

# CPE 435: OPERATING SYSTEMS LABORATORY.

## **Lab 13.**

### **Docker Containers.**

**Submitted by:** Dan Otieno.

**Date of Experiment:** 04/14/23.

**Report Deadline:** 04/23/23

**Demonstration Deadline:** 04/21/23.

## Introduction:

The goal for this lab was to learn how Dockers works. Dockers are a type of containers that provide the functionalities to help create and deploy applications.

## Lab Assignment:

- Log into the echo server, and then log into your assigned odroid as an odroid user. The first thing we are going to do is verify if you have sudo user access. Inorder to verify you have sudo user access, please perform the following command. You will use the sudo command followed by -v.

```
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.9.61+ armv7l)
Last login: Fri Apr 14 18:10:13 2023 from 172.22.0.6
odroid@odroid:~$ sudo -v
[sudo] password for odroid:
odroid@odroid:~$
```

Although we have not installed docker on our machine, let us verify one last time. Please type docker in your terminal. You should see a command not found or something of that sort.

```
odroid@odroid:~$ docker
-bash: docker: command not found
odroid@odroid:~$
```

---

## ***Assignment 1: Installation of Docker on ARM machine.***

---

The machine that we are using is odroid, which is an ARM machine. You can type cat/proc/cpuinfo to see the processor information of the machine that you are using. We also need to find out the OS version that we are using. Please use the following commands and answer the questions that follow, make sure to include a supporting screenshot for each of the questions.

- `lsb_release -a`
- `uname -a`

```
odroid@odroid:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:   Ubuntu 16.04.3 LTS
Release:      16.04
Codename:     xenial
odroid@odroid:~$ uname -a
Linux odroid 4.9.61+ #1 SMP PREEMPT Mon Feb 5 21:22:44 UTC 2018 armv7l armv7l armv7l GNU/Linux
odroid@odroid:~$
```

- What is the OS name and version in your machine? **Ubuntu** version **16.04.3**.

- What is the kernel version in your machine? **4.9.61+.**
- How many processors are there in your machine and what are their model names?  
**ARMv7 Processor rev 3. (8 processors).**

```
odroid@odroid:~$ cat /proc/cpuinfo
processor       : 0
model name     : ARMv7 Processor rev 3 (v7l)
BogoMIPS      : 84.00
Features      : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part      : 0xc07
CPU revision   : 3

processor       : 1
model name     : ARMv7 Processor rev 3 (v7l)
BogoMIPS      : 84.00
Features      : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part      : 0xc07
CPU revision   : 3

processor       : 2
model name     : ARMv7 Processor rev 3 (v7l)
BogoMIPS      : 84.00
Features      : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x0
CPU part      : 0xc07
CPU revision   : 3

processor       : 3
```

We need to remove docker, if there is any, before moving forward. So, please do the following in your terminal.

Please follow as it shows in the image below:

```
sudo apt-get remove docker docker-engine docker.io containerd runc docker-ce
```

```
odroid@odroid:~$ sudo apt-get remove docker docker-engine docker.io containerd runc docker-ce
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package docker-engine
E: Unable to locate package docker-ce
odroid@odroid:~$
```

Now, let us update.

```
sudo apt-get update
```

You might see some warnings, but ignore them for now. Let us move forward with some installations.

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties
common
```

This will take some time to complete, but there should be no problem to install. If asked to continue, select Y for yes and continue.

Before we can add docker's official GPG key we will need to update the DNS server used by the odroid if it has not been updated already. You can edit the DNS server to use UAH's current DNS server by modifying `/etc/resolv.conf` to use 146.229.1.200 and 146.229.56.2. However this change is not permanent, resetting the Odroid will undo this change. Alternatively you may use Google's public DNS server (8.8.8.8 and 8.8.4.4) if necessary.

Edit the DNS server with the following:

```
sudo nano /etc/resolv.conf
```

Make the following change:

```
GNU nano 2.5.3 File: /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
# DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 146.229.1.200
nameserver 146.229.56.2
```

After this you should be able to add docker's key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

, which should report you OK,

```
odroid@odroid:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
OK
odroid@odroid:~$
```

but we will verify that using the following command.

