How to Connect Storage Mount Using Volume Mount on Docker?

written by sysadmin | 9 June 2025

The previous article explained how to access the database installed in a container. But you must know that if you delete the container, it will automatically delete the database too. Therefore, you must create a storage to store a database so that if you delete the container, either intentionally or unintentionally, the database will remain. By default, Docker supports the following types of storage mounts for storing data outside of the container's writable layer: volume mounts, bind mounts, and tmpfs mounts, each with its unique characteristics and use cases. This article will explain how to connect storage mount using volume mount. You can see a summary of the three types of storage here.

Problem

How to connect storage mount using volume mount on Docker?

Solution

Volumes are persistent storage mechanisms managed by the Docker daemon. It is stored in the filesystem on the host in **/var/lib/docker/volumes** folder, but to interact with the data in the volume, you must mount the volume to a container. Volumes are ideal for performance-critical data processing and long-term storage needs and are also suitable for sharing data between containers since volumes can be attached to multiple containers.

A. List, remove and create the volume

To see the volume on the server, use the command below:

```
docker volume ls
```

To remove a volume, use the command below:

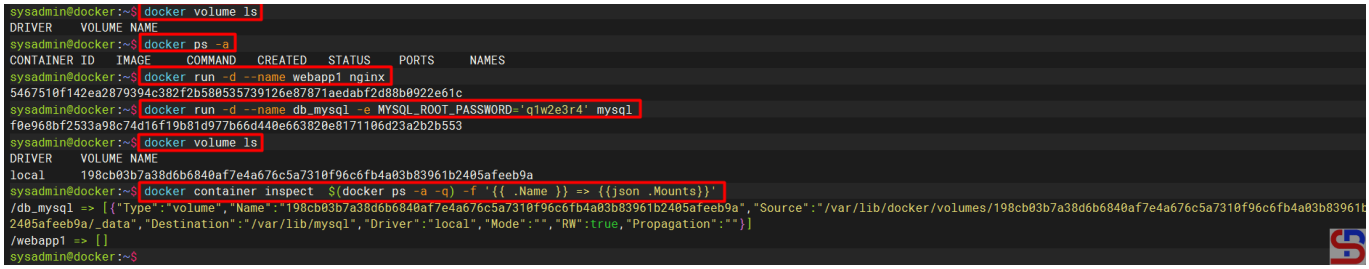
```
docker volume rm volume_name
```

Use the command below to remove all unused volumes:

```
docker volume prune
```

By default, there is no volume if there is no container on the server. But if there is a container that uses storage like a container that uses a database image, the volume will form automatically. To see which containers that use volume can use the command below:

```
docker container inspect webapp postgresdb mysqldb -f '{{ .Name }} => {{json .Mounts}}'
```



Display the container that uses volume

You can create a new volume using the format below:

```
docker volume create volume_name
```

Use the command below if you want to create a mysql_vol volume:

```
docker volume create mysql_vol
```

To see this volume information in detail, use the command below:

```
docker volume inspect mysql_vol
```

```
sysadmin@docker:~$ docker volume inspect mysql_vol
[
    {
        "CreatedAt": "2025-04-16T16:00:28Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/mysql_vol/_data",
        "Name": "mysql_vol",
        "Options": null,
        "Scope": "local"
    }
]
sysadmin@docker:~$
```

Inspect the volume

B. Connect storage mount to a container

After that, make a new container where the container mounts to the volume you created using the format below:

```
docker run -d --name container_name -v volume_name:/path/folder/in/container image_name
```

You can see more detailed information on mounting a container, using the format below:

```
docker inspect container_name -f "{{json .Mounts}}" | python3 -m json.tool
```

