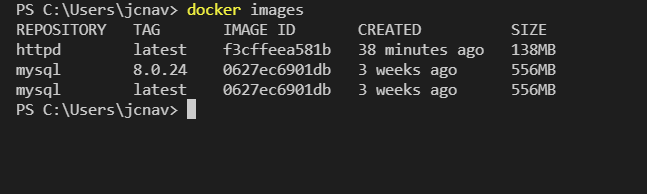
|  |
| --- |
|  |
| Actividades Docker |
|  |

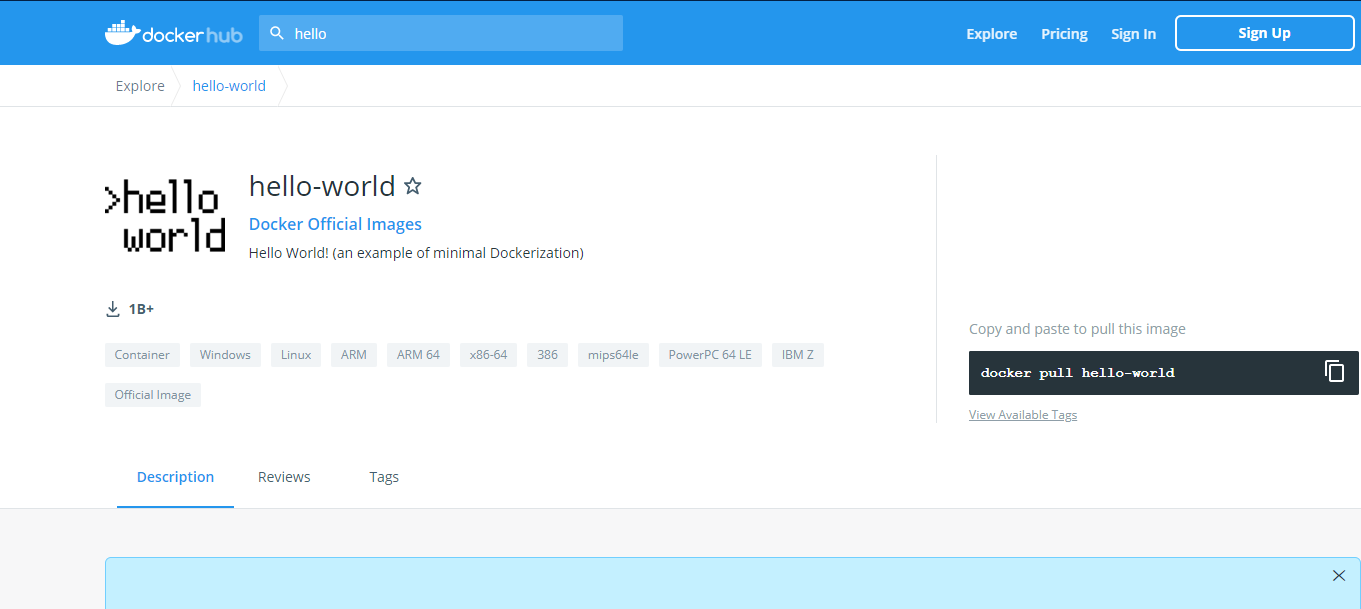
|  |
| --- |
| Juan Carlos Navidad García |

**ACTIVIDAD:**

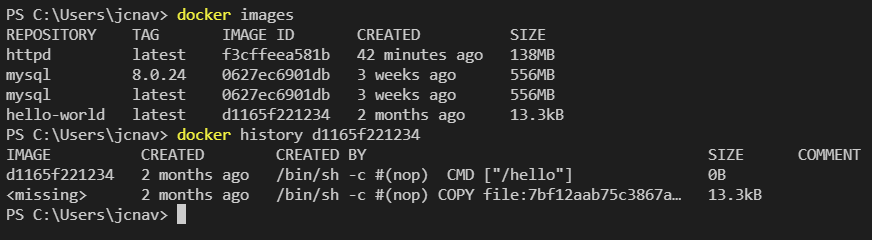
1. Verifica las imágenes que tienes en tu sistema



