Containerization Report

Abdullah McDonald *Department of Computer Sciences University of the Western Cape* [3708949@myuwc.ac.za](mailto:3708949@myuwc.ac.za)

1. INTRODUCTION

Did you ever run into a problem where your programming solution works on one computer but doesn’t work on the next because it is missing some libraries? Did you ever try to test that solution on computers that have all the necessary dependencies installed but due to permission requests you still couldn’t run the solution? Well, that ends with container environments such as Docker and Singularity.

Docker tries to solve the dependency hell [1] by running your programming solution within a container. So what is a container? Containers virtualize the host operating system and ‘contains’ all necessary dependencies that allow you to run your programming solution. They are also very lightweight [1]. Unlike virtualization, Docker containers make use of the hosts’ kernel. Docker containers provide another advantage over virtual machines, you can run multiple Docker containers at once and don’t have to worry about one container affecting the performance of the other as Docker containers are run in isolation.

On the other hand, Singularity is another container environment solution. Singularity containers differ from Docker containers in that it is not as lightweight as Docker, allows you to access the host OS whereas Docker does not and Singularity can only be installed on Linux operating systems whereas Docker can be installed on any operating system. This comes in handy when you need files that are too big to store within a container. Also, Singularity provides the added benefit of allowing you to convert your Docker images for use with Singularity [2].

In this report, I explain in detail how I made use of Docker to containerize my Twitter Analysis solution as well as how I converted my Docker image for use with Singularity.

The structure of this report is as follows; section 1 explains how Docker was used, section III explains how the Docker image was converted for use this Singularity, section IV details the problems encountered during this project and section V concludes the report.

1. DOCKER

Following Figure 1, after I successfully installed Docker I proceeded to create a Dockerfile that contains all dependencies for my Twitter Analysis solution. Figure 2 shows the details of the Dockerfile. I started by first importing a base image on

