Boletín de ejercicios Tema 3

1. Instalación de la imagen Hello-World en Docker

Los pasos que deberemos seguir son:

Comprobamos que inicialmente no hay ningún contenedor creado (la opción -a hace que se muestren también los contenedores detenidos, sin ella se muestran sólo los contenedores que estén en marcha):

```
docker ps -a
```

Comprobamos que inicialmente tampoco disponemos de ninguna imagen:

```
docker images
```

La respuesta, en forma de tabla, será la siguiente:

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
```

Docker crea los contenedores a partir de imágenes locales (ya descargadas), pero si al crear el contenedor no se dispone de la imagen local, Docker descarga la imagen de su repositorio.

La orden más simple para crear un contenedor es:

```
docker run IMAGEN
```

Cree un contenedor con la aplicación de ejemplo hello-world. La imagen de este contenedor se llama hello-world:

```
docker run hello-world
```

Como no tenemos todavía la imagen en nuestro ordenador, Docker descarga la imagen, crea el contenedor y lo pone en marcha. En este caso, la aplicación que contiene el contenedor hello-world simplemente escribe un mensaje de salida al arrancar e inmediatamente se detiene el contenedor. La respuesta será similar a esta:

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:4fe721ccc2e8dc7362278a29dc660d833570ec2682f4e4194f4ee23e415e1064
Status: Download newer image from hello-wolrd:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
...
```

Si listamos ahora las imágenes existentes ...

docker images

... se mostrará información de la imagen creada:

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              fce209e99eb9        4 weeks ago         1.84 kB
```

Si listamos ahora los contenedores existentes ...

docker ps -a

... se mostrará información del contenedor creado:

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
614B4c431ffa        hello-world         "/hello"            2 minutes ago       Exited (0) 2 minutes
ago                 hopeful_elbakyan
```

Cada contenedor tiene un identificador (ID) y un nombre distinto. Docker "bautiza" los contenedores con un nombre peculiar, compuesto de un adjetivo y un apellido.

Podemos crear tantos contenedores como queramos a partir de una imagen. Una vez la imagen está disponible localmente, Docker no necesita descargarla y el proceso de creación del contenedor es inmediato (aunque en el caso de hello-world la descarga es rápida, con imágenes más grandes la descarga inicial puede tardar un rato)

Normalmente se aconseja usar siempre la opción -d, que arranca el contenedor en segundo plano (detached) y permite seguir teniendo acceso a la shell (aunque con hello-world no es estrictamente necesario porque el contenedor hello-world se detiene automáticamente tras mostrar el mensaje).

Al crear el contenedor hello-world con la opción -d no se muestra el mensaje, simplemente muestra el identificador completo del contenedor.

```
docker run -d hello-world

1ae54736196021523a2b21c123fd671253e62150daccd882374
```

Si listamos los contenedores existentes ...

```
sudo docker ps -a
```

... se mostrarán los dos contenedores:

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
1ae547361960        hello-world         "/hello"            4 seconds ago       Exited (0) 3 seconds
ago                 distracted_banach
614B4c431ffa        hello-world         "/hello"            5 minutes ago       Exited (0) 5 minutes
ago                 hopeful_elbakyan
```

Los contenedores se pueden destruir mediante el comando rm, haciendo referencia a ellos mediante su nombre o su id. No es necesario indicar el id completo, basta con escribir los primeros carácteres (de manera que no haya ambigüedades). Borre los dos contenedores existentes:

```
docker rm 1ae

1ae

docker rm hopefukl_elbakyan

hopefukl_elbakyan
```

Comprobamos que ya no quedan contenedores:

```
docker ps -a
```

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
```

Podemos dar nombre a los contenedores al crearlos:

```
docker run -d --name=hola-1 hello-world
```

Al haber utilizado la opción -d únicamente se mostrará el ID completo del contenedor:

54e9827bd10ab2825e1b3e4d3bf7a8cbdf778b472359c655d72d9c09e753500a

Si listamos los contenedores existentes ...

