SOLUCIONES PRACTICAS DOCKER CONTAINERS

1. **Contenedores (crear, visualizar, filtrar ...)**
    a. Crear el contenedor hello-world
        i. docker run hello-world
    b. Comprobar que el contenedor no está activo
        i. docker ps
    c. Comprobar que el contenedor se ha creado, pero está parado
        i. docker ps -a
    d. Comprobar que existe la imagen
        i. docker images
    e. Creamos un nuevo contenedor a partir de la imagen y vemos que existe.
        i. docker run hello-world
    f. Crear un nuevo contenedor basado en otra imagen de Docker Hub denominada busybox, que es un contenedor con un cierto número de utilidades
        i. docker run busybox
    g. Comprobar con el comando ps el contenedor creado (también la imagen)
        i. docker ps
    h. Visualizar las ids de los contenedores
        i. docker ps -aq
    i. Ver los dos últimos contenedores lanzados.
        i. docker ps -a -n 2
    j. Obtener el espacio utilizado por los contenedores creados
        i. docker ps -a -s
    k. Utilizar la opción -f para filtrar algún dato concreto, por nombre o contenido de texto.
        i. docker ps -a -f name=un_nombre
    l. Que tengan una y en el nombre.
        i. docker ps -a -f name=y

2. **Contenedores interactivos:**
    a. Crear un contenedor interactivo de la imagen de **Fedora**
        i. docker run -it fedora bash
        ii. Al final se pasa la imagen y el comando que vamos a lanzar, bash abre una sesión con la consola o interprete de comandos.
    b. Dentro del contenedor ver los datos del S.O. Comando: **uname -a**
        i. Dentro de la sesión con el contenedor: uname -a
    c. Después de crearlo visualizar los contenedores y las imágenes.
        i. docker ps
        ii. docker images
    d. Hacer un upgrade del sistema: **yum upgrade**
        i. Teclear ese comando dentro del contenedor.
        ii. Nos avisará de los paquetes que se van a actualizar
    e. **wget** no se encuentra el comando, pero se puede instalar con:
        i. yum install wget
        ii. Instala el comando wget
    f. Salir del contenedor: tecleando **exit**
    g. Volver arrancarlo, pero utilizando las 4 primeras letras del id
        i. docker start id_cont
        ii. docker start -i id_cont → Modo interactivo
    h. Pararlo utilizando las 4 primeras letras del id
        i. docker stop id_cont

3. **Contenedores en background**
    a. Crear un contenedor en background del servidor apache con el nombre apache1, la imagen es: **httpd (buscarlo primero en Docker hub).**
        i. Lo podemos buscar con: docker search httpd
        ii. docker run -d --name apache1 httpd
        iii. Probar si funciona: http://localhost

iv. Fallará por el mapeo de puertos, hay que añadir la opción: -p 80:80
v. Borrarlo antes con: docker rm -f id_cont
vi. Luego al mapear los puertos debería de funcionar correctamente desde el navegador.

b. Consultar el contenedor, comprobar si está o no arrancado. Verlo también con la opción -a
   i. docker ps
   ii. docker ps -a

c. Crear otro contenedor basado en un S.O. ya sea fedora o Ubuntu con la opción -d y comprobar si se queda o no arrancado.
   i. docker run --name ubuntu1 -d Ubuntu

d. Buscarle con **docker ps -a**, porque no se habrá arrancado.
   i. El contenedor de Ubuntu se para a diferencia del contenedor de apache si que tiene que hacer cosas. No sólo es necesario poder la opción -d si no, que la imagen tiene que estar preparada para trabajar en modo background.

e. Probar con otro contenedor de un S.O. a lanzarlo en modo background e interactivo.
   i. Si lo arrancamos igual que antes, pero ponemos la opción -it entramos en modo interactivo, de esta forma si se mantiene arrancado.

4. Exec y borrados

a. Arrancar un contenedor en modo background basado en la imagen NGINX. Hacer las mismas pruebas que con apache. Colocarlo en el puerto 82.
   i. Colocarlo en el puerto -p 82:80, pero va del 82 al 80 del contenedor. Aquí se ve bien el aislamiento de los contenedores entre sí, tanto apache como nginx corren dentro de su contenedor por el puerto 80, pero de nuestra máquina

salimos por el puerto 80 para apache y por el puerto 82 para nginx.