1. Busca en docker-hub otra imagen de hola-mundo y descargatela



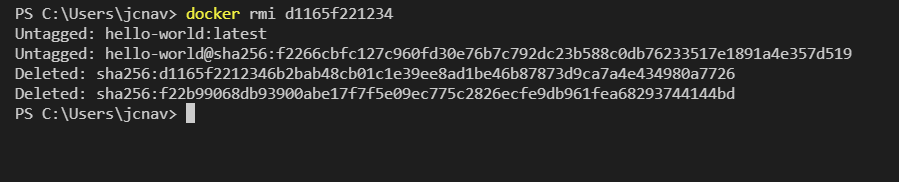
1. Muestra el historial de cambios de una de las imágenes que tengas descargadas



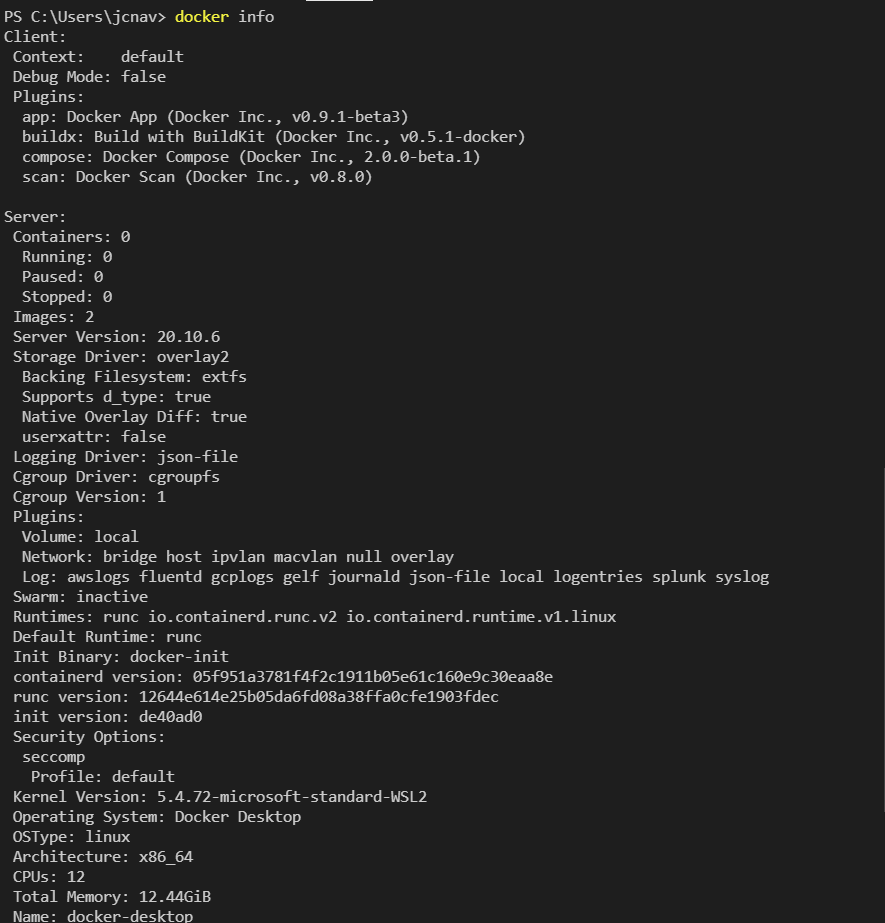
1. Muestra las características de la imagen ultima que te hayas descargado



