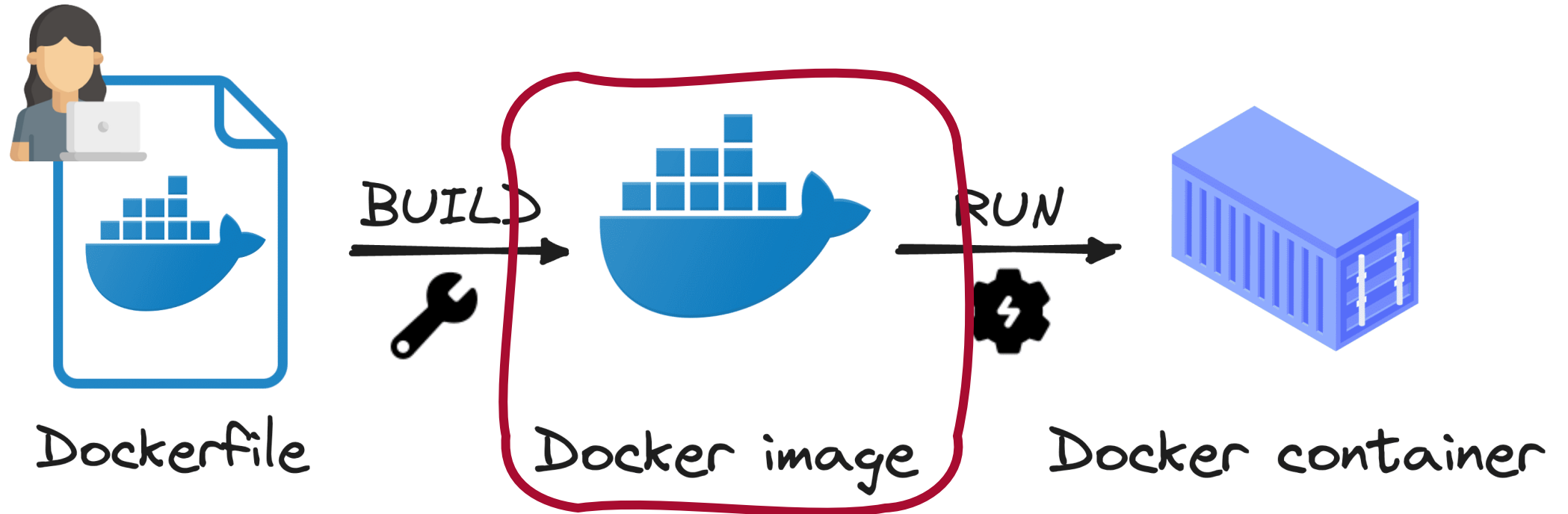


2. Imágenes



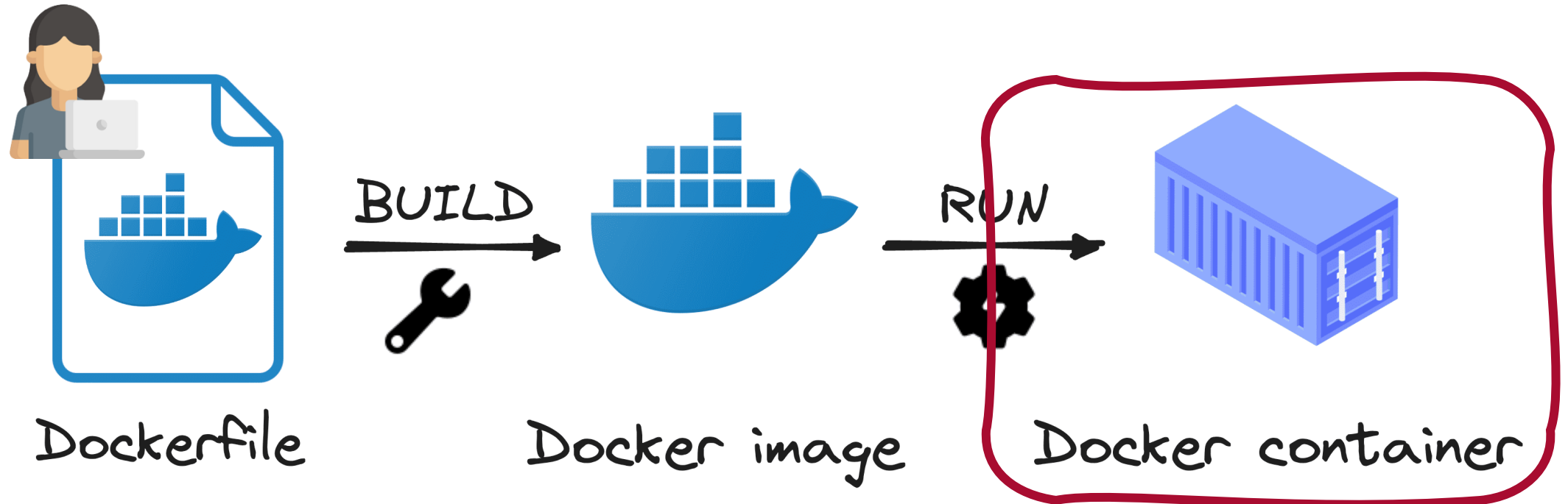
Archivo de texto que contiene las instrucciones para construir una imagen de Docker de forma automática

2. Imágenes



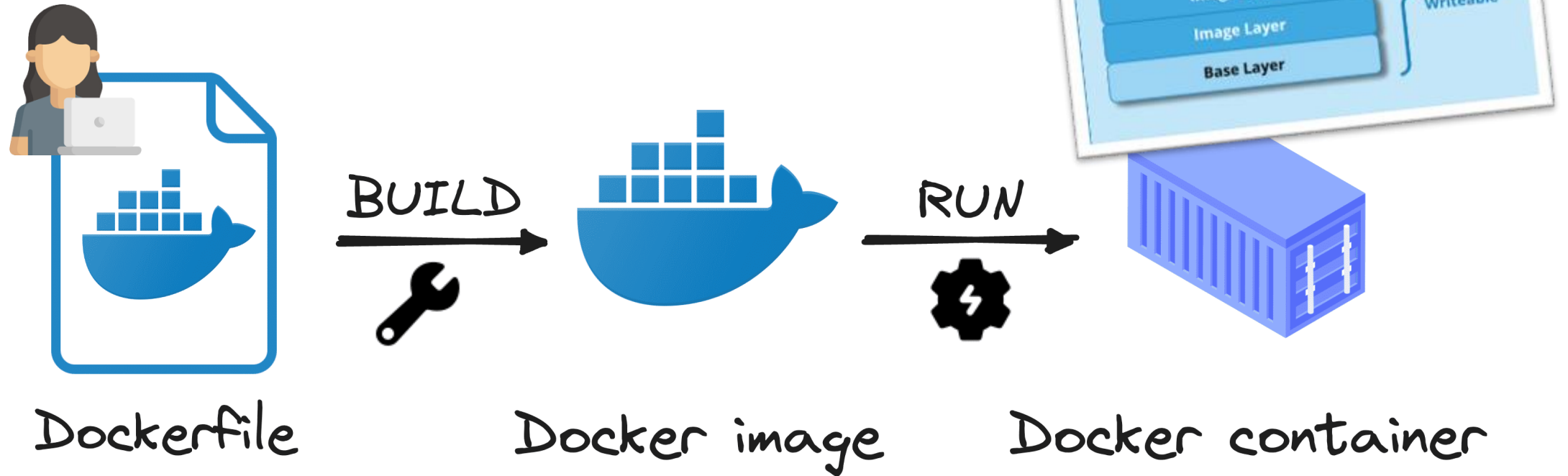
Plantillas de solo lectura que contienen el código y las dependencias necesarias para ejecutar aplicaciones

