

Desarrollo avanzando de código

Docker y DockerHub

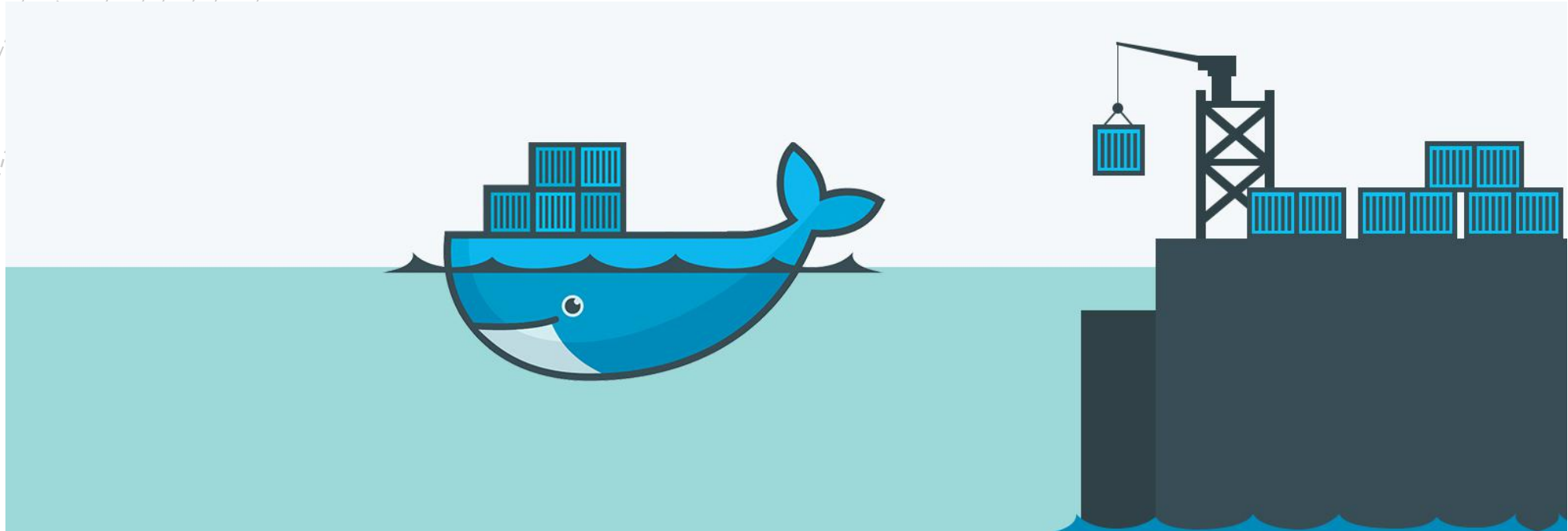
Manejo de contenedores

José Luis González Sánchez
<https://github.com/joseluisgs>



¿Qué vamos a aprender?

- Aprenderemos a manejar imagenes y contenedores y cómo aplicarlos para mejorar en el desarrollo de software



¿Qué es Docker?

- Docker es una plataforma para desarrollar, lanzar y ejecutar aplicaciones. Permite separar las aplicaciones desarrolladas de la infraestructura donde se desarrollan y trabajar así más cómoda y rápidamente.
- Docker permite empaquetar y lanzar una aplicación en un entorno totalmente aislado llamado **contenedor**. Estos contenedores se ejecutan directamente sobre el kernel de la máquina por lo que son mucho más ligeros que las máquinas virtuales.
- Con Docker podemos ejecutar mucho más contenedores para el mismo equipo que si éstos fueran maquina virtuales. De esta manera, podemos probar rápidamente uestra aplicación web, por ejemplo, en múltiples entornos distintos al de donde nos encontramos desarrollando. Realmente es mucho más rápido que hacerlo en una máquina virtual, puesto que reduce el tiempo de carga y el espacio requerido por cada uno de estos contenedores o máquinas.



