Lab 6 - Configuring Persistent Docker Containers

Introduction

A data volume is a specially designated directory within one or more containers that bypasses Docker's storage driver and interacts directly with the host file system. Data volumes provide several useful features for persistent or shared data:

- Volumes are initialized when a container is created. If the container's base image contains data at the specified mount point, that existing data is copied into the new volume upon volume initialization.
- Data volumes can be shared and reused among containers.
- Changes to a data volume are made directly to the host filesystem (vs. going through the storage driver)
- Changes to a data volume will not be included when you update an image.
- Data volumes persist even if the container itself is deleted.

Data volumes are designed to persist data, independent of the container's lifecycle. Docker therefore never automatically deletes volumes when you remove a container, nor will it "garbage collect" volumes that are no longer referenced by a container.

In this lab, we will look at **Docker Volumes**. By Docker Volumes, we are essentially going to look at how to **manage data** within your Docker containers.

1. Implementing a volume

1.1 Login as **"root"** user on **aio110** host:

Copy

```
ssh root@aio110
```

1.2 Pull down the official **Nginx** image.

Copy

```
docker pull nginx
```

1.3 Create a new volume named barcelona.

Copy

```
docker volume create --name barcelona
```

Output:

```
barcelona
```

Your new volume is created on the host in the /var/lib/docker/volumes directory.

1.4 List the volumes on your Docker host

Copy

```
docker volume ls
```

Output:

```
DRIVER              VOLUME NAME

local               barcelona
```

You should see the barcelona volume listed. You may see other volumes as well.

1.5 Instantiate a Docker container with your barcelona volume.

Copy

```
docker run -it -v barcelona:/barcelona --name volumeslab nginx
/bin/bash
```

This command creates an Nginx container named volumeslab. It also mounts your volume barcelona at **/barcelona** in the root of the container's file system. Then, the **/bin/bash** command opens an interactive shell inside the container.

Sample Output:

```
root@ccc0bd4d5237:/#
```

You are at your running container's shell prompt.

1.6 Change into the **/barcelona** directory.

Copy

```
cd /barcelona
```

1.7 Create a file.

Copy

```
touch file.txt
```

1.8 List the directory contents to make sure the file exists.

Copy

```
ls
```

Note: Press **<ctrl+P+Q>** to exit the container shell.

This key combination leaves the container running. You should return to your Docker host's shell.

1.9 Ensure your container is still running:

Copy

```
docker ps
```

The output should be similar to:

```
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES

b4536a486fe5        nginx               "/bin/bash"         18 seconds
ago      Up 17 seconds       80/tcp              volumeslab
```

Note: The STATUS should show **Up** instead of Exited.

2. Understand how Docker represents volume data in the file system

Docker manages volumes independently from the storage driver system it uses to manage the container layers. This allows for data persistence; The volume is not destroyed when the container is destroyed.

In this task, you're going to take a quick look at where Docker stores volume data. Then, you'll change the volume on the Docker host file system and see the change appear in the container.

2.1 Inspect the barcelona volume values.

Copy

```
docker volume inspect barcelona
```

Your output should be similar to:

```
[
    {
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/barcelona/_data",
        "Name": "barcelona",
        "Options": {},
        "Scope": "local"
    }
]
```

The volume mount point above is: **/var/lib/docker/volumes/barcelona/_data**

2.2 Elevate your user privileges:

Change to the barcelona data directory.

Copy

```
cd /var/lib/docker/volumes/barcelona/_data
```

2.3 List the directory contents.

Copy

```
ls
```

Output:

```
file.txt
```

Notice the file you previously created file.txt is listed in the host directory.

2.4 Create a new file.

Copy

```
touch file2.txt
```

Check that **file.txt** and **file2.txt** are in the directory.