docker ps -a

... se mostrará el contenedor con el nombre que hemos indicado:

```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
54e9827bd10a        hello-world         "/hello"            4 seconds ago       Exited (0) 3 seconds
ago                 hola-1
```

Si intentamos crear un segundo contenedor con un nombre ya utilizado ...

docker run -d --name=hola-1 hello-world

Docker nos avisará de que no es posible:

```
docker: Error response from daemon: Conflict. The container name "/hola-1" is already in use by con
tainer "54e9827bd10ab2825e1b3e4d3bf7a8cbdf778b472359c655d72d9c09e753500a". You have to remove (or re
name) that container to be able to reuse that name.
See 'docker run --help'.
```

2. Imagen apache

Pasos a seguir:

1. **Crearemos un contenedor que contenga un servidor Apache a partir de la imagen bitnami/apache**

   Para eso usaremos el comando:

   docker run -d -P --name=apache-1 bitnami/apache

   Con la opción -P hacemos que Docker asigne de forma aleatoria un puerto de la máquina virtual al puerto asignado a Apache en el contenedor.

En este caso, la imagen bitnami/apache asigna a Apache el puerto 8080 del contenedor para conexiones http y el puerto 8443 para conexiones https.

2. Mostrar la página inicial del contenedor

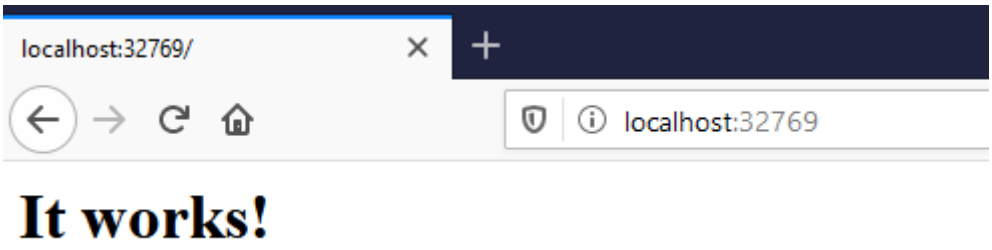
Consultamos el puerto del host utilizado por el contenedor:

docker ps -a

El resultado será:

```
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
      PORTS                                              NAMES
63476f0bf8d8        bitnami/apache      "/entrypoint.sh /run…"   41 seconds ago      Up 39 seconds
      0.0.0.0:32769->8080/tcp, 0.0.0.0:32768->8443/tcp   apache-1
```

Abrimos en el navegador la página inicial del contenedor y comprobamos que se muestra una página que dice "It works!".



3. Crearemos un contenedor nuevo donde modificaremos el fichero “index.html”

En este apartado vamos a modificar la página web inicial de Apache del contenedor Docker.

Hay que tener en cuenta que modificar el contenido de un contenedor tal y como vamos a hacer en este apartado sólo es aconsejable en un entorno de desarrollo, pero no es aconsejable en un entorno de producción porque va en contra de la "filosofía" de Docker. Los contenedores de Docker están pensados como objetos de "usar y tirar", es decir, para ser creados, destruidos y creados de nuevo tantas veces como sea necesario y en la cantidad que sea necesaria. En el apartado siguiente

realizaremos la misma tarea de una forma más conveniente, modificando no el contenedor sino la imagen a partir de la cual se crean los contenedores.