Contenedores

- Los contenedores se distinguen de las máquinas virtuales en que las máquinas virtuales emulan un ordenador físico en el que se instala un sistema operativo completo, mientras que los contenedores usan el kernel del sistema operativo anfitrión pero contienen las capas superiores (sistema de ficheros, utilidades, aplicaciones).
- Al ahorrarse la emulación del ordenador y el sistema operativo de la máquina virtual, los contenedores son más pequeños y rápidos que las máquinas virtuales. Pero al incluir el resto de capas de software, se consigue el aislamiento e independencia entre contenedores que se busca con las máquinas virtuales.



Contenedores

- Los contenedores se distinguen de las máquinas virtuales en que las máquinas virtuales emulan un ordenador físico en el que se instala un sistema operativo completo, mientras que los contenedores usan el kernel del sistema operativo anfitrión pero contienen las capas superiores (sistema de ficheros, utilidades, aplicaciones).
- Al ahorrarse la emulación del ordenador y el sistema operativo de la máquina virtual, los contenedores son más pequeños y rápidos que las máquinas virtuales. Pero al incluir el resto de capas de software, se consigue el aislamiento e independencia entre contenedores que se busca con las máquinas virtuales.



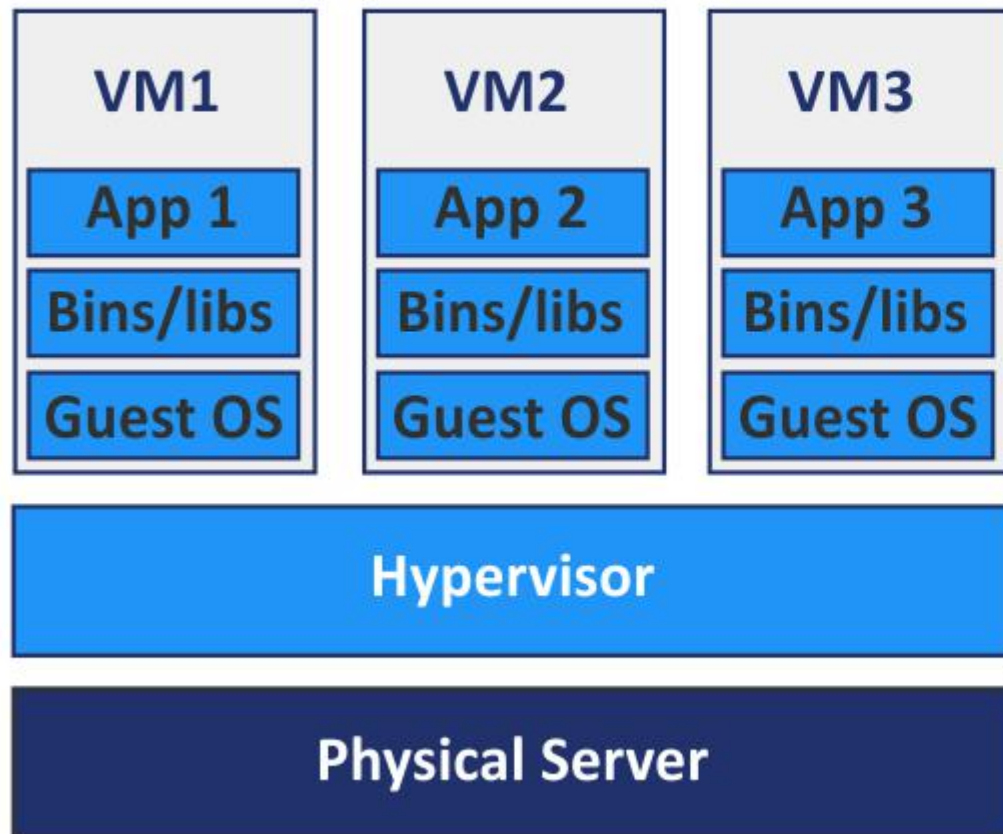
Contenedores

- Docker no es virtualizado, no hay un hipervisor. Los procesos que corren dentro de un contenedor de docker se ejecutan con el mismo kernel que la máquina anfitrión. Linux lo que hace es aislar esos procesos del resto de procesos del sistema, ya sean los propios de la máquina anfitrión o procesos de otros contenedores.
- Además, es capaz de controlar los recursos que se le asignan a esos contenedores (cpu, memoria, etc).
- Internamente, el contenedor no sabe que lo es y a todos los efectos es una distribución GNU/Linux independiente, pero sin la penalización de rendimiento que tienen los sistemas virtualizados.
- Así que, cuando ejecutamos un contenedor, estamos ejecutando un servicio dentro de una distribución construida a partir de una "receta". Esa receta permite que el sistema que se ejecuta sea siempre el mismo, independientemente de si estamos usando Docker en Ubuntu, Fedora o, incluso, sistemas privativos compatibles con Docker. De esa manera podemos garantizar que estamos desarrollando o desplegando nuestra aplicación, siempre con la misma versión de todas las dependencias.

