02 Bases de Docker

¿Qué es Docker?

Docker es una plataforma de software que permite crear, implementar y ejecutar aplicaciones en contenedores. Los contenedores son entornos ligeros, portables y consistentes que incluyen todo lo necesario para ejecutar una aplicación (código, bibliotecas, dependencias).

¿Por qué aprender Docker?

1. **Portabilidad**: Funciona en cualquier sistema operativo compatible.
2. **Escalabilidad**: Facilita el despliegue de aplicaciones en diferentes entornos.
3. **Eficiencia**: Reduce el uso de recursos frente a las máquinas virtuales.

¿Para qué sirve Docker?

- Implementar aplicaciones rápidamente.
- Probar entornos aislados.
- Facilitar el desarrollo colaborativo y continuo.

¿Cuál es la diferencia entre una imagen y un contenedor?

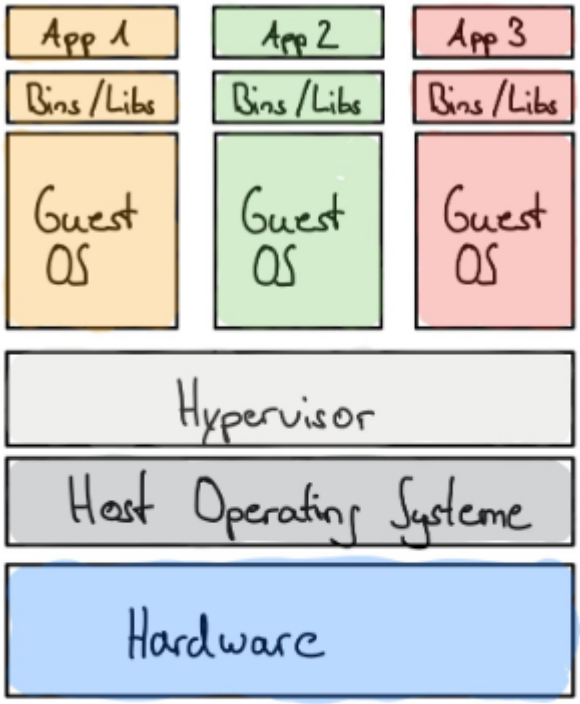
- **Imagen**: Es una plantilla inmutable que contiene todo lo necesario para ejecutar una aplicación (código, dependencias, configuraciones). Sirve como base para crear contenedores.
- **Contenedor**: Es una instancia ejecutable de una imagen. Es un entorno aislado donde corre la aplicación definida en la imagen.

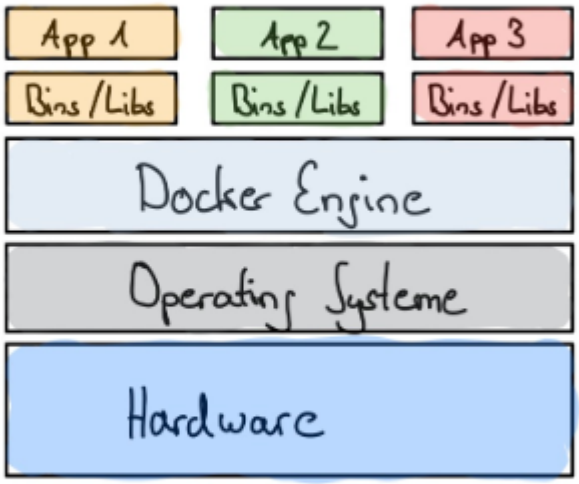
Diferencia clave:
La **imagen** es el plano o diseño, mientras que el **contenedor** es la construcción activa basada en ese diseño.

Máquinas virtuales VS Contenedores

Las máquinas virtuales (VM) emulan un hardware completo y ejecutan un sistema operativo huésped sobre un hipervisor. Cada VM tiene su propio kernel, lo que implica un mayor consumo de recursos y tiempos de arranque más largos. En contraste, los contenedores comparten el kernel del sistema operativo anfitrión y aíslan las aplicaciones a nivel de sistema operativo. Esto los hace más ligeros, rápidos de iniciar y eficientes en el uso de recursos.