2. Imágenes



Instancia en ejecución de una imagen de Docker, es decir, un entorno aislado donde esa imagen cobra vida y la aplicación se ejecuta

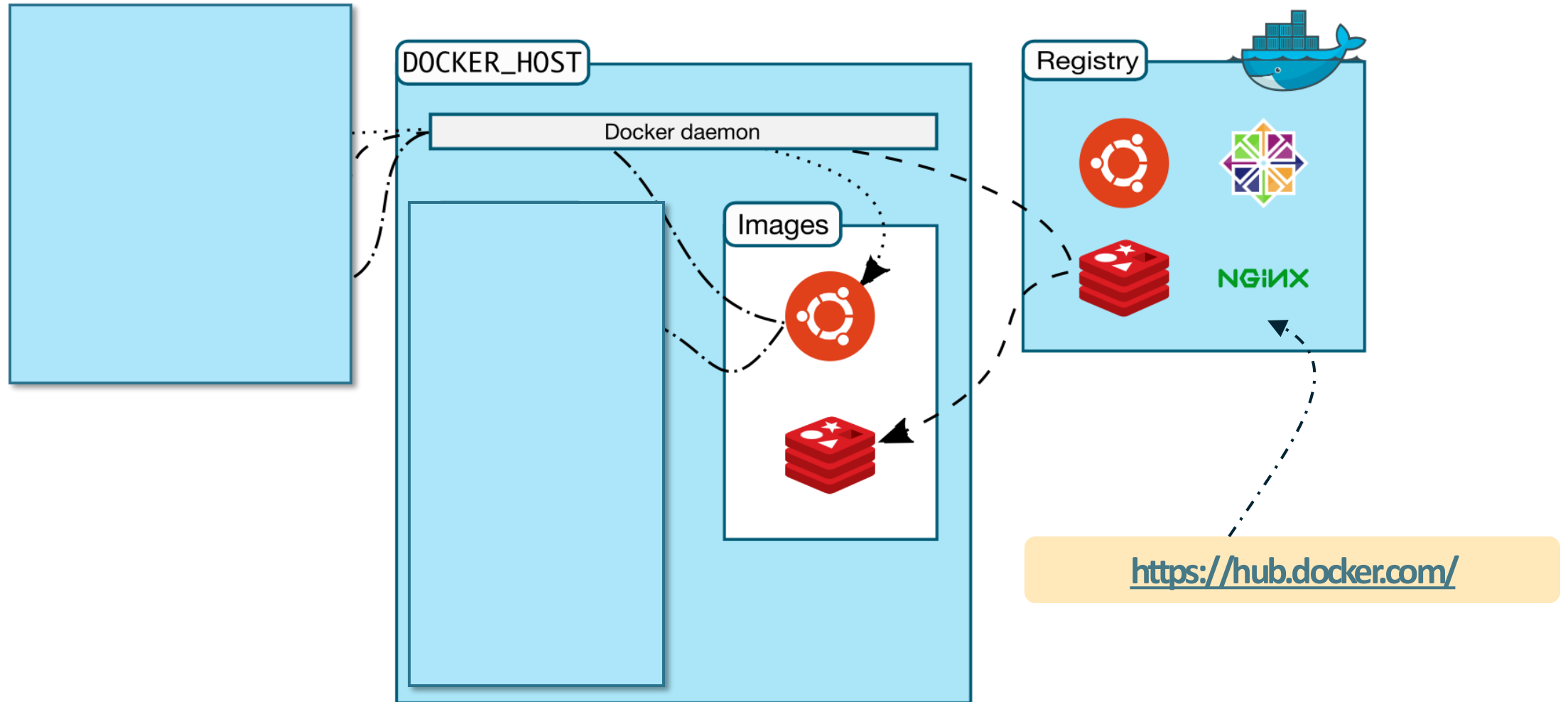
2. Imágenes



Las capas son partes apiladas de una imagen que se reutilizan para ahorrar espacio y acelerar las construcciones

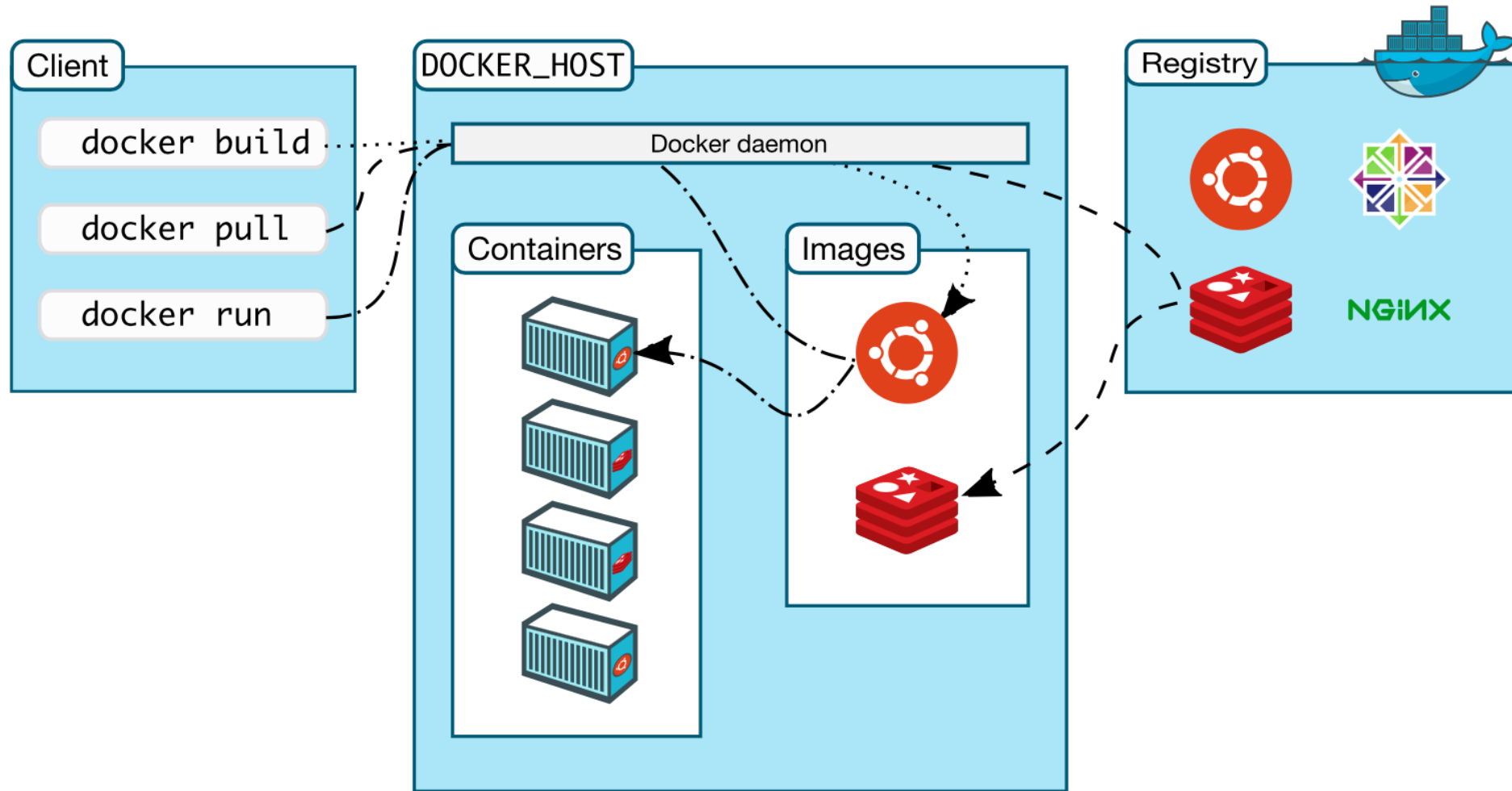
2. Imágenes

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. Dev-containers
 7. Y yo, ¿qué puedo hacer en mi proyecto?
 8. Tutorial: *dockerizando uvlhub*
- 

