Amir Mansha

CYSE 211 – DL1

Docker Lab

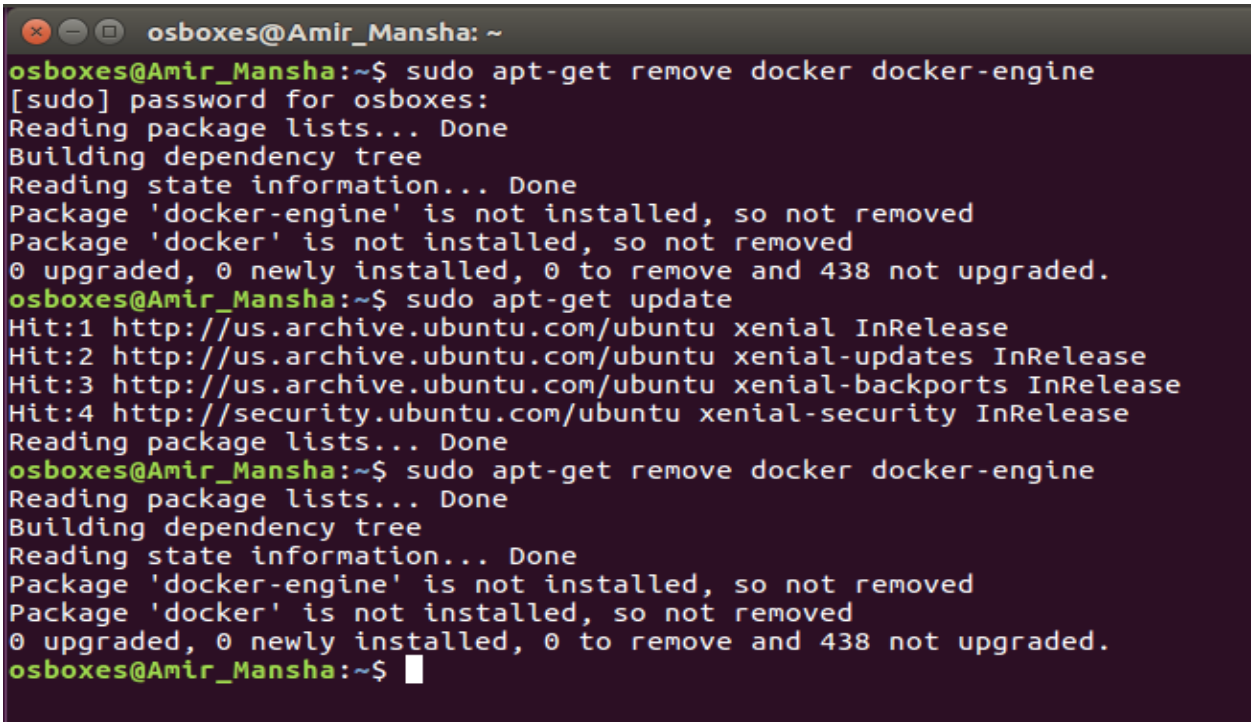
INTRO:

Docker is a tool that is efficient in handling isolated applications and the application runs in the same environment meaning the application can work in every computer if that is the situation. It is good for security because it keeps your projects separate.

