

# Gestión del almacenamiento en docker

🕒 4 minute read

Cuando un contenedor es borrado, toda la información contenida en él, desaparece. Para tener almacenamiento persistente en nuestros contenedores, que no se elimine al borrar el contenedor, es necesario utilizar volúmenes de datos ( `data volume` ). Un volumen es un directorio o un fichero en el `docker engine` que se monta directamente en el contenedor. Podemos montar varios volúmenes en un contenedor y en varios contenedores podemos montar un mismo volumen.

Tenemos dos alternativas para gestionar el almacenamiento en docker:

- Usando volúmenes de datos
- Usando contenedores de volúmenes de datos

## Volúmenes de datos

---

Los volúmenes de datos tienen las siguientes características:

- Son utilizados para guardar e intercambiar información de forma independientemente a la vida de un contenedor.
- Nos permiten guardar e intercambiar información entre contenedores.
- Cuando borramos el contenedor, no se elimina el volumen asociado.
- Los volúmenes de datos son directorios del host montados en un directorio del contenedor, aunque también se pueden montar ficheros.
- En el caso de montar en un directorio ya existente de un contenedor un volumen de datos , su contenido no será eliminado.

## Añadiendo volúmenes de datos

Vamos a empezar creando una contenedor al que le vamos a asociar un volumen:

```
$ docker run -it --name contenedor1 -v /volumen ubuntu:14.04 bash
```

Como podemos comprobar con la opción `-v` hemos creado un nuevo volumen que se ha montado en el directorio `/volumen` del contenedor. Vamos a crear un fichero en ese directorio:

```
root@d50f89458659:/# cd /volumen/
root@d50f89458659:/volumen# touch fichero.txt
root@d50f89458659:/volumen# exit
```

Podemos comprobar los puntos de montajes que tiene nuestro contenedor con la siguiente instrucción:

```
$ docker inspect contenedor1
...
"Mounts": [
  {
    "Name": "c7665edfb4505d6ac85fb0f3db118f6c7bb63958157ec722d6d3ee15ca8f3427",
    "Source": "/mnt/sda1/var/lib/docker/volumes/c7665edfb4505d6ac85fb0f3db118f6c7bb63958157ec722d6d3ee15ca8f3427/_data",
    "Destination": "/volumen",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  }
],
...
```

A continuación podemos comprobar en el docker engine el directorio correspondiente al volumen y que efectivamente contiene el fichero que hemos creado.

```
# cd /mnt/sda1/var/lib/docker/volumes/c7665edfb4505d6ac85fb0f3db118f6c7bb63958157ec722d6d3ee15ca8f3427/_data
fichero1.txt
```

Con el cliente docker también podemos comprobar los volúmenes que hemos creados ejecutando:

```
$ docker volume ls
DRIVER          VOLUME NAME
local           c7665edfb4505d6ac85fb0f3db118f6c7bb63958157ec722d6d3ee15ca8f3427
```

Y a continuación podemos obtener información del volumen:

```
$ docker volume inspect c7665edfb4505d6ac85fb0f3db118f6c7bb63958157ec722d6d3ee15ca8f3427
[
  {
    "Name": "c7665edfb4505d6ac85fb0f3db118f6c7bb63958157ec722d6d3ee15ca8f3427",
    "Driver": "local",
    "Mountpoint": "/mnt/sda1/var/lib/docker/volumes/c7665edfb4505d6ac85fb0f3db118f6c7bb63958157ec722d6d3ee15ca8f3427/_data",
    "Labels": null
  }
]
```

Por último podemos comprobar que aunque borremos el contenedor, el volumen no se borra:

```
$ docker rm contenedor1

$ docker volume ls
DRIVER          VOLUME NAME
local           c7665edfb4505d6ac85fb0f3db118f6c7bb63958157ec722d6d3ee15ca8f3427
```

## Creando volúmenes de datos

En el apartado anterior al añadir un nuevo volumen al contenedor, se ha creado un nuevo volumen pero con un nombre muy poco significativo. A la hora de gestionar una gran cantidad de volúmenes es muy importante poner nombres significativos a los volúmenes, para ello vamos a crear primero el volumen (indicando el nombre) y a continuación lo asociamos al contenedor:

```
$ docker volume create --name vol1
vol1
$ docker volume ls
DRIVER          VOLUME NAME
local           c7665edfb4505d6ac85fb0f3db118f6c7bb63958157ec722d6d3ee15ca8f3427
local           vol1
$ docker run -it --name contenedor2 -v vol1:/data ubuntu:14.04 bash
```

Hemos creado el contenedor2 que tendrá asociado el volumen `vol1` y lo tendrá montado en el directorio `/data`.

