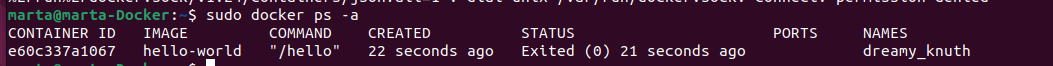
**UD11 - Activities W1**

*Docker instalado en Ubuntu Jammy. No puedo ejecutar los comandos si no uso el “sudo” delante.*

**Exercise 0**

“sudo docker run hello-world”

“docker ps -a”



**Exercise 1**

Crear los archivos index en 3 carpetas: Carpeta1, Carpeta2, Carpeta3.

Crear los archivos con “touch index.html”

- “sudo nano index.html”

Luego, para servir cada archivo en un servidor httpd, desde la carpeta donde se encuentra cada archivo, lanzar el siguiente comando:

“docker run --name pagina1 -d -p 8080:80 -v “$(pwd)”:/usr/local/apache2/htdocs/ httpd”

“docker run --name pagina2 -d -p 8081:80 -v “$(pwd)”:/usr/local/apache2/htdocs/ httpd”

“docker run --name pagina3 -d -p 8082:80 -v “$(pwd)”:/usr/local/apache2/htdocs/ httpd”

\*Cada commando deberá ser ejecutado desde la carpeta correspondiente (1, 2 o 3), cambiando el nombre del contenedor y el puerto de nuestro host.

Si entramos en nuestro navegador y escribimos “localhost:8080” o alguno de los otros puertos, deberíamos ver el contenido del archivo html.

A screenshot of a computer

Description automatically generated

**Exercise 2**

Para crear el contenedor:

*docker run -it -d --name phytonCont python:2.7.18*

También podría abrir directamente el Shell para que no se cierre automáticamente el contenedor al ejecutarlo:

*docker run -it --name pythonCont python:2.7.18 bash (o /bin/sh)*

Con el primer comando, luego para entrar el contenedor y ejecutar un Shell, y por cada Shell que queramos abrir en el contenedor:

*docker exec -it pythonCont bash*

Para instalar un editor de texto:

*apt update*

*apt install nano/vim*

Para crear el script

*touch script*

*nano script*

Editar el script y guardar. Para ejecutarlo, habría que escribir *./script* (no me deja ejecutarlo).

Para parar y eliminar el contenedor:

*docker stop pythonCont*

*docker rm phytonCont*

Al destruir el contenedor y volverlo a crear, ni el script está creado, ni el editor de texto está instalado.

**Exercise 3**

Parte 1

Para crear el contenedor mysql:

*docker run --rm --name database -env MYSQL\_ROOT\_PASSWORD=mysql -p 3037:3036 mysql*

Para abrir un shell en mysql:

*docker exec -it database bash*

Luego Podemos conectarnos a mysql escribiendo:

*mysql --user root --password=mysql*

*show databases;*

*create database;*

*etc*

Para parar y eliminar el contenedor:

*docker stop database*

*docker rm database*

Parte 2

* Para crear un volumen:

*docker volumen create volumen\_database*

* Ejecutar el servidor y montarlo en el volumen

*docker run --rm --name database-server --env MYSQL\_ROOT\_PASSWORD=password -v $(pwd):/var/lib/mysql -p 3037:3036 mysql*

* Crear una base de datos 🡪 repetir lo de la parte 1
* Destruye el contenedor y lanza uno nuevo. Para destruirlo:

*docker stop database-server*

*docker rm database-server*

Para destruir el volumen:

*docker volumen remove volume\_database*

Parte 3

* Crear un Directorio vacio

*mkdir database*

* Montar el servidor de la base de datos

*docker run --rm --name database-server --env MYSQL\_ROOT\_PASSWORD=password -p 3036:3036 --mount type=bind, src=/database, dst=/var/lib/mysql mysql*

* Para el resto de pasos, repetir los pasos de las partes anteriores.
* Quien es el creador de los archivos creados en tu ordenador? Por que?

*El propietario es el usuario 999, porque es el ID del usuario ejecutando el servidor de la base de datos en el contenedor.*

**Exercise 4**

*docker pull ibmjava:11-jdk*

*docker run --rm --it --name javaContainer ibmjava:11-jdk*

Se abre el shell automaticamente 🡪 *javac* 🡪 sí existe javac.

Al salir del Shell, el contenedor se elimina automáticamente.

Para eliminar la imagen:

*docker rmi ibmjava:11-jdk*

**Exercise 5**

Para crear la imagen:

Crear un archivo “Dockerfile” e incluir el siguiente contenido:

*FROM ubuntu:latest*

*RUN apt update && apt install -y sl 🡪* el **-y** es para que no pregunte cosas al usuario, cuando hay que instalar algún paquete o darle a Y para continuar.

*ENTRYPOINT [“/usr/games/sl”]*

Para construir la imagen:

*docker build -t exercise5Image .*

Para crear un contenedor a partir de la imagen:

*docker run --rm -it exercise5Image*

**Exercise 6**

Parte 1

Para crear la imagen, creamos el dockerfile:

*touch Dockerfile*

Editamos el contenido:

*FROM python:3*

*COPY reverse.py /usr/bin*

Para crear la imagen:

*docker build -t exercise6Imagen .*

Parte 2

Modificamos el Dockerfile:

*FROM python:3*

*COPY reverse.py /usr/bin*

*ENTRYPOINT python /usr/bin/reverse.py “!yob ananab, oY”* 🡪 hay que ejecutar el script, y luego añadir el texto

Parte 3

Para imprimir el parámetro pasado en el comando, añadir al Dockerfile:

*ENTRYPOINT [“python”,”/usr/bin/reverse.py”]*

**Exercise 7**

Parte 1

Crear el Dockerfile y añadir el siguiente contenido:

*FROM rbuntu:latest*

*RUN apt update && apt install -y iproute2 iputils-ping net-tools*

Construir la imagen:

*docker build -t ubuntu-net .*

Parte 2

Crear y ejecutar 4 contenedores basados en la imagen anterior. Como los contenedores deben ser capaces de comunicarse por una network, hay que crear las networks:

*docker network create networkAB*

*docker network create network CD*

*docker run --rm --name container\_a --network networkAB -it ubuntu-net*

*docker run --rm --name container\_b --network networkAB -it ubuntu-net*

*docker run --rm --name container\_c --network networkCD -it ubuntu-net*

*docker run --rm --name container\_d --network networkCD -it ubuntu-net*

Para eliminar las networks:

*docker network remove AB CD*