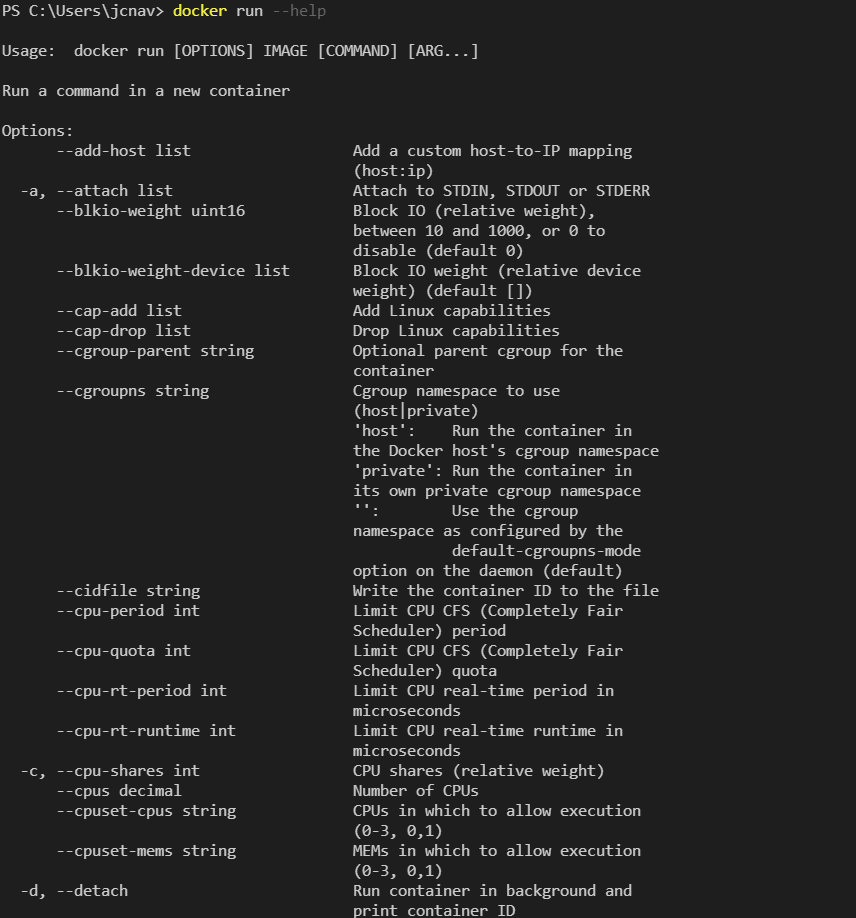
1. Elimina la última imagen descargada con un solo comando



1. Utiliza el comando info para mostrar información de sistema (imágenes, contenedores…)



1. Utiliza la opcion – help sobre el comando run para ver que realiza

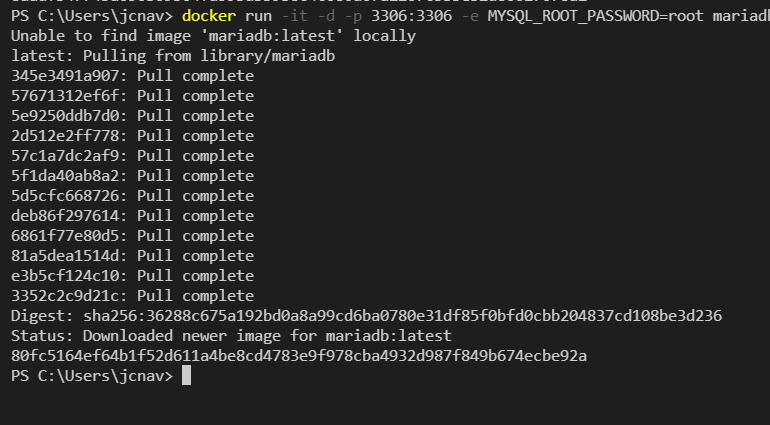


1. Del resultado anterior investiga para que sirven las siguientes opciones
   * -d: Para abrir un contenedor en segundo plano.
   * -e: Establece variables de entorno.
   * -h: Establece un nombre de red parar el contenedor.
   * -i: Establece una IP para el contenedor.
   * -t: Asigna un pseudo-TTY.
   * -w: Directorio de trabajo dentro del contenedor.
   * -u: Tienes que acompañarlo de un usuario o un UID.
   * -v: Montar en un volumen específico.
   * -p: Establecer puertos.
   * –name: Le da un nombre al contenedor.
   * –network: Conecta el contenedor a una red determinada.
   * –ip: Le da una IP concreta al contenedor.
   * –mount: Para montar un disco duro.
   * –rm: Para eliminar un contenedor