Realmente no es necesario realizar la operación de crear el volumen con anterioridad. Si creamos un contenedor y asociamos un volumen con un nombre que no existe, el volumen se creará. Vemos un ejemplo:

```
$ docker run -it --name contenedor3 -v vol2:/data ubuntu:14.04 bash
root@8f36b1c407b9:/# exit

$ docker volume ls
DRIVER          VOLUME NAME
local           c7665edfb4505d6ac85fb0f3db118f6c7bb63958157ec722d6d3ee15ca8f3427
local           vol1
local           vol2
```

De forma similar, pero utilizando el nombre largo, podemos crear un nuevo contenedor al que asociamos nuestro primer volumen y podamos comprobar que la información guardada en el volumen es persistente:

```
$ docker run -it --name contenedor4 -v c7665edfb4505d6ac85fb0f3db118f6c7bb63958157ec722d6d3ee15ca8f3427:/data ubuntu:14.04 bash
root@1fbb52788b5:/# cd data/
root@1fbb52788b5:/data# ls
fichero.txt
```

Por último para borrar un volumen tenemos que asegurarnos que no está asociado a ningún contenedor:

```
$ docker volume rm vol1
Error response from daemon: Unable to remove volume, volume still in use: remove vol1: volume is in use -
[969b3bd0ec30193b9ce2cef837b6231e3c39310caca4660ea0fcab695d99d065]

$ docker rm contenedor2
contenedor2
$ docker volume rm vol1
vol1
```

## Montando directorios del host en un contenedor

Una aplicación particular del trabajo con volúmenes de datos, es la posibilidad de montar en el contenedor un directorio del docker engine. En este caso hay que tener en cuenta que si el directorio de montaje del contenedor ya existe, no se borra su contenido, simplemente se monta encima. Veamos un ejemplo:

```
$ docker run -it --name contenedor5 -v /home/docker:/informacion ubuntu:14.04 bash
root@c1cf905f170a:/# cd informacion/
root@c1cf905f170a:/informacion# ls
log.log
```

En este ejemplo hemos montado en el directorio `/informacion` del contenedor, el directorio `/home/docker` del Docker Engine. Y comprobamos que podemos acceder a los ficheros del hosts.

## Montando ficheros del host en un contenedor

Además de poder montar un directorio, podemos montar un fichero del Docker Engine. En el siguiente ejemplo vamos a montar el fichero `/etc/hosts` en el contenedor.

```
$ docker run -it --name contenedor6 -v /etc/hosts:/etc/hosts.bak ubuntu:14.04 bash
root@ff58bc448b57:/# cat /etc/hosts.bak
```

## Contenedores de volúmenes de datos

Otra posibilidad que tenemos para conseguir el almacenamiento persistente es la creación de un contenedor donde creamos un volumen de datos y que podemos asociar a uno o varios contenedores para guardar la información en él.

Vamos a crear un contenedor con un volumen de datos asociado:

```
$ docker create -it --name data_vol_container -v /shared_folder ubuntu:14.04 bash
ec45dbf110e14f6f4097d2b2dfaec7092669c18c7424913106a46342af54ce23
```

A continuación con el parámetro `--volumes-from`, creamos un contenedor asociado al contenedor anterior y que montará el volumen de datos:

```
$ docker run -it --name container1 --volumes-from data_vol_container ubuntu:14.04 bash
root@93aed9e94393:/# cd /shared_folder/
root@93aed9e94393:/shared_folder# echo "hola">fichero.txt
root@93aed9e94393:/shared_folder# exit
```

Finalmente creamos un nuevo contenedor asociado al contenedor con el volumen de datos, para comprobar que, efectivamente, se comparte el volumen:

```
$ docker run -it --name container2 --volumes-from data_vol_container ubuntu:14.04 bash
root@a2733d38c49a:/# cat /shared_folder/fichero.txt
hola
```

## Conclusiones

---

En esta entrada se ha descrito de forma introductoria las distintas posibilidades que tenemos para conseguir que la información gestionada por nuestros contenedores sea persistentes, es decir, no sea eliminada cuando destruimos el contenedor. Para más información sobre el almacenamiento en Docker puedes consultar la [documentación oficial](https://docs.docker.com/engine/userguide/containers/dockervolumes/) (<https://docs.docker.com/engine/userguide/containers/dockervolumes/>).

 **Updated:** May 30, 2016

---

### COMMENTS

0 Comentarios PLEDIN

 Javier Martín Alon... ▾ Recomendar 2 Tweet Compartir

Ordenar por los mejores ▾



Sé el primero en comentar...

Sé el primero en comentar.

## TAMBIÉN EN PLEDIN

**GNS3, añadiendo hosts a nuestras topologías**

2 comentarios • hace un año



**José Domingo Muñoz** — Lo siento, pero hace tiempo que no trabajo con estas tecnologías. Además el error puede estar producido por

**Recursos de Kubernetes: Pods**

3 comentarios • hace un año



**felocru** — Muchas gracias por tu respuesta. Ya me ha quedado totalmente claro. Saludos.

**Crear páginas web con Bottle: Python Web Framework (1ª parte)**

3 comentarios • hace un año





**Miguel Alejandro** — Gracias, la verdad es que ya toque flask pero vi bottle y lo note mas simple jaja igual gracias por la recomendacion,

**Presentación - PLEDIN 3.0**

2 comentarios • hace un año



**José Domingo Muñoz** — Muchas gracias. Mensajes como el tuyo me dan el subidón necesario para seguir escribiendo y

 Suscríbete  Añade Disqus a tu sitio webAñade Disqus Añadir