Lab 6: Docker Storage

Learning Outcomes:

You should ensure you have watched the ONLINE video lectures each week before the lab starts. Labs require that you have watched the videos and are present in the lab. This week we want to ensure that you have your development environemt working and you are becoming more familiar and comfortable with GIT and Docker. Marks for this week are based on Part A, the work in the lab. Part B is required for feedback on the lab, but no marks will be assigned. ***Everyone should now have an IPV4 virtual machine in the cloud. Either Okeanos or Amazon EC2. If not, you need to tell your Tutor who can help assign a Virtual Machine with IPV4. Local VM will no longer be used. Azure Instances will now be allocated***

After completing the exercises in this lab you should be able:

- More Linux shell (command line interpreter) experience
- If you are having trouble with Linux commands do the code academy online tutorial.
    - https://www.codecademy.com/en/courses/learn-the-command-line
- Become more familiar with Docker Storage
- Build, run, modify, stop, move and restart a Docker Container. Look at the commandline reference tools.
    - https://docs.docker.com/reference/commandline/exec/
- Extend the API of the flask application

Organisation

Please attempt this lab individually.

Lab Assessment - 30% of overall Module

Continous assessment is based on ongoing assesment throughout the module. There are 2 parts to this weeks assessment each of which is worth 50% of the lab assessment mark. Personal circumstances will be considered on a case by case basis with supporting evidence.

- ✓ Part A. – Assessed by the **Tutor Demonstrator** before you leave the lab. Marks are allocated for completion of parts, but you must provide evidence of completion where feasible. (***Grading is between 0-100***). You must review your lab progress with the Tutor Demonstrator before you leave the lab. Failure to do so may result in a score of 0. **If you didn't complete last weeks lab, there are extra marks this week to follow the walkthough provided because you need to have Docker working to do this weeks lab.**

- ✓ Part B. – Assessed by the **Lecturer** before the next lab. you must upload your report in webcourses before the end of the day. Late reports not graded. If you have not attended then report also not graded. (Grading is between 0-0).

Part A – In Lab Assessment

1. Basics: Building a new container with a new server.py and upload a file to the container and verify the upload (25 Marks)

a. Ensure you have watched the lecture videos for this week. Cloud Storage and Cloud Databases.

b. Create new GITHUB Repo lab6 (with a Readme File in it) and clone it to a Virtual Machine

c. Clone a copy of my starter repo

https://github.com/paulfdoyle/cc-lab6-storage.git

d. Copy the my_application/ and Dockerfile from my repo into your repo

e. Build the new docker image and call it lab6

f. Create a container from lab6 image as below and call the container webserver1. The –d runs the container in the background (as a daemon), and –p maps port 8080 to the container.

```
docker run –d –p 8080:5000 –name webserver1 lab6
```

g. Make sure it works from a browser <insert machinename>

http://machinename:8080/

h. Upload a file to the container which should look similar to the following

i. Verify that the file is now in the container using the docker exec –it command using the following methods (one breaks into the VM the other looks inside it). Data is stored in the /data directory.

```
docker exec –it <containerid> /bin/bash

docker exec –it <containerid> ls /data
```

2. Storage Local to the Container (20 marks)

An image is the blueprint for creating a container. Once a container has been created and run it can be stopped and restarted. We can create multiple containers from the same image. Changes in one container will not be copied to other containers as they are distinct containers. So when we upload a file into a container it is only in that container.

a. Stop the running container webserver1 that has a file uploaded. (**`docker stop`** command)

b. Create and run a new container using the **lab6** image and call it *webserver2* and verify if the uploaded file is in the new container. (break into the container or do an ls of the /data directory in the container to check)

c. Restart container webserver1 (use **`docker ps –a`** to see the stopped conainers ID.) and verify if the container still contains the uploaded file.

3. Moving and reanimating a Containers (10)

An image can be moved to another machine using a dockerfile as the blueprint for a container, but we can also move an actual container to a different virtual machine. To move a container you have to stop it, export it, copy it to the new machine then import it again to start it running. You will need a second VM to do this. Pair up with someone else in the lab and swap containers using the **SCP** command. See https://linuxacademy.com/blog/linux/ssh-and-scp-howto-tips-tricks/ if required. All data within the container will also be copied.

a. Stop a running container webserver1 that has a file uploaded and then export it to a compressed tar file. (**`docker stop & docker export`** commands).

```
docker export –o lab6.tar.gz <containerID>
```

b. Copy the lab6.tar.gz to another virtual machine using scp.

```
scp <filename>   <user>@machinename:
```

c. Make sure docker is running on the virtual machine you copied lab6.tar.gz to. Now import it using the following command. (the –c '…' syntax adds back the Dockerfile Meta data which was removed during the export.

```
docker import –c 'EXPOSE 5000' –c 'WORKDIR /my_application' –c
'CMD mkdir /data' –c 'CMD python server.py' lab6.tar.gz
```

d. Find the newly imported image ID and give it a name lab6copy using the following

```
docker tag <imageID>  lab6copy
```

e. Generate a container based on this new image publishing port 8080:5000 as previous (**docker run**) and verify from a browser that the container is running and you can see the file within it.

f. Go back to your own VM and run a container and upload an image file. Now stop the container running and restart the container (docker restart <conainterID> to verify it can still see the uploaded file.

