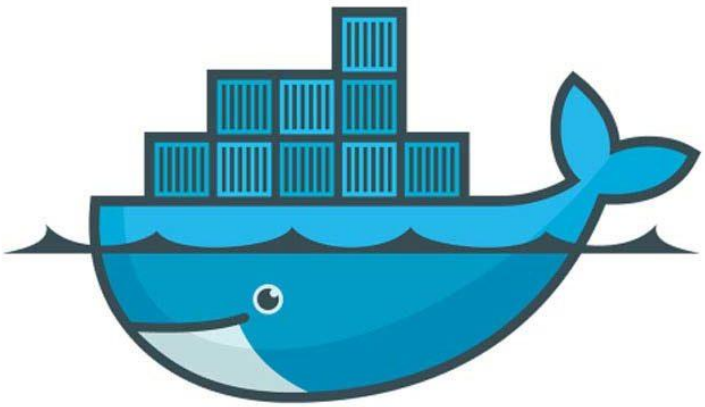
DOCKER

Sistemas Informáticos (1º DAM)



[FECHA]

[NOMBRE DE LA COMPAÑÍA]
[Dirección de la compañía]

Índice

Contenido

1. Actividad 1: descargar imágenes y operaciones básicas

1.1. Descargar imágenes de Docker Hub.

Para descargar los dockers, debemos de utilizar el comando "***docker pull <imagen[:versión]>***. Por defecto, cogerá la versión "latest" si no ponemos versión.

Ubuntu:18.04

```
usuario@usuario:~$ sudo docker pull ubuntu:18.04
[sudo] contraseña para usuario:
18.04: Pulling from library/ubuntu
7c457f213c76: Pull complete
Digest: sha256:152dc042452c496007f07ca9127571cb9c29697f42acbfad72324b2bb2e43c98
Status: Downloaded newer image for ubuntu:18.04
docker.io/library/ubuntu:18.04
usuario@usuario:~$ S
```

Centos:8

```
usuario@usuario:~$ sudo docker pull centos:8
8: Pulling from library/centos
a1d0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos:8
docker.io/library/centos:8
usuario@usuario:~$ S
```

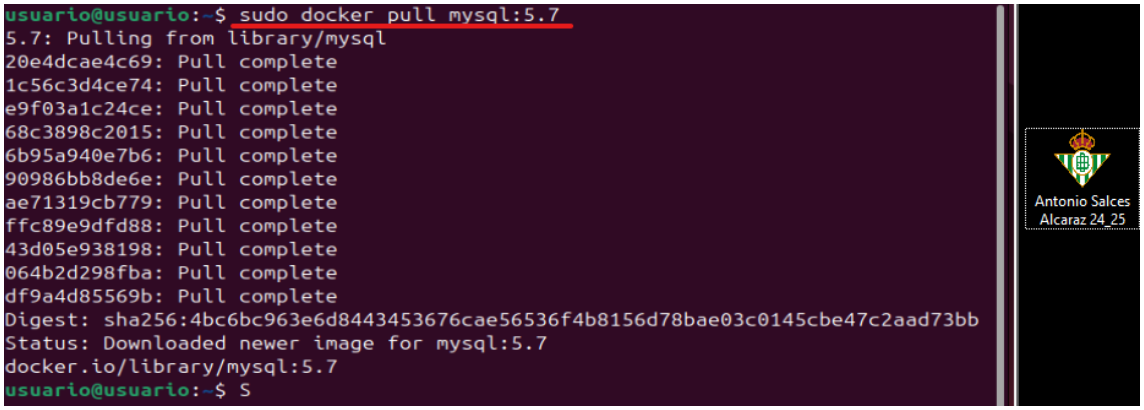
Debian: 9

```
usuario@usuario:~$ sudo docker pull debian:9
9: Pulling from library/debian
8372a04f222b: Pull complete
Digest: sha256:c5c5200ff1e9c73ffbf188b4a67eb1c91531b644856b4aefe86a58d2f0cb05be
Status: Downloaded newer image for debian:9
docker.io/library/debian:9
```