**ACTIVIDAD:**

CREA LOS SIGUIENTES CONTENEDORS

|  |
| --- |
| # Ejecuto un servidor Apache sin el flag -d ni redirección de puertos. Se bloquea el terminal mostrando los logs y tendré que salir con Ctrl+C  > docker run httpd  AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.22. Set the 'ServerName' directive globally to suppress this message AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.22. Set the 'ServerName' directive globally to suppress this message [Mon Dec 07 18:27:28.561909 2020] [mpm\_event:notice] [pid 1:tid 140253864719488] AH00489: Apache/2.4.46 (Unix) configured -- resuming normal operations [Mon Dec 07 18:27:28.562072 2020] [core:notice] [pid 1:tid 140253864719488] AH00094: Command line: 'httpd -D FOREGROUND'    # Ejecuto un servidor Apache en background y accediendo desde el exterior a través del puerto 8888 de mi máquina.  > docker run -d -p 8888:80 httpd    # Creación de un servidor de base de datos mariadb accediendo desde el exterior a través del puerto 3306 y estableciendo una contraseña de root mediante una variable de  # entorno  > docker run -it -d -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=root mariadb |



|  |
| --- |
| # Arrancar el contenedor con nombre servidorWeb (debe estar parado)  > docker start servidorWeb |

|  |
| --- |
| # Crear un contenedor de debian 9 y mostrar el contenido de una carpeta establecida con el parámetro -w  > docker run -it -w /etc debian:9 ls |

Al crear el contenedor se ejecuta la orden ls desde el directorio /etc, posteriormente el contenedor pasa a estar parado. Y ya no podremos acceder a él. Explicaremos en el próximo apartado el porqué.

Conforme vayamos creando contenedores hay dos órdenes que nos van a interesar para hacer un seguimiento de qué tenemos en nuestro sistema:

|  |
| --- |
| # Mostrar los contenedores en ejecución (Estado Up)  > docker ps    # Mostrar todos los contenedores creados ya estén en ejecución (Estado Up) o parados (Estado Exited)  > docker ps -a |

**ACTIVIDAD:**

CREA LOS SIGUIENTES CONTENEDORS

**EJECUTAR ÓRDENES EN LOS CONTENEDORES. DOCKER EXEC**

Con los contenedores en ejecución vamos a querer ejecutar ordenes en ellos. Querremos realizar operaciones como:

* Instalar paquetes.
* Modificar o ver el contenido de ciertos ficheros.
* Habilitar ciertos módulos de servicios
* etc...

Esto lo podemos hacer de **dos maneras** o bien **obteniendo un terminal** del contenedor y ejecutando las órdenes necesarias desde allí **o bien directamente** ejecutando una orden determinada "contra" el contenedor. Para ambos casos voy a necesitar la orden **docker exec y es NECESARIO QUE EL CONTENEDOR ESTÉ EN EJECUCIÓN.**

La sintaxis de esta orden es bastante sencilla y muy similar a la de docker run:

**docker exec [opciones] nombre\_contenedor orden [argumentos]**

Algunas de las opciones más importantes son:

* **-it**(-i y -t juntos) si vamos a querer tener interactividad con el contenedor ejecutando un shell (/bin/bash normamente). 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.