4. *Quick Quiz (5)*

- Name 3 ways to verify if data has been uploaded to a containers /data directory
- What is the difference between **docker run** and **docker restart**
- What does **docker –ps a** do?
- How do you find the DNSName of the virtual machine you are using to test from a browser?
- What does **docker run –d** do ?
- What does **docker tag** do ?
- What does **docker logs –f** do ?

5. *Mounting a directory from the local VM into a Container (10)*

It is possible for a local directory to be mounted inside of a container. This is handy when you only want the container to act as a conduit to the data rather than a place to store the data. Several other advantages exist also. You can now see the data from multiple containers. More details here https://docs.docker.com/userguide/dockervolumes/

a. Create a local directory on the virtual machine (not inside a container) in your current working directory and call it **data**

b. Create a new container and mount the directory from the virtual machine as follows. (watch for the double minus before name…)

```
docker run –d –p 8080:5000 –v $(pwd)/data:/data --name
givecontaineraname  lab6
```

c. Verify from the browser that there are no files stored in the data directory then add a file from the virtual machine and see if it is visible from the browser.

d. Upload a file from the browser and verify that it is visible from within the virtual machine.

6. Mounting a Data Container from a Container (10)

We may want to keep our data safe inside a container, but share that data container with other containers who are allowed to read and write data to the container. **Make sure you delete all existing containers before starting this section.** More details here https://docs.docker.com/userguide/dockervolumes/

a. We can create a data container based on the hello-world image which we should already have downloaded from lab 4.

```
docker create –v /data --name datastore1 hello-world  /bin/true
```

where /data is data location in the container

datastore1 is the name of the container

hello-world is the image base to use

/bin/true is a command but will not execute (makes sure container does not run)

You can do a docker ps –a and see the container, but it won't be running.

b. Mount this new data container into a new container based on lab6 image.

```
docker run –d --name uploader1 --volumes-from datastore1 lab6
```

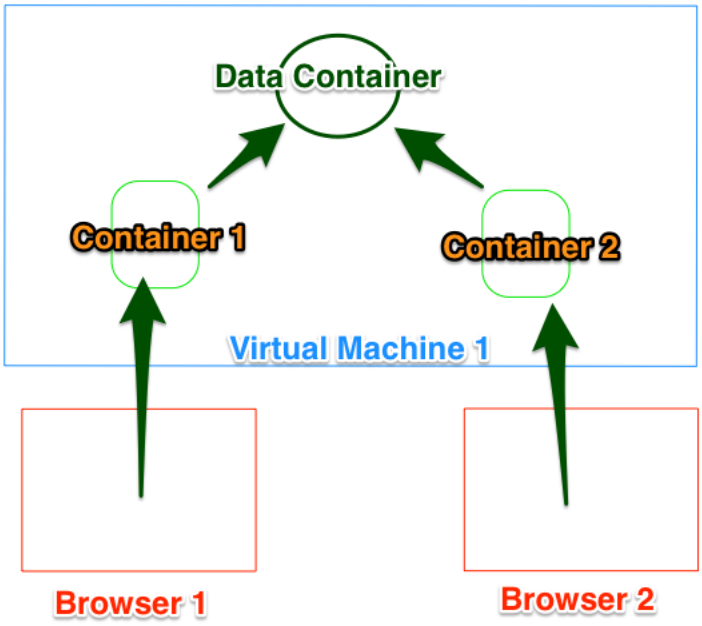
c. Break into the container uploader1 and check the /data directory exists.

d. Mount this new data container into a second container based on lab6 image and verify that inside this container the /data directory also exists.

e. Manually create a file in the /data directory in the first container and verify that it is now visible inside the second container.

7. Uploading from multiple locations (20)

Create two flask webcontainers (based on lab6 server.py) which are both connected to the same datacontainer and show how to upload files to the datacontainer from two different browsers.



Part B – Webcourses Report

---

1. Submit your Webcourses Weekly Report (0 Marks)

The lab report should contain the 4 sections as below this week.

1. Explain any areas of difficulty in this week's lab (100 words or less)
2. Explain what went well in this week's lab (100 words or less)
3. Provide a link to your GITHUB Repo
4. Any other comments?

Part C – Useful Docker Commands

Useful commands

```
docker build –t lab6 .
docker images
docker tag <imageid>

docker run lab6  (or use imageid instead of tag)
docker run –d lab6
docker run –d –p 8080:5000 lab6

docker exec –it <containerid> /bin/bash  (logs you into the container)
docker exec –it <containerid> ls /data   (lists files inside a container)

docker logs –f <containerid>  (gives running logs from the container)
docker ps
docker ps –a

docker run –d –v /local/dir:/data --name givecontaineraname <imageid>
(mount a local directory /local/dir  into a container at /data)

docker create –v /data --name datastore1 hello-world /bin/true
(create datacontainer image datastore1 using the hello-world base image.)
docker run –d --name uploader1 --volumes-from datastore1 <imageid>
(create a new container based on an imageid and mount a datacontainer)


docker stop <containerid>
docker restart <containerid>
docker kill  <containerid>

docker rmi –f <imageid>
docker rm <containerid>

docker import –c 'CMD python server.py'  <filename>
(import a container and add back meta data in the form of commands>
```