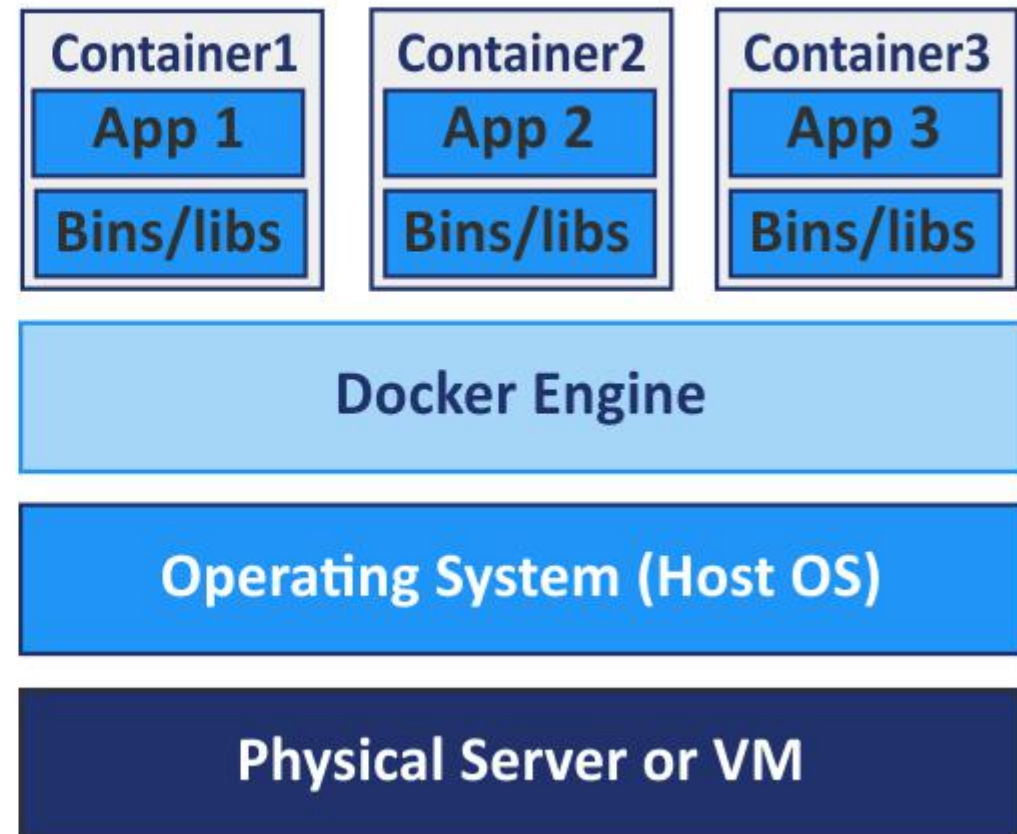


Contenedores

Virtual Machines

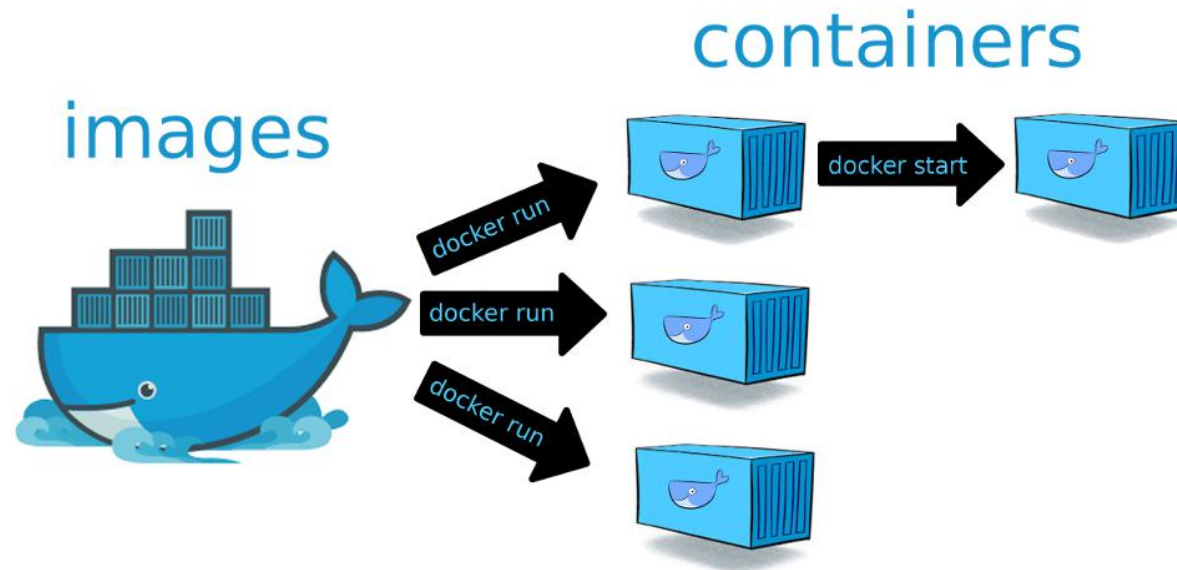


Containers



Imágenes

- Una Imagen es una plantilla de solo lectura que contiene las instrucciones para crear un contenedor Docker. Pueden estar basadas en otras imágenes, lo cual es habitual. Para ello usaremos distintos ficheros como el dockfile.
- Por ejemplo una imagen podría contener un sistema operativo Ubuntu con un servidor Apache y tu aplicación web instalada.



