Lab: Docker Volumes

Introduction:

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers.

While bind mounts are dependent on the directory structure and OS of the host machine. Volumes are completely managed by Docker. Volumes have several advantages over bind mounts.

Objective:

- **Create and Manage volumes**
- **Start a container with volume**
- **Persist a volume using container**
- **Use and read-only volume**
- **Cleanup**

1. Create and manage volumes

1.1 Let's **list** the volumes by executing the below command.

```
# docker volume ls
```

1.2 Let's **create** a volume

```
# docker volume create
```

Output:

```
[root@docker-engine ~]#docker volume create
e60d523944bda6b894cf25a539596083911b2946fb6c72b721ed3860edfb55a9
```

1.3 Let's **create** a volume name of **my-vol**.

```
# docker volume create my-vol
```

Output:

```
[root@docker-engine ~]#docker volume create my-vol
my-vol
```

1.4 Let's **List** volumes by executing the below command.

```
# docker volume ls
```

Output:

```
[root@docker-engine ~]#docker volume ls
DRIVER    VOLUME NAME
local     77631b3d7fb3ba4f79a3a6691c9c6f3cb7adf3d40727a8dd52918f163789b0ca
local     e60d523944bda6b894cf25a539596083911b2946fb6c72b721ed3860edfb55a9
local     my-vol
```

1.5 Let's **inspect** volume **my-vol** by executing the below command.

```
# docker volume inspect my-vol
```

Output:

```
[root@docker-engine ~]#docker volume inspect my-vol
[
    {
        "CreatedAt": "2023-02-06T16:31:01+05:30",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/my-vol/_data",
        "Name": "my-vol",
        "Options": null,
        "Scope": "local"
    }
]
```

1.6 Let's **remove** a volume **my-vol** by executing the below command.

```
# docker volume rm my-vol
```

Output:

```
[root@docker-engine ~]#docker volume rm my-vol
my-vol
```

2. Start a container with a volume

If you start the container with a volume that does not yet exist, Docker creates the volume for you. The following step mounts the volume **myvol2** into **/app/** in the container.

```
# docker container run --name devtest -dit -v myvol2:/app
nginx:latest
```

Output:

```
[root@docker-engine ~]#docker container run --name devtest -dit -v myvol2:/app nginx:latest
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
01b5b2efb836: Pull complete
db354f722736: Pull complete
abb02e674be5: Pull complete
214be53c3027: Pull complete
a69afcef752d: Pull complete
625184acb94e: Pull complete
Digest: sha256:c54fb26749e49dc2df77c6155e8b5f0f78b781b7f0eadd96ecfabdcdfa5b1ec4
Status: Downloaded newer image for nginx:latest
0d8cd769310819bca88c0e8c5b388a8524a1f2a4edeb0d7d71443ca773220b21
```

2.1 Let's Inspect the container **devtest** to verify that the volume was created and **mounted** correctly.

```
# docker inspect devtest | grep "Mounts" -A 10
```

Output:

```
[root@docker-engine ~]#docker inspect devtest | grep "Mounts" -A 10
        "Mounts": [
            {
                "Type": "volume",
                "Name": "myvol2",
                "Source": "/var/lib/docker/volumes/myvol2/_data",
                "Destination": "/app",
                "Driver": "local",
                "Mode": "z",
                "RW": true,
                "Propagation": ""
            }
```

2.2 Let's **stop** the container.

```
# docker container stop devtest
```

Output:

```
[root@docker-engine ~]#docker container stop devtest
devtest
```

2.3 Let's **remove** the container.

```
# docker container rm devtest
```

Output:

```
[root@docker-engine ~]#docker container rm devtest
devtest
```

2.4 Let's **remove** the **volume.**

```
# docker volume rm myvol2
```

Output:

```
[root@docker-engine ~]#docker volume rm myvol2
myvol2
```

3. Persist a volume using a container.

3.1 In this step start a **nginx container** and populates the new volume **nginx-vol** with the contents of the container's **/usr/share/nginx/html** directory, which is where Nginx store sits default HTML content.

```
# docker container run --name nginxtest -dit -v\
nginx-vol:/usr/share/nginx/html nginx:latest
```

Output:

```
[root@docker-engine ~]#docker container run --name nginxtext -dit -v nginx-vol:/usr/share/nginx/html nginx:latest
7fc8772e79287ccdcf8cf4e309c1c8ea03afc159c45670d8b31f2f2cd7ea34bd
```

```
# cat > /var/lib/docker/volumes/nginx-vol/_data/index.html <<
EOF
"HELLO FROM DOCKER"
EOF
```

3.2 Let's search the **IP address** of the container.

```
# docker container inspect nginxtest | grep -i ipaddress
```

Output:

```
[root@docker-engine ~]#docker container inspect nginxtext | grep -i ipaddress
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.2",
                    "IPAddress": "172.17.0.2",
```