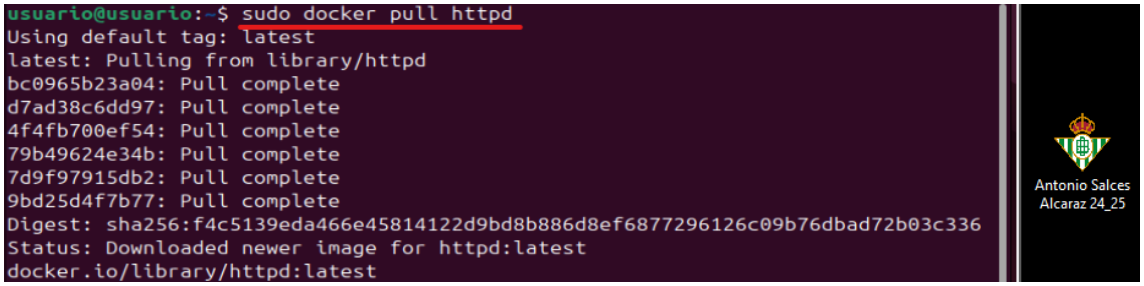
MariaDB:latest

```
usuario@usuario:~$ sudo docker pull mariadb
Using default tag: latest
latest: Pulling from library/mariadb
de44b265507a: Pull complete
ca9b21d0c985: Pull complete
041c753879ae: Pull complete
e7b5137dc4b2: Pull complete
655e5d2590bd: Pull complete
41b3170b5f12: Pull complete
95adc28016bc: Pull complete
407e9d6eefb4: Pull complete
Digest: sha256:a9547599cd87d7242435aea6fda22a9d83e2c06d16c658ef70d2868b3d3f6a80
Status: Downloaded newer image for mariadb:latest
docker.io/library/mariadb:latest
usuario@usuario:~$
```