3. Contenedores



3. Contenedores

Contenedores en la nube: Kubernetes

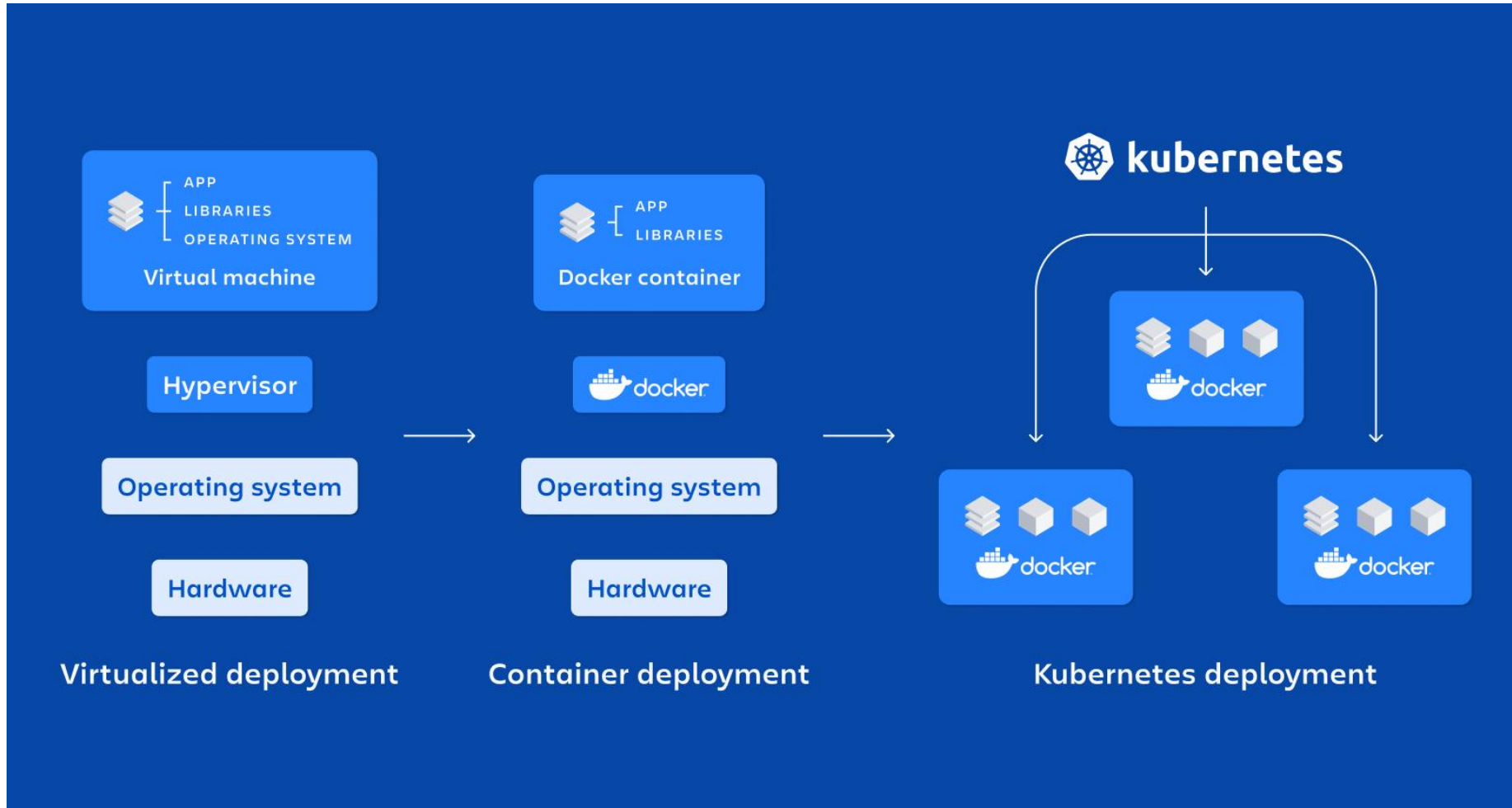


Kubernetes es una plataforma que organiza y gestiona automáticamente contenedores (como los de Docker), asegurando que las aplicaciones se ejecuten, escalen y se mantengan funcionando sin intervención manual.

Kubernetes: del griego κυβερνήτης, en español “timonel” o “gobernante”