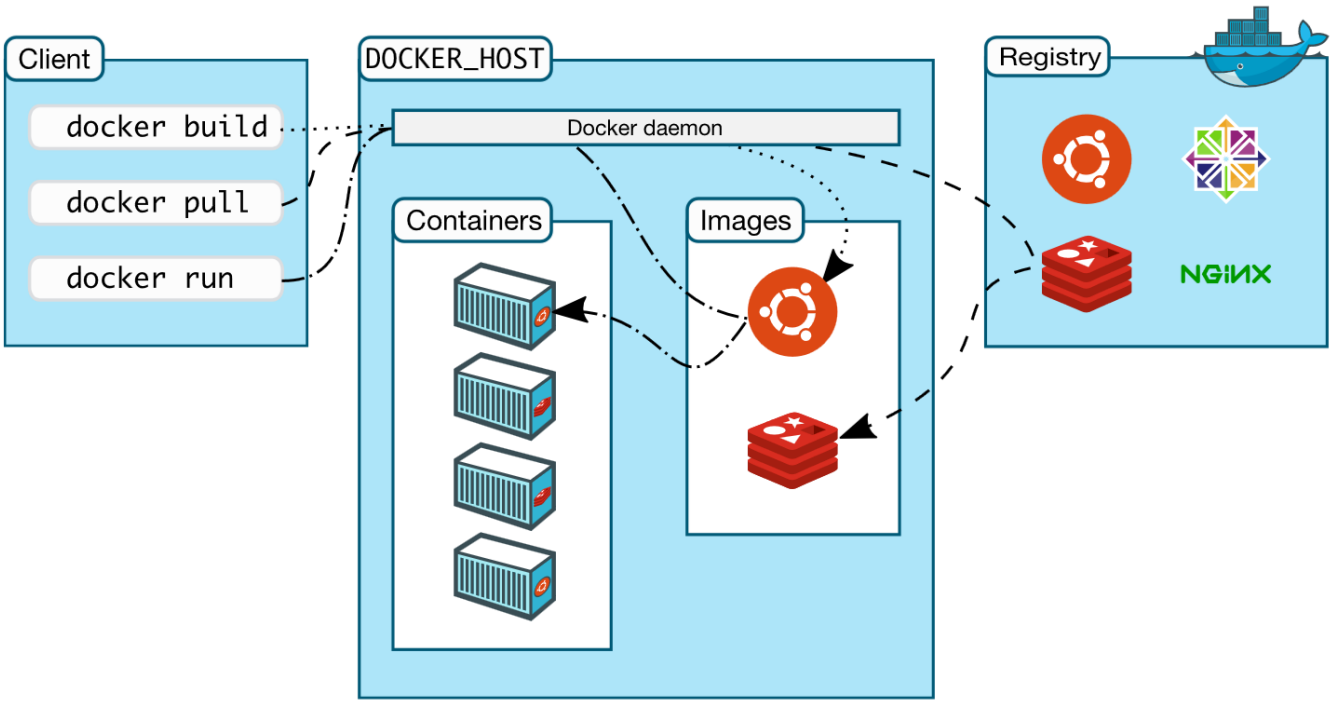
Docker es importante porque optimiza este enfoque, permitiendo ejecutar aplicaciones en entornos más livianos que las VMs. Además, ofrece portabilidad, ya que los contenedores funcionan de manera uniforme en cualquier sistema. También facilita la escalabilidad al gestionar múltiples instancias de servicios y garantiza reproducibilidad mediante el uso de imágenes predefinidas (Dockerfiles), asegurando consistencia en diferentes entornos.





Virtual Machines

Containers



Comandos básicos

1. Descargar una imagen.

A continuación mostramos el proceso de uso de un contenedor `hello-world` descargado de dockerhub.

En primer lugar vamos a descargar la imagen correspondiente de docker llamada `hello-world`. Como datos a destacar tenemos:

- `Using default tag: latest`: Hace referencia a la última versión que se desplegó en el repositorio para esta imagen.
- `Digest: sha256:5b3cc85e16e3058003c13b7821318369dad01dac3dbb877aac3c28182255c724`: Hace referencia al Hash identificador de la imagen descargada.
- `Status: Downloaded newer image for hello-world:latest`: Estado de la imagen descargada.

```
PS C:\Windows\system32> docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
c1ec31eb5944: Download complete
Digest: sha256:5b3cc85e16e3058003c13b7821318369dad01dac3dbb877aac3c28182255c724
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

2. Desplegar un contenedor

Ahora vamos a lanzar el contenedor que haga uso de esta imagen, es decir, un contenedor es una instancia de una imagen ejecutandose en un entorno aislado. Para ello realizamos el siguiente pasos:

