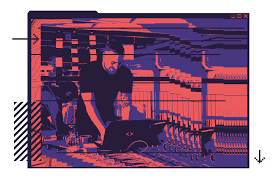
Puesta en producción segura

## **Captura la utilización de volúmenes**



[Captura la utilización de volúmenes 1](#_78omvxhg4bkk)

[Práctica 3](#_acppzwywae26)

[**Persistencia en Docker 4**](#_dz5xn19un79l)

[**Volúmenes de Docker (Docker Volumes) 4**](#_9b6it0he2je9)

[Sintaxis 5](#_f02vw9c8fnw2)

[Comandos comunes 6](#_iazm95uv1230)

[Creación de volúmenes 6](#_79ch346uje3k)

[Listado 6](#_8lie1uaxdvu9)

[Inspeccionar volúmen 7](#_m0jyrykjc44r)

[Borrar volúmen 7](#_pe5kdby1ov36)

[Iniciar container con volúmen 7](#_w6wrpe2pr4n8)

[Volúmen de solo lectura 8](#_3rj2a2lhmp4q)

[Compartir información entre máquinas 9](#_rl5x1jwlp35y)

[Instalación del gestor de volúmenes 9](#_11ra8jf5i8b9)

[Creación de un volumen 9](#_ruat886yhrdg)

[Creación de un contenedor con un volumen gestionado por driver 9](#_1z078k8vkgkv)

[Realizar copias de seguridad, restaurar o migrar volúmenes de datos 9](#_l7vp3qgslldq)

[Copia de seguridad 10](#_2753u3b3dv9k)

[Restauración de volúmen desde copia de seguridad 10](#_uj0enrwhvof7)

[Restauración 10](#_z3zy7eac4yso)

[Eliminar volúmenes anónimos 11](#_yymek7nveog7)

[**Montajes de enlace (Bind Mounts) 12**](#_n64brifmv9zc)

[Ejemplos 12](#_1c7wf8kjvwjv)

[**Tmpfs Mounts 12**](#_4f6rchaoettd)

[Casos de uso: 13](#_bwl7xqdndklq)

[Ventajas: 13](#_v6sf3djcjgr0)

[Desventajas: 13](#_k34iifrgipev)

[Ejemplo 13](#_exw9l0nt3z5u)

[Verificar que se trata de un tipo tmpfs 13](#_n5z094n1eczv)

[**Comparación entre volúmenes con nombre y montajes de enlace 14**](#_pqt044y8xxwu)

[Volúmenes con nombre 14](#_2lglqd2v4hv6)

[Montaje de enlace 14](#_1b7ay3y96ltg)

[**Bibliografía 14**](#_cbx135b9k8xq)

| | **Práctica** | | --- | | Es muy rápido montar en docker una base de datos persistente con volúmenes. Por ello, montaré un container de Mysql con persistencia y accederé a él por mi máquina host. | | **docker run -d \**  **--name mysql-c \**  **-e MYSQL\_ROOT\_PASSWORD=root\_password \**  **-e MYSQL\_USER=jj \**  **-e MYSQL\_PASSWORD=elmasguapo \**  **-e MYSQL\_DATABASE=mydatabase \**  **-v /home/ymir/Desktop/vol:/var/lib/mysql \**  **-p 3306:3306 \**  **mysql:latest** | |  | | En el host: | | **mysql -h 127.0.0.1 -P 3306 -u jj -p** | |  | |  | |  |  Persistencia en Docker |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Los containers parten de la información de la imagen, esto evita acumular datos de manera innecesaria. Pero a veces es necesario guardar cierta información para su reutilización por otro container.  Para ello se utilizan los **volumes, bind mounts, y tmpfs mount.** |
|  |