ii. docker run -d --name nginx1 -p 82:80 nginx

b. Comprobar que está funcionando
   i. docker ps
c. Lanzar el comando ls / con el comando exec de Docker
   i. docker exec nginx1 ls /
d. Entrar en la Shell indicando que es interactivo.
   i. docker exec -it nginx1 bash
e. Salir y parar el contenedor
   i. exit / docker stop nginx1/ probar a conectar con el servidor desde el navegador.
f. Comprobar que existe, pero está parado.
   i. docker ps -a
g. Borrar el contenedor
   i. docker rm nginx1
h. Consultar las imágenes que tenemos
   i. docker images
   ii. Si borramos la imagen de nginx y está parado la borra
   iii. Si borramos la imagen de apache, y el contenedor de la imagen de apache está arrancado no nos deja.
   iv. Si forzamos con docker rmi -f id_imagen. Tampoco nos deja borrar la imagen, el contenedor continua arrancado, pero la imagen se queda none:none, hay que parar el contenedor, borrarlo y luego borrar la imagen. O probar a borrar y hacer docker image prune
i. Intentar borrar una imagen que tenga contenedores. No nos tiene que dejar. Forzar como -f
j. docker ps -qa → obtener los ids de los contenedores.
   i. docker rm $(docker ps -qa) La salida del comando interno se pasa al comando.

5. **Logs, stats, kill**
    a. Arrancar en background un contenedor con nginx, ponerle nombre: nginx1.
        i. docker run -d --name nginx1 -p 80:80 nginx
    b. Comprobar que funciona
        i. docker ps
    c. Comprobar los logs del contenedor
        i. docker logs nginx1
    d. Conectarnos con una bash al contenedor
        i. docker exec -it nginx1 bash
    e. Instalar el comando wget en el contenedor
        i. apt-get update
        ii. apt-get install wget
    f. Lanzar el comando “wget localhost” (nginx)
    g. Comprobar los logs de nginx1
        i. docker logs nginx1
        ii. Deberíamos de ver la petición wget realizada antes
    h. Comprobar con top los procesos que están dentro de nginx1
        i. docker top nginx1
    i. Lanzar un comando sleep 500 desde la bash y continuar comprobando con top los procesos lanzados en nginx1
        i. docker exec nginx1 sleep 500
    j. Obtener las estadísticas de uso de nginx1
        i. docker stats nginx1
    k. Probar a matar con kill a nginx1
        i. docker kill nginx1
        ii. Lo para, pero no lo borra

6. **Inspeccionar contenedores.**
    a. Descargar la imagen de node
        i. docker pull node
    b. Comprobar que tenemos la imagen
        i. docker images | grep “node”
    c. Inspect sobre la imagen
        i. docker image inspect node
    d. Inspeccionar la imagen pero lanzar el resultado a un fichero llamado: node.json
        i. docker image inspect node > node.json
    e. Crear un contenedor con esa imagen, llamado node1
        i. docker run -d -it --name node1 node
    f. Inspeccionar el nuevo contenedor creado
        i. docker inspect node1
    g. Utilizando el comando grep de Linux mostrar la dirección IP del contenedor: IPAddress
        i. docker inspect node1 | grep “IPAddress”
    h. Lo mismo para el tamaño de la memoria compartida: ShmSize
        i. docker inspect node1 | grep “ShmSize”

7. **Crear contenedores de mysql, postgresql, etc.**
    a. Probar a crear los contenedores sin variables y probar a mirar los errores.
        i. docker run -d –name mysql1 mysql:8.0
        ii. docker logs mysql1
    b. Mirar las variables de entorno de mysql, crear user, base de datos, password, etc.
    c. Conectar con MySQL: imagen → mysql:8.0
        i. Mirar en Docker Hub
            1. MYSQL_ROOT_PASSWORD
            2. MYSQL_DATABASE
            3. MYSQL_USER

4. MYSQL_PASSWORD
5. docker run -d --name mysql1
-e MYSQL_ROOT_PASSWORD=root
-e MYSQL_DATABASE=prueba1
-e MYSQL_USER=antonio
-e MYSQL_PASSWORD=antonio

ii. docker exec -it mysql1 bash

1. Conectar con mysql:
    a. mysql -u antonio -p
    b. Teclear password
    c. Probar comandos de MySQL: show databases; use prueba1; show tables; etc.

d. **PostGreSQL: imagen → postgres**

i. Mirar en Docker Hub, variables de entorno:
1. POSTGRES_PASSWORD
2. POSTGRES_USER
3. POSTGRES_DB

ii. Crear el contenedor y luego conectar con bash
1. docker run -d --name postgre1 postgres
2. docker logs postgres1
3. docker run -d --name postgre1
-e POSTGRES_PASSWORD=antonio
-e POSTGRES_USER=antonio
-e POSTGRES_db=prueba1
postgres
4. docker exec -it postgres1 bash
5.

iii. psql -U <<USER>> <<DATABASE>>

iv. Dentro → \l

v. \q