3.3 Let's **access** the web container to verify the data.

```
# curl 172.17.0.2
```

Output:

```
[root@docker-engine ~]#curl 172.17.0.2
"HELLO FROM DOCKER"
```

3.4 Let's **Stop** the container.

```
# docker container stop nginxtest
```

Output:

```
[root@docker-engine ~]#docker container stop nginxtext
nginxtext
```

3.5 Let's **remove** the volume.

```
# docker container rm nginxtest
```

Output:

```
[root@docker-engine ~]#docker container rm nginxtext
nginxtext
```

3.6 Let's **verify** if the data persist even after the container is deleted.

```
# cat /var/lib/docker/volumes/nginx-vol/_data/index.html
```

Output:

```
[root@docker-engine ~]#cat /var/lib/docker/volumes/nginx-vol/_data/index.html
"HELLO FROM DOCKER"
```

3.7 Let's **remove** the volume.

```
# docker volume rm nginx-vol
```

Output:

```
[root@docker-engine ~]#docker volume rm nginx-vol
nginx-vol
```

4. Use a read-only volume

4.1 In this step let's **mount** a volume as read-only.

```
# docker container run --name nginxtest -dit -v\
nginx-vol:/usr/share/nginx/html:ro nginx:latest
```

Output:

```
[root@docker-engine ~]#docker container run --name nginxtext -dit -v nginx-vol:/usr/share/nginx/html:ro nginx:latest
6d67c8652e59ed078159228229c3e4b40d56a655ff978966bca1959b667b400b
```

4.2 Let's **Inspect** the volume to verify that the volume was created and **mounted** correctly.

```
# docker container inspect nginxtest | grep "Mounts" -A 10
```

Output:

```
[root@docker-engine ~]#docker container inspect nginxtext | grep "Mounts" -A 10
        "Mounts": [
            {
                "Type": "volume",
                "Name": "nginx-vol",
                "Source": "/var/lib/docker/volumes/nginx-vol/_data",
                "Destination": "/usr/share/nginx/html",
                "Driver": "local",
                "Mode": "ro",
                "RW": false,
                "Propagation": ""
            }
```

4.3 Let's **verify** if its readonly by creating a file.

```
# docker container exec nginxtest touch \
/usr/share/nginx/html/testfile
```

Output:

```
[root@docker-engine ~]#docker container exec nginxtext touch /usr/share/nginx/html/testfile
touch: cannot touch '/usr/share/nginx/html/testfile': Read-only file system
```

4.4 Let's **stop** the container.

```
# docker container stop nginxtest
```

Output:

```
[root@docker-engine ~]#docker container stop nginxtext
nginxtext
```

4.5 Let's **remove** the container.

```
# docker container rm nginxtest
```

Output:

```
[root@docker-engine ~]#docker container rm nginxtext
nginxtext
```

4.6 Let's **remove** the volume.

```
# docker volume rm nginx-vol
```

Output:

```
[root@docker-engine ~]#docker volume rm nginx-vol
nginx-vol
```

Cleanup:

5.1 Let's **cleanup**, by executing the below command.

```
# docker container rm `docker container ls -a -q` -f
```

```
# docker image rm `docker image ls -q` -f
```

Output:

```
[root@docker-engine ~]#docker image rm `docker image ls -q` -f
Untagged: nginx:latest
Untagged: nginx@sha256:c54fb26749e49dc2df77c6155e8b5f0f78b781b7f0eadd96ecfabdcdfa5b1ec4
Deleted: sha256:9eee96112defa9d7b1880f18e76e174537491bcec6e31ad9a474cdea40f5abab
Deleted: sha256:9cadfb09eeeb8021f9bcda73607dc61ccff20c8fdc44af61267c788f174e9bdb
Deleted: sha256:f35b4ecd12ae382160d9a00f3b6956fec245c644e79e71d2bd0998403ea93a00
Deleted: sha256:0023ddac86d5e471a726b8b20d0422defc75eadbf85b3f98bfd02351f263bc57
Deleted: sha256:4bb459f6844bfd227980b8037560c81a5bf565fc81fd288052b84acbcf660b42
Deleted: sha256:0e16e930455b149d8ce0b75aaaa31b168c82d658527bfe184db954073457cb60
Deleted: sha256:bd2fe8b74db65d82ea10db97368d35b92998d4ea0e7e7dc819481fe4a68f64cf
```

```
# docker volume prune -f
```

Output:

```
[root@docker-engine ~]#docker volume prune -f
Deleted Volumes:
e60d523944bda6b894cf25a539596083911b2946fb6c72b721ed3860edfb55a9

Total reclaimed space: 0B
```