| Volúmenes de Docker (Docker Volumes) |
| --- |
| Los "volúmenes" son el mecanismo preferido para persistir datos generados y utilizados por los contenedores Docker. Mientras que los "bind mounts" dependen de la estructura de directorios y del sistema operativo de la máquina anfitriona, los volúmenes son gestionados completamente por Docker.  Los volúmenes tienen varias ventajas:   * Los volúmenes son más fáciles de respaldar o migrar que los "bind mounts". * Puedes gestionar los volúmenes utilizando comandos de la CLI de Docker o la API de Docker. * Los volúmenes funcionan tanto en contenedores Linux como en contenedores Windows. * Los volúmenes pueden compartirse de manera más segura entre múltiples contenedores. * Los controladores de volumen te permiten almacenar volúmenes en máquinas remotas o proveedores de servicios en la nube, encriptar el contenido de los volúmenes o agregar otras funcionalidades. * Los nuevos volúmenes pueden tener su contenido prellenado por un contenedor. * Los volúmenes en Docker Desktop tienen un rendimiento mucho mayor que los "bind mounts" desde máquinas anfitrionas Mac y Windows. |
| Sintaxis |
| En general, --mount es más explícito y detallado.  La mayor diferencia es que la sintaxis de -v combina todas las opciones juntas en un solo campo, mientras que la sintaxis de --mount las separa. |
| Si necesitas **especificar opciones del controlador** de volumen, debes utilizar **--mount.**  **-v** o **--volume**:  Consta de tres campos, separados por dos puntos (:).  Los campos deben estar en el orden correcto y el significado de cada campo no es inmediatamente evidente.   1. En el caso de volúmenes con nombre, el primer campo es el **nombre del volumen** y es **único** en una máquina hospedadora dada. Para **volúmenes anónimos, se omite el primer campo**. 2. El segundo campo es la r**uta donde se monta el archivo** o **directorio en el contenedor**. 3. El tercer campo es **opcional** y es una lista separada por comas de opciones, como ro. Estas opciones se discuten a continuación. |
| **--mount:**  Consiste en múltiples pares de clave-valor, separados por comas y cada uno compuesto por una tupla <clave>=<valor>.  La sintaxis de --mount es más detallada que -v o --volume, pero el orden de las claves no es significativo y el valor de la bandera es más fácil de entender.   * El tipo de montaje, (**type**) que puede ser **bind, volume o tmpfs**. Este tema trata sobre volúmenes, por lo que el tipo es siempre volume. * La fuente (**source**) del montaje. Para **volúmenes con nombre**, esto es el nombre del volumen. **Para volúmenes anónimos, se omite** este campo. Puede especificarse como source o src. * El destino (**destination**) toma como valor la ruta donde se monta el archivo o directorio en el contenedor. Puede especificarse como destination, dst o target. * La opción **readonly**, si está presente, hace que el montaje de tipo **Bind** se monte en el contenedor como de solo lectura. Puede especificarse como readonly o ro. * La opción **volume-opt**, que puede especificarse más de una vez, toma un par clave-valor que consiste en el nombre de la opción y su valor. |
| Comandos comunes |
| Creación de volúmenes |
| **docker volume create my-vol** |
| Listado |
| **docker volume ls** |
| Inspeccionar volúmen |
| **docker volume inspect my-vol**  Resultado:  [  {  "Driver": "local",  "Labels": {},  "Mountpoint": "/var/lib/docker/volumes/my-vol/\_data",  "Name": "my-vol",  "Options": {},  "Scope": "local"  }  ] |
| Borrar volúmen |
| docker volume rm my-vol |
| Iniciar container con volúmen |
| Si inicias un contenedor con un volúmen que aún no existe, Docker creará el volumen por ti.  El siguiente ejemplo monta el volumen myvol2 en /app/ en el contenedor.  Los ejemplos de -v y --mount a continuación producen el mismo resultado.  No puedes ejecutar ambos a menos que elimines el contenedor devtest y el volumen myvol2 después de ejecutar el primero. |
| Mount :  **docker run -d \**  **--name devtest \**  **--mount source=myvol2,target=/app \**  **nginx:latest** |
| -v  **docker run -d \**  **--name devtest \**  **-v myvol2:/app \**  **nginx:latest** |
| Resultado:  "Mounts": [  {  "Type": "volume",  "Name": "myvol2",  "Source": "/var/lib/docker/volumes/myvol2/\_data",  "Destination": "/app",  "Driver": "local",  "Mode": "",  "RW": true,  "Propagation": ""  }  ], |
| \*\*\*Si se inicia docker en una carpeta ya poblada del container, ésta información será copiada en el volúmen. |
| Volúmen de solo lectura |
| Para algunas aplicaciones de desarrollo, el contenedor necesita escribir en el montaje de tipo bind para que los cambios se propaguen de vuelta al host de Docker.  En otras ocasiones, el contenedor solo necesita acceso de lectura a los datos.  **Varios contenedores pueden montar el mismo volumen**.  Puedes montar simultáneamente un único volumen como **lectura-escritura para algunos contenedores** y como solo lectura para otros.  El siguiente ejemplo modifica el anterior.  Monta el directorio como un volumen de solo lectura, agregando ro a la lista de opciones (vacía por defecto), después del punto de montaje dentro del contenedor. Cuando hay varias opciones presentes, puedes separarlas con comas. |
