Lab 1. Installing and Using Docker

The Docker installation package available in the official Ubuntu repository may not be the latest version. To ensure we get the latest version, we'll install Docker from the official Docker repository. To do that, we'll add a new package source, add the GPG key from Docker to ensure the downloads are valid, and then install the package.

Step 1. Install Docker

1. First, update your existing list of packages:

```
sudo apt update
```

2. Next, install a few prerequisite packages which let apt use packages over HTTPS:

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

3. Then add the GPG key for the official Docker repository to your system:

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

Add the Docker repository to APT sources:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
```

4. Next, update the package database with the Docker packages from the newly added repo:

```
sudo apt update
```

5. Make sure you are about to install from the Docker repo instead of the default Ubuntu repo:

```
apt-cache policy docker-ce
```

You'll see output like this, although the version number for Docker may be different:

```
                    Output of apt-cache policy docker-ce

docker-ce:
  Installed: (none)
  Candidate: 18.03.1~ce~3-0~ubuntu
  Version table:
     18.03.1~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64
Packages
```

Notice that docker-ce is not installed, but the candidate for installation is from the Docker repository for Ubuntu 18.04 (bionic).

```
sudo apt install docker-ce
```

Docker should now be installed, the daemon started, and the process enabled to start on boot. Check that it's running:

```
sudo systemctl status docker
```

```
Output
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor
preset: enabled)
   Active: active (running) since Thu 2018-07-05 15:08:39 UTC; 2min 55s ago
     Docs: https://docs.docker.com
 Main PID: 10096 (dockerd)
    Tasks: 16
   CGroup: /system.slice/docker.service
           ├─10096 /usr/bin/dockerd -H fd://
           └─10113 docker-containerd --config
/var/run/docker/containerd/containerd.toml
```

Step 2. Running Docker without needing sudo

By default, the docker command can only be run the root user or by a user in the docker group, which is automatically created during Docker's installation process. If you attempt to run the docker command without prefixing it with sudo or without being in the docker group, you'll get an output like this:

```
Output
docker: Cannot connect to the Docker daemon. Is the docker daemon running
on this host?.
See 'docker run --help'.
```

If you want to avoid typing sudo whenever you run the docker command, add your username to the docker group:

```
sudo usermod -aG docker ${USER}
```

To apply the new group membership, log out of the server and back in, or type the following:

```
su - ${USER}
```

Confirm that your user is now added to the docker group by typing:

```
id -nG
```

Step 3. Using Docker

Docker containers are built from Docker images. By default, Docker pulls these images from Docker Hub, a Docker registry managed by Docker, the company behind the Docker project. Anyone can host their Docker images on Docker Hub, so most applications and Linux distributions you'll need will have images hosted there.

To check whether you can access and download images from Docker Hub:

First, you must login to the docker repo using docker login like so:

```
docker login
```

Docker will prompt you for your login name and password.

Once you are logged in, run the following command:

```
docker run hello-world
```

You should see output that looks something like this:

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest:
sha256:92695bc579f31df7a63da6922075d0666e565ceccad16b59c3374d2cf4e
8e50e
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.
    (amd64)
 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/
```

Docker was initially unable to find the hello-world image locally, so it downloaded the image from Docker Hub, which is the default repository. Once the image downloaded, Docker created a container from the image and the application within the container executed, displaying the message.

Search for the *ubuntu* image by running the following command:

```
docker search ubuntu
```