Pasos a seguir:

- Creamos un segundo contenedor que contenga un servidor Apache a partir de la imagen bitnami/apache

docker run -d -P --name=apache-2 bitnami/apache

- Consultamos el puerto del host utilizado por el contenedor:

docker ps –a

Se mostrarán los dos contenedores creados:

```
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
      PORTS                                                NAMES
3c879106e610        bitnami/apache      "/entrypoint.sh /run…"   50 seconds ago      Up 49 seconds
      0.0.0.0:32771->8080/tcp, 0.0.0.0:32770->8443/tcp   apache-2
63476f0bf8d8        bitnami/apache      "/entrypoint.sh /run…"   8 minutes ago       Up 8 minutes
      0.0.0.0:32769->8080/tcp, 0.0.0.0:32768->8443/tcp   apache-1
```

- Creamos la nueva página index.html:

notepad index.html

- Le pondremos el siguiente contenido:

<h1>Hola, Montecastelo!</h1>

- Guardamos el fichero.
- Entramos en la shell del contenedor para averiguar la ubicación de la página inicial:

docker exec -it apache-2 /bin/bash

- Se abrirá una shell en el contenedor.

```
I have no name!@3c879106e610:/app$
```

- El DocumentRoot de Apache está en el directorio /opt/bitnami/apache/htdocs, por lo que nos movemos hacia esa ruta:

```
I have no name!@3c879106e610:/app$ cd /opt/bitnami/apache/htdocs/
```

En ese directorio se encuentra el fichero index.html que queremos modificar. Usamos el siguiente comando para ver su contenido:

cat index.html

```
I have no name!@3c879106e610:/opt/bitnami/apache/htdocs$ cat index.html
<html><body><h1>It works!</h1></body></html>
```

- Salimos de la shell del contenedor:

exit

- Copiamos el fichero index.html en el contenedor:

docker cp index.html apache-2:/opt/bitnami/apache/htdocs/index.html

4. Mostraremos la página modificada

- Abrimos en el navegador la página inicial del contenedor y compruebe que se ha modificado.

5. Crearemos una imagen nueva que incluya la página modificada, de manera que cada vez que se cree el contenedor, la página inicial sea la modificada.

Si queremos cambiar la página inicial, la forma correcta de hacerlo en Docker es crear una nueva imagen que incluya la página modificada, de manera que cada vez que se cree el contenedor, la página inicial sea la modificada.

Las imágenes se crean a partir de Dockerfiles, ficheros que describen los elementos que forman la imagen. Los Dockerfiles pueden ser muy extensos. En este caso, se trata de un Dockerfile mínimo.

Seguiremos los pasos:

- Creamos un directorio que contendrá el Dockerfile

```
mkdir mi-apache
```

- Copiamos el fichero index.html creado anteriormente

```
move index.html mi-apache
```

- Entramos en el directorio y creamos un fichero Dockerfile

```
cd mi-apache
notepad Dockerfile
```

El contenido del Dockerfile será:

```
FROM bitnami/apache
COPY index.html /opt/bitnami/apache/htdocs/index.html
```

- Guardamos el fichero Dockerfile. **ASEGURAMOS QUE SE GUARDA COMO Dockerfile, SIN LA EXTENSIÓN .TXT.**
- Generamos la nueva imagen

```
docker build –t [NOMBRE_USUARIO]/mi-apache .
```

El último argumento (y el único imprescindible) es el nombre del archivo Dockerfile que tiene que utilizar para generar la imagen. Como en este caso se encuentra en el mismo directorio y tiene el nombre predeterminado Dockerfile, se puede escribir simplemente punto (.).

Para indicar el nombre de la imagen se debe añadir la opción -t. El nombre de la imagen debe seguir el patrón nombre-de-usuario/nombre-de-imagen. Si la imagen sólo se va a utilizar localmente, el nombre de usuario y de la imagen pueden ser cualquier palabra.

- Creamos un contenedor a partir de la nueva imagen:

```
docker run -d -P --name=mi-apache-1 [NOMBRE_USUARIO]/mi-apache
```

- Abrimos en el navegador la página inicial del contenedor y comprobamos que se ha modificado.

3. Directorios enlazados

En este ejercicio vamos a aprovechar que la imagen bitnami/apache está configurada para que el directorio htdocs habitual enlace al directorio /app.

Si al crear el contenedor enlazamos el directorio /app con un directorio de la máquina virtual, el contenedor servirá las páginas contenidas en el directorio de la máquina virtual.

