Prácticas Docker

1. Volumenes que ya vienen con el contenedor. OwnCloud

- Vamos a lanzar un contenedor y vamos a ver como se crea un volumen de forma automática y para ver donde lo deja.
- En este caso vamos a usar una imagen de OwnCloud, que nos permite disponer de nuestro almacenamiento privado en la nube.

- Por defecto, usa el volumen situado en **/var/www/html** del contenedor
- Primero nos descargamos la imagen

```
docker pull owncloud
```

- Ahora vamos a ver los volúmenes que tenemos. Para este ejemplo, es posible que tengamos múltiples volúmenes ya creados de las prácticas que hemos ido haciendo a lo largo del curso, ya que muchos contenedores crean sus propios volúmenes.
- Ejecutamos el comando siguiente. En mi caso no tengo ningún volumen

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

- Es posible que a ti te salga bastantes más. Si quieres empezar desde cero puedes ejecutar estos dos comandos. NOTA: se borran todos los contenedores y volúmenes que tienes. El primero borra los contenedores y el segundo los volúmenes que están sin usar.

```
docker rm `docker ps -qa`
docker volume prune
```

- Primero nos situamos en /var/lib/Docker/volumes

```
cd /var/lib/docker/volumes/


ls -l
total 44
-rw-------. 1 root root 65536 mar 24 17:17 metadata.db
```

- Si has dejado algún volumen, a ti te saldrán mas directorios
- Para probar que funciona correctamente, vamos a crear un contenedor que se elimine al salir

```
docker run -d --rm -p 80:80 --name cloud1 owncloud
09108071010a45ae3a3335167468a4498d90f7b9564190e3a44f6560dc
63fc94
```

- Podemos comprobar que tenemos un nuevo directorio, asociado al volumen del contenedor

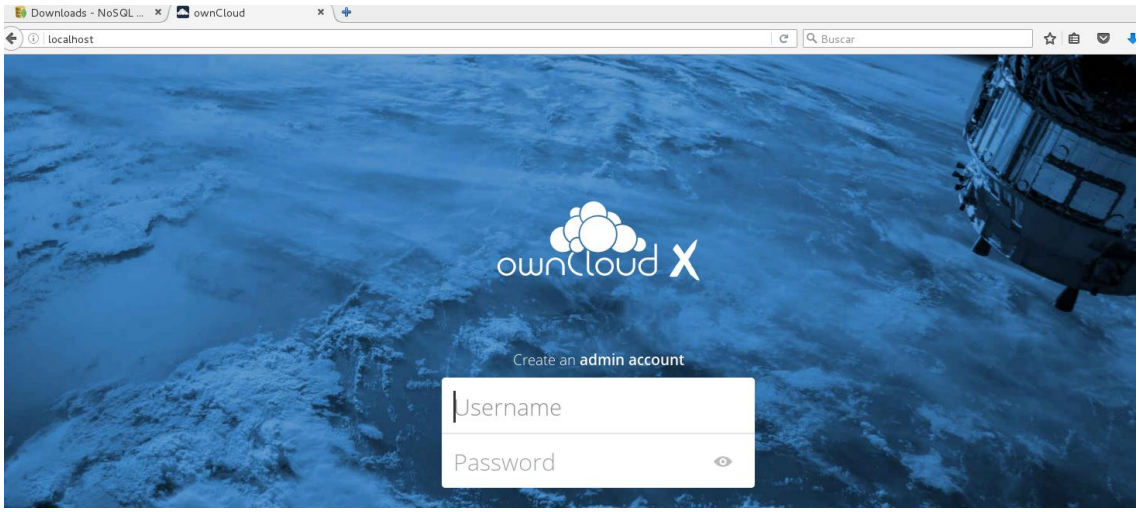
```
ls -l
total 44
drwxr-xr-x. 3 root root    19 mar 24 17:26
8900e560bd60b64da85cc2700c5d8ec8a670cd05c0dfb4fdfa827e0e99
fa0eb5
-rw-------. 1 root root 65536 mar 24 17:26 metadata.db
```