For example, use the command below if you want to create a db_mysql container with a password q1w2e3r4 using mysql_vol volume in the folder /var/lib/mysql :

```
docker run -d --name db_mysql -e MYSQL_ROOT_PASSWORD='q1w2e3r4' -v mysql_vol:/var/lib/mysql  mysql
docker inspect db_mysql -f "{{json .Mounts}}" | python3 -m json.tool
```

```
sysadmin@docker:~$ docker run -d --name db_mysql -e MYSQL_ROOT_PASSWORD='q1w2e3r4' -v mysql_vol:/var/lib/mysql  mysql
6ad689b31c9d559aac3b360c23434b591bf650fbec017629ff26b273da5e7d7e
sysadmin@docker:~$
sysadmin@docker:~$ docker inspect db_mysql -f "{{json .Mounts}}" | python3 -m json.tool
[
    {
        "Type": "volume",
        "Name": "mysql_vol",
        "Source": "/var/lib/docker/volumes/mysql_vol/_data",
        "Destination": "/var/lib/mysql",
        "Driver": "local",
        "Mode": "z",
        "RW": true,
        "Propagation": ""
    }
]
sysadmin@docker:~$
```

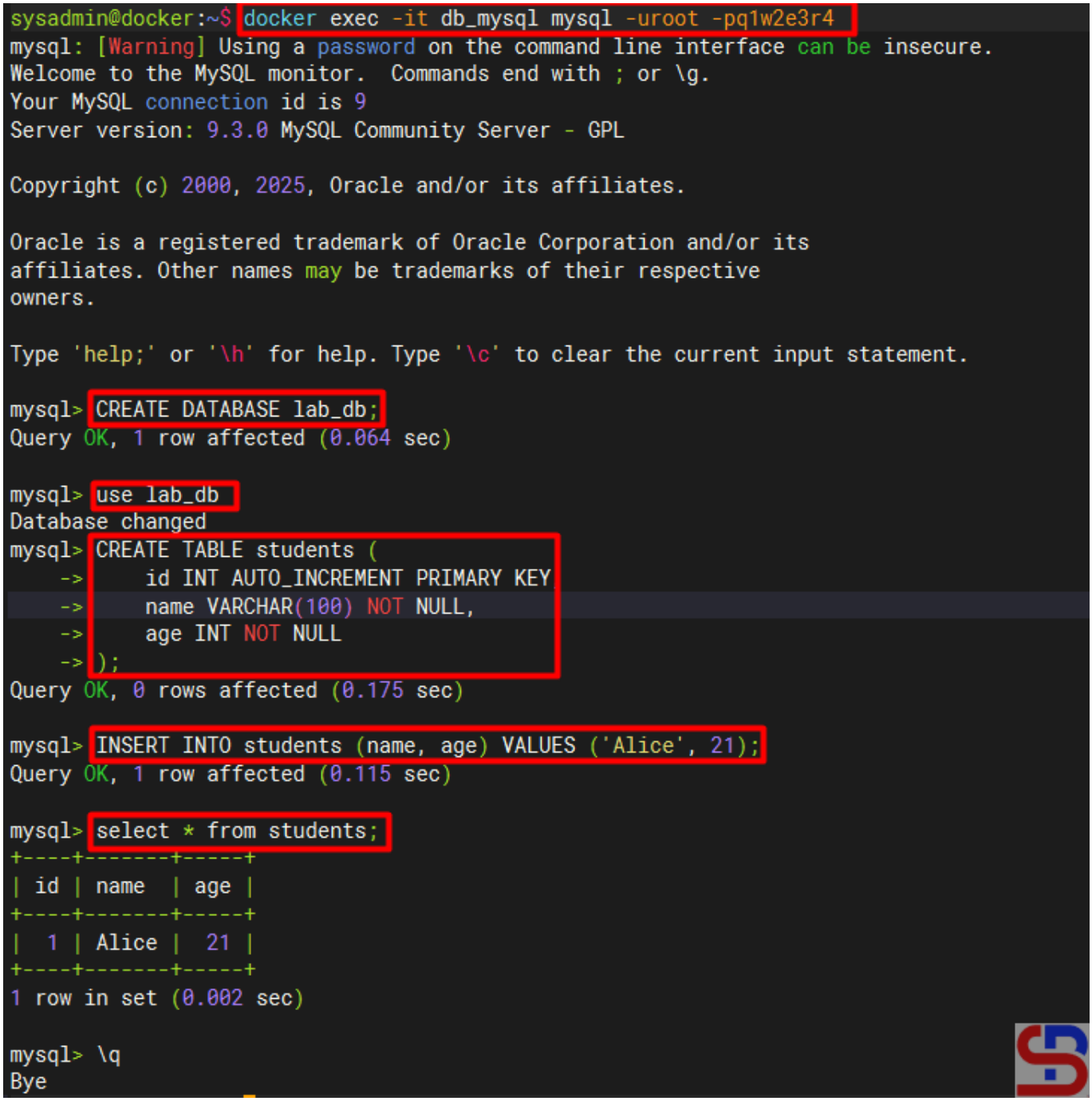
Run the container and check the mount

C. Simulate remove the container

After that, try to enter the container by using the command below:

```
docker exec -it db_mysql mysql -uroot -pq1w2e3r4
```

Create a database and fill it as shown in the image below:



Create a database

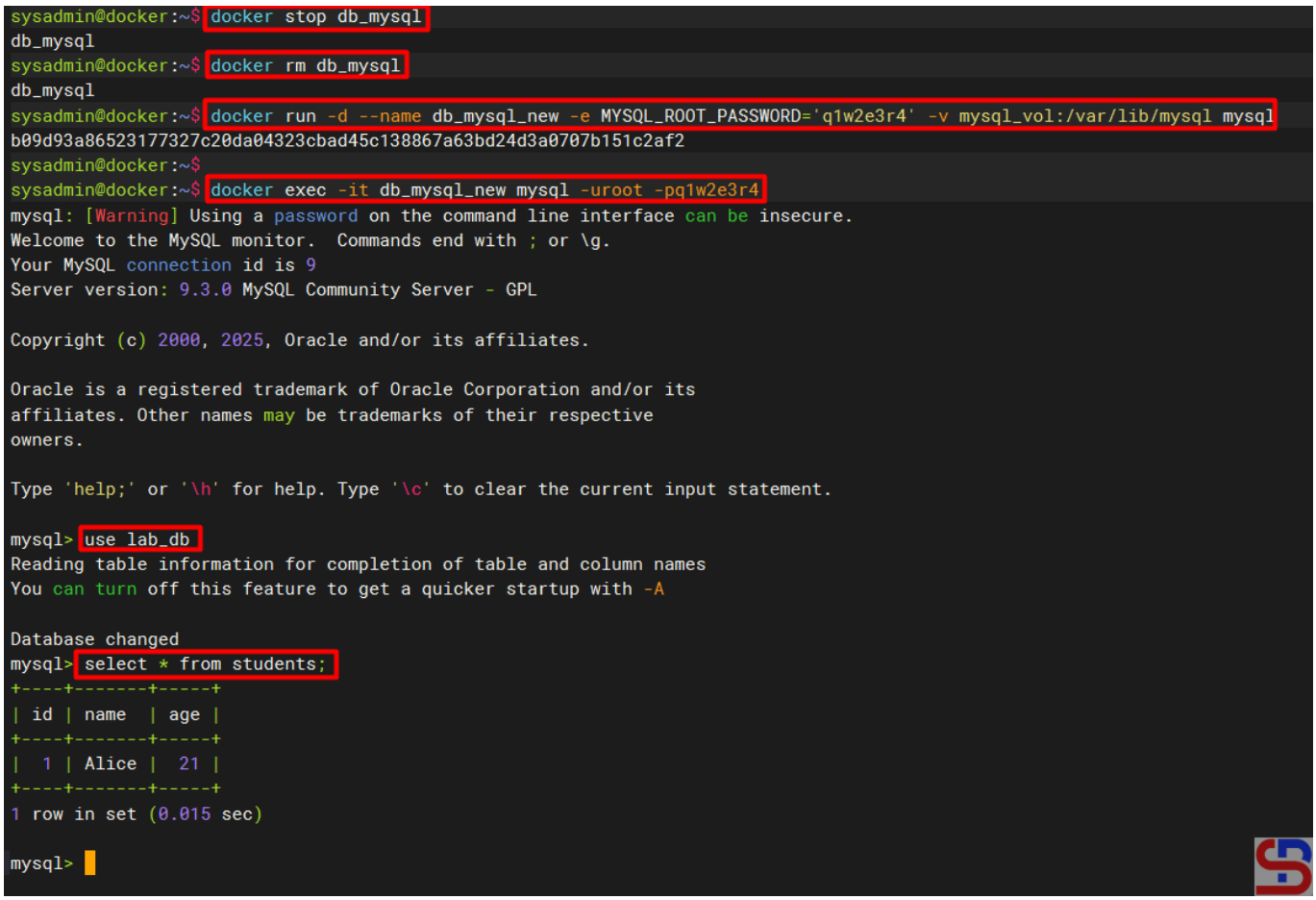
Then, try to simulate by deleting the container and creating a new container where the data is put into the mysql_vol volume in the /var/lib/mysql folder using the command below:

```
docker stop db_mysql
docker rm db_mysql
docker run -d --name db_mysql_new -e MYSQL_ROOT_PASSWORD='q1w2e3r4' -v
mysql_vol:/var/lib/mysql mysql
```