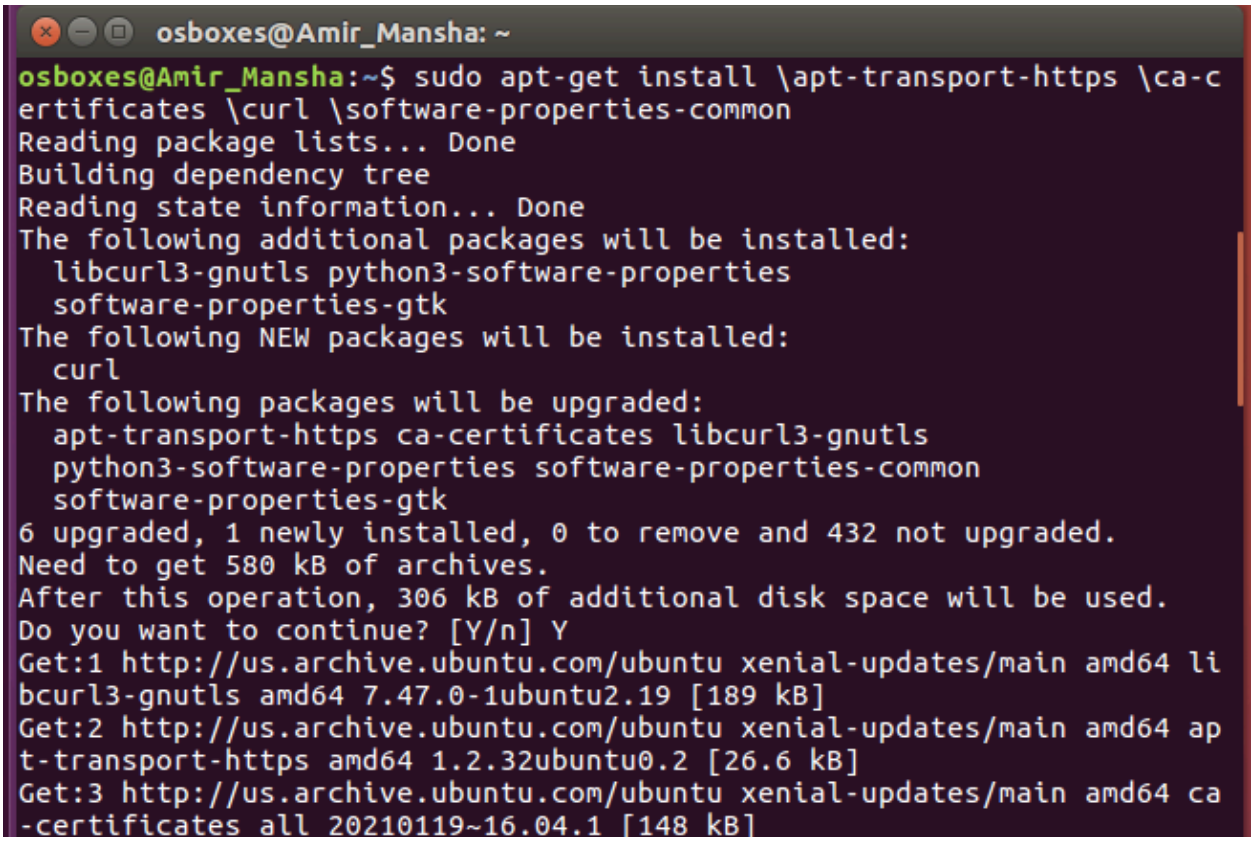
A container in the docker is where codes and the environment are in a “container.” In a virtual machine, it has its own kernel and uses its kernel. But docker container uses the host computer kernel. An “image” contains the OS, software, apps etc. which runs in the container.