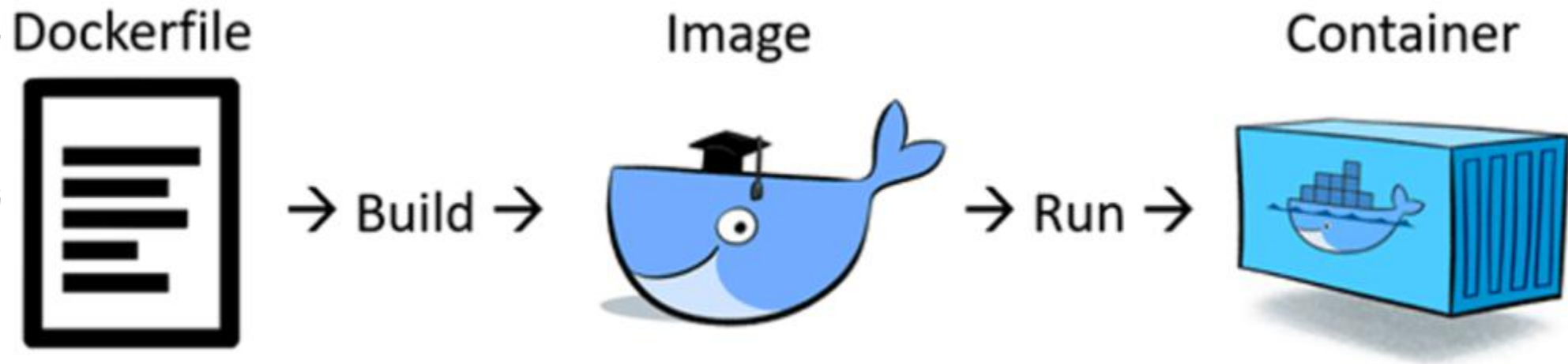
Imágenes y contenedores

- Por lo tanto, un contenedor es s una instancia ejecutable de una imagen.
- Esta instancia puede ser creada, iniciada, detenida, movida o eliminada a través del cliente de Docker o de la API.
- Las instancias se pueden conectar a una o más redes, sistemas de almacenamiento, o incluso se puede crear una imagen a partir del estado de un contenedor.
- Se puede controlar cómo de aislado está el contenedor del sistema anfitrión y del resto de contenedores.
- El contenedor está definido tanto por la imagen de la que procede como de las opciones de configuración que permita.
- Por ejemplo, la imagen oficial de MariaDb permite configurar a través de opciones la contraseña del administrador, de la primera base de datos que se cree, del usuario que la maneja, etc.

