The Department of Electrical and Computer Engineering

The University of Alabama in Huntsville

CPE 435 Lab-13

Docker Containers

Introduction

Dockers or containers are tools that help you to create and deploy applications with ease. This means, you can package up the parts that are needed for the applications to run without worrying about the systems at the users' side or deployment site. Containers are somewhat similar to virtual machines, but instead of requiring a complete OS as in a virtual machine, you will perform OS virtualization.

In this lab, you will see an example where we will write a program that has different dependencies that are not installed in the deployment machine.

Procedures

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.

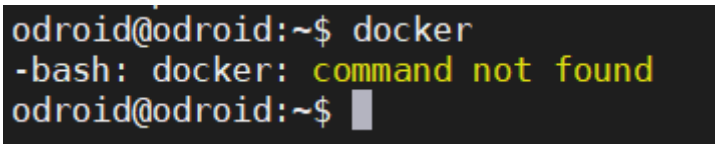
```
[prawar.DESKTOP-NLP2FUA] ▸ ssh pp0030@echo.ece.uah.edu
Last login: Wed Apr  3 09:39:20 2019 from 146.229.165.177
pp0030@echo:~> ssh odroid@odroid20
odroid@odroid20's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.9.61+ armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

479 packages can be updated.
311 updates are security updates.

New release '18.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed Apr  3 14:39:53 2019 from 172.22.0.6
odroid@odroid:~$
```

Inorder to verify you have sudo user access, please perform the following command. You will use the sudo command followed by -v.

```
odroid@odroid:~$ sudo -v
[sudo] password for odroid:
odroid@odroid:~$
```

Now that you have verified you have sudo access, things will be easier.

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.



Assignment 1: Installation of Docker on ARM machine

Following contents are derived from official docker website at [Install Docker Engine on Ubuntu | Docker Documentation](). If you have any problems during the installation process, please visit the link.

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.

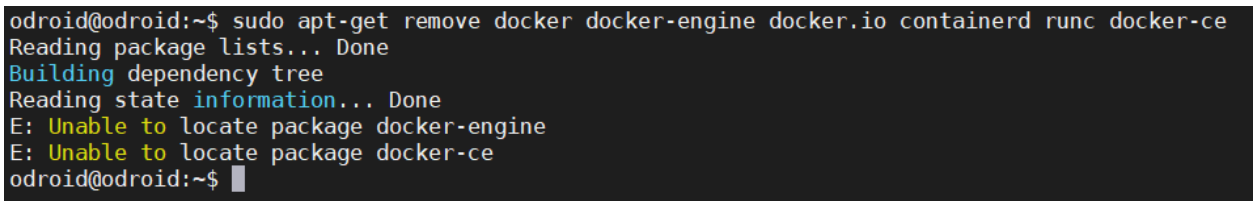
`lsb_release -a`

`uname -a`

1. What is the OS name and version in your machine?
2. What is the kernel version in your machine?
3. How many processors are there in your machine and what are their model names?

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`



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 Odriod will undone 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:

```
# 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 69.1.30.42
#nameserver 69.1.30.43
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, but we will verify that using the following command.

`sudo apt-key fingerprint 0EBFCD88`

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.

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.")
```

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.

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.

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 a take a **screenshot**:

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

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.

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.

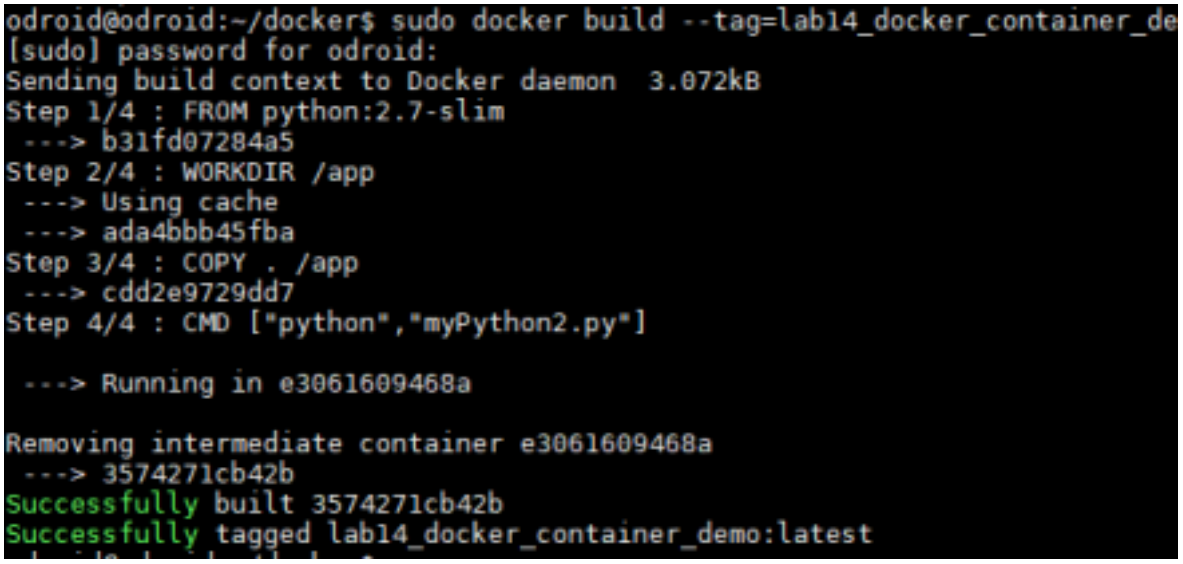
1. Create a new directory named myDocker
2. Go inside the folder that you just created. This will be our directory for docker files that we need for our image.
3. Create a text file named Dockerfile and add the following contents in the file.

```
FROM python:2.7-slim

WORKDIR /app

COPY . /app

CMD ["python","myPython2.py"]
```

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.

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. This will take some time to run and you will get something like the following.



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**.
9. Run the image that you just created using `sudo docker run mylab13_<yourchargerID>`. Take a **screenshot**.
10. Go to 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`.
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?
    1. Image name
    2. TAG
    3. image id
    4. size
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**.

Assignment 5: Distribution

1. 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.

Deliverables

Lab Report

The following material in each section is expected:

1. Cover page with your name, lab number, course name, and dates
2. Sections labeled as in the assignment document, along with any answers and screenshots asked for

The report should be submitted as a single pdf document.

Demonstration

The following material in each section is expected:

1. Introduce yourself and give the name of the lab
2. Show that your python program will not run since you have removed python 2 from your odroid
3. Show that your python program can run by using docker

The demonstration may be in person or recorded and submitted as an mp4 file alongside the report.