Task 1 – set up

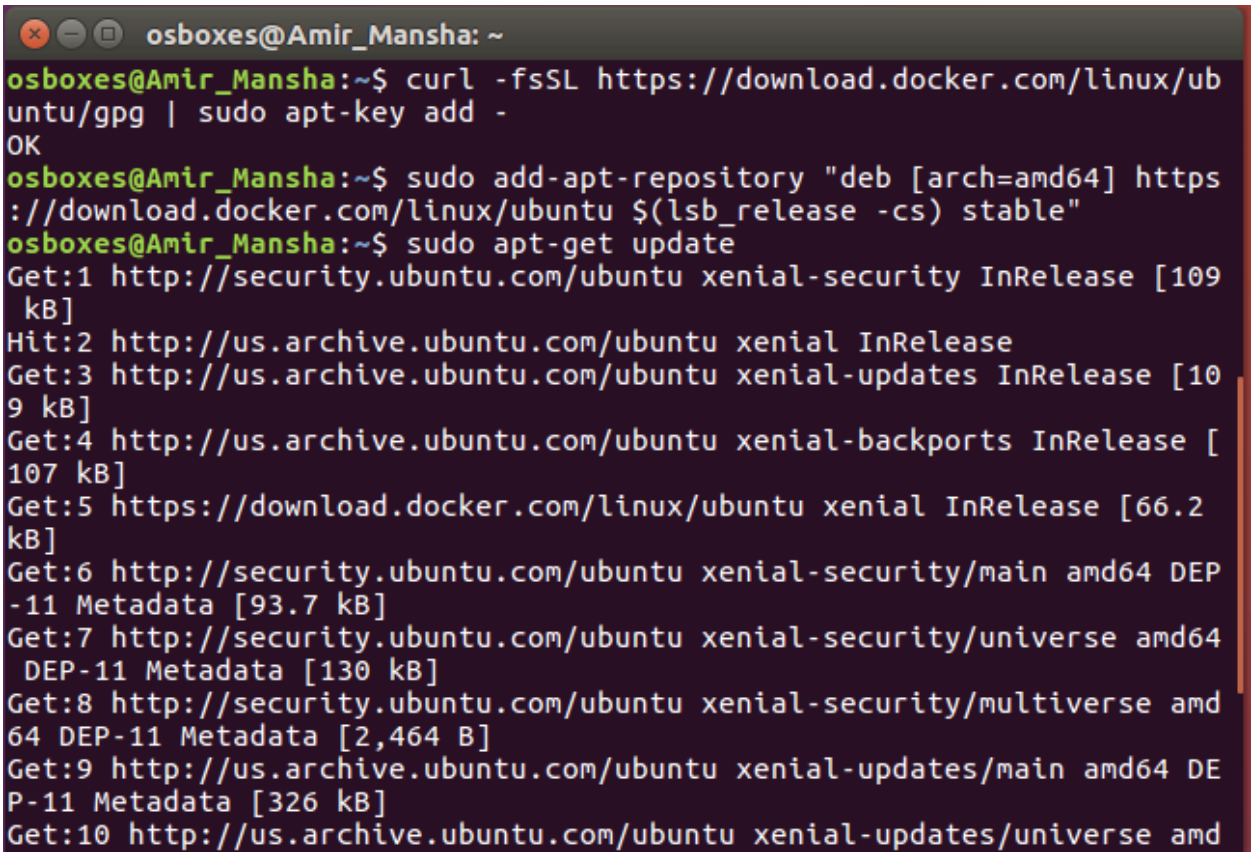
In this task we are setting up and installing docker.



We install docker in a 64-bit machine because docker works on that only. So,in this 1st screenshot, we are removing any prior installed docker image first. Next, we are just updating our system and then checking again if any default docker image installed and removing that again.



In this 2nd screenshot, we are installing some required packages such as the certificates.



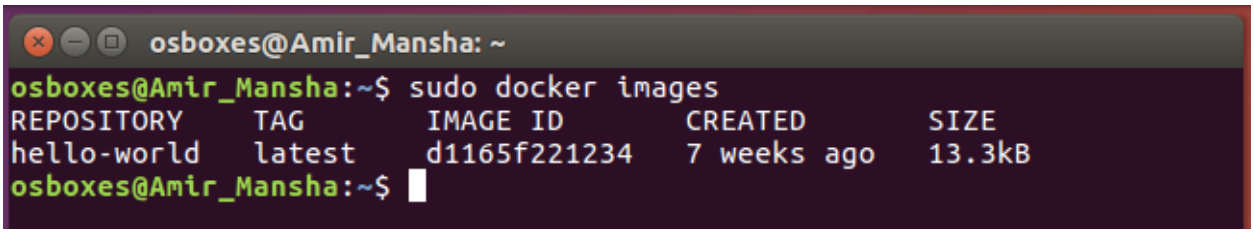
In this 3rd screenshot, we use the curl command we install the docker gpg key and then we add the docker repository command where it stores the docker images. Next, we want to update everything again so the docker packages are included and updated in the system.

```
osboxes@Amir_Mansha:~$ sudo apt-get install docker-ce
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-ce-cli docker-ce-rootless-extras
  docker-scan-plugin git git-man liberror-perl libseccomp2 pigz
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run
  | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb
  git-arch git-cvs git-mediawiki git-svn
Recommended packages:
  slirp4netns
The following NEW packages will be installed:
  containerd.io docker-ce docker-ce-cli docker-ce-rootless-extras
  docker-scan-plugin git git-man liberror-perl pigz
The following packages will be upgraded:
  libseccomp2
1 upgraded, 9 newly installed, 0 to remove and 431 not upgraded.
Need to get 111 MB of archives.
After this operation, 491 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu xenial/universe amd64 pigz a
```