```
sudo apt-key fingerprint 0EBFCD88
```

```
odroid@odroid:~$ sudo apt-key fingerprint 0EBFCD88
pub 4096R/0EBFCD88 2017-02-22
    Key fingerprint = 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88
uid                               Docker Release (CE deb) <docker@docker.com>
sub 4096R/F273FCD8 2017-02-22

odroid@odroid:~$
```

This should show you some installed keys.

Now is the main part where we will add the link to the repository of docker release based on our linux version and architecture of hardware we are using. Please use the following command:

```
sudo add-apt-repository "deb [arch=armhf] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable"
```

Now you have set up the repository. We will update once again, and install docker on our machine.

```
sudo apt-get update
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

You might get similar messages while updating but you can ignore them again. Continue with the installation with the second command above. This will again take some time to complete.

---

## ***Assignment 2: Verify Docker Installation.***

---

Now that docker is installed properly, we have to verify installation. Please run the following:

```
sudo docker run hello-world
```

This will download an image and run it in a container. Please run this command, take a screenshot and read the contents of the output that you see.

```
odroid@odroid:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
04341b189be6: Pull complete
Digest: sha256:4e83453afed1b4fa1a3500525091dbfca6ce1e66903fd4c01ff015dbcb1ba33e
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm32v7)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

odroid@odroid:~$
```

## **Program and Dependency**

Following is the program that has some different dependencies.

```
# file: myPython2.py
import sys

def python2_stuff():
    print("Yeah!!! This is function called only by python2.x")
    print("This is an important function")
if sys.version_info[0]<3:
    print("This will only work with python version 2.x")
    python2_stuff()
else:
    print("Error!!!You are not supposed to run it like this.")
```

```
odroid@odroid:~$ cat myPython2.py
# file: myPython2.py
import sys

def python2_stuff():
    print("Yeah!!! This is function called only by python2.x")
    print("This is an important function")
if sys.version_info[0]<3:
    print("This will only work with python version 2.x")
    python2_stuff()
else:
    print("Error!!!You are not supposed to run it like this.")
odroid@odroid:~$
```

If you run this with python2, you will make a call to the function python2\_stuff(). If you run with python3, the function will not be called. We want the function to be called because that is an important function. Please run this code using `python2 myPython2.py`. Take a screenshot of the output.

```
odroid@odroid:~$ python2 myPython2.py
This will only work with python version 2.x
Yeah!!! This is function called only by python2.x
This is an important function
odroid@odroid:~$
```

Let us remove python2 from our machine. Please use the following command to remove python 2.7. `sudo apt purge python2.7-minimal`

This will take some time and will remove python2.7. Now, try running the code again using python2. Do you notice any difference? Take a screenshot of the output.

```
odroid@odroid:~$ python2 myPython2.py
-bash: /usr/bin/python2: No such file or directory
odroid@odroid:~$
```

---

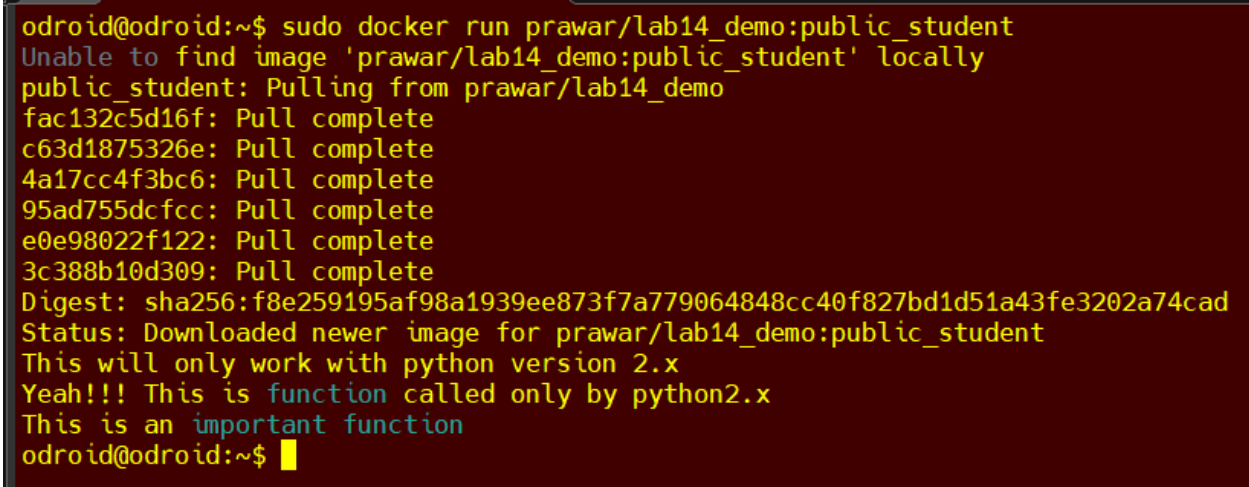
### *Assignment 3: Running using a container.*