Access the container by using the command:

```
docker exec -it db_mysql_new mysql -uroot -pq1w2e3r4
```

The lab_db database should still be accessible using the container you just created, as shown in the image below:



The database can still be accessed

D. Share data in the volume among containers

Let’s do a simulation to share the data that is in the volume with more than one container on a single server. In this article, 2 containers will be created that are connected to a mysql_vol volume. Type the command below to make 2 containers:

```
docker run -d --name db_mysql1 -e MYSQL_ROOT_PASSWORD='q1w2e3r4' -v
mysql_volume:/var/lib/mysql  mysql
docker run -d --name db_mysql2 -e MYSQL_ROOT_PASSWORD='q1w2e3r4' -v
mysql_volume:/var/lib/mysql  mysql
docker ps
```

```
sysadmin@docker:~$ docker run -d --name db_mysql1 -e MYSQL_ROOT_PASSWORD='q1w2e3r4' -v mysql_vol:/var/lib/mysql  mysql
c1ea3cf4e789f13b3d9ccd5df8e4f08e11f26209ca68752c1559a09fae276938
sysadmin@docker:~$
sysadmin@docker:~$ docker run -d --name db_mysql2 -e MYSQL_ROOT_PASSWORD='q1w2e3r4' -v mysql_vol:/var/lib/mysql  mysql
62e62f11319726962f8c697a2b0bd9ba361331e516f6a85078f08cc465c4855a
sysadmin@docker:~$
sysadmin@docker:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                 NAMES
62e62f113197   mysql     "docker-entrypoint.s…"   9 seconds ago    Up 8 seconds    3306/tcp, 33060/tcp   db_mysql2
c1ea3cf4e789   mysql     "docker-entrypoint.s…"   21 seconds ago   Up 21 seconds   3306/tcp, 33060/tcp   db_mysql1
sysadmin@docker:~$
```

Create 2 containers

After that, type the command below to create a database and insert the data via container db_mysql1:

```
docker exec db_mysql bash -c "mysql -uroot -pq1w2e3r4 -e \"CREATE DATABASE db_school; USE db_school; CREATE TABLE students (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, age INT NOT NULL); INSERT INTO students (name, age) VALUES ('bob', 21), ('John', 22);select * from students;\""
```

Then there will be a display below:

```
sysadmin@docker:~$ docker exec db_mysql1 bash -c "mysql -uroot -pq1w2e3r4 -e \"CREATE DATABASE db_school; USE db_school; CREATE TABLE students (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, age INT NOT NULL); INSERT INTO students (name, age) VALUES ('bob', 21), ('John', 22);select * from students;\""
mysql: [Warning] Using a password on the command line interface can be insecure.
id      name    age
1       bob     21
2       John    22
sysadmin@docker:~$
```

Execute the command to create a database and insert data

From the picture above, there are 2 data in the student table in the db_school database. Type the command below to add data to the table from container db_mysql2:

```
docker exec db_mysql1 bash -c "mysql -uroot -pq1w2e3r4 -e \"USE db_school; INSERT INTO students (name, age) VALUES ('Laura', 20), ('Andrew', 23);select * from students;\""
```