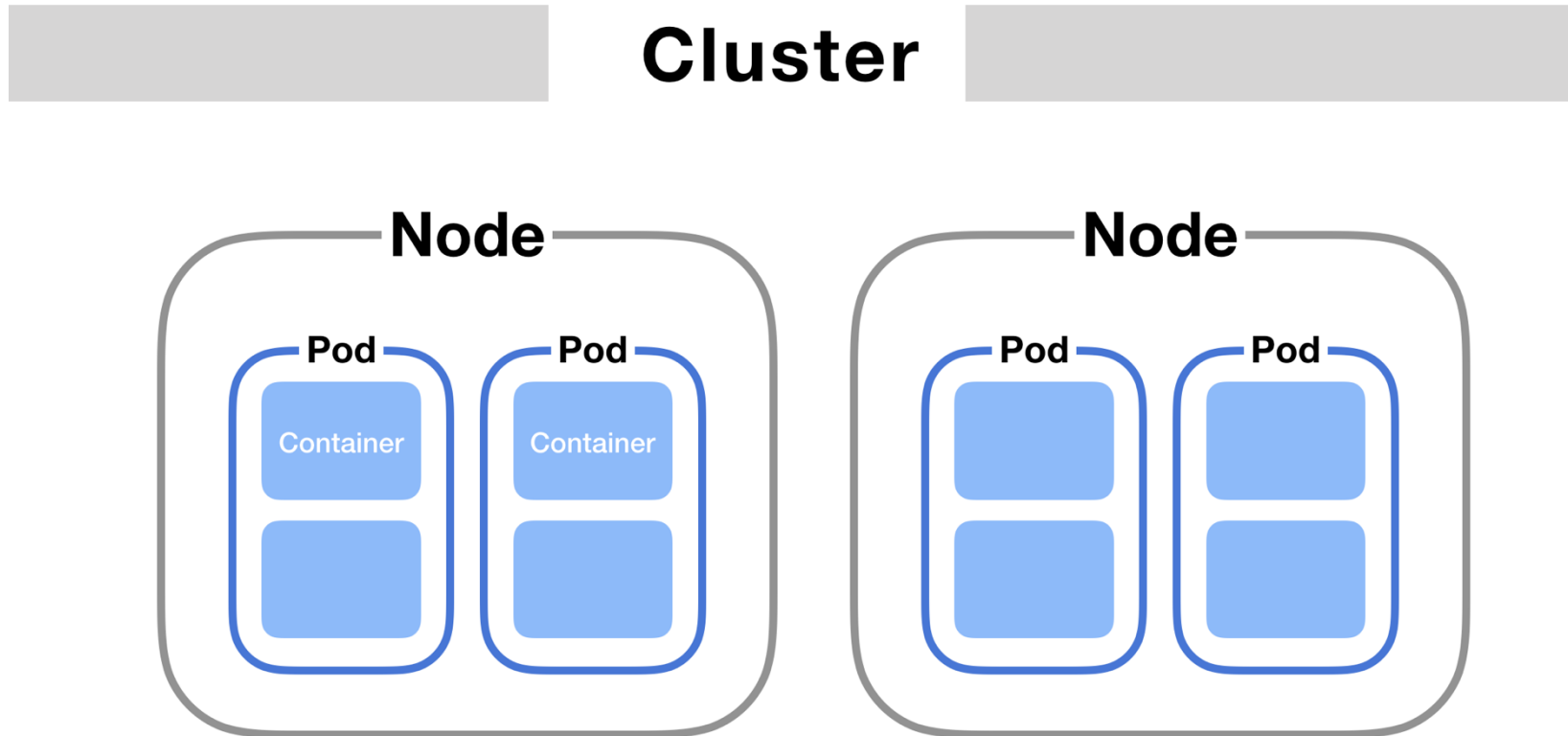
3. Contenedores

Contenedores en la nube: Kubernetes



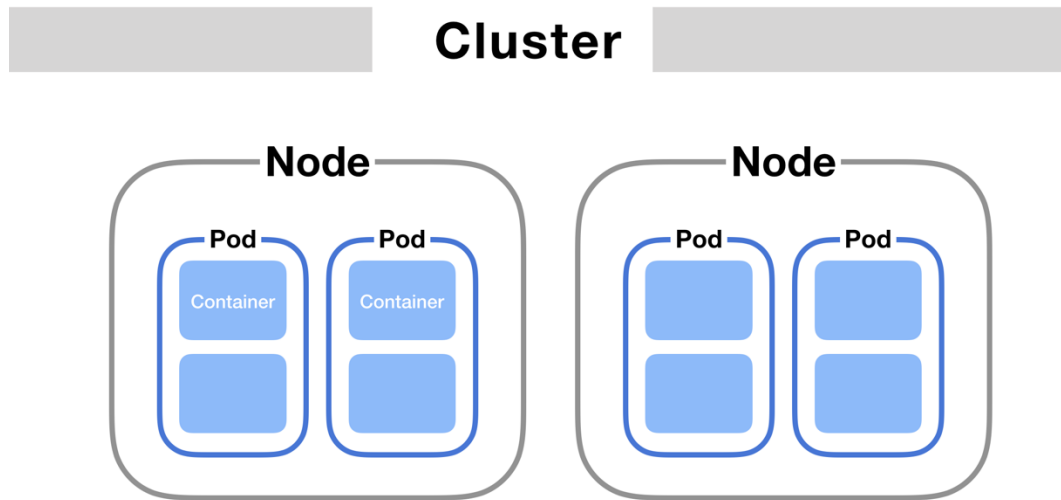
3. Contenedores

Contenedores en la nube: Kubernetes (jerarquía)



3. Contenedores

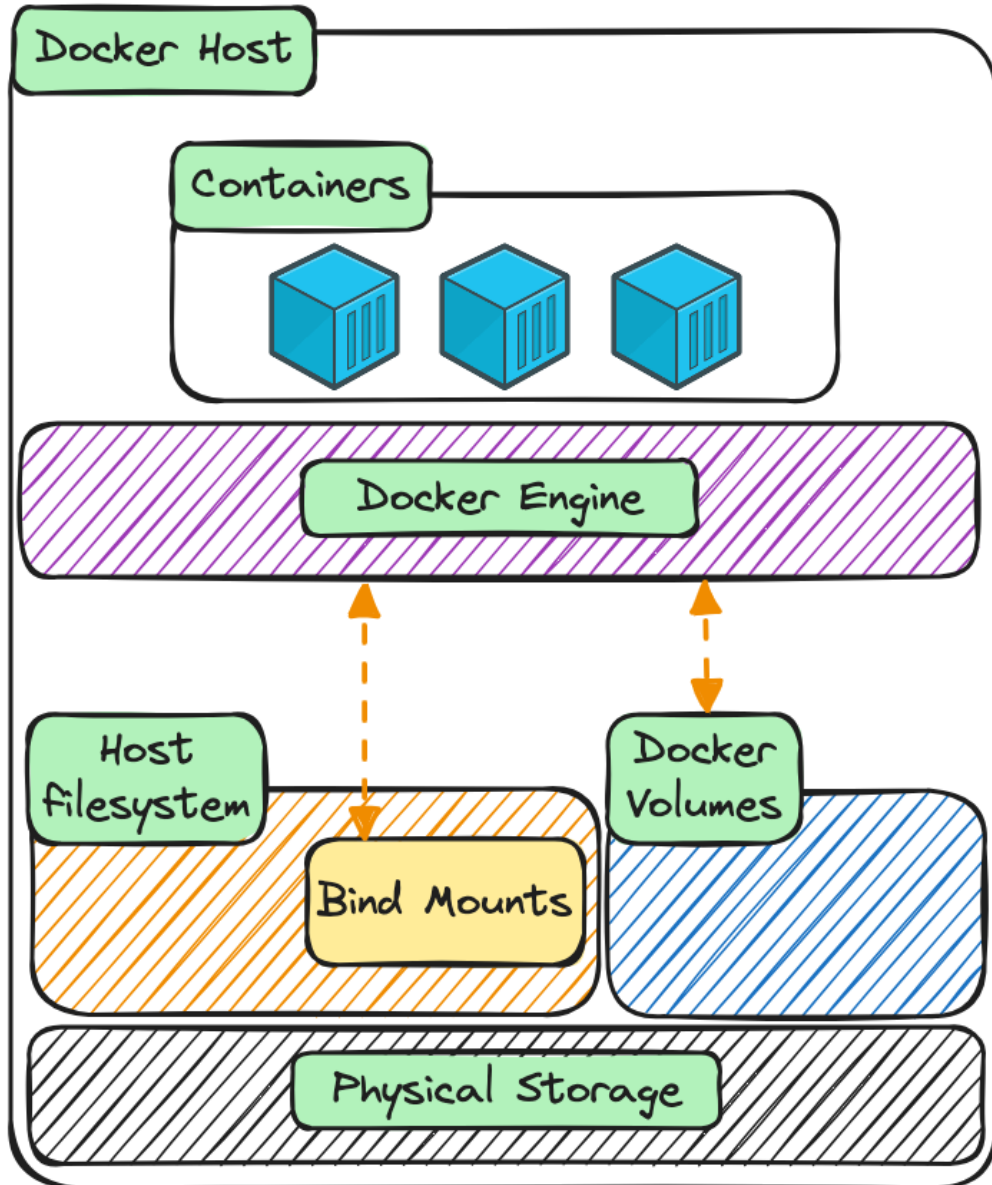
Contenedores en la nube: Kubernetes (jerarquía)



- **Contenedor:** ejecuta una aplicación o servicio concreto
- **Pod:** agrupa uno o varios contenedores que comparten red y almacenamiento
- **Nodo:** es una máquina (física o virtual) donde se ejecutan los pods
- **Cluster:** es el conjunto de nodos gestionados por Kubernetes que trabajan juntos para ejecutar todas las aplicaciones