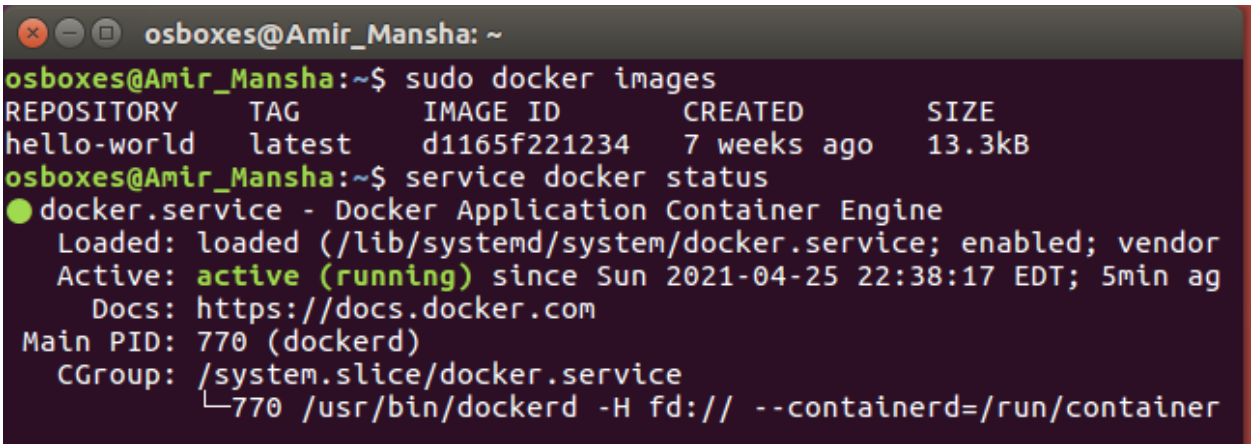
In this 4th screenshot, now we finally install docker -ce version.

```
osboxes@Amir_Mansha:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
b8dfde127a29: Pull complete
Digest: sha256:f2266cbfc127c960fd30e76b7c792dc23b588c0db76233517e1891a
4e357d519
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correc
tly.

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 H
ub.
    (amd64)
 3. The Docker daemon created a new container from that image which ru
ns 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 verify we have docker installed, we run the docker image of “hello world” and as you can see it printed “hello from docker!” which means we have successfully installed docker.



Here we use the command “sudo docker images” to see the image we created of the “hello world” and as you can see it prints out the hello – world under repository along with other information of the image such as the image id.



In this last screenshot of task 1, we run the status of the docker image I installed, and it prints out a lot of information about the docker image such as its active status and other things.

Task 2

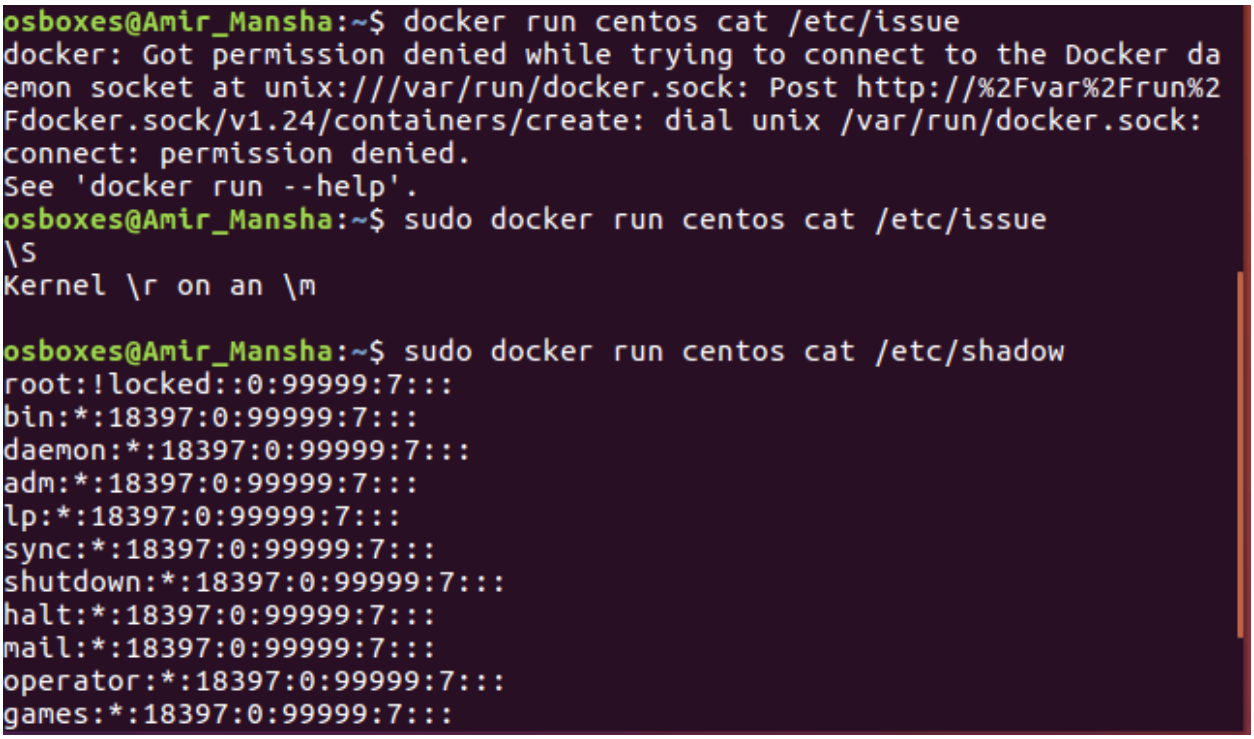
In this task 2, we mainly want to create a docker image.

