Sistemas Operativos 2020 UEM

Docker LAB 2 : Persistencia de datos en Docker

En este lab vamos a crear un contenedor que incluya un servidor Apache en funcionamiento a partir de una imagen pública, modificar el contenido del contenedor y crear una nueva imagen para poder crear contenedores personalizados. Posteriormente veremos cómo se logra la persistencia de datos mediante directorios enlazados y volúmenes.

1.- Creación de un contenedor Apache

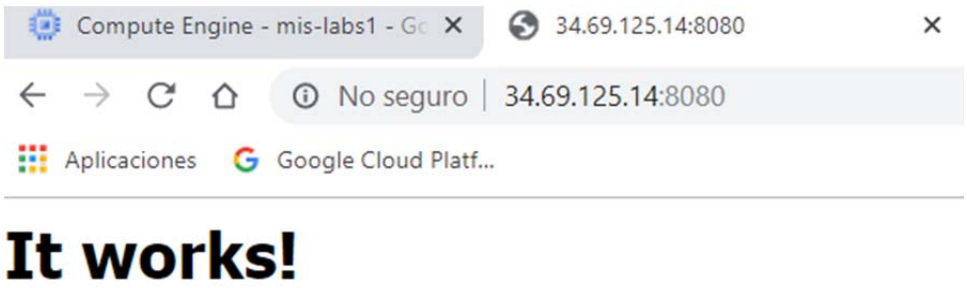
Crearemos un contenedor que contenga un servidor Apache a partir de la imagen bitnami/apache

La imagen bitnami/apache asigna a Apache (servidor web) el puerto 8080 del contenedor para conexiones http y el puerto 8443 para conexiones https. Los mapearemos a los puertos 8080 y 8443. Debemos comprobar si están abiertos en la red de la UEM y si no utilizar nuestros datos móviles.

¡! AL COPIAR LOS COMANDOS TENER ESPECIAL CUIDADO CON LOS CARACTERES - y -- ¡! Si el comando no es funciona reescribidlo manualmente

```
iegogachet@iserv-docker:~$  docker run -d –p 8080:8080 –p 8443:8443 --name=apache-1 bitnami/apache
```

SDebemos crear una regla firewall que permita tráfico por los puertos establecidos por docker, en el ejemplo anterior serían 8080 y 8443 . Abrimos el navegador usando la IP externa de la instancia de GCP y el puerto anteriormente mostrado, (en el ejemplo el 8080) y compruebamos que se muestra una página que con "It works!".



Modificar la página inicial del contenedor Apache

Tengamos 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.

Crearemos una nueva página index.html utilizando nano o bien un editor en Windows y luego subiéndolo a la instancia mediante la consola SSH.

```
<!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, mundo!</h1>
</body>
</html>
```

Entremos en la *shell* del contenedor para averiguar la ubicación de la página inicial:

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

Se abrirá una *shell* en el contenedor. Fíjese que nos lleva al directorio /app.

Para ver la página original index.html debemos ir al directorio /opt/bitnami/apache/htdocs (usar el comando cd)

```
I have no name!@c5041d12aabd:/opt/bitnami/apache/htdocs$
```

En ese directorio se encuentra el fichero index.html que queremos modificar, estando dentro del contenedor podemos mirar el contenido de index.html (en el directorio : /opt/bitnami/apache/htdocs)

```
$ cat index.html
<html><body><h1>It works!</h1></body></html>
```

Salimos de la shell del contenedor:

```
$ exit
```

Copiamos el nuevo fichero index.html en el contenedor:

```
$ docker cp index.html apache-1:/opt/bitnami/apache/htdocs/index.html
```

Abrimos el navegador y refrescamos la página inicial del contenedor comprobando que se ha modificado.

2.- Crear una nueva imagen

Si queremos cambiar la página inicial de froma permanente, 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.

1. Creeamos un directorio que contendrá el Dockerfile

```
$ mkdir mi-apache
```

2. Copie el fichero index.html creado anteriormente

```
$ mv index.html mi-apache
```

3. Entre en el directorio y cree un fichero Dockerfile

```
$ cd mi-apache

$ nano Dockerfile
```

El contenido del Dockerfile puede ser el siguiente:

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

Genere la nueva imagen. El último argumento 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 (.):

```
$ docker build -t mi-apache .
```

Dado que el anterior contenedor (apache-1) estará ejecutándose y utiliza los mismos puertos que el contenedor que vamos a crear lo detenemos y y borramos:

$ docker stop $(docker ps –aq)

$ docker rm $(docker ps –aq)

Creamos un contenedor a partir de la nueva imagen:

```
$ docker run -d –p 8080:8080 –p 8443:8443 --name=mi-apache-1 mi-apache
```

3.- Docker - Volúmenes

Docker simplifica enormemente la creación de contenedores, y eso lleva a tratar los contenedores como un elemento efímero, que se crea cuando se necesita y que no importa que se destruya puesto que puede ser reconstruido una y otra vez a partir de su imagen.

