PRACTICAS DOCKER CONTAINERS

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

2. **Contenedores interactivos:**
    a. Crear un contenedor interactivo de la imagen de **Fedora**
    b. Dentro del contenedor ver los datos del S.O. Comando: **uname -a**
    c. Después de crearlo visualizar los contenedores y las imágenes.
    d. Hacer un upgrade del sistema: **yum upgrade**
    e. **wget** no se encuentra el comando, pero se puede instalar con:
        i. **yum install wget**
    f. Salir del contenedor: tecleando **exit**
    g. Volver arrancarlo, pero utilizando las 4 primeras letras del id
    h. Pararlo utilizando las 4 primeras letras del id

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. Probar desde un navegador: http://localhost
        ii. Fallará, hay que mapear los puertos.
        iii. Borrar el contenedor anterior de apache con la opción -f
        iv. Volverlo a crear con el mapeo de puertos: -p 80:80
    b. Consultar el contenedor, comprobar si está o no arrancado. Verlo también con la opción -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.
    d. Buscarle con **docker ps -a**, porque no se habrá arrancado.
    e. Probar con otro contenedor de un S.O. a lanzarlo en modo background e interactivo.

4. **Exec y borrados**
    a. Arrancar un contenedor en modo background basado en la imagen NGINX. Hacer las mismas pruebas que con apache.
    b. Comprobar que está funcionando
    c. Lanzar el comando **ls /** con el comando exec de Docker
    d. Entrar en la Shell indicando que es interactivo.
    e. Salir y parar el contenedor
    f. Comprobar que existe, pero está parado.
    g. Borrar el contenedor
    h. Consultar las imágenes que tenemos
    i. Intentar borrar una imagen que tenga contenedores. No nos tiene que dejar. Forzar como -f, por ejemplo, borrar la imagen de Apache (httpd) y con el contenedor de apache arrancado. Comprobar si se ha borrado o no la imagen.
    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.
    b. Comprobar que funciona
    c. Comprobar los logs del contenedor
    d. Conectarnos con una bash al contenedor
    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
    h. Comprobar con top los procesos que están dentro de nginx1
    i. Lanzar un comando sleep 500 desde la bash y continuar comprobando con top los procesos lanzados en nginx1
    j. Obtener las estadísticas de uso de nginx1
    k. Probar a matar con kill a nginx1

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

7. **Crear contenedores de mysql, postgresql, etc.**
    a. Probar a crear los contenedores sin variables y probar a mirar los errores.
    b. Mirar las variables de entorno de mysql, crear user, base de datos, password, etc.
    c. MySQL: imagen mysql:8.0
        i. Mirar en Docker Hub
            1. MYSQL_ROOT_PASSWORD
            2. MYSQL_DATABASE
            3. MYSQL_USER
            4. MYSQL_PASSWORD
        ii. Conectar con mysql → mysql -u USER -p
        iii. Teclear el password, probar comandos: show databases; use data_base, 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
        iii. psql -U <<USER>> <<DATABASE>>
        iv. Dentro → \l
        v. \q