---

You only have python3 in your machine, but let us assume that the software that you wrote is in python2. You might be in a situation like this when you inherit a large software code base written in python2. Or in a different situation when you bought a software that is written in python2, but you are not allowed to install python2 on your machine due to company policy. The worst situation is when you do not know what dependencies are being used, or you do not want your customer to install dependencies and want to make things easier for them. A docker image can be created to manage and install any missing dependencies required while creating the image, and these dependencies are irrespective of what is installed in the machine.

Please run the following command and take a screenshot:

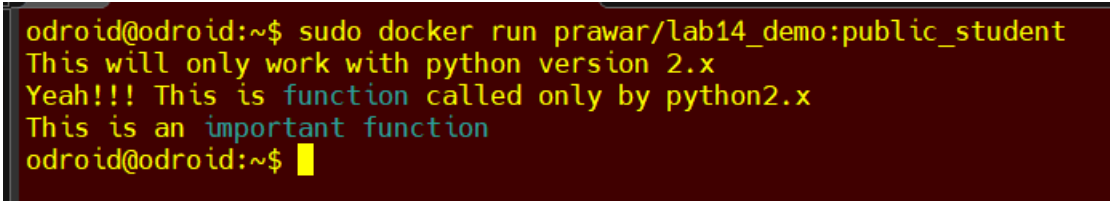
```
sudo docker run prawar/lab14_demo:public_student
```



```
odroid@odroid:~$ sudo docker run prawar/lab14_demo:public_student
Unable to find image 'prawar/lab14_demo:public_student' locally
public_student: Pulling from prawar/lab14_demo
fac132c5d16f: Pull complete
c63d1875326e: Pull complete
4a17cc4f3bc6: Pull complete
95ad755dcfcc: Pull complete
e0e98022f122: Pull complete
3c388b10d309: Pull complete
Digest: sha256:f8e259195af98a1939ee873f7a779064848cc40f827bd1d51a43fe3202a74cad
Status: Downloaded newer image for prawar/lab14_demo:public_student
This will only work with python version 2.x
Yeah!!! This is function called only by python2.x
This is an important function
odroid@odroid:~$
```

This will take some time to run. What docker run command does is it searches for the image by the name prawar/lab14\_demo:public-student locally. If it does not find it in the local machine, as in our case, it downloads the image from docker hub and runs it. This is what it does for the first time you run it. Note that the output is the same as what you saw before you uninstalled python2.

If you run the command again, it will not download but instead it will run the local image. Please take a screenshot of this output.



```
odroid@odroid:~$ sudo docker run prawar/lab14_demo:public_student
This will only work with python version 2.x
Yeah!!! This is function called only by python2.x
This is an important function
odroid@odroid:~$
```

---

## *Assignment 4: Creating Docker Image.*

---

For the assignment above, someone had already built an image and we ran it as a customer. But what if we are software developers and want to release a software but do not want customers to go through the hassle of installing all the dependencies? We have to create a docker image and release it to the public.

The docker image requires at least two files. One is called Dockerfile and the other is your source file. You can create other files based on your need, but we will create a minimal example for demonstration purposes. Dockerfile is the one that specifies all the commands a user will require to build an image. We will base our image on python2 so there is not much package installation that we need to do. But you can install other libraries and packages using the RUN instruction in the Dockerfile.