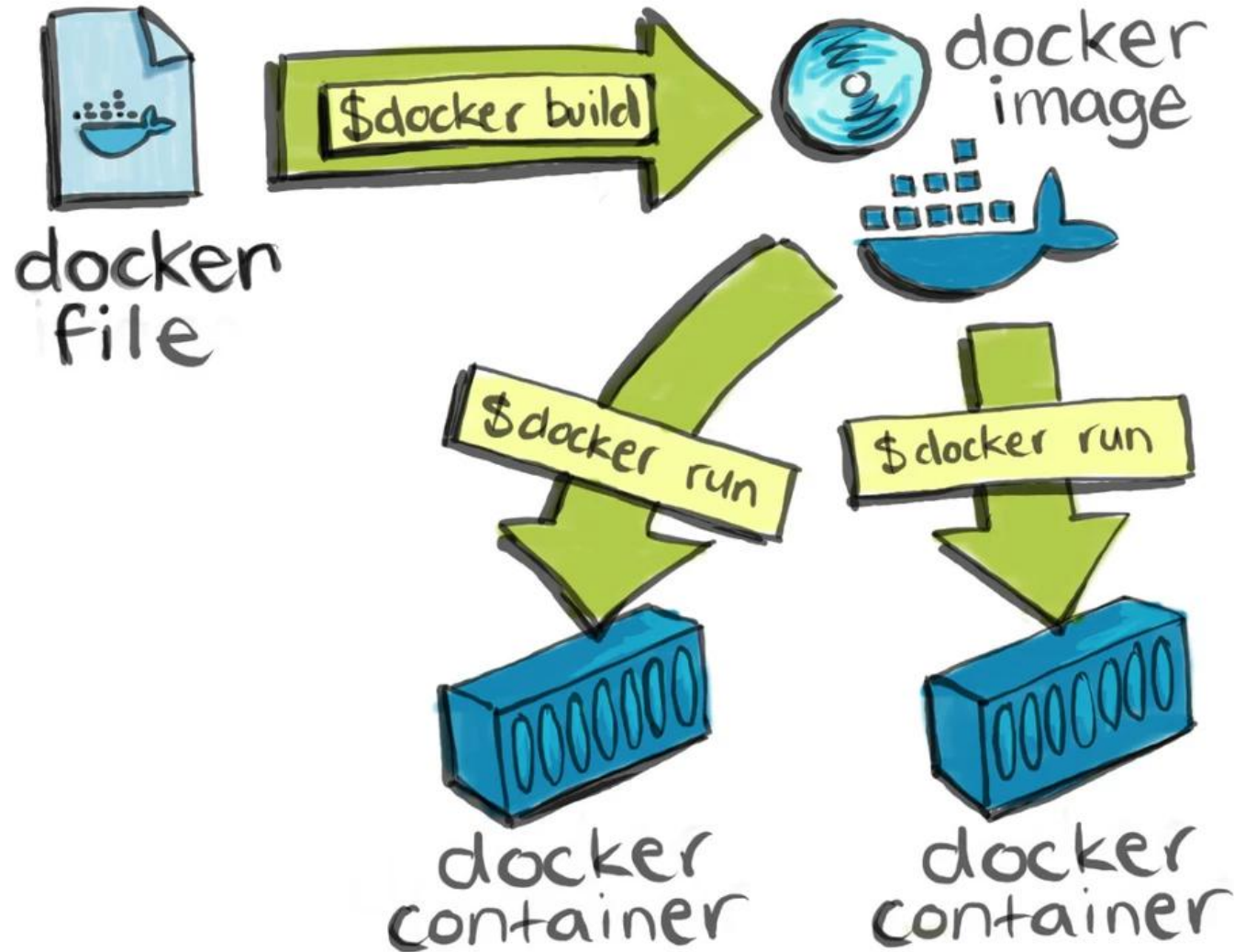


Dockerfile

- Un Dockerfile es un archivo de texto plano que contiene una serie de instrucciones necesarias para crear una imagen que, posteriormente, se convertirá en los contenedores que ejecutamos en el sistema
- Es como la receta para definir la imagen, que posteriormente lanzaremos como contenedor