MySQL:5.7

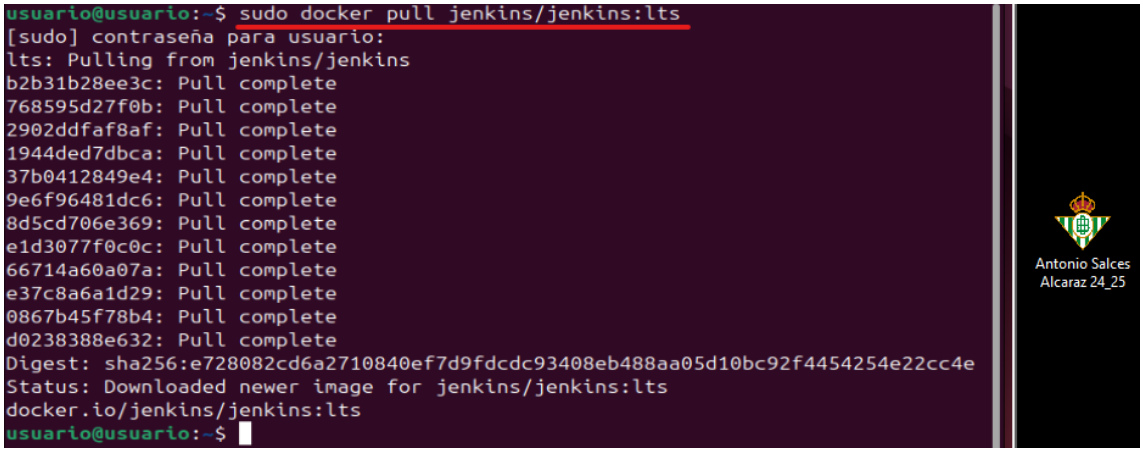


httpd

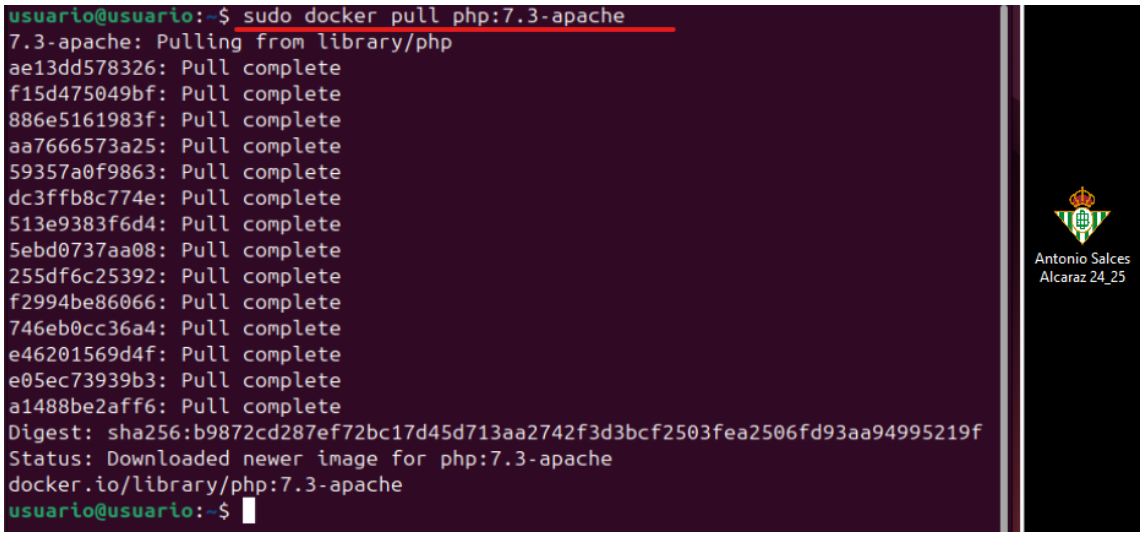


tomcat:9.0.39-jdk11

jenkins/jenkins:lts



php:7.3-apache

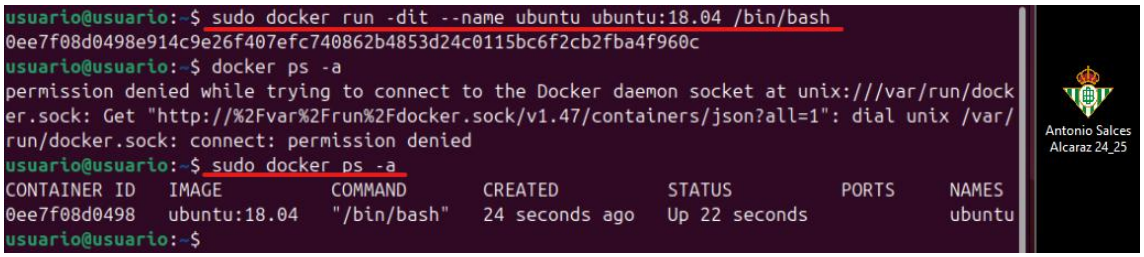


1.2. Mostrar imágenes

Para hacer esto, debemos de hacer uso del comando "***docker images***"

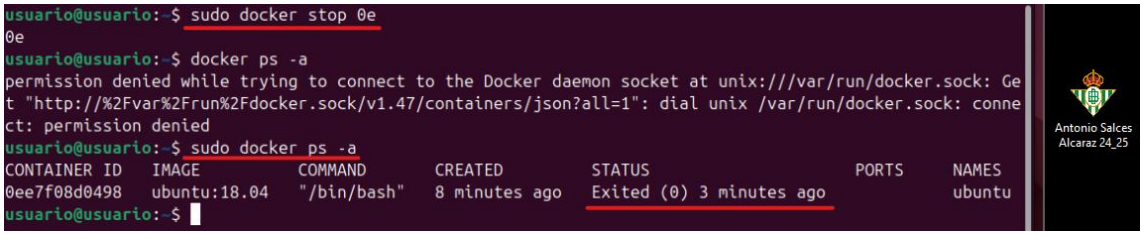
1.3. Contenedor con Ubuntu 18.04, nombre “ubuntu”

Para arrancar el contenedor y darle un nombre, utilizaremos el comando “***docker run [-dit] [--name <nombre>] <imagen:versión> [comando]***”. También utilizaremos el comando “***docker ps -a***” para ver todos los contenedores del sistema y asi poder saber el ID del docker de ubuntu.



