Docker Homework
Due Date: Mon. February 15th 5:30pm CST

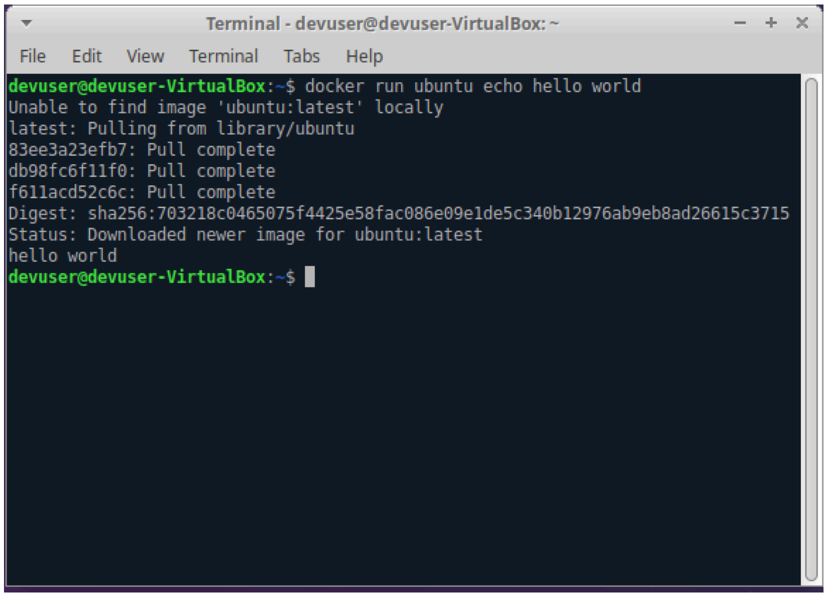
Introduction

In this assignment you will learn how to start existing Docker containers and create your own containers with your applications.

Running an existing container

1. Open a terminal window in your VM.
2. Pull and run a new ubuntu distribution container by typing:

```
docker run ubuntu echo hello world
```

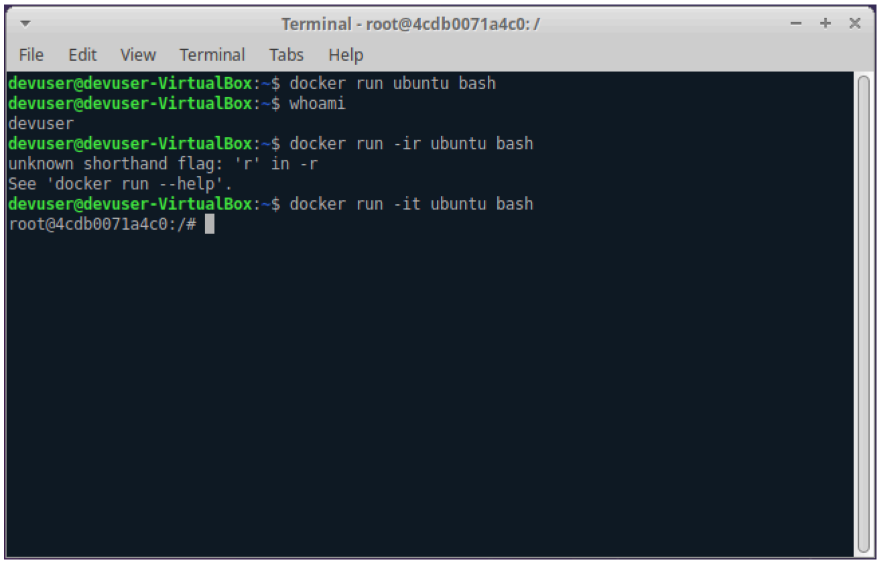

Terminal - devuser@devuser-VirtualBox: ~
File Edit View Terminal Tabs Help

```
devuser@devuser-VirtualBox:~$ docker run ubuntu echo hello world
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
83ee3a23efb7: Pull complete
db98fc6f11f0: Pull complete
f611acd52c6c: Pull complete
Digest: sha256:703218c0465075f4425e58fac086e09e1de5c340b12976ab9eb8ad26615c3715
Status: Downloaded newer image for ubuntu:latest
hello world
devuser@devuser-VirtualBox:~$
```

This will run the ubuntu container and execute the command "`echo hollo world`"

3. Run the container again with the bash command:

```
docker run -it ubuntu bash
```


Terminal - root@4cdb0071a4c0: /
File Edit View Terminal Tabs Help

```
devuser@devuser-VirtualBox:~$ docker run ubuntu bash
devuser@devuser-VirtualBox:~$ whoami
devuser
devuser@devuser-VirtualBox:~$ docker run -ir ubuntu bash
unknown shorthand flag: 'r' in -r
See 'docker run --help'.
devuser@devuser-VirtualBox:~$ docker run -it ubuntu bash
root@4cdb0071a4c0:/#
```

Note: You in the image above I am now running a container named 4cdb0071a4c0 as user root.

4. Type exit to quit the container.

5. Now, in your ubuntu VM, run a centos based container and, in the bash shell, execute the following. Make sure to replace <YOUR STUDENT ID> with your student ID number:

```
cat /etc/redhat-release ; echo <YOUR STUDENT ID>
```

5. Screenshot the terminal. This will be a screenshot you submit.

Now it's time to create our own Docker image. Our goal in this section will be to create an image that sandboxes a simple C application.

A Dockerfile is a simple text file that contains a list of commands that the Docker client calls while creating an image. It's a simple way to automate the image creation process. The best part is that the commands you write in a Dockerfile are almost identical to their equivalent Linux commands. This means you don't really have to learn new syntax to create your own dockerfiles.

6. Create a new blank file in a text-editor and save it with the name Dockerfile
7. Copy the the fork.c program from the course GitHub to your current directory.

8. Add the following to your Dockerfile:

```
FROM gcc:4.9
COPY . /usr/src/myapp
WORKDIR /usr/src/myapp
RUN gcc -o myapp main.c
CMD [“./myapp"]
```

This Docker file pulls the gcc Docker image, runs the gcc command to build your application and then runs the resulting executable.

9. Build the Docker image by typing:

```
docker build -t my-fork-app .
```

Run your Docker image:

```
docker run -it --rm --name my-fork-app my-fork-app
```

10. Create a C program, named main.c, that forks a child. The child process will count down, once per second, from 10 to 0. Each second it will print the time remaining. Once it is complete the parent will print "Countdown complete". For 10 bonus points use `alarm()` and a signal handler.
11. Build your Dockerfile and image.
12. Run the image and verify it works correctly.
13. Submit your Dockerfile, main.c, and screenshot from step 5.

Academic Integrity

This assignment must be 100% your own work. No code may be copied from friends, previous students, books, web pages, etc. All code submitted is automatically checked against a database of previous semester's graded assignments, current student's code and common web sources. By submitting your code on Canvas you are attesting that you have neither given nor received unauthorized assistance on this work. **Code that is copied from an external source or used as inspiration, excluding the course github or Canvas, will result in a 0 for the assignment and referral to the Office of Student Conduct.**