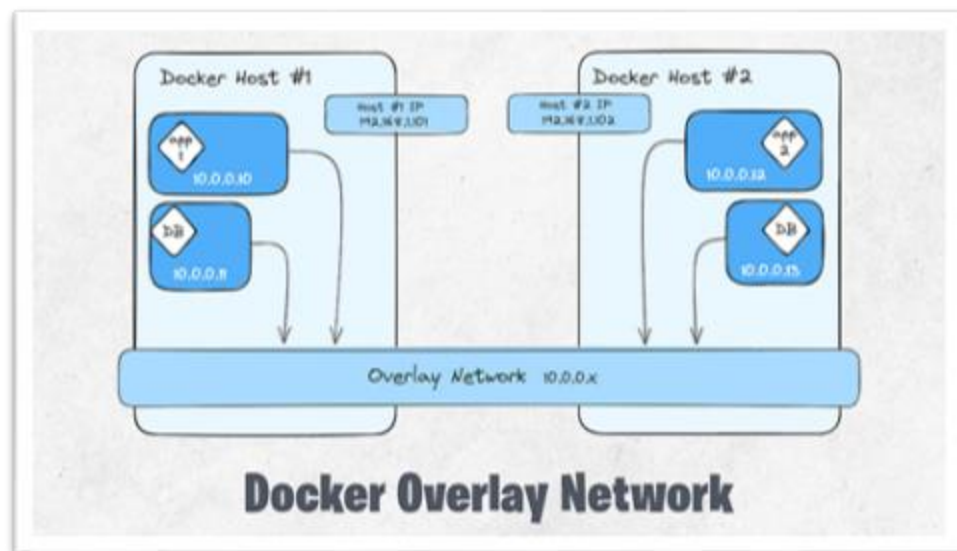
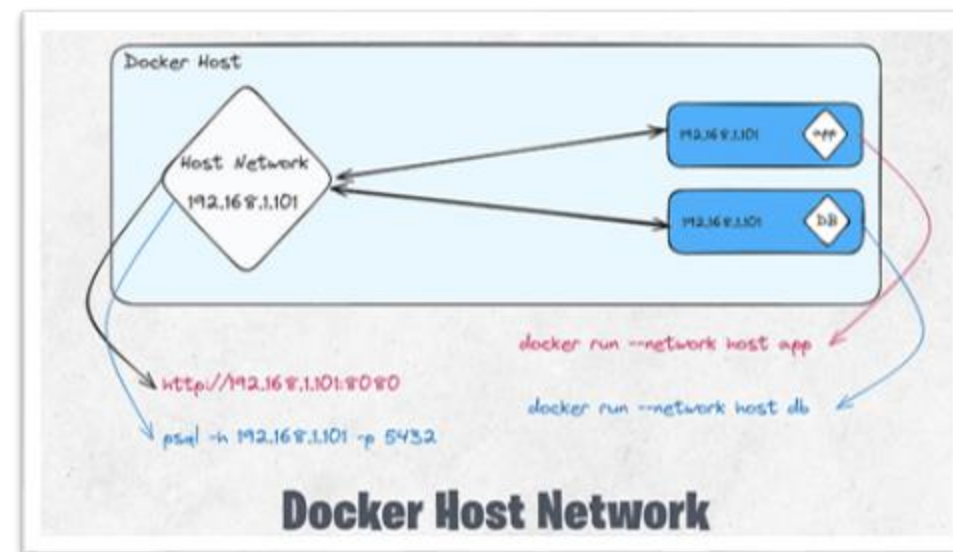
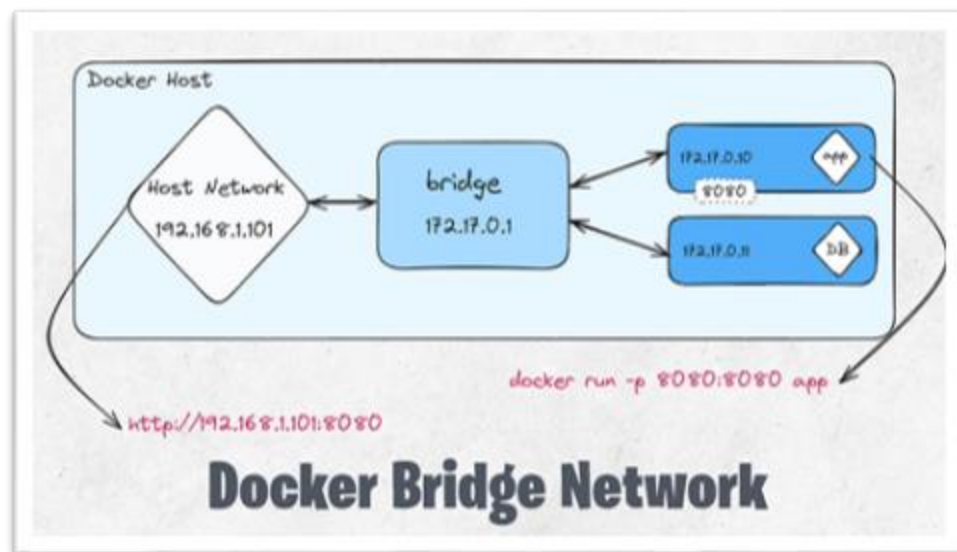
1. Introducción a Docker y DockerHub
 2. Imágenes
 3. Contenedores
 4. **Volúmenes y redes**
 5. Composición de servicios: Docker Compose
 6. Dev-containers
 7. Y yo, ¿qué puedo hacer en mi proyecto?
 8. Tutorial: *dockerizando uvlhub*
- 

4. Volúmenes y redes



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.

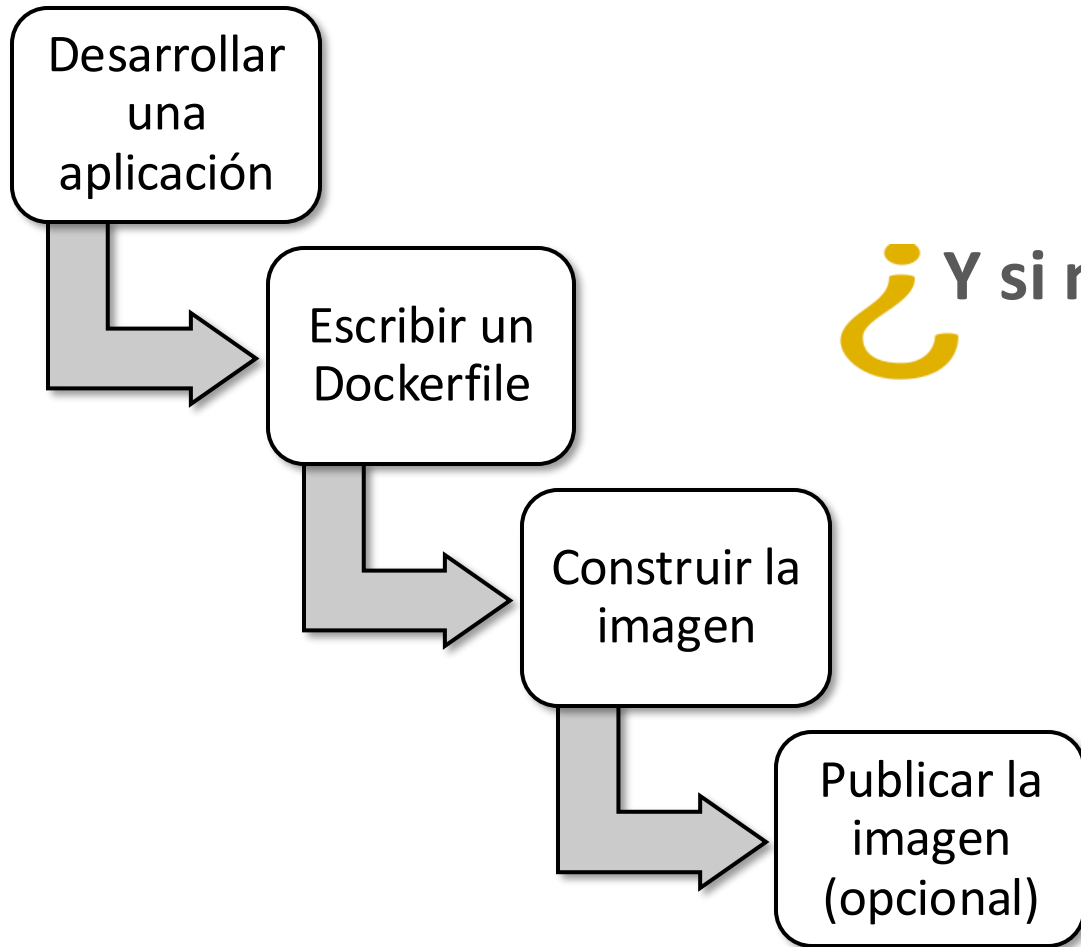
4. Volúmenes y redes



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. Dev-containers
 7. Y yo, ¿qué puedo hacer en mi proyecto?
 8. Tutorial: *dockerizando uvlhub*
- 