```
osboxes@Amir_Mansha:~$ sudo docker search centos
[sudo] password for osboxes:
NAME                                DESCRIPTION
             STARS     OFFICIAL   AUTOMATED
centos                              The official build of CentOS.
             6519      [OK]
ansible/centos7-ansible             Ansible on Centos7
             133                   [OK]
consol/centos-xfce-vnc              Centos container with "headless" VN
C session…   128                   [OK]
jdeathe/centos-ssh                  OpenSSH / Supervisor / EPEL/IUS/SCL
 Repos - …   117                   [OK]
centos/systemd                      systemd enabled base container.
             98                    [OK]
imagine10255/centos6-lnmp-php56     centos6-lnmp-php56
             58                    [OK]
tutum/centos                        Simple CentOS docker image with SSH
 access      47
kinogmt/centos-ssh                  CentOS with SSH
             29                    [OK]
pivotaldata/centos-gpdb-dev         CentOS image for GPDB development.
Tag names…   13
```

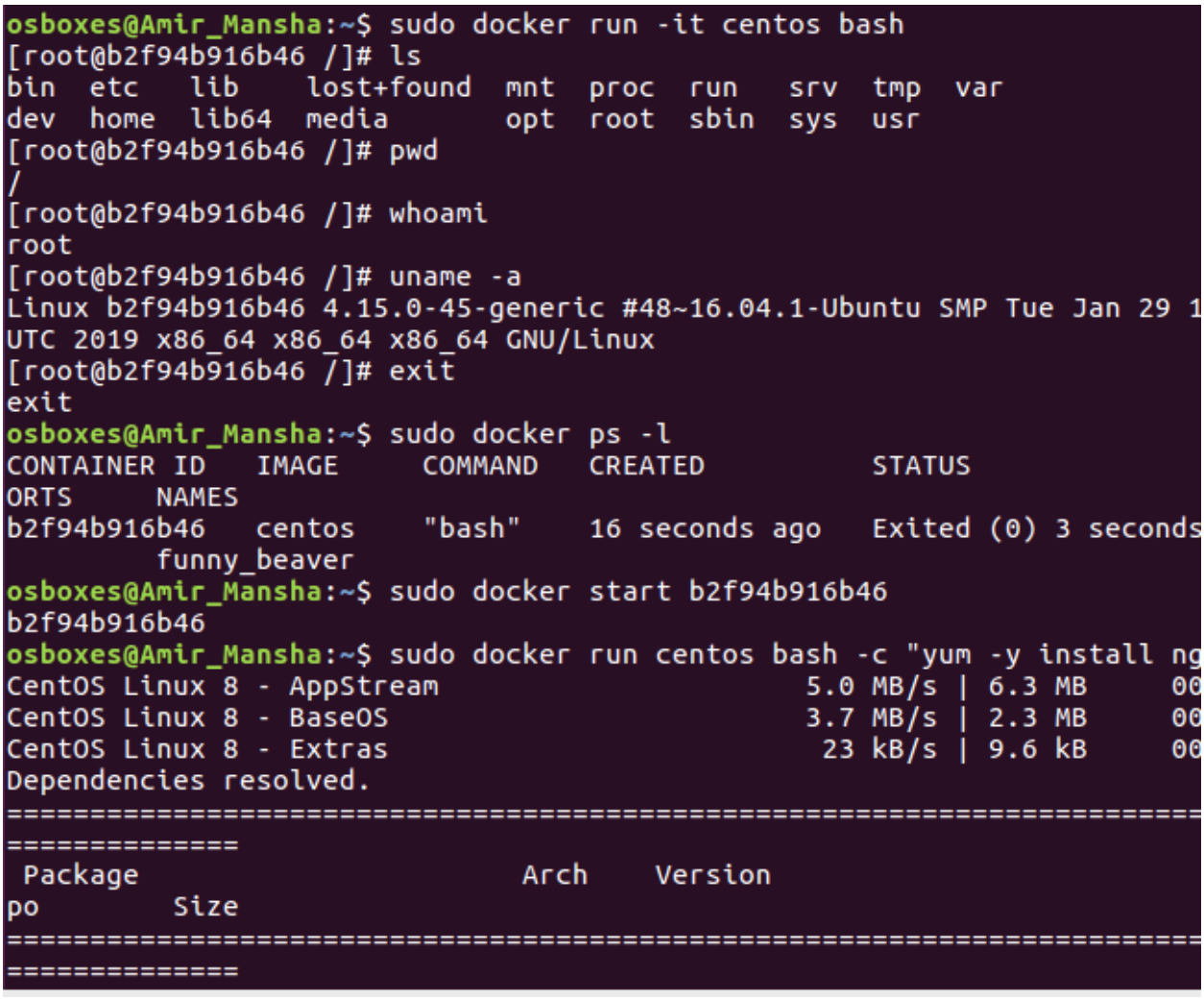
In this 1st screenshot of Task 2, we find the image container of centos operating system, so we use the “search” command and see the list of the repositories for centos. We select the 1st one.