Daremos los siguientes pasos:

- **Creamos un directorio que contendrá las páginas web:**

```
mkdir web
```

- **Creamos un contenedor Apache.**

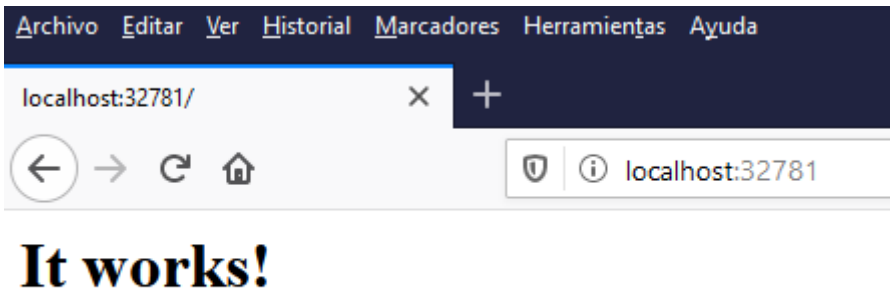
  La opción --mount permite crear el enlace entre el directorio de la máquina virtual y el contenedor. La opción tiene tres argumentos separados por comas, pero sin espacios:

  type=bind,source=ORIGEN-EN-MÁQUINA-VIRTUAL,target=DESTINO-EN-CONTENEDOR

  Ambos directorios deben existir previamente.

```
docker run -d -P --name=apache-bind-1 --mount type=bind,source=C:\Usuarios\[NOMBRE_USUARIO]\web,target=/app bitnami/apache
```

- **Abrimos en el navegador la página inicial del contenedor y comprobamos que se muestra el contenido de un fichero index.**



- **Creamos un fichero index.html:**

notepad web/index.html

Con el contenido siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Apache en Docker</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<h1>¡Hola, Montecastelo!</h1>
</body>
</html>
```

- **Actualizamos el navegador y comprobamos que se muestra la página recién creada.**

Gracias a los directorios enlazados, podemos trabajar en la máquina virtual con los ficheros del directorio /web haya o no haya contenedores en marcha y al crear un contenedor, podremos acceder al contenido del directorio a través del servidor web del contenedor

- **Detenemos y eliminamos el contenedor apache-bind-1:**

```
docker stop apache-bind-1
docker rm apache-bind-1
```

- **Creamos un nuevo contenedor que enlace al directorio /web:**

```
docker run -d -P --name=apache-bind-2 --mount type=bind,source=
C:\Usuarios\[NOMBRE_USUARIO]\web,target=/app bitnami/apache
```

- **Abrimos en el navegador la página inicial del nuevo contenedor y comprobamos que se muestra la página que está en la máquina virtual.**

La imagen bitnami/apache incluye un directorio /app que enlaza a /opt/bitnami/apache/htdocs como podemos comprobar entrando en el contenedor:

- **Entramos en la shell del contenedor:**

```
docker exec -it apache-bind-2 /bin/bash
```

- **Listamos los directorios en la raíz del contenedor:**

```
ls -al /
```

- **Comprobamos que aparece la carpeta que creamos y que tiene el fichero index.html**

4. Volúmenes

En este ejercicio utilizaremos un volumen en vez de un directorio enlazado. En este caso, enlazaremos el directorio /app con un volumen de Docker. directorio de la máquina virtual, el contenedor servirá las páginas contenidas en el directorio de la máquina virtual.

Daremos los siguientes pasos:

- **Creamos un contenedor Apache.**

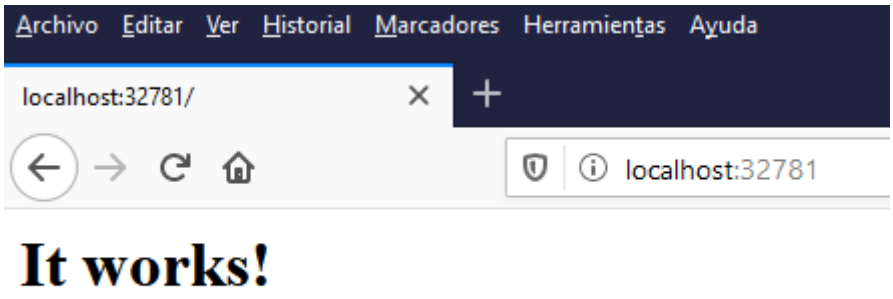
  La opción --mount permite crear el volumen . La opción tiene tres argumentos separados por comas pero sin espacios:

  type=volume,source=NOMBRE-DEL-VOLUMEN,target=DESTINO-EN-CONTENEDOR.

  El directorio de destino debe existir previamente.

