Docker

Objectives:

After completing this lab, you should be able to:
1. Explain Linux containers and differentiate between a container and a VM.
2. list solutions that could be used to implement containers.
3. understand the use cases of containers.
4. install and configure Docker.
5. use pre-built container images.
6. build your own container image.

Pre-reading:

Please read about the topics listed here. Your pre-lab quiz will be based on the information in these pages.
1. Linux containers LXC.
https://en.wikipedia.org/wiki/LXC
2. Docker Documentation.
https://docs.docker.com/get-started/overview/
3. snap packages:
https://en.wikipedia.org/wiki/Snap_(package_manager)
4. Dockerfile:
https://docs.docker.com/engine/reference/builder/
Required resources:
- Ubuntu server VM with a bridged adapter.
- Internet connection.

Install Docker

- In this lab we will use the snap store to install Docker. Use the following command to install Docker on your VM:

```
sudo snap install docker
```

to verify that docker installed successfully use the following command:

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

- set you current user account to use docker without root permissions:

```
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
```

- verify you have permission to use docker without root permissions:

```
docker run hello-world
```

if the last command fails reboot your machine and run the command again.

Download images

docker hub (https://hub.docker.com/) is the main source of docker images, these are prebuilt containers that we can download and use for creating containers we need. Visit the site and search for Ubuntu.
To download that image, use the following command:

```
docker pull ubuntu
```

this will download the latest version of that image. If you wish to download another version, you can specify the version number (tag) after the image name. Tags are found in the details of the image when you click the image on docker hub website.
For example, if you wish to download Ubuntu 18.04 use the following command:

```
docker pull ubuntu:18.04
```

to list the images, you have downloaded use the command:

```
docker images
```

Container management

create a container from an image:

```
docker run -dt --name ubuntu01 ubuntu
```

-d: detached so that we don’t enter the command line of the container immediately which will be a problem since we did not specify a shell to use meaning we will be stuck on an empty screen.
-t: attach a tty for the container. Without this command the container will stop immediately after starting. This is because containers are supposed to run apps inside they exist for only that. So if a containers’ app exits the containers gets destroyed and if no app is running the container stops after starting.

check if the container is running or not:

```
docker ps
```

notice that ubuntu01 is listed as running.
Let's create another container using the same image but in a slightly different way as follows:

```
docker run -d --name ubuntu02 ubuntu
```

run the **docker ps** command and see if ubuntu02 is listed or not. Explain.
Use the **docker ps -a** command and notice the output. What is the status of ubuntu02?
execute a command inside the container:

```
docker exec ubuntu01 pwd
docker exec ubuntu01 ls /home
docker exec ubuntu01 touch /home/test.txt
docker exec ubuntu01 ls /home
```

to enter the shell of the container:

```
docker exec -it ubuntu01 sh
or
docker attach ubuntu01
```

-t will attach a tty and -i will enable standard input so that we can input commands. Create a few files inside the home directory.

Stopping and Starting a Container:
To stop a container, use the following command:

```
docker stop ubuntu01
```

verify that the container has been stopped. What command did you use?
Now let's start the container again.

```
docker start ubuntu01
```

verify that the container is running. Connect to the container shell and check the contents of the home directory. Are your files still there?
Deleting a container:
to delete a container simply use the following set of commands:

```
docker stop ubuntu01
docker rm ubuntu01
```

verify that the container has been deleted.
Now recreate the container again with the same name then attach to its shell and check the contents of the home directory. Are your files still there? Explain.

Persistent Storage Containers:

the idea is very simple, containers are volatile and to make their storage persistent we need to map a directory inside the container to a directory inside our host machine.
First create a directory in your home directory named containerdir. The second step is to create a container with directory mapping, the command is as follows:

```
docker run --name ubuntu01 -dt -v  /home/$USER/containerdir:/home ubuntu
docker exec -ti ubuntu01 sh
touch /home/testfile
```

Networking

If we have running services on our container, then they are only accessible by the host machine. Let's try that with an http server.

```
docker run -dt --name webserver httpd
docker exec -ti webserver sh
hostname -i
exit
```

now on your host access the site using **curl http://[container_IP]** this should display the html code of the start page of the server. This address is a host only network address that can be accessed only by the host. This is not practical we need other hosts to be able to access the site. Let's create a new container with port mapping enabled.

```
docker run -dt -p 80:80 --name webserver httpd
```

where httpd is an image on docker hub.
display the running containers and notice the difference in the output.
now open the VM IP address from another machine [your PS host OS]. This should display a web page that says "It works!".
Stop all containers before proceeding.

Building your own images

Dockerfile:

To understand the syntax of a dockerfile let's do a simple example. In this example we need to create our own container image that has a web server app running. This image should include a static website that we provide to be included inside the container image so that whenever we need to run a container it comes with the site already built-in.

Start here:

```
mkdir dockerexample
cd dockerexample
nano Dockerfile
```

add the following inside the dockerfile:

```
FROM ubuntu
RUN apt update
RUN apt install -y apache2
RUN apt clean
RUN rm -rf /var/lib/apt/lists/*
WORKDIR /var/www/html
COPY index.html index.html
EXPOSE 80
CMD ["apachectl","-D","FOREGROUND"]
```

Create index.html:

```
<!doctype html>
<html>
  <head>
    <title>Docker Images</title>
  </head>
  <body>
      <h1>NIS students are the best</h1>
    <p>This is a custom apache start page from inside a container made with a
custom image.</p>
  </body>
</html>
```

Build the image:

```
docker build -t mysrvrimg .
```

Now let’s create a container using that image:

```
docker run -dit -p 80:80 –name mysrvr01 mysrvrimg
```

from your host open a browser and open the sample site using the ip address of your VM.

Clean Up

Stop and remove all containers.

To Do:

Create 2 containers with the following specs:
1. Container 1 should be a web server. The website must be persistent with the index.html page containing the names of your group members and must be accessible from other machines.
2. Container 2 based on httpd with all default settings. This also should be accessible from all other machines.

END