1. Create a new directory named myDocker.
2. Go inside the folder that you just created. This will be our directory for docker files that we need for our image.
3. Create a text file named Dockerfile and add the following contents in the file.

```
FROM python:2.7-slim
WORKDIR /app
COPY . /app
CMD ["python", "myPython2.py"]
```

```
odroid@odroid:~$ mkdir myDocker
odroid@odroid:~$ cd myDocker/
odroid@odroid:~/myDocker$ touch Dockerfile
odroid@odroid:~/myDocker$ vi Dockerfile
odroid@odroid:~/myDocker$ cat Dockerfile
FROM python:2.7-slim

WORKDIR /app

COPY . /app

CMD ["python", "myPython2.py"]
odroid@odroid:~/myDocker$
```

The commands in the file use python2.7 as the base for our image. it sets up /app as the working directory and copies the content of this current folder to /app.

4. Save the file.



5. The above file also mentions that the image should run python myPython2.py. Copy the code that runs for python2 from Assignment2 to the current folder (i.e. to the folder myDocker).
6. If your source file myPython2.py contains any other dependencies, you can install them using the RUN command in the Dockerfile.
7. Build the image using the following command `sudo docker build --tag=mylab13_<yourchargerID> .` .Replace yourchargerID with your charger id. Do not forget the '.' after a space after your charge id.

```
odroid@odroid:~/myDocker$ sudo docker build --tag=mylab13_dpo0002 .
Sending build context to Docker daemon 28.67kB
Step 1/4 : FROM python:2.7-slim
2.7-slim: Pulling from library/python
838dc71644c1: Pull complete
6d3a2bd251b9: Pull complete
a2da0ab9c4b0: Pull complete
9b7d8020f876: Pull complete
Digest: sha256:6c1ffdf499e29ea663e6e67c9b6b9a3b401d554d2c9f061f9a45344e3992363
Status: Downloaded newer image for python:2.7-slim
--> b532061a7f5b
Step 2/4 : WORKDIR /app
--> Running in 81f606140517
Removing intermediate container 81f606140517
--> a3194b405a9b
Step 3/4 : COPY . /app
--> 18a67bcf7c97
Step 4/4 : CMD ["python","myPython2.py"]
--> Running in 53ed768c3349
Removing intermediate container 53ed768c3349
--> 6ca4ccb5ae5d
Successfully built 6ca4ccb5ae5d
Successfully tagged mylab13_dpo0002:latest
odroid@odroid:~/myDocker$
```

8. Perform `docker image ls` to see the images that are in your machine. You should see the image that you just created also. Take a screenshot.

```
odroid@odroid:~/myDocker$ docker image ls
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get http://%2F%2Frun%2Fdocker.sock/v1.24/images/json: dial unix /var/run/docker.sock: connect: permission denied
odroid@odroid:~/myDocker$
```

9. Run the image that you just created using `sudo docker run mylab13_<yourchargerID>`. Take a screenshot

```
odroid@odroid:~/myDocker$ sudo docker run mylab13_dpo0002
This will only work with python version 2.x
Yeah!!! This is function called only by python2.x
This is an important function
odroid@odroid:~/myDocker$
```

10. Go to [hub.docker.com](https://hub.docker.com) and create an account for you. After you have created an account, go to your working terminal and login using `sudo docker login`.

```
odroid@odroid:~/myDocker$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: dpo0002
Password:
WARNING! Your password will be stored unencrypted in /home/odroid/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

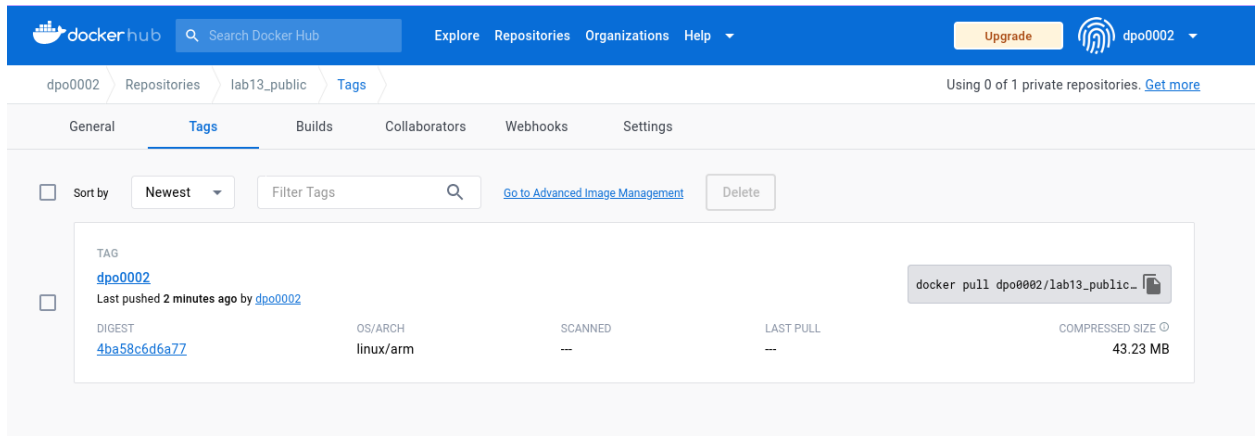
Login Succeeded
odroid@odroid:~/myDocker$
```