| Con –mount: |
| **docker run -d \**  **--name=nginxtest \**  **--mount source=nginx-vol,destination=/usr/share/nginx/html,readonly \**  **nginx:latest** |
| Con -v: |
| **docker run -d \**  **--name=nginxtest \**  **-v nginx-vol:/usr/share/nginx/html:ro \**  **nginx:latest** |
| Compartir información entre máquinas |
| Los controladores de volumen te permiten abstraer el sistema de almacenamiento subyacente de la lógica de la aplicación.  Por ejemplo, si tus servicios utilizan un volumen con un controlador NFS, puedes actualizar los servicios para utilizar un controlador diferente. Por ejemplo, para almacenar datos en la nube, sin cambiar la lógica de la aplicación. |
| Docker en su guía recomienda el plugin **vieux/sshfs como** gestor de volúmenes |
| Instalación del gestor de volúmenes |
| **docker plugin install --grant-all-permissions vieux/sshfs** |
| Creación de un volumen |
| **docker volume create --driver vieux/sshfs \**  **-o sshcmd=test@node2:/home/test \**  **-o password=testpassword \**  **sshvolume** |
| Creación de un contenedor con un volumen gestionado por driver |
| **docker run -d \**  **--name sshfs-container \**  **--volume-driver vieux/sshfs \**    **--mount src=sshvolume,target=/app,volume-opt=sshcmd=test@node2:/home/test,volume-opt=password=testpassword \**  **nginx:latest** |
| Realizar copias de seguridad, restaurar o migrar volúmenes de datos |
| Los volúmenes son útiles para copias de seguridad, restauraciones y migraciones. Utiliza la bandera **--volumes-from** para crear un nuevo contenedor que monte ese volumen. |
| Copia de seguridad |
| **docker run -v /dbdata --name <container> ubuntu /bin/bash** |
| **docker run --rm --volumes-from <container> -v $(pwd):/backup ubuntu tar cvf /backup/backup.tar /dbdata** |
| 1. **docker run:** Inicia un nuevo contenedor. 2. **--rm**: Elimina el contenedor después de que se detiene. Esto es útil para evitar la acumulación de contenedores no utilizados. 3. **--volumes-from <container>:** Monta los volúmenes del contenedor <container> en el nuevo contenedor. Esto permite acceder a los datos almacenados en esos volúmenes desde el nuevo contenedor. 4. **-v $(pwd):/backup:** Monta el directorio actual del host (donde se ejecuta el comando) en el directorio /backup del nuevo contenedor. Esto se utiliza para almacenar la copia de seguridad en el sistema de archivos del host. 5. **ubuntu:** Especifica la imagen de Docker que se utilizará para el nuevo contenedor, en este caso, la imagen de Ubuntu. 6. **tar cvf /backup/backup.tar /dbdata:** Ejecuta un comando dentro del contenedor. En este caso, utiliza tar para crear un archivo llamado backup.tar en el directorio /backup dentro del contenedor. El contenido que se está empaquetando proviene del directorio /dbdata en el contenedor. |
| Restauración de volúmen desde copia de seguridad |
| **docker run -v /dbdata --name dbstore2 ubuntu /bin/bash** |
| Restauración |
| **docker run --rm --volumes-from dbstore2 -v $(pwd):/backup ubuntu bash -c "cd /dbdata && tar xvf /backup/backup.tar --strip 1"** |
| 1. **docker run**: Inicia un nuevo contenedor. 2. **--rm:** Elimina el contenedor después de que se detiene. Esto asegura que no se acumulen contenedores innecesarios. 3. **--volumes-from dbstore2:** Monta los volúmenes del contenedor llamado "dbstore2" en el nuevo contenedor. Esto permite acceder a los datos almacenados en esos volúmenes desde el nuevo contenedor. 4. **-v $(pwd):/backup:** Monta el directorio actual del host (donde se ejecuta el comando) en el directorio /backup del nuevo contenedor. Esto se utiliza para proporcionar la ubicación de la copia de seguridad almacenada en el sistema de archivos del host. 5. **ubuntu:** Especifica la imagen de Docker que se utilizará para el nuevo contenedor, en este caso, la imagen de Ubuntu. 6. **bash -c "cd /dbdata && tar xvf /backup/backup.tar --strip 1"**: Ejecuta un comando dentro del contenedor.   En este caso, utiliza bash para cambiar al directorio /dbdata y luego utiliza tar para extraer el contenido del archivo de respaldo backup.tar ubicado en el directorio /backup.  La opción --strip 1 se utiliza para omitir el nivel superior del directorio al extraer, lo que significa que los archivos se extraerán directamente en /dbdata. |
| Eliminar volúmenes anónimos |
| # Lista todos los volúmenes para identificar el volumen anónimo que deseas eliminar  **docker volume ls**  # Encuentra el volumen anónimo que deseas eliminar y reemplaza [VOLUME\_ID] con el ID real  **docker volume rm [VOLUME\_ID]** |