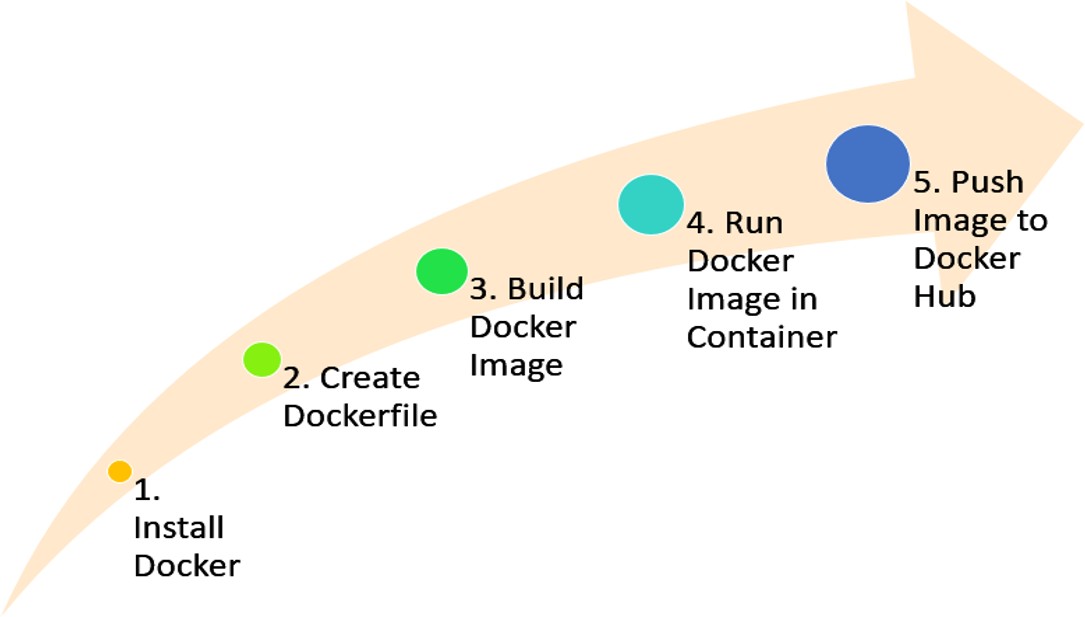


Fig. 1: Docker process

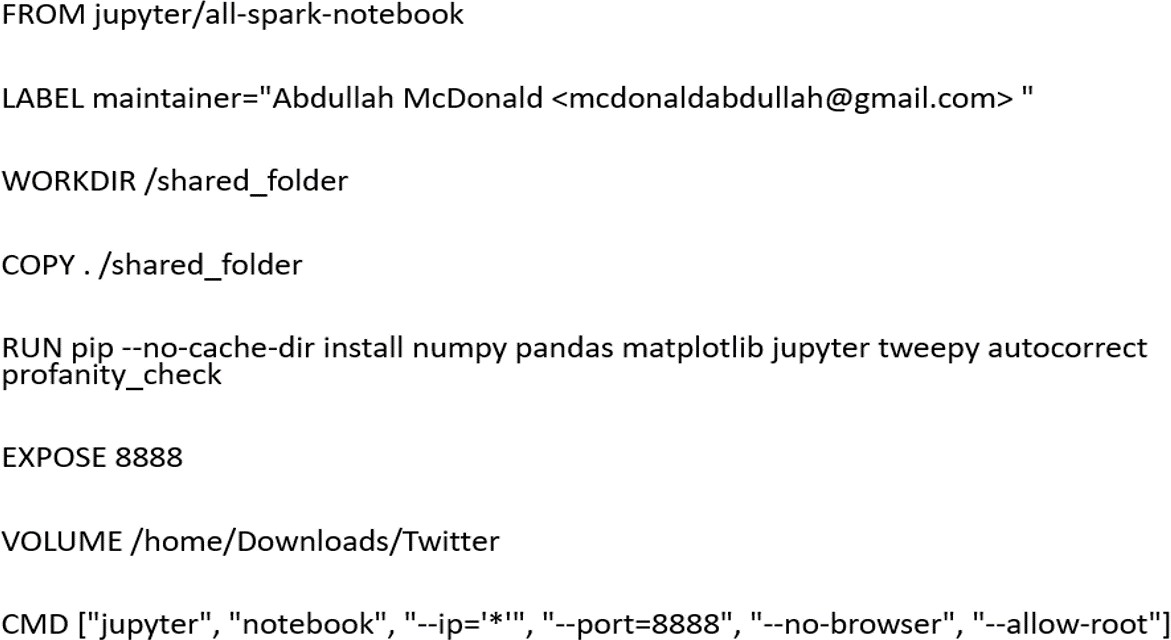


Fig. 2: Details of Dockerfile

which my Docker image would be built. The jupyter/all-spark- notebook base image contains a jupyter notebook as well as all the spark libraries that are needed for the solution. Next, I mention a Maintainer Label, this specifies the creator of the Docker image. The WORKDIR command specifies the working directory that we would like to work in. All libraries that I need that have not been included in the based image is then mentioned in the RUN command. Since Jupyter Notebook makes use of port number 8888, we need to be able to access

this port and this is exactly what the EXPOSE command does. The VOLUME command is to bind any directory to the image. Since the /home/abdullah/Downloads/Twitter directory contains all the files needed to run my Twitter Analysis solution, I choose to bind this directory to my image. The CMD command specifies the command needed to start up Jupyter Notebook once we are in our container.

Now for Step 3 and 4 shown in Figure 1, the Docker Image is ready to be built using the Dockerfile. To build the Docker Image, we run the command:



Fig. 3: Building Image from Dockefile command

The ‘–t’ command specifies the tag associated with the Image and the ‘.’ specifies that the Dockerfile is in the current working directory. Next, we run the Docker image in a container with the command:



