

# Section 9 - Persistent Data and Volumes

## 2 Persistent Data: Volumes

# Volume in Dockerfile

- A VOLUME can be also defined in a Dockerfile.

```
FROM debian
...
VOLUME /path/to/data-dir
...
```

- The *Dockerfile* of official images is good reference to see how *volumes* are used.

# Example - mysql (1a)

- [https://hub.docker.com/\\_/mysql](https://hub.docker.com/_/mysql)
- <https://github.com/docker-library/mysql/blob/26380f33a0fcd07dda35e37516eb24eaf962845c/5.7/>

```
FROM debian:stretch-slim
...
VOLUME /var/lib/mysql
...
```

# Example - mysql (1b)

- `/var/lib/mysql` => The MySQL "data directory". The MySQL "data directory" (a.k.a., "datadir") is the area where the Retain database would be stored
- `VOLUME /var/lib/mysql` => When we start a new container, a new *volume* with the content of the `/var/lib/mysql` directory will be created.
- Any file created in the `/var/lib/mysql` directory will be preserved even after the container is deleted.
- The contents of the `volume` can be delete only "manually" by executing the related `docker volume delete` command.

## Example - mysql (2)

- We can also see information about the *volume* by inspecting the mysql docker image:

```
# docker image pull mysql:5.7
# docker image inspect mysql:5.7
...
"Volumes": {
    "/var/lib/mysql": {}
},
...
```

# Example - mysql (3a)

- Create and inspect a mysql container:

```
# docker container run -d -e MYSQL_ALLOW_EMPTY_PASSWORD=true --name mysql
# docker container inspect mysql
"Mounts": [
  {
    "Type": "volume",
    "Name": "b256cc6cc11f6392d44db73f1b24e2925e7063dcc479701a56",
    "Source": "/var/lib/docker/volumes/b256cc6cc11f6392d44db73f1b24e2925e7063dcc479701a56/_data",
    "Destination": "/var/lib/mysql",
    "Driver": "local",
    ...
  },
],
"Config": {
  ...
"Volumes": {
  "/var/lib/mysql": {}
},
...
```

# Example - mysql (3b)

- From the output of the inspect command we can see the location of the volume:
  - in the host file system and => Source: `/var/lib/docker/volumes/b25.../`
  - in the container file system => Destination: `/var/lib/mysql`

# Example - mysql (4)

- Use the `docker volume ls` command to list the volumes available on a Docker host:

```
# docker volume ls
DRIVER      VOLUME NAME
local       b256cc6cc11f6392d44db73f1b24e2925e7063dcc479701a5e6ef28
```

Note:

The volume created from the mysql container is listed



# Example - mysql (5)

- Use `docker volume inspect` command to see detailed information about a volume

```
# docker volume inspect b256cc6cc11f6392d44db73f1b24e2925e7063dcc479701
...
  "CreatedAt": "2019-05-01T20:37:23+03:00",
  "Driver": "local",
  "Labels": null,
  "Mountpoint": "/var/lib/docker/volumes/b256cc6cc11f6392d44db73f1b24e2925e7063dcc479701a5e6ef2a6a1",
  "Name": "b256cc6cc11f6392d44db73f1b24e2925e7063dcc479701a5e6ef2a6a1",
  "Options": null,
  "Scope": "local"
...
```

Note: From the output of the `docker volume inspect` command we **cannot** see which container is using this volume.

# Example - mysql (6)

- On a linux host we could actually navigate to the volume location (`/var/lib/docker/volumes/b25.../_data`) and access the files that have been created from the container.
- On a Windows/MAC host running Docker toolbox we cannot do this because the file system of linux VM running the Docker daemon is not accessible (Note: This actually is NOT completely true).

# Example - mysql (7)

- Delete the mysql container and verify that the volume is preserved:

```
# docker container stop mysql
# docker container rm mysql
# docker volume ls
```

DRIVER	VOLUME NAME
local	b256cc6cc11f6392d44db73f1b24e2925e7063dcc479701a5e6

## *Note:*

It is not very user friendly handling volumes with such long names (IDs)  
The solution to this problem are the **Named Volumes**

# Named Volumes

- Named Volume => Friendly way to assign volume to container
- Use the `--volume` or `-v` option to defined a named volume.

```
-v <volume_name>:<container_path>
```

# Example mysql - Named Volume

```
# docker container run -v mysql-db:/var/lib/mysql -d -e MYSQL_ALLOW_EMPTY_PASSWORD=1 ...  
  
# docker volume ls  
DRIVER          VOLUME NAME  
local           b256cc6cc11f6392d44db73f1b24e2925e7063dcc479701a5e6ef28  
local           mysql-db
```

Note:

We can see that the new volume has a friendly name **mysql-db**.

# Example mysql - Named Volume (2a)

- Inspect the mysql container:

```
# docker container inspect mysql
...
"Mounts": [
  {
    "Type": "volume",
    "Name": "mysql-db",
    "Source": "/var/lib/docker/volumes/mysql-db/_data",
    "Destination": "/var/lib/mysql",
    "Driver": "local",
    "Mode": "z",
    "RW": true,
    "Propagation": ""
  }
]
...
```

# Example mysql - Named Volume (2b)

- Inspect the named volume *mysql-db*:

```
# docker volume inspect mysql-db
[
  {
    "CreatedAt": "2019-05-01T21:27:12+03:00",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/mysql-db/_data",
    "Name": "mysql-db",
    "Options": null,
    "Scope": "local"
  }
]
```

Note:

The name volume is created at a specific predictable location

**`/var/lib/docker/volumes/mysql-db/_data`**.

# Example mysql - Named Volume (3a)

- Use an existing volume with a new container.
- In this example we will delete the mysql container and create a new mysql2 container specifying the existing named volume "mysql-db".

```
docker container rm -f mysql # <== The "-f" option \
    is used to force the removal of a running container
```

```
docker container run -v mysql-db:/var/lib/mysql -d -e MYSQL_ALLOW_EMPTY_PASSWORD
```



# Example mysql - Named Volume (3b)

- Inspect the mysql2 container:

```
# docker container inspect mysql2
...
  "Mounts": [
    {
      "Type": "volume",
      "Name": "mysql-db",
      "Source": "/var/lib/docker/volumes/mysql-db/_data",
      "Destination": "/var/lib/mysql",
      "Driver": "local",
      "Mode": "z",
      "RW": true,
      "Propagation": ""
    }
  ]
...
```

*Note:* All data created from mysql container are preserved and available from mysql2 container.

# docker volume create (1)

- "Empty" named volumes can be also created with `docker volume create` command ahead of time.
- We can then configure a container to use it.

```
# docker volume create --help
```

```
Usage:  docker volume create [OPTIONS] [VOLUME]
```

```
Create a volume
```

```
Options:
```

<code>-d, --driver string</code>	Specify volume <b>driver name</b> (default <b>"local"</b> )
<code>--label list</code>	<b>Set</b> metadata <b>for</b> a volume
<code>-o, --opt map</code>	<b>Set driver</b> specific options (default <b>map[]</b> )

## docker volume create (2)

- The main reason we may want to create a volume is because this is the only way to specify a different volume driver.
- In this case we can use the `--driver` option to specify a volume driver other than the default one ("local").
- An example of a very common third party volume driver is REX-Ray.

REX-Ray driver can be used to access storage such as an Amazon EBS storage.