- Y dentro tenemos un directorio denominado "_data" donde podemos ver lo que el contenedor persiste

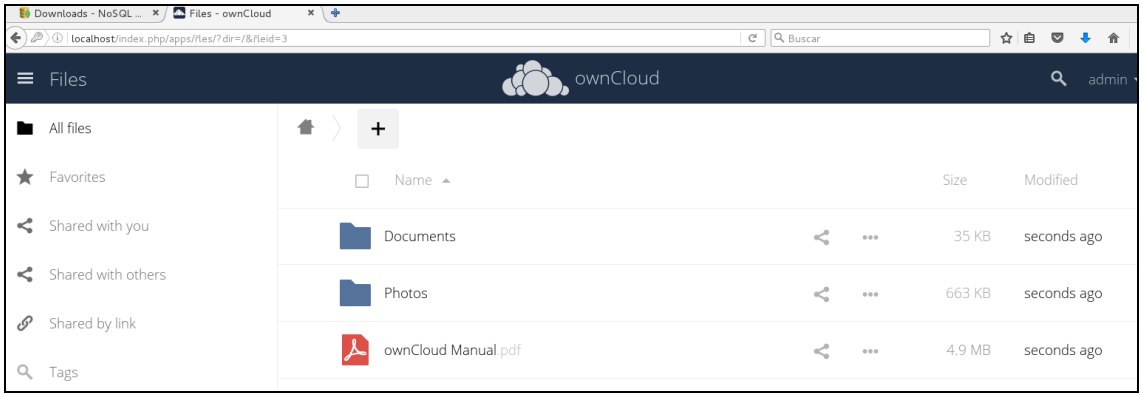
```
#cd
8900e560bd60b64da85cc2700c5d8ec8a670cd05c0dfb4fdfa827e0e99
fa0eb5/_data/
ls -l
total 180
drwxrwxrwx. 24 33 nfsnobody  4096 feb 19 17:28 apps
-rw-r--r--.  1 33 nfsnobody  8859 feb 19 17:27 AUTHORS
-rw-r--r--.  1 33 nfsnobody 41944 feb 19 17:27
CHANGELOG.md
drwxrwxrwx.  2 33 nfsnobody    31 feb 19 17:27 config
-rw-r--r--.  1 33 nfsnobody  4353 feb 19 17:27 console.php
-rw-r--r--.  1 33 nfsnobody 34520 feb 19 17:27 COPYING
drwxr-xr-x. 17 33 nfsnobody  4096 feb 19 17:30 core
```

```
-rw-r--r--.  1 33 nfsnobody  4969 feb 19 17:27 cron.php
-rw-r--r--.  1 33 nfsnobody 30898 feb 19 17:27
db_structure.xml
-rw-r--r--.  1 33 nfsnobody   179 feb 19 17:27 index.html
-rw-r--r--.  1 33 nfsnobody  3689 feb 19 17:27 index.php
drwxr-xr-x.  3 33 nfsnobody    32 feb 19 17:27 l10n
drwxr-xr-x.  6 33 nfsnobody   101 feb 19 17:27 lib
-rwxr-xr-x.  1 33 nfsnobody   283 feb 19 17:27 occ
drwxr-xr-x.  2 33 nfsnobody    73 feb 19 17:27 ocs
drwxr-xr-x.  2 33 nfsnobody    23 feb 19 17:27 ocs-
provider
-rw-r--r--.  1 33 nfsnobody  3197 feb 19 17:27 public.php
-rw-r--r--.  1 33 nfsnobody  5481 feb 19 17:27 remote.php
drwxr-xr-x.  4 33 nfsnobody    39 feb 19 17:27 resources
drwxr-xr-x. 12 33 nfsnobody   225 feb 19 17:27 settings
-rw-r--r--.  1 33 nfsnobody  1967 feb 19 17:27 status.php
drwxr-xr-x.  6 33 nfsnobody   130 feb 19 17:28 updater
-rw-r--r--.  1 33 nfsnobody   271 feb 19 17:30 version.php
```

- Si entramos en la máquina principal por el puerto 80…



- Nos pide crear una cuenta de administración. Podemos poner cualquier usuario y password. Al entrar aparece la siguiente pantalla.



