OPS535 Lab 8

Purpose

In this lab you will learn to run docker containers, and to make changes to the images containers are built from.

Pre-Requisites

This lab does not have any pre-requisites.

Investigation 1: Installing Docker and Running a Container

Perform the following steps as root on your VM2

1. Install the the necessary packages, then install docker.

```
> yum install -y epel-release lvm2 device-mapper-persistent-data
> yum install yum-utils
> yum-config-manager –add-repo \
https://download.docker.com/linux/centos/docker-ce.repo
> yum install docker-ce
```

2. Start and enable the service.

3. Test that the service is working properly

```
> docker run hello-world
```

   - This downloads a simple image from a public repo and runs it, making sure your service is working properly.

4. You can find more publicly available images on the docker hub. You don't need an account to download images, but you will need one to upload, so create one now.

5. Run a container built from an apache image:

```
> docker run httpd
```

   - This time you'll notice you don't get your terminal back immediately. This is because most containers don't immediately end, but keep running until you stop them.
   - In the output it should give you an ip address you can use to contact the web-service running in the container. Direct your browser to that address to receive a message stating that the service works.
   - Use <ctrl>-c to shut down the container.

6. To run a container in the background (and get our terminal back while it is running), add -d (for detached) to your docker run command.

```
> docker run -d httpd
```

   - Point you browser at the same ip address as before. You should see the same webpage.

7. To stop the image you can either use the UUID that is presented as output, or find out its name and use that.

```
> docker container ls
```

   - Also known as docker ps, will display information about all running containers (or all containers if you add -a to it). Use that information to fill in the following command to stop the running container.

```
> docker stop <container-name>
```

8. Instead of depending on the semi-random names generated by docker, you can use the --name option in your docker run commands to give your container a name of your choice.

9. Generally, you will want the services running in your container to be reachable by other machines. To do this, you can connect the ports on your machine to ports in your containers with -p.

```
> docker run -d -p 80:80 httpd
```

   - Now point the browser on your host at the address of your vm2 (you may have to update your firewall to allow the traffic in). You should be able to see the same webpage. Once you have done so, stop that container.
   - Note that -p defaults to tcp. Find out how to make it connect udp ports.
   - Also, you can use multiple copies of the -p option in the same docker run command if you want to connect multiple host ports to container ports.

10. You now have several containers present on your machine. The docker ls -a command will let you find them, and the docker rm command will let you delete them.

```
> docker ls -a
```

   - For each container in that list, run

```
> docker rm <container name>
```

Investigation 2: Customizing an Image

Perform the following steps as root on your VM2

1. The container you ran, works, but the webpage doesn't have much in the way of content. There are several ways to change this.

2. The -v option for the docker run command allows you to mount volumes from your host into a container when it is created.

    - Create a directory /etc/docker/http and create a simple webpage called index.html in it.
    - Run the following command to create a new container from the httpd image, mounting your /etc/docker/http directory into the DocumentRoot in the container.

        ```
        > docker run -d -p 80:80 -v /etc/docker/http:/usr/local/apache2/htdocs httpd
        ```

    - Now use your browser to view the page again. This time you should see your page instead of the initial default.

3. You can use the docker exec command to interact with a running container, running extra commands against it. The ability to run a bash shell (or other shell) is particularly useful.

    - The -i option for docker exec allows for interactive commands, while -t allocates a terminal.
    - Fill the name of your container into the following command.

        ```
        > docker exec -it <containername> /bin/bash
        ```

    - Try to edit the /usr/local/apache2/htdocs/index.html file. You'll probably find you don't have access to the commands you are used to. This is because containers have only the minimal resources necessary to fulfill their tasks, and a web-service does not rely on text editors. However you can use -v to mount them from your machine.

4. Stop and remove your container, then run a new one using another -v option to mount /usr/bin from your host into /usr/bin in the container.

    - Now try docker exec again to get bash access inside the container, then use an editor of your choice to modify the index.html page.
    - Refresh your browser, and you should see the changes.

5. Changes you make to a running container are not persistent. They will exist in that one container, but will not be carried over to other containers built from the same image.

    - Stop and remove your container.
    - Start a new one, and refresh your browser. Your page will be back to what it was before you made changes.

6. If you want changes to be copied to other containers, you will need to commit them into an image then build new containers from that image.

    - Use the docker exec command to get back into the container and make some changes to the index.html page, then exit.
    - Stop the container, but do not remove it.
    - Use docker commit to save your changes into an image. This will require the username you created for dockerhub earlier.

```
>docker commit -a “<insert your name and email>” -m “<insert a message describing your changes>” <containername> <imagename>
```

- Note that the image name is a combination of your account and the name of the image you want to create (e.g. petercallaghan/httpd).

7. This creates an image on your own machine, but it will not be accessible to anyone else. Pushing the image to a repo will make it available to others.

   - Use docker login to authenticate to the docker hub, using the account name and password you created earlier.
   - Once you are logged in, use docker push to upload your image to the repo.

Investigation 3: Deploying an Image to a Swarm

Perform the following steps as root on your VM1

1. Use the steps from Investigation 1 to install docker on your vm1 and vm3.

2. So far you have only deployed your image onto a single machine. This is nice, but doesn’t reveal the true power of docker. We can cause it to deploy containers built from our image across any number of machines.

3. First we need to create a swarm, using vm2 as the swarm manager.

   ```
   > docker swarm init
   ```

   - This reveals a swarm token that can be used by other machines to join the swarm. It will also reveal a port number you need to allow through your firewall on the swarm manager.
   - If you need to reveal this token again (perhaps because you want to ssh from your host to vm2 so you can copy the token) use:

     ```
     > docker swarm join-token worker
     ```

4. Add your other vms to the swarm using the token and address from your vm2.

   ```
   > docker swarm join --token <token> <manager’s ip>:2377
   ```

5. Using the compose file from the tutorial linked in the slides as a reference, create a new compose file that will deploy one copy of your image onto every machine in your swarm.

Completing the Lab

You now have a custom image running on multiple machines in your network. If something goes wrong with one of the containers, it is easily replaced. If you find you need more resources devoted to that service, you can deploy more instances of it, or remove some that you no longer need. When you need to make a change, you only need to change the base image and it will replicate the change to each container the next time they start. All of these capabilities will make your network far more versatile.

Follow the instructions on blackboard to submit the lab.