

Docker - Funcionamiento

Gestión de imágenes y contenedores

Pull

Entendemos por Pull **descargar una imagen** de docker hub (registros de imágenes) y almacena en local.

Vía terminal, la estructura del comando Pull es:

docker pull [Nombre_imagen]:[Etiqueta_version]

Ejemplo de Pull para preparar un servidor web **nginx** (similar a apache) versión :

docker pull nginx:1.27

Para ver las etiquetas de versiones y más info sobre esta imagen:

https://hub.docker.com/_/nginx



Pull

Por defecto el Pull se realiza sobre docker hub (docker.io). Se pueden buscar las imágenes de aplicaciones comunes directamente vía navegador:

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

Es posible usar el tag “latest”, o no indicar tag, para descargar la última versión, pero siempre es recomendable emplear un tag de versión específico (buena práctica).

Para **ver las imágenes descargadas** y sus tags (etiquetas), emplear el comando;

docker images



Run


Mediante el comando Run **se crea y se arranca un contenedor** con la imagen que se le especifique:

docker run [Nombre_imagen]:[Etiqueta]

En el caso de la imagen de nginx descargada:

docker run nginx:1.27

Al ejecutar el comando, por consola se mostraran logs de nginx inicializando y en funcionamiento. El terminal queda reservado a nginx.



ps

El comando “ps” representa “**process status**” y sirve para ver el estado de los contenedores en marcha:

docker ps

Muestra el ID de contenedor, la imagen sobre la que se ha arrancado, cuando fue creado, su estado, puertos en uso y su “nombre”.

Docker genera nombres aleatorios para los contenedores automáticamente si no se especifica uno.



Run -d


Mediante el comando Run -d se crea y se arranca un contenedor con la imagen que se le especifique, pero en **modo “detach”** para no bloquear el terminal:

docker run -d [Nombre_imagen]:[Etiqueta]

En el caso de la imagen de nginx descargada:

docker run -d nginx:1.27

Al ejecutar el comando, por consola se mostrará el ID completo del contenedor, pero sin bloqueo.



logs

Con un contenedor arrancado en modo detach ya no es posible ver los logs con consola. En caso de que queramos verlos, es posible mostrarlos vía comando:

docker logs [ID_Contenedor]

Nota: Basta con la ID de contenedor corta (primeros caracteres, ver docker ps).

Esto nos muestra por pantalla los registros generados por el contenedor hasta ese momento, sin bloquear la terminal.

También es posible emplear el **nombre de contenedor**.



stop

Para detener uno o más contenedores en marcha:

docker stop [ID_Container]

Nota: Basta con la ID de contenedor corto (primeros caracteres, ver docker ps).

Se pueden poner sucesivos IDs para detener varios contenedores simultáneamente.

También es posible emplear el **nombre de contenedor**.



Sobre Pull & Run

No es necesario realizar manualmente la descarga de imagen mediante Pull:
Run realiza un Pull de la imagen sobre docker hub si no la localiza en local.

Si queremos arrancar otra versión anterior de nginx, la 1.23, que no está descargada, basta con hacer:

```
docker run nginx:1.23
```



Container Port vs Host Port

La aplicación en funcionamiento dentro de un contenedor se encuentra en una red de docker **aislada**, separada de la red del equipo local.

Esto **permite poder tener varias aplicaciones en funcionamiento trabajando sobre el mismo puerto.**



Container Port vs Host Port

Para poder acceder a la aplicación, se debe **exponer** el puerto que usa el **contenedor al puerto del host** (la máquina local sobre la que se ejecuta el contenedor).

Para ello se deben enlazar los puertos (**port binding**).

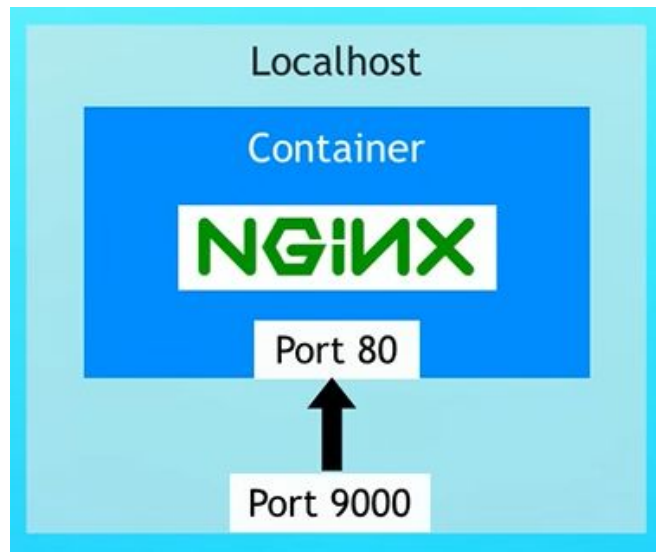


Port Binding

Se puede emplear **cualquier puerto del host** al enlazar con los puertos de un contenedor.

En el ejemplo de Nginx, igual que apache, funciona sobre el puerto 80 por defecto. Se puede asignar el puerto del host que se desee (ej. 9000) para vincularlo con el puerto 80 del contenedor.

Aun así, el estándar consiste en enlazar el mismo puerto local que se requiere en el contenedor (buena práctica. ej, 80:80).



Port Binding - run -p

El enlace de puertos se debe configurar al arrancar un contenedor. Para esto, al comando **Run** se añade el flag “-p” o “--publish” para publicar el puerto del contenedor al host.

```
docker run -p [Puerto_Host]:[Puerto_contenedor] [Nombre_Imagen]:[Etiqueta_ver]
```

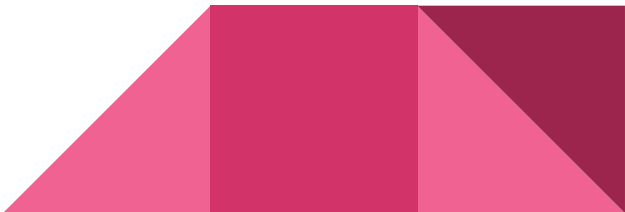
Para hacer accesible un contenedor de Nginx mediante el puerto 9000:

```
docker run -d -p 9000:80 nginx:1.27
```

Este comando permite acceder a la página por defecto de Nginx vía: localhost:9000. Tras acceder, si se visualizan los logs del contenedor se pueden observar peticiones GET.

Solo se puede definir **un puerto del host por servicio**.

El comando ps muestra este enlace de puertos.



ps -a

Cada vez que se ejecuta un “docker run” se está creando un nuevo contenedor, **NO se están reutilizando**. docker ps solo nos muestra los contenedores en funcionamiento. Sin embargo, si añadimos el flag “-a” o “--all” se **mostrará la lista completa de contenedores**, tanto en funcionamiento como parados.

docker ps -a



start

Para reutilizar un contenedor ya creado previamente, se puede emplear el comando start:

docker start [ID_Contenedor]

Nota: Basta con la ID de contenedor corto (primeros caracteres, ver docker ps).

Se pueden poner sucesivos IDs para iniciar varios contenedores simultáneamente.

También es posible emplear el **nombre de contenedor**.



Nombrar un contenedor - run --name

Al crear un contenedor, mediante el flag “**--name**” es posible **otorgar un nombre específico** (no aleatorio) al contenedor que se vaya a crear. Esto facilita mucho su gestión (por ejemplo, para arrancar y detener por consola, o ver logs).

```
docker run --name [Nombre_contenedor] [Nombre_Imagen]:[Etiqueta_ver]
```

Aplicado sobre el ejemplo de Nginx queda:

```
docker run --name web-server -d -p 9000:80 nginx:1.27
```