The script will crawl Docker Hub and return a listing of all images whose name match the search string. In this case, the output will be similar to this:

```
NAME                                                      DESCRIPTION
STARS               OFFICIAL            AUTOMATED
ubuntu                                                    Ubuntu is a
Debian-based Linux operating sys…   9435                 [OK]
dorowu/ubuntu-desktop-lxde-vnc                            Docker
image to provide HTML5 VNC interface …   292
[OK]
rastasheep/ubuntu-sshd                                    Dockerized
SSH service, built on top of offi…   212
[OK]
consol/ubuntu-xfce-vnc                                    Ubuntu
container with "headless" VNC session…   173
[OK]
ubuntu-upstart                                            Upstart is
an event-based replacement for th…   97                   [OK]
ansible/ubuntu14.04-ansible                               Ubuntu
14.04 LTS with ansible                    96
[OK]
neurodebian                                               NeuroDebian
provides neuroscience research s…   56                    [OK]
1and1internet/ubuntu-16-nginx-php-phpmyadmin-mysql-5   ubuntu-16-
nginx-php-phpmyadmin-mysql-5          50
[OK]
ubuntu-debootstrap                                        debootstrap
--variant=minbase --components=m…   40                    [OK]
nuagebec/ubuntu                                           Simple
always updated Ubuntu docker images w…   23
[OK]
i386/ubuntu                                               Ubuntu is a
Debian-based Linux operating sys…   17
ppc64le/ubuntu                                            Ubuntu is a
Debian-based Linux operating sys…   13
1and1internet/ubuntu-16-apache-php-7.0                    ubuntu-16-
apache-php-7.0                           13
[OK]
eclipse/ubuntu_jdk8                                       Ubuntu,
JDK8, Maven 3, git, curl, nmap, mc, …   10
[OK]
darksheer/ubuntu                                          Base Ubuntu
Image -- Updated hourly             5
[OK]
codenvy/ubuntu_jdk8                                       Ubuntu,
JDK8, Maven 3, git, curl, nmap, mc, …   5
[OK]
pivotaldata/ubuntu                                        A quick
freshening-up of the base Ubuntu doc…   2
paasmule/bosh-tools-ubuntu                                Ubuntu
based bosh-cli                            1
[OK]
smartentry/ubuntu                                         ubuntu with
```

```
smartentry                            1
[OK]
1and1internet/ubuntu-16-sshd                                ubuntu-16-
sshd                                  1
[OK]
1and1internet/ubuntu-16-php-7.1                             ubuntu-16-
php-7.1                               1
[OK]
ossobv/ubuntu                                               Custom
ubuntu image from scratch (based on o…   0
pivotaldata/ubuntu-gpdb-dev                                 Ubuntu
images for GPDB development              0
1and1internet/ubuntu-16-healthcheck                         ubuntu-16-
healthcheck                           0
[OK]
1and1internet/ubuntu-16-rspec                               ubuntu-16-
rspec                                 0
[OK]
```

In the **OFFICIAL** column, **OK** indicates an image built and supported by the company behind the project. Once you've identified the image that you would like to use, you can download it to your computer using the pull subcommand.

Run the following command:

```
docker pull ubuntu
```

You'll see the following output:

```
Using default tag: latest
latest: Pulling from library/ubuntu
898c46f3b1a1: Pull complete
63366dfa0a50: Pull complete
041d4cd74a92: Pull complete
6e1bee0f8701: Pull complete
Digest:
sha256:017eef0b616011647b269b5c65826e2e2ebddbe5d1f8c1e56b3599fb14f
abec8
Status: Downloaded newer image for ubuntu:latest
```

After an image has been downloaded, you can then run a container using the downloaded image with the run subcommand. As you saw with the hello-world example, if an image has not been downloaded when docker is executed with the run subcommand, the Docker client will first download the image, then run a container using it.

```
docker images
```

The output should look similar to the following:

```
REPOSITORY          TAG                 IMAGE ID                                  
CREATED             SIZE
ubuntu              latest              94e814e2efa8        5
weeks ago         88.9MB
hello-world         latest              fce289e99eb9        3
months ago        1.84kB
```

Step 4. Running a docker container.

The hello-world container you ran in the previous step is an example of a container that runs and exits after emitting a test message. Containers can be much more useful than that, and they can be interactive. After all, they are similar to virtual machines, only more resource-friendly.

As an example, let's run a container using the latest image of Ubuntu. The combination of the **-i** and **-t**switches gives you interactive shell access into the container. Run the following:

```
docker run -it ubuntu
```

Your command prompt should change to reflect the fact that you're now working inside the container and should take this form:

```
root@d3f35ad6e4ff:/#
```

Note the container id in the command prompt. In this example, it is d9b100f2f636. You'll need that container ID later to identify the container when you want to remove it.