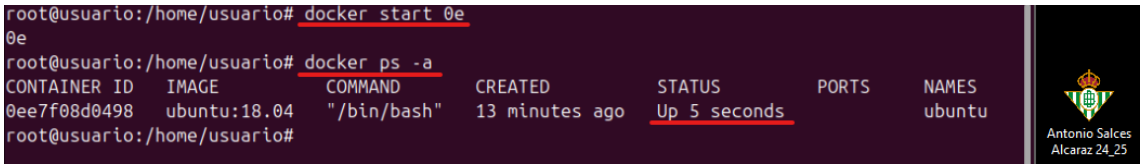
1.3.1. Parar contenedor

Para parar el contenedor, ejecutaremos el comando “***docker stop <ID>***”. Basta con poner el inicio del ID, no es necesario ponerlo entero. Para comprobar que está parado, basta con poner “***docker ps -a***” y comprobar el estado, donde se ve que está parado.



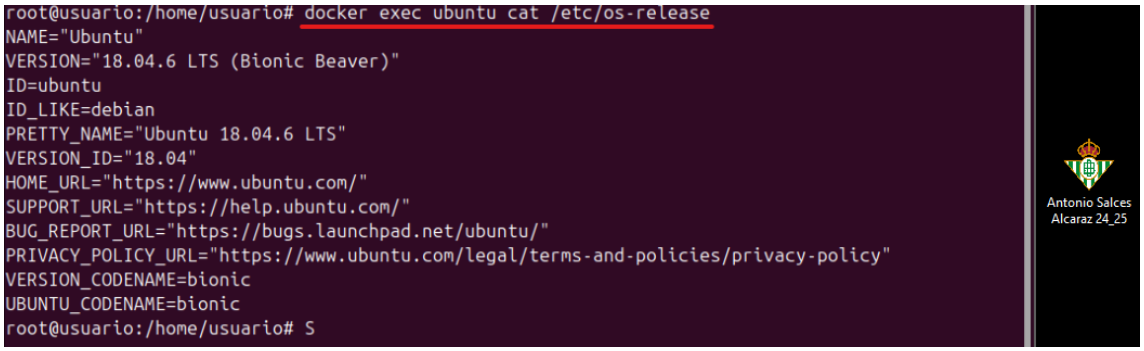
1.3.2. Rearrancar contenedor

Para rearrancar el contenedor, utilizaremos “***docker start <ID>***”, y para comprobar si se ha iniciado, “***docker ps -a***”.



1.3.3. Mostrar fichero sin entrar al contenedor

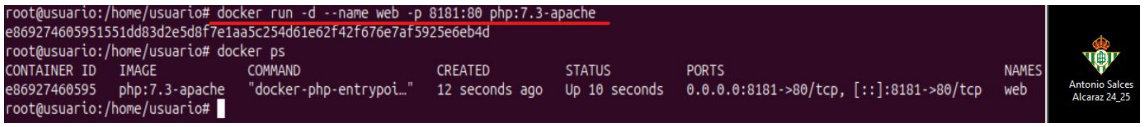
Para ello, debemos utilizar el comando "***docker exec <nombre> cat /etc/os-release***". Docker exec nos permite ejecutar comandos del contenedor desde fuera del mismo, mientras que "cat" se utiliza para visualizar archivos. En lugar del nombre, también podemos poner el ID.



2. Actividad 2: trabajar con PHP y MariaDB

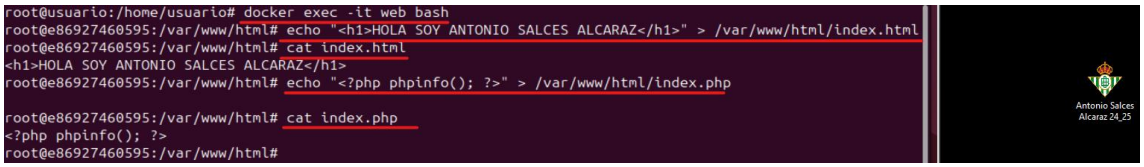
2.1. Contenedor con PHP, nombre “web” y puerto 8181

Para iniciar el contenedor, utilizaremos el comando “***docker run [-d] [--name <nombre>] [-p <puerto>] <imagen>***”. Con “-d” ejecutaremos el docker en segundo plano, y con “-p” podremos elegir el puerto para el docker, en este caso el 8181.