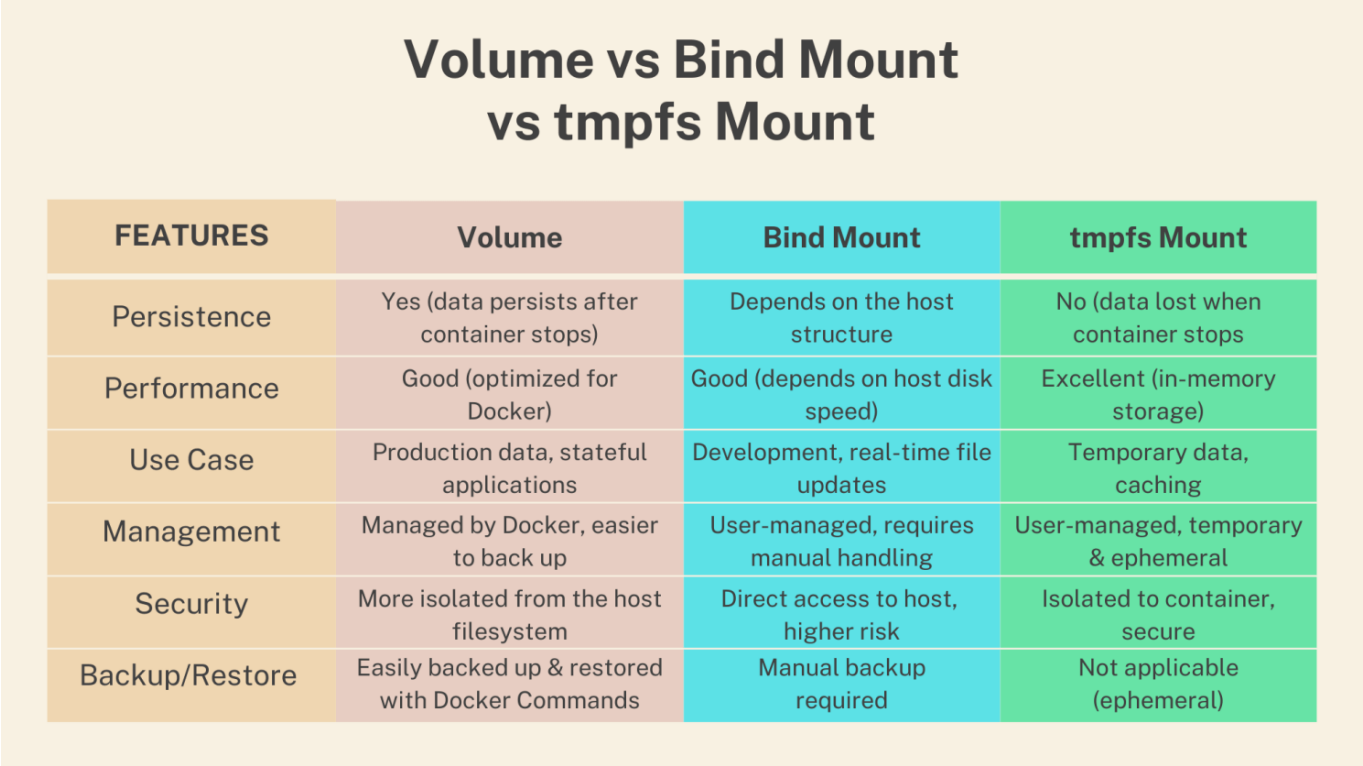
```
sysadmin@docker:~$ docker exec db_mysql1 bash -c "mysql -uroot -pq1w2e3r4 -e \"USE db_school; INSERT INTO students (name, age) VALUES ('Laura', 20), ('Andrew', 23);select * from students;\""
mysql: [Warning] Using a password on the command line interface can be insecure.
id      name    age
1       bob     21
2       John    22
3       Laura   20
4       Andrew  23
sysadmin@docker:~$
```

Execute the command to add data

In the picture above, there are 4 data in the student table so that the Docker volume data can be shared between containers on a server well.

Note

You can see the picture below to see the difference between the three storage:



Types of storage in Docker (Image credit from linkedin.com)

References

docs.docker.com
medium.com
linkedin.com
hub.docker.com
youtube.com