- Por tanto, si usamos este contenedor, los ficheros que subamos se almacenan en un volumen persistente y perviven a través de apagados y cierres de contenedores.

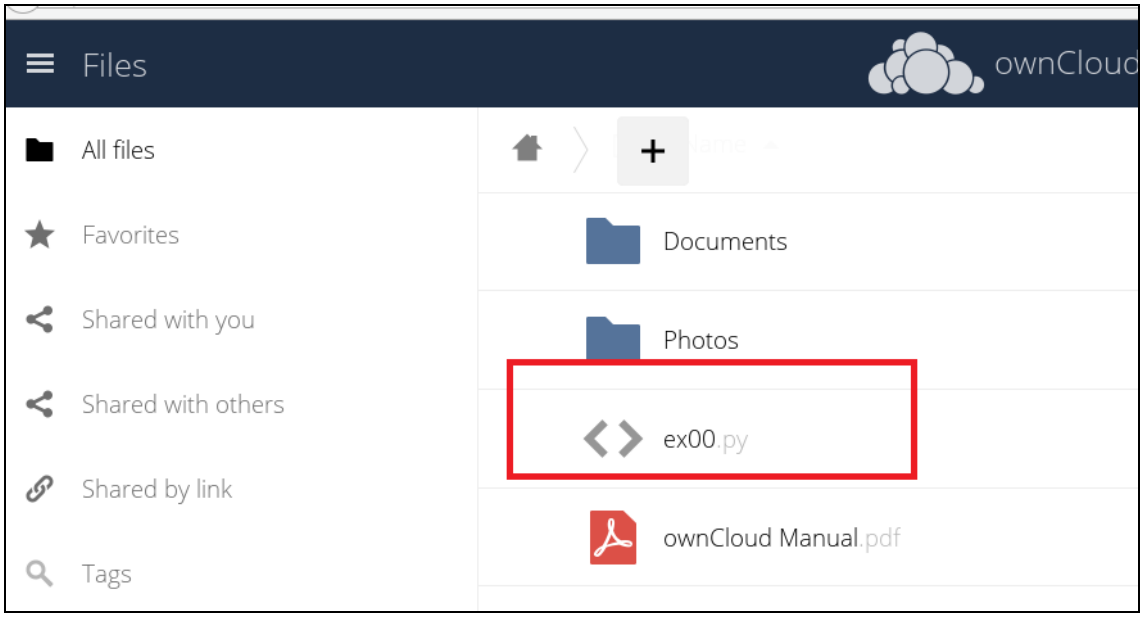
- Desde dentro del directorio _data nos situamos en el directorio data/admin/files. Vemos que está el contenido de los ficheros que salen en la página web

- cd data/admin/files

```
ls -l
total 4980
drwxr-xr-x. 2 33 33      25 mar 24 17:30 Documents
-rw-r--r--. 1 33 33 5097391 mar 24 17:30 ownCloud
Manual.pdf
drwxr-xr-x. 2 33 33      68 mar 24 17:30 Photos
```

- Si subimos algo a través del navegado para probar, debe dejarlo en este directorio

- Yo he subido un programa Python



- Debe haberlo dejado dentro del directorio

```
ls -l
total 4984
drwxr-xr-x. 2 33 33      25 mar 24 17:30 Documents
-rw-r--r--. 1 33 33      18 oct 17  2016 ex00.py
-rw-r--r--. 1 33 33 5097391 mar 24 17:30 ownCloud
Manual.pdf
drwxr-xr-x. 2 33 33      68 mar 24 17:30 Photos
```