2.2. Colocar ficheros index.html e index.php en el directorio web del contenedor

En primer lugar, accederemos al directorio web del contenedor con el primer comando que se ve en la imagen (“***docker exec -it [nombre o ID] bash***”). Una vez estemos en el contenedor, utilizaremos “***echo <contenido del archivo> > <ubicación del archivo>***”. Para comprobar el contenido del archivo, podemos utilizar “***cat <nombre>***”. Haremos lo mismo con el fichero “index.php”.



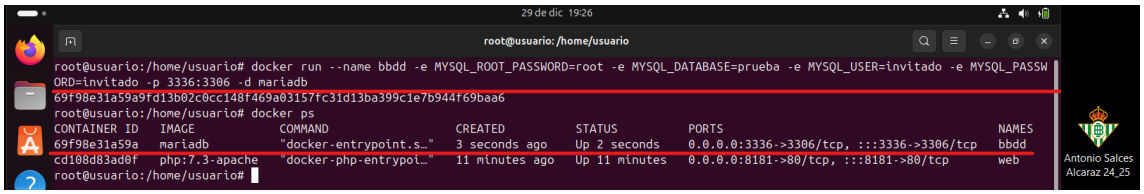
Si buscamos la siguiente dirección en la maquina virtual, podremos ver el fichero index.html que hemos creado.

Y esto saldrá si buscamos el fichero index.php.

2.3. Contenedor con mariadb con diferentes parámetros

Debemos arrancar un contenedor que contenga “mariadb” con los siguientes parámetros:

- Nombre: bbdd.
- Puerto: 3336.
- Contraseña de root: root.
- Crear base de datos con nombre: prueba.
- Contraseña del usuario: usuario.

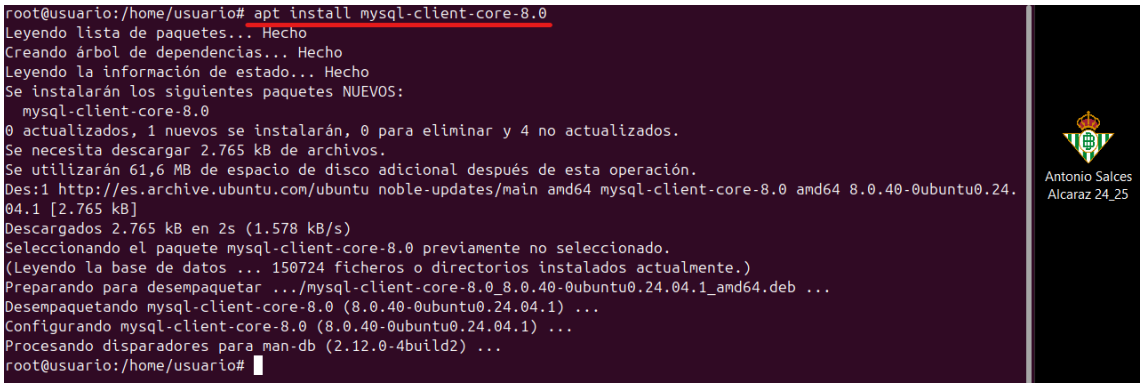


Los parámetros que se han utilizado son los siguientes:

- --name bbdd: el nombre del docker es “bbdd”.
- -e MYSQL_ROOT_PASSWORD=root: la constraseña del root será “root”.
- -e MYSQL_DATABASE=prueba: crea la base de datos llamada “prueba”.
- -e MYSQL_USER=invitado: crea el usuario "invitado".
- -e MYSQL_PASSWORD=invitado: la contraseña del usuario “invitado” será “invitado”.
- -p 3336:3306: indica el puerto a utilizar.
- -d mariadb: lanzaremos “mariadb” en segundo plano.