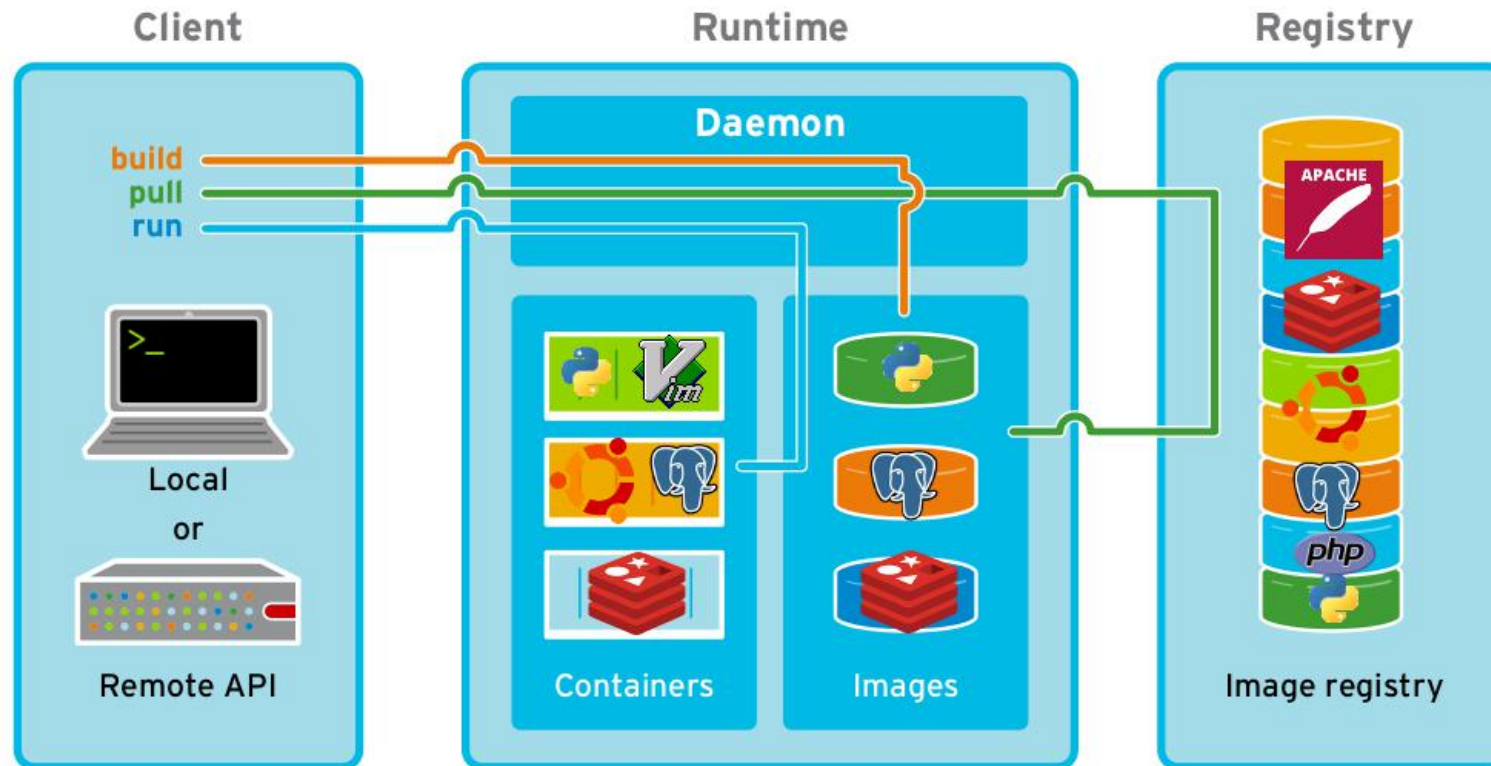


Resumiendo



DockerHub

- [DockerHub](#) es el registro de imagenes donde podemos subir nuestras imágenes o encontrar imágenes ya hechas con la que trabajar sobre ellas. Podemos usarlo de manera muy similar a la que usamos GitHub y nos podemos dar de alta en dicho servicio para almacenar nuestras imágenes



Instalar Docker

- Debido a que, dependiendo de la distribución, la forma de instalarlo difiere, es mejor consultar la documentación oficial para saber como instalar Docker en tu máquina.
- <https://docs.docker.com/get-docker/>
- Desinstalar versiones anteriores
`sudo apt-get remove docker docker-engine docker.io containerd runc`
- Configurando el repositorio
`sudo apt-get update`
`sudo apt-get install \`
`apt-transport-https \`
`ca-certificates \`
`curl \`
`gnupg-agent \`
`software-properties-common`



Instalar Docker

- Añadimos la clave GPG

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

- Añadimos el repositorio

```
sudo add-apt-repository \  
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \  
$(lsb_release -cs) \  
stable"
```

- Instalamos

```
sudo apt-get update  
sudo apt-get install docker-ce docker-ce-cli containerd.io
```



Instalar Docker

- Añadimos el usuario, de esta manera nos ahorramos hacer sudo siempre

```
sudo usermod -aG docker $USER
```

- Instalamos Docker Compose

```
sudo apt install docker-compose
```

- Probamos el primer contenedor

```
sudo docker run hello-world
```

```
$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adbf7777e9aacf18357296e799f81cab9fde470971e499788
Status: Downloaded newer image for hello-world:latest
```

```
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/
```



Comandos Docker

- Para ver los comandos disponibles (en general y en particular):

```
sudo docker
```

```
sudo docker image
```

```
sudo docker network
```

- Para ver las opciones de cada comando:

```
sudo docker cp --help
```

- Los comandos tienen a su vez opciones. Los nombres de las opciones van precedidas de los caracteres -- y entre la opción y su valor se puede escribir un espacio o el carácter =.

```
sudo docker COMANDO --OPCIÓN=VALOR
```

```
sudo docker COMANDO --OPCIÓN VALOR
```

- Si el valor de una opción contiene espacios, escriba el valor entre comillas

```
sudo docker COMANDO --OPCIÓN="VALOR CON ESPACIOS"
```

```
sudo docker COMANDO --OPCIÓN "VALOR CON ESPACIOS"
```