```
PS C:\Windows\system32> docker container run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

3. Listar contenedores del sistema

Podemos listar los contenedores que tenemos pero por defecto unicamente se mostrarán los contenedores activos. Por compatibilidad, si bien se trata de mantener comandos como el `ps`, es corriente encontrar una sintaxis muy similar a la utilizada en sistemas UNIX.

```
PS C:\Windows\system32> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Windows\system32> docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Windows\system32> docker container ls -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS
PORTS     NAMES
718d7b4c72a0   hello-world   "/hello"   17 minutes ago   Exited (0) 14 minutes ago
eager_roentgen
```

4. Eliminar contenedores

Podemos eliminar el contenedor tanto por su nombre como por su identificador. El nombre de un contenedor se establece de forma aleatoria (si bien podemos establecerlo nosotros) siendo la unión aleatoria de nombres de personalidades famosas en la informática.

```
PS C:\Windows\system32> docker rm 718d7b4c72a0
718d7b4c72a0
PS C:\Windows\system32> docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Nota: No es necesario eliminar de uno en uno ya que podríamos eliminar varios haciendo uso del comando siguiente (suponinedo que existan dos que empiecen por los siguientes dígitos).

```
PS C:\Windows\system32> docker container ls -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS
PORTS     NAMES
7d445848888d   hello-world   "/hello"   20 seconds ago   Exited (0) 19 seconds ago
flamboyant_banach
f0ee6fbfca84   hello-world   "/hello"   22 seconds ago   Exited (0) 21 seconds ago
gifted_faraday
PS C:\Windows\system32> docker container rm 7d4 f0e
7d4
f0e
PS C:\Windows\system32> docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Nota 2: A mayores, mencionar que podemos eliminarlos todos con el comando `prune`.

```
PS C:\Windows\system32> docker container prune
WARNING! This will remove all stopped containers.
```

```
Are you sure you want to continue? [y/N] y
Deleted Containers:
7c0aebd6f728d44a2788f260d7d0de837d1159ed98db5b7467fa5470a735ebc4
d95b7355de1c7ff6189442ca9e397853d90d6353f57e0878ef6e3036d10d4e38

Total reclaimed space: 8.192kB
PS C:\Windows\system32> docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

5. Eliminar imagenes

Podemos listar y eliminar imágenes de los contenedores como se ve a continuación.

```
PS C:\Windows\system32> docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
hello-world   latest    5b3cc85e16e3   19 months ago   24.4kB
PS C:\Windows\system32> docker image rm hello-world
Untagged: hello-world:latest
Deleted: sha256:5b3cc85e16e3058003c13b7821318369dad01dac3dbb877aac3c28182255c724
PS C:\Windows\system32> docker image ls
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

6. Ejecutar un contenedor de forma desenlazada

Vamos a ejecutar un contenedor de forma desenlazada o detach lo que va a permitir que se lance y vamos a poder seguir escribiendo comandos en nuestra terminal.

Como podemos ver partimos sin imagenes ni contenedores.

```
PS C:\Windows\system32> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
PS C:\Windows\system32> docker image ls
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
PS C:\Windows\system32> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Windows\system32> docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Windows\system32> docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Vamos a crear un contenedor a partir de la imagen getting-started que se nos ofrece para realizar esta práctica. Entonces lo que hacemos es descargar imagen, crear contenedor y lanzarlo de forma desenlazada.

```
PS C:\Windows\system32> docker container run -d docker/getting-started
Unable to find image 'docker/getting-started:latest' locally
latest: Pulling from docker/getting-started
```

```
4f7e34c2de10: Download complete
b6334b6ace34: Download complete
9b6f639ec6ea: Download complete
ee68d3549ec8: Download complete
c158987b0551: Download complete
f1d1c9928c82: Download complete
33e0cbbb4673: Download complete
1e35f6679fab: Download complete
cb9626c74200: Download complete
Digest: sha256:d79336f4812b6547a53e735480dde67f8f8f7071b414fbd9297609ffb989abc1
Status: Downloaded newer image for docker/getting-started:latest
c986be1faface65eacb6cacc2566ef4b3222e8c1dc661d1456f525094c9ab5c5
```

```
PS C:\Windows\system32> docker image ls
REPOSITORY               TAG       IMAGE ID       CREATED         SIZE
docker/getting-started   latest    d79336f4812b   24 months ago   73.9MB
PS C:\Windows\system32> docker container ls
CONTAINER ID   IMAGE                    COMMAND                  CREATED
STATUS         PORTS     NAMES
c986be1fafac   docker/getting-started   "/docker-entrypoint.…"   2 minutes ago
Up 2 minutes   80/tcp    determined_solomon
```

7. Lanzar y detener un contenedor

Una vez lanzado el contenedor, podemos detenerlo de la siguiente forma.