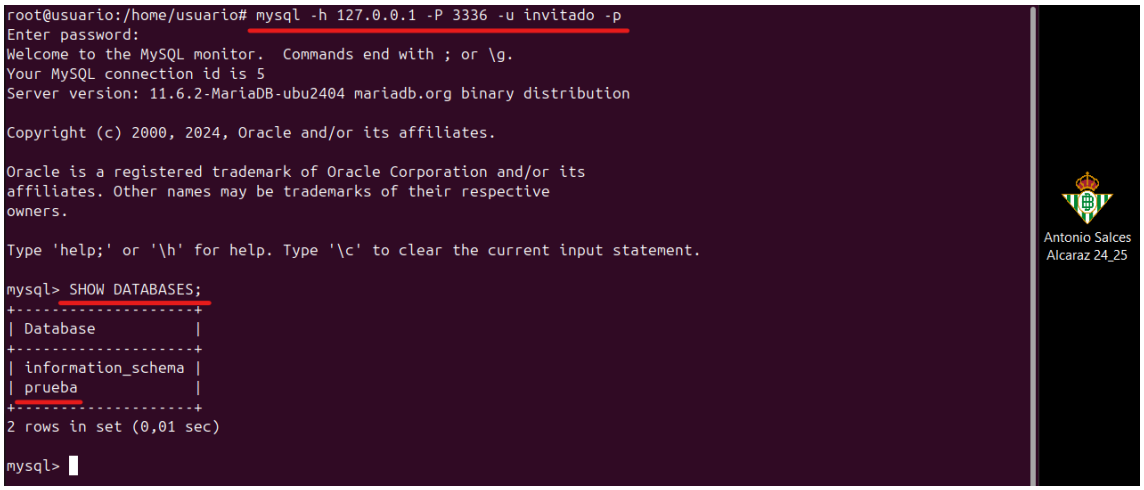
2.4. Conectar con un cliente de base de datos

En primer lugar, debemos de instalar un cliente de base de datos. En mi caso, utilizaré un cliente de terminal (mysql-client). Utilizaremos el siguiente comando para instalarlo: “***apt install <paquete>***”.



Con el primer comando conseguiremos conectarnos con la base de datos del contenedor. El comando es el siguiente: "***mysql <-h IP> <-P puerto> <-u usuario> [-p]***". "-p" indica que tenemos que introducir una contraseña para entrar.

Una vez hayamos conectado con la base de datos, podemos utilizar el comando "***SHOW DATABASES;***" para ver las bases de datos creadas. Podemos ver en la imagen que la base de datos "prueba" se ha creado.



3. Actividad 3: gestión de imágenes

3.1. Descargar imagen Ubuntu:20.04

Para descargar la imagen, utilizaremos el comando "***docker pull ubuntu:20.04***", y una vez se haya completado la descarga, para comprobar que está en nuestro sistema, utilizaremos el comando "***docker images*****".**

```
root@usuario:/home/usuario# docker pull ubuntu:20.04
20.04: Pulling from library/ubuntu
d9802f032d67: Pull complete
Digest: sha256:8e5c4f0285ecbb4ead070431d29b576a530d3166df73ec44affc1cd27555141b
Status: Downloaded newer image for ubuntu:20.04
docker.io/library/ubuntu:20.04
root@usuario:/home/usuario# docker images
REPOSITORY        TAG           IMAGE ID       CREATED         SIZE
jenkins/jenkins   lts           e699ac6c40dc   5 weeks ago     467MB
mariadb           latest        6722945a6940   5 weeks ago     407MB
ubuntu            20.04         6013ae1a63c2   2 months ago    72.8MB
httpd             latest        4ce47c750a58   5 months ago    147MB
mysql             5.7           5107333e08a8   12 months ago   501MB
ubuntu            18.04         f9a80a55f492   19 months ago   63.2MB
debian            9             662c05203bab   2 years ago     101MB
php               7.3-apache    35da9118b3c0   2 years ago     451MB
centos            8             5d0da3dc9764   3 years ago     231MB
tomcat            9.0.39-jdk11  2703bbe9e9d4   4 years ago     648MB
root@usuario:/home/usuario#
```

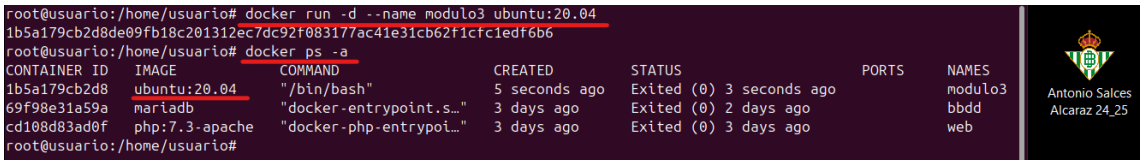
3.2. Volcar la información de la nueva imagen en un archivo