Comandos con imágenes

- Para gestionar las imágenes, se utiliza el comando:
`sudo docker image OPCIONES`
- Para descargar una imagen:
`sudo docker image pull REPOSITORIO`
`sudo docker pull REPOSITORIO`
- Para ver las imágenes ya descargadas:
`sudo docker image ls`
- Para borrar una imagen (se deben borrar previamente los contenedores basados en esa imagen):
`sudo docker image rm IMAGEN`



Comandos con contenedores

- Para crear un contenedor (y ponerlo en marcha):

```
sudo docker run --name=CONTENEDOR REPOSITORIO
```

El problema de este comando es que dejamos de tener acceso a la shell y sólo se puede parar el proceso desde otro terminal.

Lo habitual es poner en marcha el contenedor en modo separado (detached), es decir, en segundo plano, y así podemos seguir utilizando la shell:

```
sudo docker run -d --name=CONTENEDOR REPOSITORIO
```

- Si queremos ver la secuencia de arranque del contenedor, podemos poner en marcha el contenedor en modo pseudo-tty, que trabaja en primer plano, pero del que podemos salir con Ctrl+C.

```
sudo docker run -t --name=CONTENEDOR REPOSITORIO
```



Comandos con contenedores

- Al crear el contenedor se pueden añadir diversas opciones:

Para incluir el contenedor en una red privada virtual (y que se pueda comunicar con el resto de contenedores incluidos en esa red):

```
sudo docker run --name=CONTENEDOR --net=RED REPOSITORIO
```

Para que el contenedor atienda a un puerto determinado, aunque internamente atienda un puerto distinto:

```
sudo docker run --name=CONTENEDOR -p PUERTO_EXTERNO:PUERTO_INTERNO REPOSITORIO
```

Para establecer variables de configuración del contenedor:

```
sudo docker run --name=CONTENEDOR -e VARIABLE=VALOR REPOSITORIO
```

Las variables de configuración se pueden consultar en el repositorio del que obtenemos la imagen.



Comandos con contenedores

- Para ver los contenedores en funcionamiento:
`sudo docker ps`
- Para ver los contenedores en funcionamiento o detenidos:
`sudo docker ps -a`
- Para detener un contenedor:
`sudo docker stop CONTENEDOR`
- Para detener todos los contenedores:
`sudo docker stop $(sudo docker ps -aq)`
- Para borrar un contenedor:
`sudo docker rm CONTENEDOR`
- Para poner en marcha un contenedor detenido:
`sudo docker start CONTENEDOR`



Comandos con contenedores

- Para entrar en la shell de un contenedor:
`sudo docker exec -it CONTENEDOR /bin/bash`
- Para entrar en la shell del contenedor como root:
`sudo docker exec -u 0 -it CONTENEDOR /bin/bash`
- Para salir de la shell del contenedor:
`exit`
- Para copiar (o mover) archivos entre el contenedor y el sistema anfitrión o viceversa:
`sudo docker cp CONTENEDOR:ORIGEN DESTINO`
`sudo docker cp ORIGEN CONTENEDOR:DESTINO`



Comandos de red

- Para gestionar las redes, se utiliza el comando:
`sudo docker network OPCIONES`
- Para crear una red:
`sudo docker network create RED`
- Para ver las redes existentes:
`sudo docker network ls`
- Para ver información detallada de una red (entre ella, los contenedores incluidos y sus IP privadas):
`sudo docker network inspect RED`
- Para borrar una red:
`sudo docker network rm RED`



Comandos de sistema

- Para ver el espacio ocupado por las imágenes, los contenedores y los volúmenes:

`sudo docker system df`

- Para eliminar los elementos que no están en marcha:

Contenedores:

`sudo docker container prune`

Imágenes:

`sudo docker image prune`

Volúmenes:

`sudo docker volume prune`

Redes:

`sudo docker network prune`

todo:

`sudo docker system prune`



Referencias

- [Docker](#): Sitio oficial
- [DockerHub](#): Imágenes