```
PS C:\Windows\system32> docker container ls
CONTAINER ID   IMAGE                    COMMAND                  CREATED
STATUS         PORTS     NAMES
c986be1fafac   docker/getting-started   "/docker-entrypoint.…"   5 minutes ago
Up 5 minutes   80/tcp    determined_solomon
PS C:\Windows\system32> docker container stop c98
c98
PS C:\Windows\system32> docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Windows\system32> docker container ls -a
CONTAINER ID   IMAGE                    COMMAND                  CREATED
STATUS                      PORTS     NAMES
c986be1fafac   docker/getting-started   "/docker-entrypoint.…"   5 minutes ago
Exited (0) 6 seconds ago             determined_solomon
PS C:\Windows\system32> docker container start c98
c98
PS C:\Windows\system32> docker container ls
CONTAINER ID   IMAGE                    COMMAND                  CREATED
STATUS         PORTS     NAMES
c986be1fafac   docker/getting-started   "/docker-entrypoint.…"   5 minutes ago
Up 5 minutes   80/tcp    determined_solomon
```

8. Lanzar un contenedor y publicarlo en un puerto

Para ello ahora vamos a arrancar de nuevo el contenedor y vamos a publicarlo `-d -p 8080:80` con respecto al puerto 80 del equipo con el 80 del contenedor. Si accedemos a localhost:8080 tendremos el contenedor en funcionamiento en la URL localhost:8080.

```
PS C:\Windows\system32> docker container run -d -p 8080:80 docker/getting-started
52c508ce5de450ae113e29978cbc3ca0e8f10209ed390fd7a9e148bbd3f0bfec
PS C:\Windows\system32> docker container ls
CONTAINER ID   IMAGE                  COMMAND                  CREATED
STATUS         PORTS                NAMES
52c508ce5de4   docker/getting-started   "/docker-entrypoint.…"   8 seconds ago
Up 7 seconds   0.0.0.0:8080->80/tcp   mystifying_clarke
```

Una vez llegados a este punto vamos a eliminar el contenedor y la imagen. Ya que el contenedor está corriendo, en lugar de pararlo vamos a forzar la eliminación.

```
PS C:\Windows\system32> docker container ls -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED
STATUS         PORTS                NAMES
5ae32f145a90   docker/getting-started   "/docker-entrypoint.…"   17 minutes ago
Up 2 minutes   0.0.0.0:8080->80/tcp   inspiring_black
PS C:\Windows\system32> docker container rm -f 5ae
5ae
PS C:\Windows\system32> docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS C:\Windows\system32> docker images
REPOSITORY               TAG       IMAGE ID       CREATED         SIZE
docker/getting-started   latest    d79336f4812b   24 months ago   73.9MB
PS C:\Windows\system32> docker image rm docker/getting-started
Untagged: docker/getting-started:latest
Deleted: sha256:d79336f4812b6547a53e735480dde67f8f8f7071b414fbd9297609ffb989abc1
PS C:\Windows\system32> docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

9. Variables de entorno

Vamos a lanzar un contenedor a partir de una imagen de una base de datos postgres. Para ello vamos a utilizar variables de entorno para nuestro contenedor lanzando este de una forma concreta. En primer lugar en PostgreSQL en Docker Hub tenemos una imagen oficial de una base de datos postgres que vamos a emplear.

