COMANDOS DOCKER

docker run imagen	Arranca una imagen (si no existe, la descarga también).
docker run 'imagen:versión'	Arrancar una imagen con una versión concreta.
docker pull imagen	Descarga la imagen, pero no la arranca (por defecto la más reciente).
docker images	Muestra las imágenes que tenemos descargadas.
docker images | head	Muestra las primeras líneas de las imágenes descargadas.
docker ps	Muestra las imágenes que están arrancadas, activas.
docker ps -a	Muestra las últimas imágenes utilizadas.
control C	Frena una imagen arrancada.
docker start ID	Si queremos reiniciar/recuperar el contenedor que hemos paralizado.
docker log -f ID	Muestra las salidas de la imagen.
docker exec ID	Ejecuta un comando dentro de un contenedor que está arrancado.
docker exec -it ID sh	-i crea una sesión interactiva, -t emula una terminal sh es una shell.
docker stop ID	Para un contenedor.
docker run -d imagen	Arranca una imagen "background".
docker rm ID	Borra un contenedor.

- **docker pull nombre_imagen:version** → **Descargar imagen**
  Descarga desde el repositorio una imagen con la versión indicada o la última versión **(latest)** si no indicamos versión.

- **docker run [options] nombre_imagen [command] [args]** → **Ejecutar contenedor en base a una imagen**
  Si vamos a ejecutar un contenedor que usa como base una imagen que no tenemos ésta se descargará de manera automática.

  ❖ [options]: opciones para el arranque, por ejemplo, nombre, ip, puertos, volúmenes, I/O hostname…
  - ➢ **-d o --detach** para ejecutar un contenedor (normalmente porque tenga un servicio) en background. Si no lo hacemos se bloqueará el terminal mostrando el log del servicio y tendremos que salir del mismo con Ctrl+C. Esto para el contenedor, aunque podremos arrancarlo posteriormente.
  - ➢ **-e o --env** para establecer variables de entorno en la ejecución del contenedor. **-e NOMBRE_VARIABLE=VALOR**
  - ➢ **-h o --hostname** para establecer el nombre de red parar el contenedor.
  - ➢ **--help** para obtener ayuda de las opciones de docker.
  - ➢ **--interactive o -i** para mantener la STDIN abierta en el contenedor.
  - ➢ **--ip** si quiero darle una ip concreta al contenedor.
  - ➢ **--name** para darle nombre al contenedor. Si usamos nombre elegidos por nosotros serán más fáciles de recordar que los asignados por defecto. Además podemos elegir nombres que tenga relación con la función que va a desempeñar dicho contenedor. Ejemplo: *docker run -d --name servidorWeb -p 80:80 httpd*
  - ➢ **--net o --network** para conectar el contenedor a una red determinada.
  - ➢ **-p o --publish** para conectar puertos del contenedor con los de nuestro host. **-p PUERTO_EN_HOST:PUERTO_EN_CONTENEDOR** es una **redirección de puertos.** No podemos tener dos servicios escuchando en el mismo puerto.
  - ➢ **--restart** que permite reiniciar un contenedor si este se "cae" por cualquier motivo.
  - ➢ **--rm** que destruye el contenedor al pararlo.
  - ➢ **--tty o -t** para que el contenedor que vamos a ejecutar nos permita un acceso a un terminal para poder ejecutar órdenes en él.
  - ➢ **--user o -u** para establecer el usuario con el que vamos a ejecutar el contenedor.
  - ➢ **--volume o -v** para montar un bind mount o un volumen en nuestro contenedor.
  - ➢ **--wordirk o -w** para establecer el directorio de trabajo en un contenedor.

  ❖ **Siempre usar el flag -it** (-i + -t) al ejecutar una orden docker run si es un contenedor que no tiene servicios.
  ❖ **No debemos añadir al final otra orden que no sea /bin/bash** (u otro shell que puedan contener los contenedores) ya que eso sobreescribe la orden de arranque de algunos contenedores y será imposible volver a arrancarlos una vez parados.

> docker run -d -p 8888:80 httpd → Ejecuto un servidor Apache en background y accediendo desde el exterior a través del puerto 8888 de mi máquina.
> docker run -it -d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=root mariadb → Creación de un servidor de BBDD mariadb accediendo desde el exterior a través del puerto 3306 y estableciendo una contraseña de root mediante una variable de entorno.

- **docker exec [options] nombre_contenedor orden [args] → Ejecutar órdenes en los contenedores.**
  Para ejecutar este comando, el contenedor debe estar en ejecución (docker run).
  - **[options]**
    - **-it** (-i y -t juntos) si vamos a querer tener interactividad con el contenedor ejecutando un shell (/bin/bash normalmente). Una vez tenemos el terminal ya podremos trabajar desde dentro del propio sistema.
    - **-u o --user** si quiero ejecutar la orden como si fuera un usuario distinto del de root.
    - **-w o --workdir** si quiero ejecutar la orden desde un directorio concreto.