| Montajes de enlace (Bind Mounts) |
| --- |
| * Permiten compartir un directorio del sistema de archivos del host en el contenedor. * Útil al trabajar en una aplicación, puedes utilizar un "bind mount" para montar el código fuente en el contenedor. * El contenedor percibe los cambios que realizas en el código de inmediato, tan pronto como guarda un archivo.   Esto significa que puedes ejecutar procesos en el contenedor que estén atentos a los cambios en el sistema de archivos y respondan a ellos. |
| Ejemplos |
| **docker run -it --mount type=bind,src="$(pwd)",target=/src ubuntu bash** |
| **docker run -d \**  **--name devtest \**  **--mount source=myvol2,target=/app \**  **nginx:latest** |

| Tmpfs Mounts |
| --- |
| Las "tmpfs mounts" en Docker se refieren a montar un sistema de archivos en m**emoria temporal compartida con el host** dentro de un contenedor. En resumen: Casos de uso:  1. Almacenamiento temporal: Útil para almacenar datos temporales que no necesitan ser persistentes, como archivos temporales o cachés. 2. Mejora del rendimiento: Puede acelerar las operaciones de lectura/escritura al aprovechar la velocidad de la memoria RAM en lugar de acceder al disco.  Ventajas:  1. **Rendimiento:** Acceso más rápido a datos al residir en la memoria. 2. **Efímero:** Los datos almacenados en tmpfs no persisten después de reiniciar el contenedor, lo que puede ser beneficioso para ciertos casos de uso.  Desventajas:  1. **Limitación de tamaño**: Al estar en la memoria, existe una limitación inherente del tamaño del sistema de archivos tmpfs. Si se supera, podría afectar el rendimiento del sistema host. 2. **Pérdida de datos**: Los datos almacenados en tmpfs se perderán si se detiene o se reinicia el contenedor. 3. **No se puede compartir**: No es la mejor opción para almacenar datos que deben persistir más allá del ciclo de vida del contenedor. 4. **Solo disponible en Linux** |
| Ejemplo |
| **docker run -d \**  **-it \**  **--name tmptest \**  **--mount type=tmpfs,destination=/app \**  **nginx:latest** |
| Verificar que se trata de un tipo tmpfs |
| **docker inspect tmptest --format '{{ json .Mounts }}'**  **[{"Type":"tmpfs","Source":"","Destination":"/app","Mode":"","RW":true,"Propagation":""}]** |

# Comparación entre volúmenes con nombre y montajes de enlace

| **Volúmenes con nombre** | | **Montaje de enlace** |
| --- | --- | --- |
| Localización | Docker elige | Tú decides |
| Ejemplo (con --mount) | type=volume,src=my-volume,target=/usr/local/data | type=bind,src=/path/to/data,target=/usr/local/data |
| Añade información del container | Si | No |
| Admite Gestores | Si | No |

| Bibliografía |
| --- |
| * [https://docs.docker.com/get-started](https://docs.docker.com/get-started/05_persisting_data/) |