Descargamos la imagen en cuestión y como no indicamos una versión en concreto descarga la imagen con etiqueta latest.

```
PS C:\Windows\system32> docker pull postgres
Using default tag: latest
```

```
latest: Pulling from library/postgres
2cd360f3b7db: Download complete
a24f300391ed: Download complete
bc0965b23a04: Download complete
2cb801c39436: Download complete
627f580b7ad7: Download complete
c9cdd1fe82e4: Download complete
8f152c4aceed: Download complete
002e1a8eb6f9: Download complete
cfb3c2203f88: Download complete
c5fdb20d8658: Download complete
9e592465b243: Download complete
67c5fe618f0c: Download complete
8d4265d09d9c: Download complete
e3a8293e92fd: Download complete
Digest: sha256:fe4efc6901dda0d952306fd962643d8022d7bb773ffe13fe8a21551b9276e50c
Status: Downloaded newer image for postgres:latest
docker.io/library/postgres:latest
```

Tal y como se dice en la documentación vamos a ejecutar el siguiente comando que establece variables de entorno para una base de datos postgres.

- --name some-postgres: El nombre del contenedor.
- -e POSTGRES_USER=martin: Variable de entorno que hace referencia al usuario de conexión a la base de datos.
- -e POSTGRES_PASSWORD=abc123.: Variable de entorno que hace referencia a la contraseña.
- -e POSTGRES_DB=miBD: Nombre de la base de datos.
- -dp 5432:5432 postgres: Lanza de forma desatendida el contenedor en base a la imagen postgres anteriormente descargada estando el puerto 5432 del equipo en escucha con el puerto 5432 habilitado por el contenedor.

```
PS C:\Users\Carballeira\Documents\Docker> docker container run `
>> --name miBD `
>> -e POSTGRES_USER=martin `
>> -e POSTGRES_PASSWORD=abc123. `
>> -e POSTGRES_DB=miBD `
>> -dp 5432:5432 `
>> postgres
7a885490b13c6e3f700bbfa5aacb7ec50c76d1d40c0caecc9045e52b1da3f999

PS C:\Windows\system32> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS
PORTS                    NAMES
7a885490b13c   postgres   "docker-entrypoint.s…"   15 seconds ago   Up 14 seconds
0.0.0.0:5432->5432/tcp   miBD
```

Llegados a este punto podría conectarme a mi contenedor que tiene una instancia postgres y crear bases de datos pero en el momento que eliminemos el contenedor se perderá dicha información (posteriormete introduciremos el término volumen).

10. Varias instancias postgres

Sería interesante tener diferentes instancias de postgres ya que pudiera darse el caso de que una aplicación estuviera migrando de una versión postgres a otra diferente. Para ello vamos a crear dos contenedores diferentes.

```
PS C:\Users\Carballeira\Documents\Docker> docker container run `
>> --name postgres-alpha `
>> -e POSTGRES_PASSWORD=mypass1 `
>> -dp 5432:5432 `
>> postgres
1fa43402bc8d46f1fca86ac1454cf5fc537c4b7cf4c7229271401c9353a150cc

PS C:\Users\Carballeira\Documents\Docker> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS
PORTS                    NAMES
1fa43402bc8d   postgres   "docker-entrypoint.s…"   44 seconds ago   Up 43 seconds
0.0.0.0:5432->5432/tcp   postgres-alpha
```

A continuación, vamos a crear otro contenedor pero en base a una imagen diferente de postgres a la que podremos conectarnos.

```
PS C:\Users\Carballeira\Documents\Docker> docker container run `
>> --name postgres-alpine `
>> -e POSTGRES_PASSWORD=mypass1 `
>> -dp 5433:5432 `
>> postgres:14-alpine3.21
fb69ad382009dc589a753f763a20ee0834b97e395183de8439d0a9c95ad5efdc
PS C:\Users\Carballeira\Documents\Docker> docker ps
CONTAINER ID   IMAGE                    COMMAND                  CREATED
STATUS         PORTS                    NAMES
fb69ad382009   postgres:14-alpine3.21   "docker-entrypoint.s…"   8 seconds ago
Up 7 seconds   0.0.0.0:5433->5432/tcp   postgres-alpine
1fa43402bc8d   postgres                 "docker-entrypoint.s…"   8 minutes ago
Up 8 minutes   0.0.0.0:5432->5432/tcp   postgres-alpha
```

Propuesta de actividad

Realiza los siguientes pasos:

1. Montar la base de datos Postgres con usuario="Tu nombre", contraseña="FP123." bd="docker".
2. Conectarse a la base de datos con la extensión de VScode PostgreSQL.
3. Hacer un informe con los pasos explicados en markdown y una imagen de la conexión.