Fig. 4: Running Image in a Container command

The ‘-it’ command is to make the container interactive, the ‘-e GRANT SUDO=yes –user root’ allows us to save files to the volume mentioned in the Dockerfile and the ‘-p 8888:8888’ allows us to access the Jupyter Notebook which uses port 8888. For the ‘[Image ID]’ section, the correct Image ID needs to be specified and this can be gathered using the command ‘docker images’. The correct image ID will be associated with the name ‘twitter solution’. Step 5 involves pushing the Docker Image to Docker Hub. After logging in to Docker Hub, we push the image using the following command:

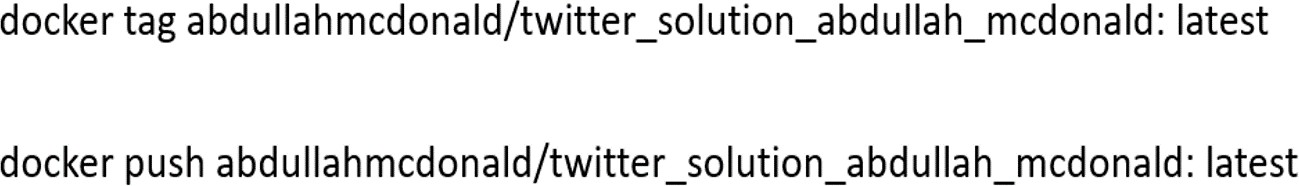


Fig. 5: Pushing Image to Docker Hub commands

The ‘tag’ section in the first allows us to name the repository where the image will be stored. The ‘push’ section in the second command allows the image to be published on Docker Hub.

1. SINGULARITY

After Singularity was installed, it was quite easy to convert my Docker image to a Singularity Image. Figure 6 All I needed to do was to pull the image from Docker Hub and then convert it to a Singularity Image File (SIF). This was done using two commands:

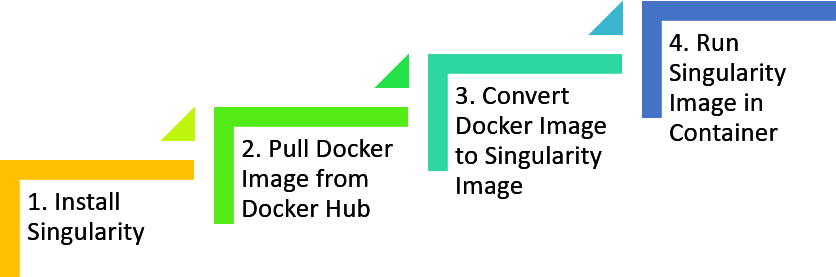


Fig. 6: Singularity process

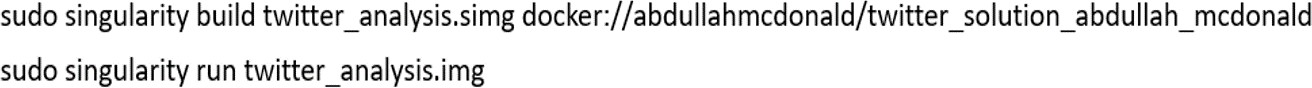


Fig. 7: Singularity commands

The first command pulls the image from Docker Hub and converts it to a Singularity image. The second command just runs the Singularity image in a container.

1. ISSUES ENCOUNTERED

The only issue I encountered during this project was that I was not able to save files in my Docker container. This was due to my Docker container not have write permissions. The ‘-e GRANT SUDO=yes –user root’ solved this problem as it allowed me full access to all the files in my container and I was able to save files then.

1. CONCLUSION

In this project, I created a Docker image for my Twitter Analysis solution using a Dockerfile. This image then ran in a Docker container and was also pushed to Docker Hub so that it could be publicly available. Once the image was pushed to Docker Hub, I was able to pull it to Singularity and convert it to a Singularity image.

REFERENCES

1. D. Merkel, “Docker: lightweight linux containers for consistent devel- opment and deployment,” Linux Journal, vol. 2014, no. 239, p. 2, 2014.
2. A´ . Kova´cs, ”Comparison of different Linux containers,” 2017 40th International Conference on Telecommunications and Signal Processing (TSP), Barcelona, 2017, pp. 47-51, doi: 10.1109/TSP.2017.8075934.