5. Composición de servicios: Docker compose

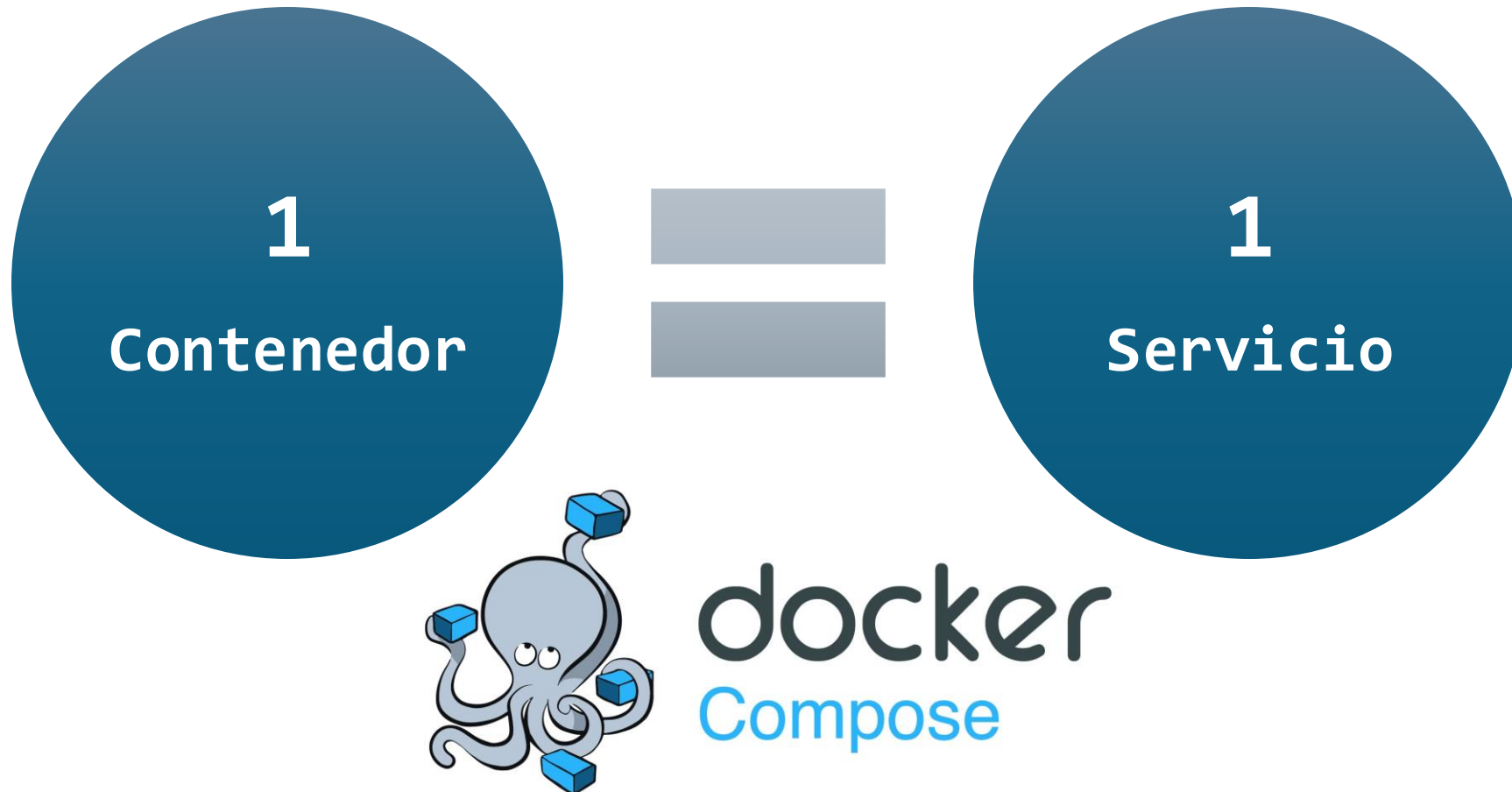


¿Y si necesito varias imágenes y varios contenedores?



5. Composición de servicios: Docker compose

Principio de responsabilidad única

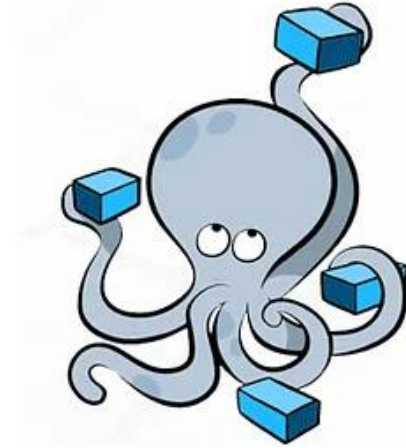


5. Composición de servicios: Docker compose

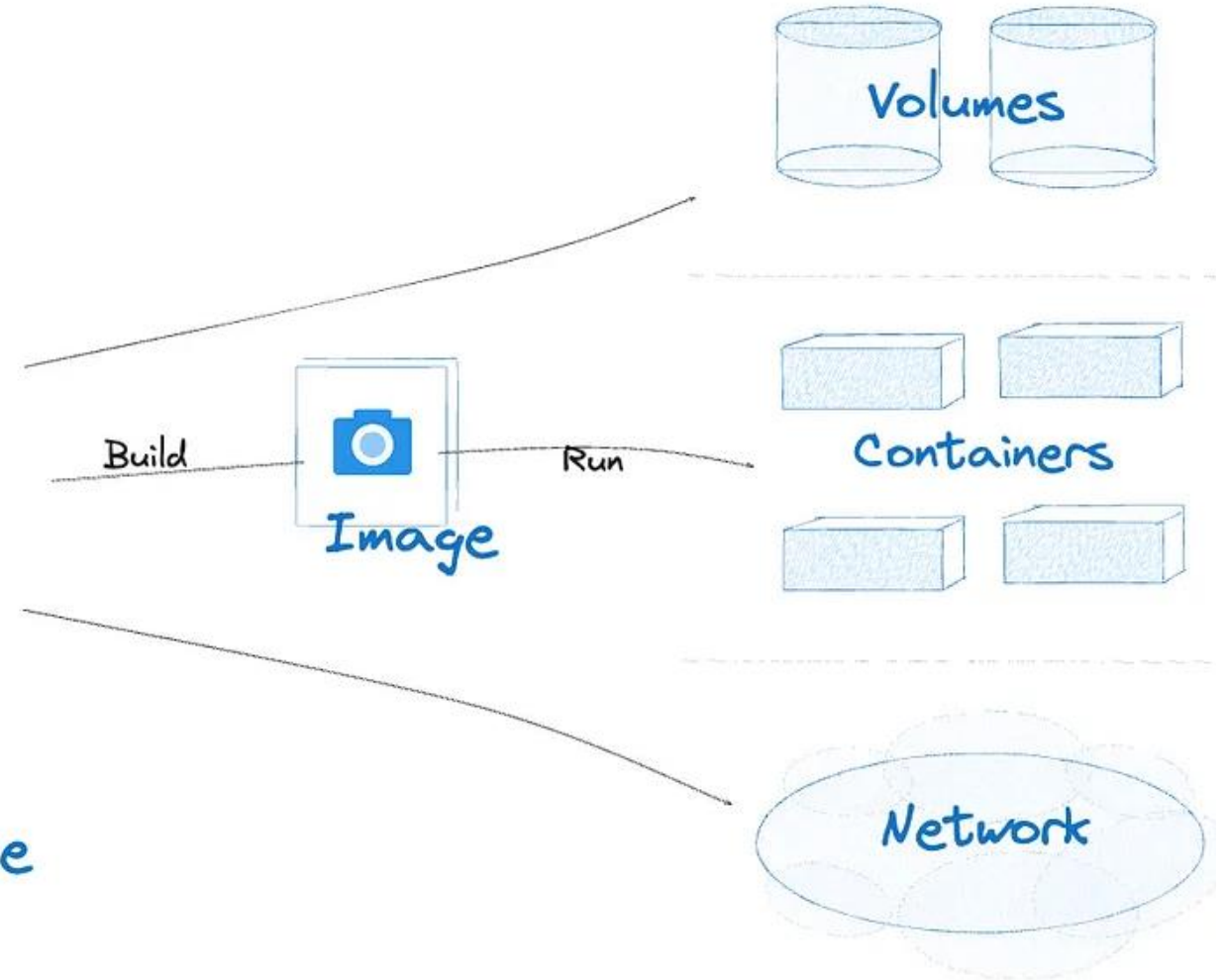
```
docker-compose.yml

services:
  app:
    image: node:18-alpine
    command: sh -c 'yarn install && yarn run dev'
    ports:
      - 127.0.0.1:3000:3000
    working_dir: /app
    volumes:
      - ./app
    environment:
      MYSQL_HOST: mysql
      MYSQL_USER: root
      MYSQL_PASSWORD: secret
      MYSQL_DB: todos
  mysql:
    image: mysql:8.0
    volumes:
      - todo-mysql-data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: secret
      MYSQL_DATABASE: todos
volumes:
  todo-mysql-data:
```