Copy

```
ls
```

Output:

```
file2.txt file.txt
```

2.5 Log back into the Nginx container’s shell.

Copy

```
docker exec -it volumeslab /bin/bash
```

2.6 List the contents of the /barcelona directory.

Copy

```
ls /barcelona
```

Output:

```
file.txt file2.txt
```

The file you created from the Docker host shell file2.txt should appear inside your running container.

2.7 Exit the container and return to your Docker host.

Copy

```
exit
```

3. Deleting a volume

By default, when you destroy a container Docker does not remove any volumes associated with the container. You can, however, delete a given volume with the **docker volume rm** command.

3.1 Stop the **volumeslab** container you created.

Copy

```
docker stop volumeslab
```

Output:

```
volumeslab
```

3.2 Remove the container.

Copy

```
docker rm volumeslab
```

Output:

```
volumeslab
```

Removing the containers does not remove the barcelona volume the container was using.

3.3 Ensure the barcelona volume still exists.

Copy

```
docker volume ls
```

Output:

```
 DRIVER              VOLUME NAME

 local               barcelona
```

You may see other volumes listed in addition to barcelona.

3.4 List the contents of the barcelona volume directory.

Copy

```
ls /var/lib/docker/volumes/barcelona/_data
```

Output:

```
file.txt file2.txt
```

The volume and its data are still intact.

3.5 Remove the volume.

Copy

```
docker volume rm barcelona
```

Output:

```
 barcelona
```

3.6 Ensure the volume was removed.

Copy

```
docker volume ls
```

Output:

```
  DRIVER              VOLUME NAME
```

The barcelona volume is no longer listed, although other volumes may be.

4. Volumes Usecase: Recording Logs

4.1 Setting up an app with logging

4.1.1 Create a volume called nginx_logs:

Copy

```
docker volume create --name nginx_logs
```

Output:

```
nginx_logs
```

4.1.2 Create a folder called **public_html** inside your home directory

Copy

```
mkdir ~/public_html && cd ~/public_html
```

4.1.3 Inside your **public_html** folder, create a file called **index.html** and write some lines of text on the file

Copy

```
cat > index.html <<EOF

Hello web application

EOF
```

4.1.4 Run the custom **training/nginx:17.06** image and map your public_html host folder to a directory at **/usr/share/nginx/html**. Also mount your **nginx_logs** volume to the /var/log/nginx folder. Name the container **nginx_server**:

Copy

```
docker container run -d -p 8080:80 --name nginx_server -v
~/public_html:/usr/share/nginx/html -v nginx_logs:/var/log/nginx
training/nginx:17.06
```

4.1.5 Run the below command , to find the host port which is mapped to **port 80** on the container. In your browser, access the URL and **port nginx** is exposed on.

Copy

```
docker container ls
```

Output:

```
CONTAINER ID        IMAGE                  COMMAND
CREATED             STATUS              PORTS                    NAMES

d34f3c580ab8        training/nginx:17.06   "nginx -g 'daemon ..."   10
seconds ago      Up 9 seconds        0.0.0.0:8080->80/tcp
nginx_server
```

4.1.6 Verify you can see the contents of your index.html file from your public_html folder on your browser

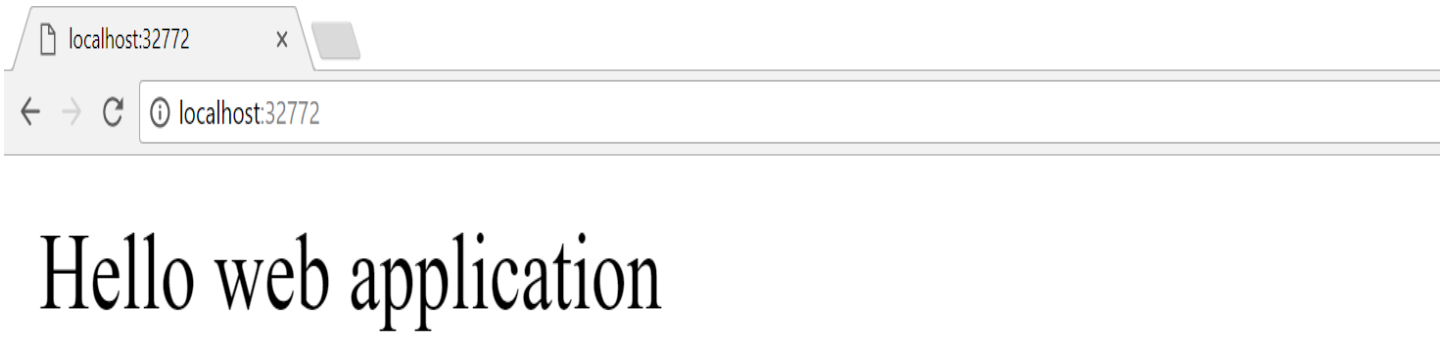
Copy

```
curl http://localhost:8080
```