Para ver la información de la nueva imagen y volcarla en un archivo, utilizaremos el comando "***docker inspect <imagen> > <ruta>***". Para ver la información guardada en el archivo, utilizamos el comando "***cat <ruta>***".

```
root@usuario:/home/usuario# docker inspect ubuntu:20.04 > /home/usuario/Escritorio/info.txt
root@usuario:/home/usuario# cat /home/usuario/Escritorio/info.txt
[
    {
        "Id": "sha256:6013ae1a63c2ee58a8949f03c6366a3ef6a2f386a7db27d86de2de965e9f450b",
        "RepoTags": [
            "ubuntu:20.04"
        ],
        "RepoDigests": [
            "ubuntu@sha256:8e5c4f0285ecbb4ead070431d29b576a530d3166df73ec44affc1cd27555141b
"
        ],
        "Parent": "",
        "Comment": "",
        "Created": "2024-10-11T03:38:27.357079367Z",
        "DockerVersion": "24.0.7",
        "Author": "",
        "Config": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
```

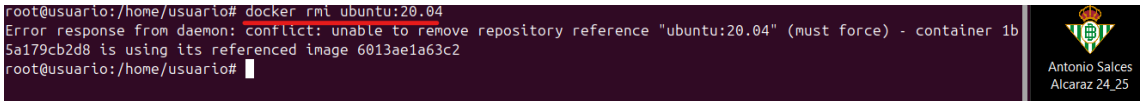
3.3. Crear contenedor con Ubuntu:20.04 llamada “modulo3”

Utilizaremos el comando “***docker run -d –name modulo3 ubuntu:20.04***” para crear el contenedor con los parámetros especificados. Con el comando “***docker ps -a***” veremos el contenedor.



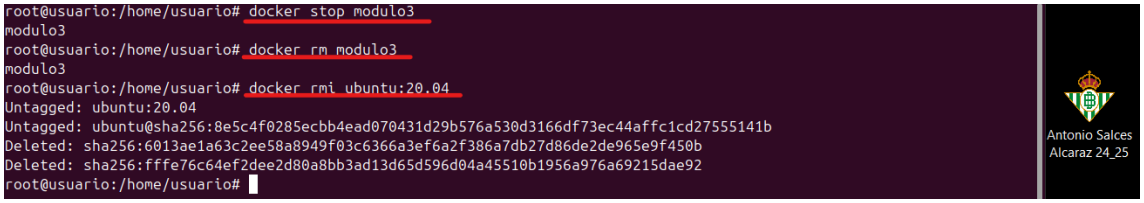
3.4. Borrar contenedor con Ubuntu:20.04

Para borrar una imagen, utilizaremos el comando “***docker rmi <imagen>***”. Como podemos ver, no podemos borrar la imagen. Esto es debido a que la imagen se está utilizando en un contenedor, por lo que no podremos borrarla mientras el contenedor exista.



3.5. Hacer acciones necesarias para borrar el contenedor Ubuntu:20.04

Para borrar la imagen, en primer lugar debemos parar el contenedor con “***docker stop <contenedor>***”, luego borrar con “***docker rm <contenedor>***”, y por ultimo borrar la imagen con “***docker rmi <imagen>***”.



4. Actividad 4: operaciones con volúmenes

4.1. Crear volúmenes

Con el comando "***docker volume create <nombre>***" podremos crear los volúmenes con el nombre que queramos. Luego, utilizaremos "***docker volume ls***" para ver los volúmenes creados.

```
root@usuario:/home/usuario# docker volume create volumen_datos
volumen_datos
root@usuario:/home/usuario# docker volume create volumen_web
volumen_web
root@usuario:/home/usuario# docker volume ls
DRIVER    VOLUME NAME
local     e6893b55a515fce92167af832f8e9fe5ca29cb3911b67d258916613f6732f019
local     volumen_datos
local     volumen_web
root@usuario:/home/usuario#
```


4.2. Instanciar contenedores en los volúmenes

Con el comando "***docker run [-d] [--name <nombre>] [-v <volumen>:][ruta] <imagen>***" podremos crear un contenedor y elegir en que volumen y que ruta queremos asignarle. Con "***docker ps***" veremos que el contenedor está activo