Pero si la aplicación o aplicaciones incluidas en el contenedor generan datos y esos datos se guardan en el propio contenedor, en el momento en que se destruyera el contenedor perderíamos esos datos. Para conseguir la persistencia de los datos, se pueden emplear dos técnicas:

- Los directorios enlazados, en la que la información se guarda fuera de Docker, en la máquina *host* (en nuestro caso, en la máquina virtual de Ubuntu)
- Los volúmenes, en la que la información se guarda mediante Docker, pero en unos elementos llamados *volúmenes*, independientes de las imágenes y los contenedores

Los volúmenes son la mejor solución cuando la información es generada por el propio contenedor y los directorios enlazados pueden ser más adecuados cuando la información no es generada por ningún contenedor.

3.1 Directorios enlazados (*bind*)

Docker permite asociar directorios del contenedor a directorios de la máquina *host* (en nuestro caso, de la máquina virtual de Ubuntu). Es decir, que cuando el contenedor lea o escriba en su directorio, donde leerá o escribirá será en el directorio de la máquina virtual.

Si el directorio enlazado es el directorio en el que la aplicación guarda los datos generados por la propia aplicación, de esta manera conseguimos que los datos estén realmente fuera del contenedor. Eso significa que podemos conservar los datos aunque se destruya el contenedor, reutilizarlos con otro contenedor, etc.

En este ejemplo, vamos a volver a aprovechar el hecho que la imagen bitnami/apache está configurada para que el directorio htdocs habitual se encuentra en **/opt/bitnami/apache/htdocs**.

Si al crear el contenedor enlazamos el directorio /opt/bitnami/apache/htdocs. con un directorio de la máquina virtual (instancia), el contenedor servirá las páginas contenidas en el directorio de la máquina virtual.

En la máquina virtual, ejecute los comandos siguientes en un terminal:

Cree un directorio que contendrá las páginas web:

```
$ mkdir /home/TU-USUARIO/web
```

Creemos un contenedor Apache nuevo a partir de la imagen bitnami/apache con un directorio enlazado.

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.

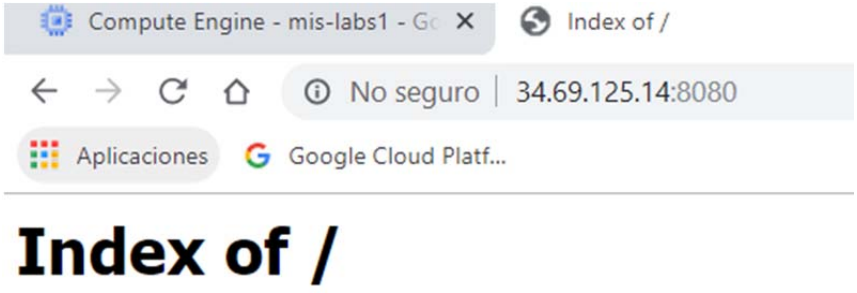
Parar y borrar los contenedores que se estén jecutando

$ docker stop $(docker ps –aq)

$ docker rm $(docker ps –aq)

$ docker run -d -p 8080:8080 -p 8443:8443 --name=apache-2 --mount type=bind,source=/home/TU_DIRECTORIO/web,target=/opt/bitnami/apache/htdocs bitnami/apache

Abra en el navegador la página inicial del contenedor y compruebe que se muestra el contenido de un directorio vacío.



Cree un fichero index.html:

- `sudo nano /home/TU-USUARIO/web/index.html`

```
<!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, amigo!</h1>
</body>
</html>
```

Actualice el navegador y compruebe 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

Detenga y elimine el contenedor apache-2

```
$ docker stop apache-2
$ docker rm apache-2
```

3.2 Volúmenes (*volume*)

En vez de guardar los datos persistentes en la máquina *host*, Docker dispone de unos elementos llamados volúmenes que podemos asociar también a directorios del contenedor, de manera que cuando el contenedor lea o escriba en su directorio, donde leerá o escribirá será en el volumen.