Config YAML



Docker Compose



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

6. Dev-containers

¿Pero mi app web está en Docker y yo sigo trabajando con el código en mi host?

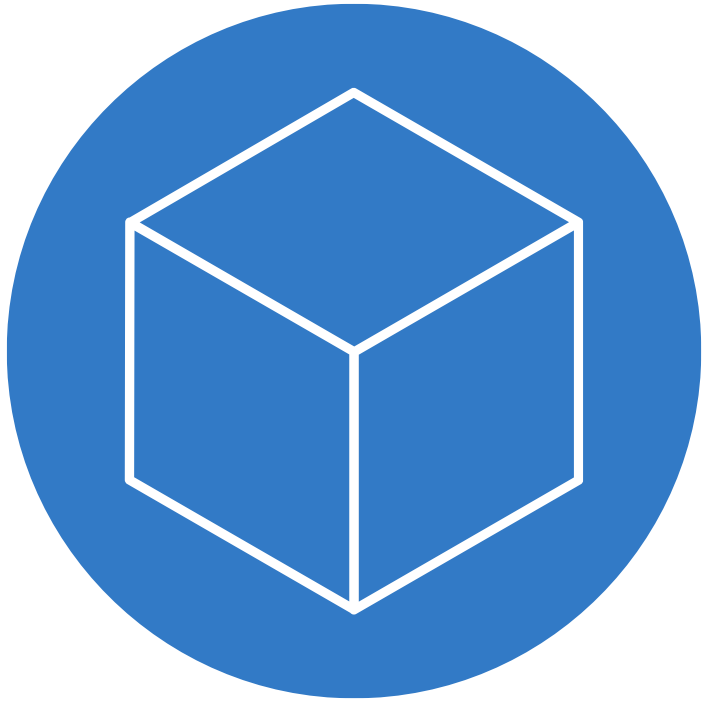
Mi VS Code no detecta las librerías al estar fuera del contenedor web...

Cada miembro de mi equipo tiene una configuración de VS Code distinta...



6. Dev-containers

Definición

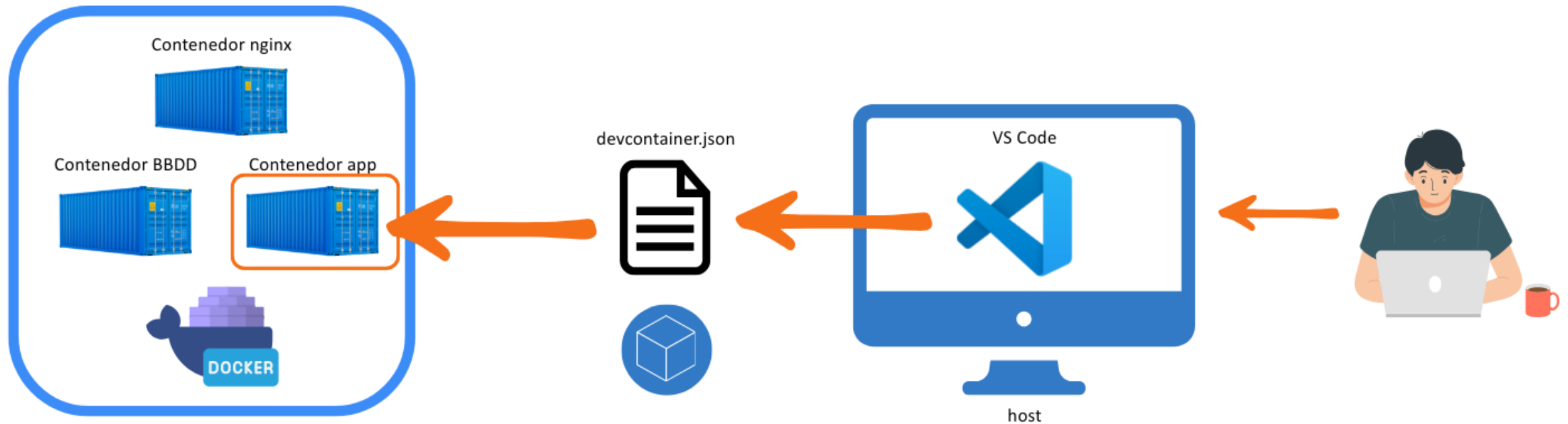


Un **Dev Container** es un **entorno de desarrollo preconfigurado** dentro de un contenedor Docker.

Garantiza que todos los desarrolladores trabajen con las mismas herramientas, dependencias y configuración.