Now you can run any command inside the container. For example, let's update the package database inside the container. You don't need to prefix any command with sudo, because you're operating inside the container as the **root** user

Run the apt-update command inside your container.

```
root@d3f35ad6e4ff:/# apt-update
```

Then install the node.js application.

```
root@d3f35ad6e4ff:/# apt install nodejs
```

This installs Node.js in the container from the official Ubuntu repository. When the installation finishes, verify that Node.js is installed by running:

```
root@d3f35ad6e4ff:/#  node -v
```

You'll see the version number displayed in your terminal:

```
v8.10.0
```

Any changes you make inside the container only apply to that container.

Exit the container, type exit or hit control-d at the prompt.

Step 5. Managing Docker Containers

After using Docker for a while, you'll have many active (running) and inactive containers on your computer. To view the active ones, run:

```
docker ps
```

You will see output similar to the following:

```
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS               NAMES
```

In this lab, you started two containers; one from the hello-world image and another from the ubuntu image. Both containers are no longer running, but they still exist on your system.

To view all containers — active and inactive, run docker ps with the -a switch:

```
docker ps -a
```

You should see output that looks like the following:

```
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                          PORTS
NAMES
df5462befd5b        ubuntu              "/bin/bash"         About
an hour ago   Exited (127) About a minute ago
reverent_aryabhata
44421b5b1be3        ubuntu              "/bin/bash"         3
hours ago         Exited (0) About an hour ago
silly_tu
d3f35ad6e4ff        ubuntu              "/bin/bash"         3
hours ago         Exited (0) 3 hours ago
ecstatic_meitner
```

```
23bf15635b8b        hello-world         "/hello"            3
hours ago          Exited (0) 3 hours ago
unruffled_chebyshev
```

To view the latest container you created, pass it the `-l` switch:

```
docker ps -l
```

You should see output similar to this:

```
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                          PORTS
NAMES
df5462befd5b        ubuntu              "/bin/bash"         About
an hour ago    Exited (127) 3 minutes ago
reverent_aryabhata
```

To start a stopped container, use docker start, followed by the container ID or the container's name. Let's start the Ubuntu-based container with the ID of d3f35ad6e4ff

```
docker start <docker_id>
```

The container will start, and you can use `docker ps` to see its status:

```
docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS               NAMES
d3f35ad6e4ff        ubuntu              "/bin/bash"         3
hours ago          Up 33 seconds
ecstatic_meitner
```

To stop a running container, use docker stop, followed by the container ID or name. for example, we'd use the name that Docker assigned the container in the previous example, which is ecstatic_meitner.

Type in the following:

```
docker stop <container_name>
```

Once we've decided we no longer need a container anymore, remove it with the docker rm command, again using either the container ID or the name. Use the docker ps -a command to find the container ID or name for the container associated with the hello-world image and remove it.

Run the following:

```
docker ps -a
```

To stop a running container, use docker stop, followed by the container ID or name. This time, we'll use the name that Docker assigned the container.

To remove the container, run the docker rm command.

```
docker rm <container_name>
```

You can start a new container and give it a name using the --name switch. You can also use the --rm switch to create a container that removes itself when it's stopped.

Step 6. Commiting changes in a Docker container.

When you start up a Docker image, you can create, modify, and delete files just like you can with a virtual machine. The changes that you make will only apply to that container. You can start and stop it, but once you destroy it with the docker rm command, the changes will be lost for good.

After installing Node.js inside the Ubuntu container, you now have a container running off an image, but the container is different from the image you used to create it. But you might want to reuse this Node.js container as the basis for new images later.

Then commit the changes to a new Docker image instance using the following command.

```
docker commit -m "What you did to the image" -a "Author Name" container_id
repository/new_image_name
```

For this lab, commit the changes to the container where you installed node.js

Run the following commands:

```
docker commit -m "Added node.js" -a "<your_id>" <container_id>
repository/new_image_name
```

For repository use your docker login name.

Note that running docker images will show you the new image.

Step 7. Building and executing a Dockerfile.

For this step, you will be given a directory that contains the following:

1. A simple python flask application, "app.py"
2. A requirements.txt file
3. a templates directory for the application

In this exercise, you will build a dockerfile that will do the following:

Using a docker image called python:3-onbuild,

Expose port 5000 as the ingress and egress port for the application

In the image, run the following command:

"python ./app.py"

First, build the image using the docker build command.

Next, run the image using docker run.