Homework 5

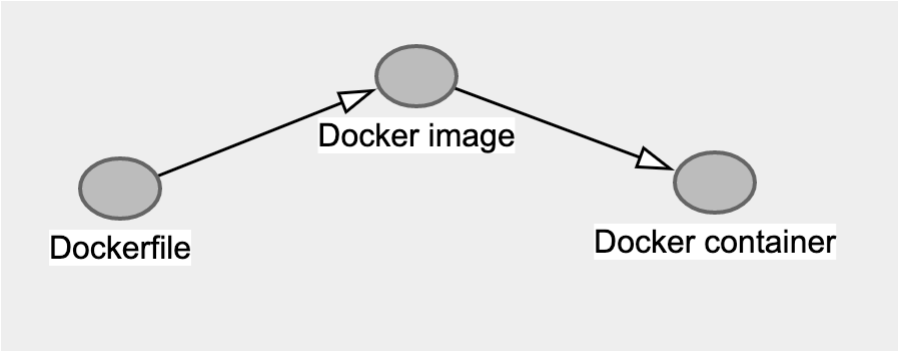
Code

Scott Sun

Part 1:

Q1

- Docker container: an object instance of a Docker image
- Docker image: the template (analogous to class in the context of programming) for creating the realization, Docker container
- Dockerfile: the blueprint to build Docker image



Q2

`jupyter/pyspark-notebook` has a size of 4.04GB.

Q3

The following command binds port `8888` in the container to port `823` on the host machine.

Code

The url used to connect the `JupyterLab` is:
http://127.0.0.1:823/lab?token=b845fac44c27006664b1919e20d36514c131cce8a85088fc (http://127.0.0.1:823/lab?token=b845fac44c27006664b1919e20d36514c131cce8a85088fc)

Q4

The code return `0.977408`

Q5

`docker ps` does not return any container in the output table. `docker ps –a` return all the containers. The difference is due to the fact that `docker ps` by default only show containers that are running and `–a` displays all the containers.

Q6

The file saved before does not exist because by running the image we instantiate a completely new container. The file saved previously is saved in the first container.

Q7

The `CONTAINER ID` of the original container is `fb67bf8dbae4` , so we run the following command.

Code

Then, we use the following command to get the token of the JupyterLab, which is:
3f5232de749dc4fcb90b5241083308c5dbc067edfde492ad

Code

Finally, we use the url with the updated token to launch JupyterLab and access our original file :
http://127.0.0.1:823/lab?token=3f5232de749dc4fcb90b5241083308c5dbc067edfde492ad
(http://127.0.0.1:823/lab?token=3f5232de749dc4fcb90b5241083308c5dbc067edfde492ad)

Q8

Assuming the target directory on the local machine is `/usr/jupyter/pyspark` , we copy the files from the container using the following command.

Code

Part 2

(1)

After `cd` to the directory `flask` , use the following commands to build the image and instantiate an container based on the image.

Code

The link to Flask web-app is: localhost:8000

(2)

After `cd` to the directory `dockerfile2` , use the following commands to build the image and instantiate an container based on the image.

Hide

```
        docker image build –t py–r–hw–example .
        docker run ––name p2–2 py–r–hw–example
```

Part 3

Q9

First, we build the image based on the Dockerfile. After we log into Docker Hub, we need to re-tag the image with the username at the front (i.e., `scotsun/<image-name>`) and finally push as a repo.

Code

The link to the image is: https://hub.docker.com/repository/docker/scotsun/py-r-hw-example (https://hub.docker.com/repository/docker/scotsun/py-r-hw-example)

Q10

Code

```
[ms1008@dcc-login-02  ~/bios823_hw5_part3 $ ls -l
total 4616616
-rwxr-xr-x. 1 ms1008 dukeusers 4125341413 Nov 15 17:15 py-r-hw-example_1.0.sif
-rw-r--r--. 1 ms1008 dukeusers         81 Nov 12 19:12 requirements.R
-rw-r--r--. 1 ms1008 dukeusers         23 Nov 12 19:11 requirements.txt
-rw-r--r--. 1 ms1008 dukeusers        439 Nov 12 19:11 test.R
-rw-r--r--. 1 ms1008 dukeusers       1618 Nov 12 19:11 test.py
```

The `SIF` container file has a size of 4GB. It has been uploaded to `Sylabs` as a signed repo. Here is the link: https://cloud.sylabs.io/library/scotsun/bios823/py-r-hw-example.sif (https://cloud.sylabs.io/library/scotsun/bios823/py-r-hw-example.sif)