Los volúmenes son independientes de los contenedores, por lo que también podemos conservar los datos aunque se destruya el contenedor, reutilizarlos con otro contenedor, etc. La ventaja frente a los directorios enlazados es que pueden ser gestionados por Docker. Otro detalle importante es que **el *acceso al contenido de los volúmenes*** sólo se puede hacer a través de algún contenedor que utilice el volumen.

Vamos a repetir un ejemplo similar al anterior, pero utilizando un volumen en vez de un directorio enlazado. En este caso, enlazaremos el directorio /opt/bitnami/apache/htdocs con un volumen de Docker..

En la máquina virtual, ejecute los comandos siguientes en un terminal:

1. Cree 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 8080:8080 –p 8443:8443 --name=apache-3 --mount
type=volume,source=vol-apache,target=/opt/bitnami/apache/htdocs
bitnami/apache
```

Abra en el navegador la página inicial del contenedor y compruebe que se muestra la página inicial habitual de esta imagen.

Compruebe que se ha creado un volumen con el nombre asignado al crear el contenedor:

```
$ docker volume ls
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 del contenedor, más exactamente a través del directorio indicado al crear el contenedor.

Entre en el contenedor y liste el directorio /opt/bitnami/apache/htdocs. (utilice el comando "cd" para navegar sobre el sistema de ficheros)

- `sudo docker exec -it apache-3 /bin/bash`

```
I have no name!@3b1bcc5a67f3:$ ls /opt/bitnami/apache/htdocs
index.html
```

El directorio /opt/bitnami/apache/htdocs contiene únicamente el fichero index.html. Pero tenga en cuenta que la página web index.html se encuentra en el volumen, no en el contenedor.

Modifique esa página web. Para ello, salga del contenedor`:

```
I have no name!@3b1bcc5a67f3:$ exit
```

y cree un nuevo fichero index.html:

```
$ sudo nano index.html
```

```
<!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 de nuevo, amigo!</h1>
</body>
</html>
```

Copie el fichero index.html en el directorio /opt/bitnami/apache/htdocs del contenedor (aunque realmente se copiará en el volumen):

```
$ docker cp index.html apache-3:/opt/bitnami/apache/htdocs
```

- Actualice el navegador y compruebe que se muestra la página recién creada.

Detenga los contenedores que se estén ejecutando

$ docker stop $(docker ps –aq)
$ docker rm $(docker ps –aq)

Cree ahora un nuevo contenedor que use el mismo volumen:

```
$ sudo docker run -d –p 8080:8080 –p 8443:8443 --name=apache-4 --mount
type=volume,source=vol-apache,target=/opt/bitnami/apache/htdocs
bitnami/apache
```

Abra en el navegador la página inicial del nuevo contenedor y compruebe 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 intenta borrar el volumen del ejemplo anterior mientras los contenedores están en marcha, Docker muestra un mensaje de error que indica los contenedores afectados:

```
$ docker volume rm vol-apache
Error response from daemon: remove vol-apache: volume is in use -
[a6c8a30f7b1dc7a4ef165046daff226ee
1d6a69573269ca24d57b5b4b6802881,
3b1bcc5a67f38853810972b1da8a67148fad78c6cd6f22b2c823d141be59c81c]
```

Detenga todos los contenedores:

$ docker stop $(docker ps –aq)

Si intenta de nuevo borrar el volumen del ejemplo anterior ahora que los contenedores están detenidos, Docker sigue mostrando el mensaje de error que indica los contenedores afectados:

```
$ docker volume rm vol-apache
Error response from daemon: remove vol-apache: volume is in use -
[a6c8a30f7b1dc7a4ef165046daff226ee
1d6a69573269ca24d57b5b4b6802881,
3b1bcc5a67f38853810972b1da8a67148fad78c6cd6f22b2c823d141be59c81c]
```

Eliminar los contenedores:

$ docker rm $(docker ps –aq)

Si intenta de nuevo borrar el volumen del ejemplo anterior ahora que no hay contenedores que utilicen el volumen, Docker ahora sí que borrará el volumen:

```
$ sudo docker volume rm vol-apache
```

Compruebe que el volumen ya no existe:

```
$  docker volume ls
DRIVER              VOLUME NAME
```

Tenga en cuenta que al borrar un volumen, los datos que contenía el volumen se pierden para siempre, salvo que hubiera realizado una copia de seguridad.