```
docker run -d -P --name=apache-volume-1 --mount type=volume,source=vol-apache,target=/app bitnami/apache
```

- **Abrimos en el navegador la página inicial del contenedor y comprobamos que se muestra el contenido de un fichero index.**



- **Comprobamos que se ha creado un volumen con el nombre asignado al crear el contenedor:**

```
docker volumen ls
```

Nos dará este resultado:

```
DRIVER              VOLUME NAME
local               vol-apache
```

Los volúmenes son entidades independientes de los contenedores, pero para acceder al contenido del volumen hay que hacerlo a través contenedor, más exactamente a través del directorio indicado al crear el contenedor.

- **Entramos en el contenedor y listamos el directorio /app.**

```
docker exec -it apache-volume-1 /bin/bash
```

El directorio /app contiene únicamente el fichero index.html. Pero debemos tener en cuenta que la página web index.html se encuentra en el volumen, no en el contenedor.

- **Modificaremos esa página web. Salimos del contenedor:**

exit

Creamos un nuevo fichero index.html:

notepad index.html

Con el contenido siguiente:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Apache en Docker</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<h1>¡Hola desde un volumen, Montecastelo!</h1>
</body>
</html>
```

- **Copiamos el fichero index.html en el directorio /app del contenedor (aunque realmente se copiará en el volumen):**

docker cp index.html apache-volume-1:app/

- **Actualizamos el navegador y comprobamos que se muestra la página recién creada.**

- **Creamos ahora un nuevo contenedor que use el mismo volumen:**

docker run -d -P --name=apache-volume-2 --mount type=volume,source=vol-apache,target=/app bitnami/apache

- **Comprobamos que se ha creado el contenedor y consultamos el puerto asignado:**

```
docker ps
```

- **Abrimos en el navegador la página inicial del nuevo contenedor y comprobamos que este contenedor muestra también la página que ha copiado antes en el volumen.**

Los volúmenes son independientes de los contenedores, pero Docker tiene en cuenta qué volúmenes están siendo utilizados por un contenedor.

- **Si intentamos borrar el volumen anterior mientras los contenedores están en marcha, Docker muestra un mensaje de error que indica los contenedores afectados:**

```
docker volume rm vol-apache
```

- **Detenemos los contenedores:**

```
docker stop apache-volume-1
docker stop apache-volume-2
```

- **Si intentamos de nuevo borrar el volumen anterior ahora que los contenedores están detenidos, Docker sigue mostrando el mensaje de error que indica los contenedores afectados:**

```
docker volume rm vol-apache
```

- **Borramos los contenedores:**

```
docker rm apache-volume-1
```

```
docker rm apache-volume-2
```

- **Si intentamos de nuevo borrar el volumen anterior ahora que no hay contenedores que utilicen el volumen, Docker ahora sí que borrará el volumen:**

```
docker volume rm vol-apache
```

- **Comprobamos que el volumen ya no existe:**

```
docker volume ls
```

Se debe tener en cuenta que, al borrar un volumen, los datos que contenía el volumen se pierden para siempre, salvo que hubiéramos realizado una copia de seguridad.