Open a web browser, paste the below URL to verify the contents of **index.html**.

Copy

```
http://localhost:8080
```



Note: If your unable to access, please verify the port-forwarding is configured.

4.1.7 Get terminal access to your container:

Copy

```
docker container exec -it nginx_server bash
```

Sample Output:

```
root@8cf0916bdb00:/#
```

4.1.8 Put some additonal text into **/usr/share/nginx/html/index.html**, and then exit the terminal.

Copy

```
cat > /usr/share/nginx/html/index.html <<EOF

Recording logs

EOF
```

Copy

```
exit
```

4.1.9 Verify you can see the contents of your updated **index.html** file from your public_html folder on your
browser.

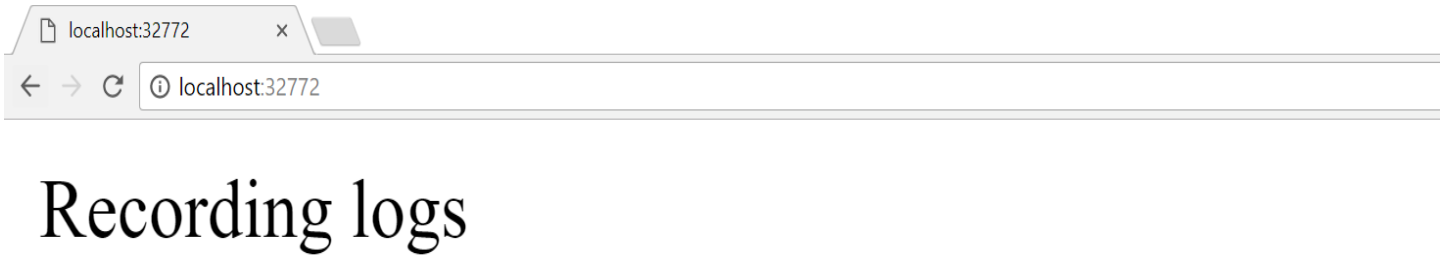
Copy

```
curl http://localhost:8080
```

Refresh your browser to Verify the updated text.

Copy

```
http://localhost:8080
```



4.2 Inspecting Logs on the Host

4.2.1 Get terminal access to your container again:

Copy

```
docker container exec -it nginx_server bash
```

Sample Output:

```
root@8cf0916bdb00:/#
```

4.2.2 Change directory to **/var/log/nginx**:

Copy

```
cd /var/log/nginx
```

4.2.3 Check that you can see the **access.log** and **error.log** files.

Copy

```
ls
```

Output:

```
access.log  error.log
```

4.2.4 Run the below tail command and Refresh your browser a few times and observe the log entries being written to the file. Exit the container terminal after seeing a few live log entries.

Copy

```
tail -f access.log
```

Open a web browser, paste the below url and keep refresh the page to observe the log entries being written to the file in the terminal.

Copy

```
http://localhost:8080
```

Note: Press **<ctrl+c>** to exit from the tail command.

Copy

```
exit
```

Copy

```
cd
```

4.2.5 Run the below command and copy the path indicated by the “Mountpoint” field; path should be **/var/lib/docker/volumes/nginx_logs/_data**.