```
osboxes@Amir_Mansha: ~
osboxes@Amir_Mansha:~$ sudo docker pull centos
Using default tag: latest
latest: Pulling from library/centos
7a0437f04f83: Pull complete
Digest: sha256:5528e8b1b1719d34604c87e11dcd1c0a20bedf46e83b5632cdeac91
b8c04efc1
Status: Downloaded newer image for centos:latest
docker.io/library/centos:latest
osboxes@Amir_Mansha:~$ sudo docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS
               PORTS     NAMES
25bd7e8c5634   hello-world   "/hello"   20 minutes ago   Exited (0) 20
 minutes ago             upbeat_blackburn
osboxes@Amir_Mansha:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    d1165f221234   7 weeks ago    13.3kB
centos        latest    300e315adb2f   4 months ago   209MB
osboxes@Amir_Mansha:~$
```

In this 2nd screenshot, we pull and load the image of centos using the “pull” command. Using the “docker images” command we see the image of the centos that I created along with the image id.



In this 3rd screenshot, we just explore different commands within the centos operating system. So, we run the /etc commands and get the /etc/shadow and /etc/issue information.



In this 4th screenshot, we are still exploring commands. We run the bash command within the centos operating system to open the shell and run multiple commands such as “ls, pwd, whoami” commands. Next, we use the “ps -l” command to find out more information about the centos image such as the name and container id etc. I used the container id instead of the name to start the docker container. We use the “yum” command to install packages in the centos operating system. We installed the “nginx” web server that uses HTTP in the centos OS.

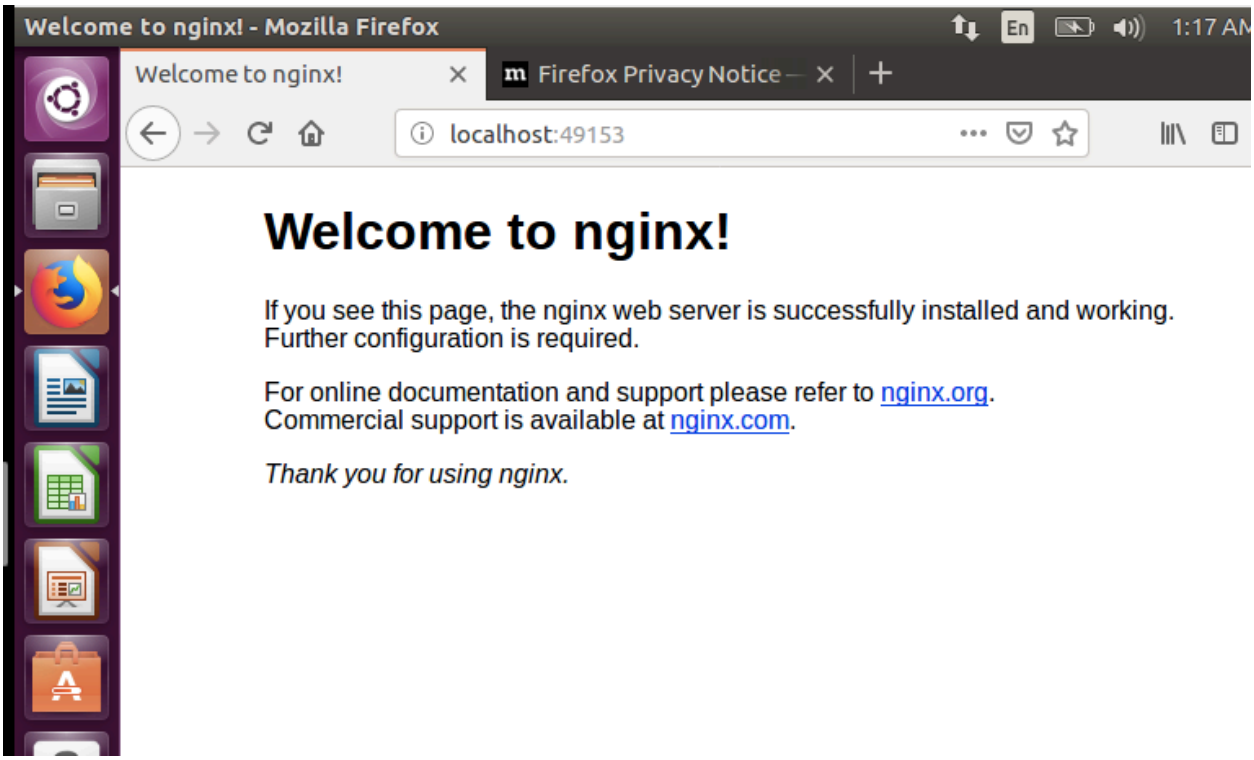
Task 3

In this task 3, we want to create our own webpage using nginx and docker by creating a html file.