11. You will make this image available publicly, and download it again to run it in your machine. You need to tag it first. So what you will do is tag your repository using `sudo docker tag mylab13_<yourchargerID> <yourusernameindockerhub>/lab13_public:<yourchargerID>`.
12. Perform `sudo docker image ls` to see all the local repositories that you have. Take a screenshot. You should see a new one with the new tag. For the image that you created, what are the following values?
- Image name – `dpo0002/lab13_public`.
  - TAG – `dpo0002`.
  - image id – `6ca4ccb5ae5d`.
  - size – `107MB`.

```
odroid@odroid:~/myDocker$ sudo docker tag mylab13_dpo0002 dpo0002/lab13_public:dpo0002
odroid@odroid:~/myDocker$ sudo docker image ls
REPOSITORY          TAG          IMAGE ID      CREATED       SIZE
mylab13_dpo0002     latest      6ca4ccb5ae5d  10 minutes ago  107MB
dpo0002/lab13_public dpo0002     6ca4ccb5ae5d  10 minutes ago  107MB
hello-world         latest      7066d68bd2f2  3 months ago   4.85kB
python              2.7-slim    b532061a7f5b  2 years ago    107MB
prawar/lab14_demo   public_student 3574271cb42b  4 years ago    93.5MB
odroid@odroid:~/myDocker$
```

13. Push this image to the docker hub using `sudo docker push <yourusernameindockerhub>/lab13_public:<yourchargerID>`. After this operation is completed, go to your account in the docker hub and verify that it is available there. Take a screenshot.

```
odroid@odroid:~/myDocker$ sudo docker push dpo0002/lab13_public:dpo0002
The push refers to repository [docker.io/dpo0002/lab13_public]
925297ff13d3: Pushed
30049f8b2eef: Pushed
ce6efa381dfa: Mounted from library/python
ac4ca466d0da: Mounted from library/python
1ca64abfb30b: Mounted from library/python
4e534c010dc1: Mounted from library/python
dpo0002: digest: sha256:4ba58c6d6a7787d66f0df70110d81f9e5f1dee64661ce51ec438d47887f68692 size: 1576
odroid@odroid:~/myDocker$
```



## Assignment 5: Distribution.

Now that you have created a software that runs regardless of the dependencies available in the user machine, you would want to distribute it. Find at least one customer of your software (you can find a friend in CPE435) and tell them to download the image as you did in Assignment 3. Ask for a screenshot from them of the final run and post it in your report.

```
odroid@odroid:~/myDocker$ sudo docker pull jonathanswindell/lab13_public:jes0057
jes0057: Pulling from jonathanswindell/lab13_public
838dc71644c1: Already exists
6d3a2bd251b9: Already exists
a2da0ab9c4b0: Already exists
9b7d8020f876: Already exists
6304304f63db: Pull complete
6d92f5f3ab65: Pull complete
Digest: sha256:9164c51c02a80f4e688f13d065d8788aa5f693382a94836bccf4e6b46d96836a
Status: Downloaded newer image for jonathanswindell/lab13_public:jes0057
docker.io/jonathanswindell/lab13_public:jes0057
odroid@odroid:~/myDocker$ sudo docker run jonathanswindell/lab13_public:jes0057
This will only work with python version 2.x
Yeah!!! This is function called only by python2.x
This is an important function
odroid@odroid:~/myDocker$
```

- For this assignment, Jonathan downloaded my image, and I downloaded his image, a screenshot of him pulling my docker image should be in his report.