Copy

```
docker volume inspect nginx_logs
```

Output:

```
[

    {

        "Driver": "local",
```

```
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/nginx_logs/_data",
        "Name": "nginx_logs",
        "Options": {},
        "Scope": "local"
    }
]
```

4.2.6 Check for the presence of the access.log and error.log files, then follow the tail of access.log:

Copy

```
ls /var/lib/docker/volumes/nginx_logs/_data
```

Output:

```
access.log  error.log
```

Copy

```
 tail -f /var/lib/docker/volumes/nginx_logs/_data/access.log
```

Output:

```
172.17.0.1 - - [04/Mar/2018:04:45:25 +0000] "GET / HTTP/1.1" 403 169
"-" "curl/7.29.0" "-"

172.17.0.1 - - [04/Mar/2018:04:48:36 +0000] "GET / HTTP/1.1" 200 15 "-
" "curl/7.29.0" "-"

172.17.0.1 - - [04/Mar/2018:04:50:13 +0000] "GET / HTTP/1.1" 200 15 "-
" "curl/7.29.0" "-"
```

```
172.17.0.1 - - [04/Mar/2018:04:50:26 +0000] "GET / HTTP/1.1" 200 15 "-
" "curl/7.29.0" "-"

172.17.0.1 - - [04/Mar/2018:04:50:40 +0000] "GET / HTTP/1.1" 200 15 "-
" "curl/7.29.0" "-"

172.17.0.1 - - [04/Mar/2018:04:50:42 +0000] "GET / HTTP/1.1" 200 15 "-
" "curl/7.29.0" "-"

172.17.0.1 - - [04/Mar/2018:04:50:43 +0000] "GET / HTTP/1.1" 200 15 "-
" "curl/7.29.0" "-"

172.17.0.1 - - [04/Mar/2018:04:51:22 +0000] "GET / HTTP/1.1" 200 15 "-
" "curl/7.29.0" "-"

10.1.1.91 - - [04/Mar/2018:04:54:26 +0000] "GET / HTTP/1.1" 200 15 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/64.0.3282.186 Safari/537.36" "-"

10.1.1.91 - - [04/Mar/2018:04:54:26 +0000] "GET /favicon.ico HTTP/1.1"
404 571 "http://localhost:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186
Safari/537.36" "-"

...

...
```

Open a web browser, paste the below url and keep refresh the page to observe the log entries being written to the file in the terminal.

Copy

```
http://localhost:8080
```

Note: Press **<ctrl+c>** to exit from the tail command.

4.3 Sharing Volumes

4.3.1 Make sure that your nginx_server container from the last step is still running; if not, restart it.

Copy

```
docker container ls
```

Output:

```
CONTAINER ID        IMAGE                  COMMAND
CREATED             STATUS              PORTS                    NAMES

a5fcc60f6491        training/nginx:17.06   "nginx -g 'daemon ..."   7
minutes ago       Up 7 minutes        0.0.0.0:8080->80/tcp
nginx_server
```

4.3.2 Run a new centos container and mount the nginx_logs volume to the folder **/data/mylogs** as read only,
with bash as your process:

Copy

```
docker container run -it -v nginx_logs:/data/mylogs:ro centos:7 bash
```

4.3.3 In your new container's terminal, change directory to **/data/mylogs**

Copy

```
cd /data/mylogs
```

4.3.4 Confirm that you can see the access.log and error.log files.

Copy

```
ls
```

Output:

```
access.log  error.log
```

4.3.5 Try and create a new file called **text.txt**

Copy

```
touch test.txt
```

Notice: How it fails, because we mounted the volume as **Read only**.

Copy

```
exit
```

Copy

```
cd
```

5. Cleanup

5.1 To remove all the containers run the below commands.

Copy

```
docker rm `docker ps -a -q` -f
```

5.2 To remove all the images run the below commands.

Copy

```
docker rmi `docker images -q` -f
```