  > docker exec -it web /bin/bash → Obtener un terminal en un contenedor que ejecutar un servidor Apache (httpd) y que se llama web
  > docker exec web ls /usr/local/apache2/htdocs → Mostrar el contenido de la carpeta /usr/local/apache2/htdocs del contenedor web. Como no hace falta interactividad no es necesario -it
  > docker exec -it web sh -c "echo 'HOLA MUNDO' > /usr/local/apache2/htdocs/index.html" → Mostrar el contenido de la carpeta /usr/local/apache2/htdocs del contenedor web. Como no hace falta interactividad no es necesario -it
  > docker cp prueba.html web:/usr/local/apache2/htdocs/index.html → Copiar mi fichero prueba.html al fichero /usr/local/apache2/htdocs/index.html de mi contenedor llamado web que es un servidor Apache (httpd)

- **docker ps [options] → obtener información de los contenedores ya arrancados (estado, imagen de la que deriva, tamaño actual, orden que ejecuta al arrancar, nombre, fecha de creación y redirección de puertos).**
  - **[options]**
    - **-a o --all** muestra todos los contenedores, estén parados (EXITED) o en ejecución.
    - **-s o --size** añade la información del tamaño del contenedor a la información por defecto.
    - **-l o --latest** muestra información del último contenedor que se ha creado. Da igual el estado.
    - **-f o --filter** filtra los contenedores de acuerdo a algún criterio.
      > docker ps --filter name=servidor_web → filtrado por nombre
      > docker ps --filter publish=8080 → filtrado por puerto. Contenedores que hacen público el puerto 8080

- **docker inspect [contenedor_nombre/contenedor_id] → obtiene información más detallada del contenedor seleccionado (id, puertos abiertos y redirecciones, bind mounts y volúmenes usados, tamaño, configuración de red, entrypoint, valor de variables de entorno)**
  - Por nombre: *docker inspect jenkins*
  - Por id: *docker inspect 5e5adf6815bc*

  # Mostrar la ip del contenedor
  >docker inspect --format 'La ip es {{.NetworkSettings.Networks.bridge.IPAddress}}' jenkins
  >La ip es 172.17.0.2
  # Mostrar las redirecciones de puertos del contenedor
  > docker inspect --format 'Las redirecciones de puertos son {{.NetworkSettings.Ports}}' jenkins
  > Las redirecciones de puertos son map[50000/tcp:[{0.0.0.0 50000}] 8080/tcp:[{0.0.0.0 9090}]]

- **docker logs [contenedor_nombre/contenedor_id] → obtiene información de lo que está pasando en el contenedor, tanto si está parado o en ejecución**
  - Por nombre: *docker logs jenkins*
  - Por id: *docker logs 5e5adf6815bc*

  # Opción **-f o --follow** . Sigue escuchando la salida que pueden dar los logs del contenedor
  > docker logs -f jenkins
  # Opción **--tail** 5. Muestra las 5 últimas líneas de los logs del contenedor en cuestión
  > docker logs --tail 5 jenkins

- **docker stop** →para detener el contenedor, ya sea por nombre o por ID. Si hago docker stop y el contenedor ya está parado, no pasa nada.
  docker stop servidorWeb → Para un contenedor en ejecución que se llame servidorWeb
  docker stop -t 10 ea9b922190d8 → Para un contenedor en ejecución cuyo ID es ea9b922190d8 pero esperando 10 segundo (-t o --time)

- **docker rm** →para borrar el contenedor, ya sea por nombre o por ID. Es importante destacar que SI UN CONTENEDOR ESTÁ EN EJECUCIÓN NO PODEMOS BORRARLO salvo que usemos la opción -f.
  docker rm servidorBD → Borrar un contenedor que se llama servidorBD
  docker rm -f jenkins → Borrado un contenedor que se llame jenkins aunque esté en ejecución (--force o -f)

- **docker start** → iniciar un contenedor que estaba parado previamente, ya sea por nombre o por ID. Si hago **docker start** y el contenedor ya está iniciado, no pasa nada.
  docker start jenkins → Inicio de un contenedor con nombre jenkins
  docker start -i jenkins → Inicio de un contenedor con nombre jenkins pero haciendo el attach de la entrada estándar para poder interactuar con él (-i o --interactive)

- **docker restart** → para reiniciar un contenedor que previamente ya estaba en ejecución. Si hago docker restart y el contenedor ya está parado, es lo mismo que si ejecutar un docker start.
  docker restart ea9b922190d8 → Reinicio de un contenedor con ID ea9b922190d8
  docker restart -t 10 ea9b922190d8 → Reinicio de un contenedor con ID ea9b922190d8 pero esperando 10 segundo (-t o --time)