Lo vamos a ver mejor con algunos ejemplos:

|  |
| --- |
| # Obtener un terminal en un contenedor que ejecutar un servidor Apache (httpd) y que se llama web  > docker exec -it web /bin/bash  root@5d96ce1f7374:/usr/local/apache2#    # Mostrar el contenido de la carpeta /usr/local/apache2/htdocs del contenedor web. Como no hace falta interactividad no es necesario -it  > docker exec web ls /usr/local/apache2/htdocs    # Crear directamente un fichero "HOLA MUNDO" en el directorio raíz del servidor apache. Utilizo sh -c para ordenes compuestas o complejas  > docker exec -it web sh -c "echo 'HOLA MUNDO' > /usr/local/apache2/htdocs/index.html" |

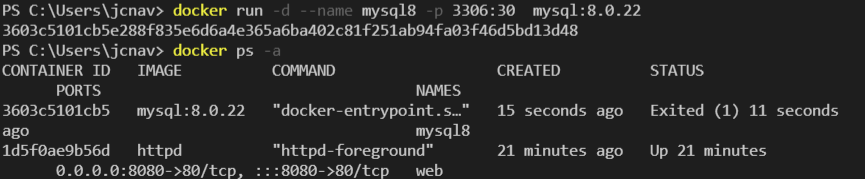
Adicionalmente existe otra orden que nos va a ser de mucha utilidad cuando trabajemos con contenedores. la orden **docker cp** que me permite mover ficheros desde mi sistema al contenedor y desde el contenedor a mi sistema. Su sintaxis es muy sencilla y la vamos a ilustrar con dos ejemplos, uno en cada sentido:

|  |
| --- |
| # 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 cp prueba.html web:/usr/local/apache2/htdocs/index.html |

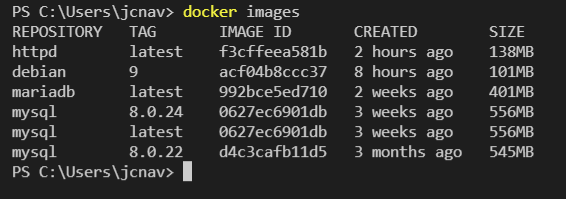
**NOTA: LOS CONTENEDORES VIENEN CON SOLO LO IMPRESCINDIBLE INSTALADO. SI QUIERO INSTALAR ALGO DEBO NORMALMENTE HACER ANTES UN APT UPDATE (ya que la mayoría son basados en Debian).**

**EJERCICIOS**

1. **Crear un contenedor con el servicio mysql versión 8.0.22 con el nombre mysql8 y que se pueda acceder desde el puerto 3306 en el host. Ademas ejecútalo en segundo plano de forma iterativa.**



1. **Lista las imágenes descargadas**



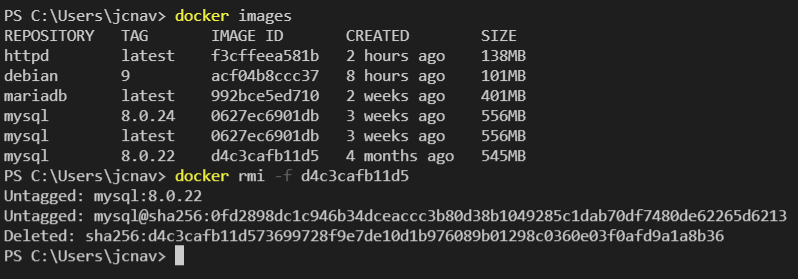
1. **¿Qué información nos muestra el comando Docker images?**

Nos muestra las imágenes descargadas.

1. **¿Que realiza la opción -q? sobre Docker pull o Docker run**

Le asigna un puerto al contenedor.

1. **Borra la imagen msql8 usando su image id.**



1. **¿Cuál de las siguientes instrucciones son correctas?**

* > docker rmi mysql:8.0.22, **es correcta**
* > docker rmi dd7265748b5d, **es correcta**
* > docker image rm mysq:8.0.2, **no es correcta**
* > docker image rm dd7265748b5d, **no es correcta**
* > docker rmi mysql:8.0.22 mysql:5.7**, es correcta**

1. **¿Es posible borrar una imagen si un contenedor la esta usando?**

Es posible forzándola a borrarse, es decir, utilizando -f en el comando.

1. **Borra todas las imágenes**