- Podemos ver que todo lo que hacer el contenedor lo deja en ese volumen
- Ahora vamos a inspeccionar la información del volumen y del contenedor
- Podemos ver el nombre del volumen

```
docker volume ls
DRIVER              VOLUME NAME
local
8900e560bd60b64da85cc2700c5d8ec8a670cd05c0dfb4fdfa827e0e99
fa0eb5
```

- Podemos hacer un inspect del volumen. Dado que genera bastante información la mandamos a un fichero

```
docker volume inspect
8900e560bd60b64da85cc2700c5d8ec8a670cd05c0dfb4fdfa827e0e99
fa0eb5 > v1.txt
```

- Editamos el fichero para ver su contenido. Podemos ver toda su infomación

```
[
    {
        "CreatedAt": "2018-03-24T17:33:06+01:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint":
"/var/lib/docker/volumes/8900e560bd60b64da85cc2700c5d8ec8a
670cd05c0dfb4fdfa827e0e99fa0eb5/_data",
        "Name":
"8900e560bd60b64da85cc2700c5d8ec8a670cd05c0dfb4fdfa827e0e9
9fa0eb5",
        "Options": {},
        "Scope": "local"
    }
]
```

- Ahora hacemos lo mismo con el contenedor

```
docker inspect cloud1 > cloud1.json
```

- Editamos su contenido y buscamos la zona donde están los volúmenes. Podemos observar que el volumen está asociado al directorio /var/www/html

```
"Mounts": [
            {
                "Type": "volume",
                "Name":
"8900e560bd60b64da85cc2700c5d8ec8a670cd05c0dfb4fdfa827e0e9
9fa0eb5",
                "Source":
"/var/lib/docker/volumes/8900e560bd60b64da85cc2700c5d8ec8a
670cd05c0dfb4fdfa827e0e99fa0eb5/_data",
                "Destination": "/var/www/html",
                "Driver": "local",
                "Mode": "",
                "RW": true,
                "Propagation": ""
```

```
            }
        ],
```

- Paramos el contenedor. Dado que lo hemos creado con la opción “--rm” también borra el volumen

```
docker stop cloud1
cloud1


docker volume ls
DRIVER              VOLUME NAME
```

2. Crear un volumen en el contenedor

- Ahora vamos a hacer un ejemplo de crear un volumen en un contenedor
- Nos vamos a /var/docker/lib/volumes
- Comprobamos los directorios que hay, si tenemos alguno
- Lanzamos un contenedor y le asociamos un volumen, por ejemplo con Fedora. El directorio del contenedor es /datos

```
docker run -it --name fedora1 -v /datos fedora bash
[root@61e44cd000f1 /]#
```

- Nos habrá creado un directorio y un volumen

```
ls -l
total 44
drwxr-xr-x. 3 root root    19 mar 24 17:57
44787bbc29f93859ab2b6830b1ae39f678c0b13fdadf1f6e2db4a6314f
bec92b
-rw-------. 1 root root 65536 mar 24 17:57 metadata.db+



docker volume ls
DRIVER              VOLUME NAME
local
44787bbc29f93859ab2b6830b1ae39f678c0b13fdadf1f6e2db4a6314f
bec92b
```

- Nos vamos al contenedor y creamos un fichero en el directorio /datos

```
[root@61e44cd000f1 /]# cd datos
[root@61e44cd000f1 datos]# touch f1.txt
```

- Si nos vamos al directorio _data del volumen debe aparecer

```
#cd
44787bbc29f93859ab2b6830b1ae39f678c0b13fdadf1f6e2db4a6314f
bec92b/_data/


# ls
f1.txt
```

- Si ahora creamos un fichero en ese directorio debe aparecer en /datos del contenedor
- Por último salimos del contenedor
- Si comprobamos los volúmenes, debe seguir existiendo, ya que n o se borra cuando el contenedor se par

```
docker volume ls
DRIVER              VOLUME NAME
local
44787bbc29f93859ab2b6830b1ae39f678c0b13fdadf1f6e2db4a6314f
bec92b
```

-