6. Dev-containers

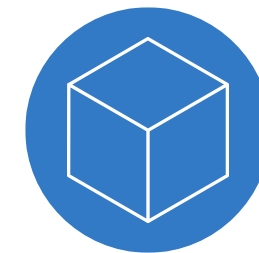
Diferencia entre container y dev-container



Un dev-container nos “sitúa” dentro de un contenedor para aprovechar el aislamiento y la potencia de VS Code

6. Dev-containers

Ejemplo



```
devcontainer.json X
devcontainer > {} devcontainer.json > ...
1 {
2   "name": "IPV4Hub Dev",
3   "dockerComposeFile": "../docker/docker-compose.dev.yml",
4   "service": "web",
5   "workspaceFolder": "/app",
6   "runServices": [
7     "web",
8     "db",
9     "nginx"
10  ],
11  "forwardPorts": [
12    5000,
13    3306,
14    80
15  ],
16  "shutdownAction": "stopCompose",
17  "customizations": {
18    "vscode": {
19      "extensions": [
20        "ms-azuretools.vscode-docker",
21        "ms-python.python",
22        "ms-python.debugpy",
23        "KevinRose.vsc-python-indent",
24        "ms-python.flake8",
25        "dbaeumer.vscode-eslint",
26        "twixes.pyli-assistant",
27        "cstrap.flask-snippets"
28      ],
29      "settings": {
30        "python.defaultInterpreterPath": "/usr/local/bin/python",
31        "python.formatting.provider": "black",
32        "python.linting.enabled": true,
33        "python.linting.flake8Enabled": true,
34        "python.linting.flake8Path": "flake8",
35        "editor.formatOnSave": true
36      }
37    }
38  },
39  "remoteEnv": {
40    "FLASK_ENV": "development",
41    "PYTHONUNBUFFERED": "1"
42  },
43  "mounts": [
44    {
45      "source": "${localWorkspaceFolder}/uploads",
46      "target": "/app/uploads",
47      "type": "bind"
48    }
49  ],
50  "remoteUser": "root"
51 }
```

Para cargar nuestro dev-container

Mac: ⌘ + Shift + P

Windows / Linux: Ctrl + Shift + P

Escribir en el cuadro de búsqueda:
Dev Containers: Reopen in Container
y elegimos abrir la raíz del proyecto

→ `.devcontainer/devcontainer.json`

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

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

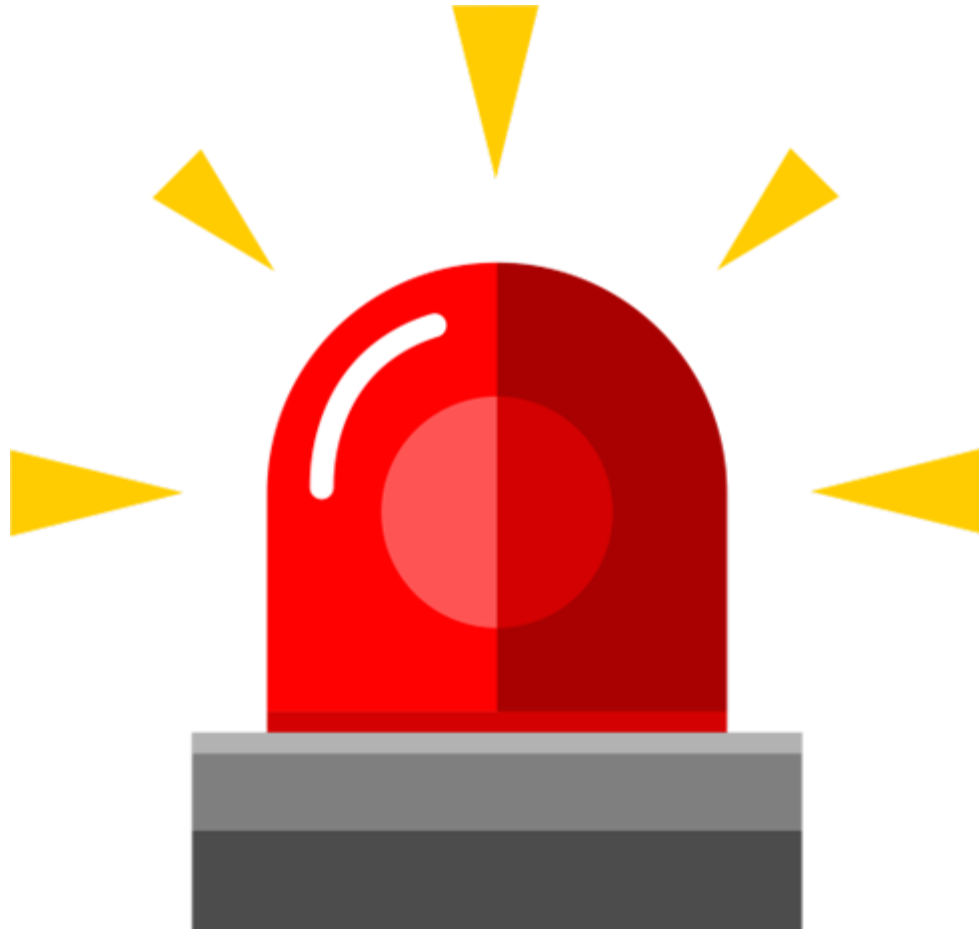


¡Diseña tu propio
despliegue en Docker!

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. Dev-containers
 7. Y yo, ¿qué puedo hacer en mi proyecto?
 8. Tutorial: ***dockerizando uvlhub***
- 



Antes de continuar, hay que actualizar nuestro proyecto con el fork de la asignatura

1º) **Añadir** repositorio original como upstream

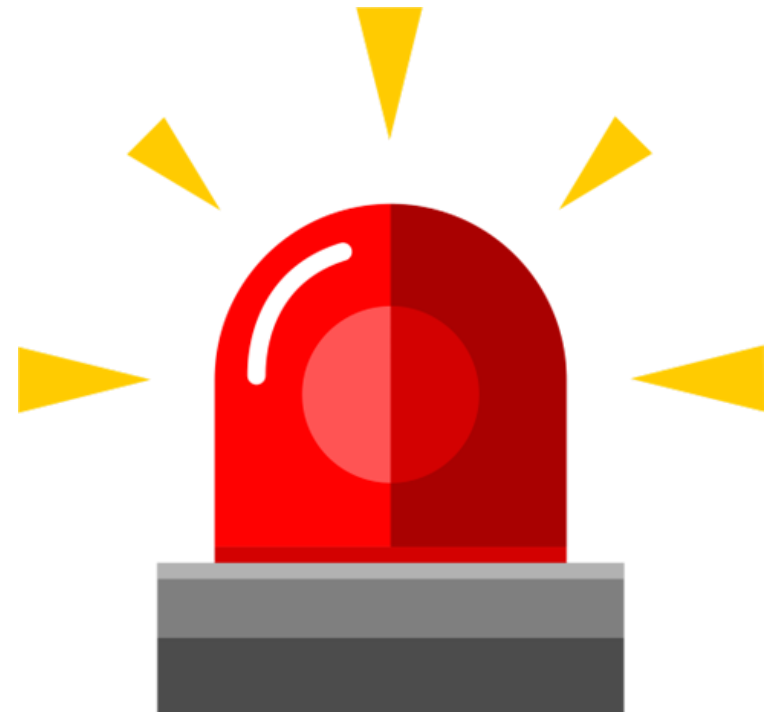
```
git remote add upstream https://github.com/EGCETSII/uvlhub
```

2º) **Fetch** del repositorio original

```
git fetch upstream
```

3º) **Fusionar** los cambios en tu fork

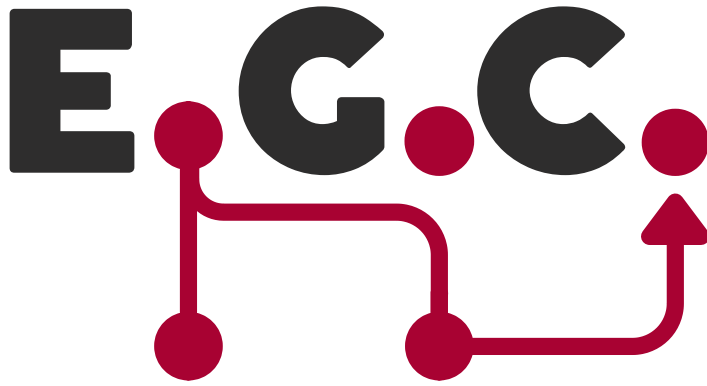
```
git checkout main  
git merge upstream/main
```



8. Tutorial: *dockerizando uvlhub*



<https://1984.lsi.us.es/wiki-egc/index.php/Tutorial-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)