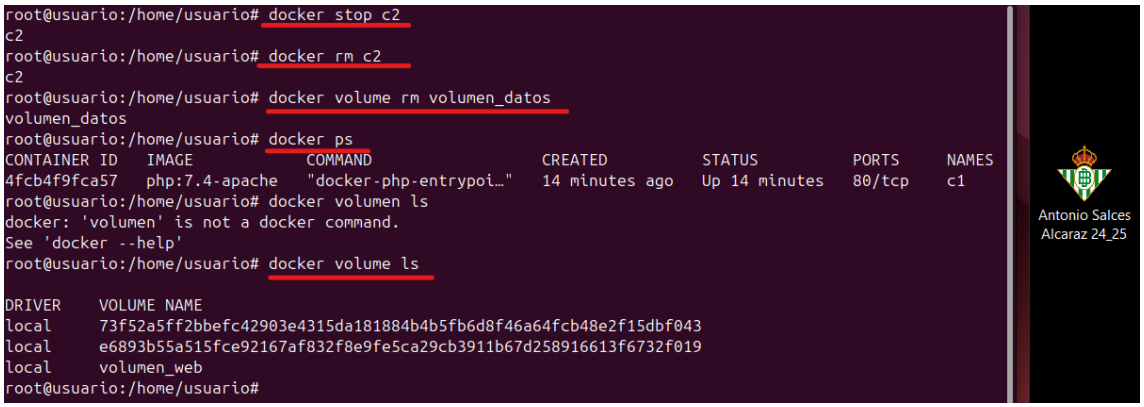
```
root@usuario:/home/usuario# docker run -d --name c1 -v volumen_web:/var/www/html php:7.4-apache
4fcb4f9fca57a0fa960701bd6405a1a17efb3e3be00ac8319eb777fb7bcd3920
root@usuario:/home/usuario# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
4fcb4f9fca57   php:7.4-apache   "docker-php-entrypoi…"   9 seconds ago   Up 7 seconds   80/tcp    c1
root@usuario:/home/usuario#
```


Con el comando "***docker run [-d] [--name <nombre>] >] [-v <volumen>:][ruta] [-e<MYSQL_ROOT_PASSWORD>=<contraseña>] <imagen>***" podremos crear un contenedor, elegir el volumen y la ruta donde se aloja y, en este caso (se trata de MariaDB) podemos asignar también una contraseña al usuario "root".

```
root@usuario:/home/usuario# docker run -d --name c2 -v volumen_datos:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=admin mariadb
fea2b1eb1d73e6e259b0295058b05b4f7af654c00d07e711676af81e869a4192
root@usuario:/home/usuario# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS      NAMES
fea2b1eb1d73   mariadb        "docker-entrypoint.s…"   15 seconds ago   Up 14 seconds   3306/tcp   c2
4fcb4f9fca57   php:7.4-apache   "docker-php-entrypoi…"   4 minutes ago    Up 4 minutes    80/tcp     c1
root@usuario:/home/usuario#
```

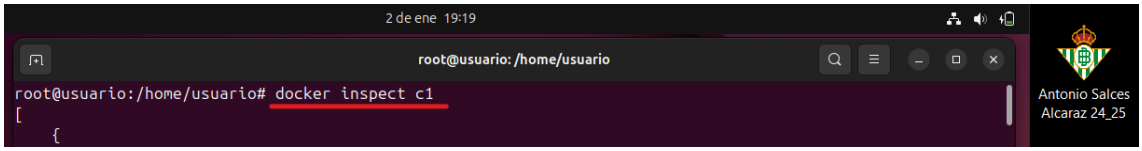
4.3. Parar y borrar contenedor “c2” y volumen “volumen_datos”

En primer lugar, paramos el contenedor “c2” con “***docker stop <nombre o ID>***”, luego, lo borramos con “***docker rm <nombre o ID***”, y a continuación borramos el volumen con “***docker volume rm <nombre>***”. Para acabar, comprobaremos que se han eliminado con “***docker ps***” y “***docker volume ls***” y que ya solo queda el contenedor “c1”.



4.4. Comprobar que “c1” está montado sobre “volumen_web”

Debemos utilizar la orden “***docker inspect <nombre o ID>***”.



Ahora bajamos hasta la parte de “Mounts” y comprobamos en que volumen está. En este caso, está sobre “volumen_web”.