```
osboxes@Amir_Mansha:~$ sudo docker run --name mynginx -P -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
f7ec5a41d630: Pull complete
aa1efa14b3bf: Pull complete
b78b95af9b17: Pull complete
c7d6bca2b8dc: Pull complete
cf16cd8e71e0: Pull complete
0241c68333ef: Pull complete
Digest: sha256:75a55d33ecc73c2a242450a9f1cc858499d468f077ea942867e
662c247b5e412
Status: Downloaded newer image for nginx:latest
ac728aab82074a2218bd1dbb279912501e8f50d9768527888dd666ca7b54a2a7
```

```
osboxes@Amir_Mansha: ~
osboxes@Amir_Mansha:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS
 PORTS                                    NAMES
b2f94b916b46   centos    "bash"                   3 minutes ago   Up 3 minute
                                          funny_beaver
0fe19b790330   nginx     "/docker-entrypoint.…"   8 minutes ago   Up 8 minute
 0.0.0.0:49153->80/tcp, :::49153->80/tcp  mynginx
osboxes@Amir_Mansha:~$
```

In the 1st screenshot of task 3- step 1, we are creating a docker container for nginx using the 1st command. Using the “ps” command we see the nginx image we created and the webserver container where it includes the port # 49153. We will use this port # to open the webserver using firefox. The “-P” means port which lets us have specific port so we don’t get conflicted of multiple nginx instances.

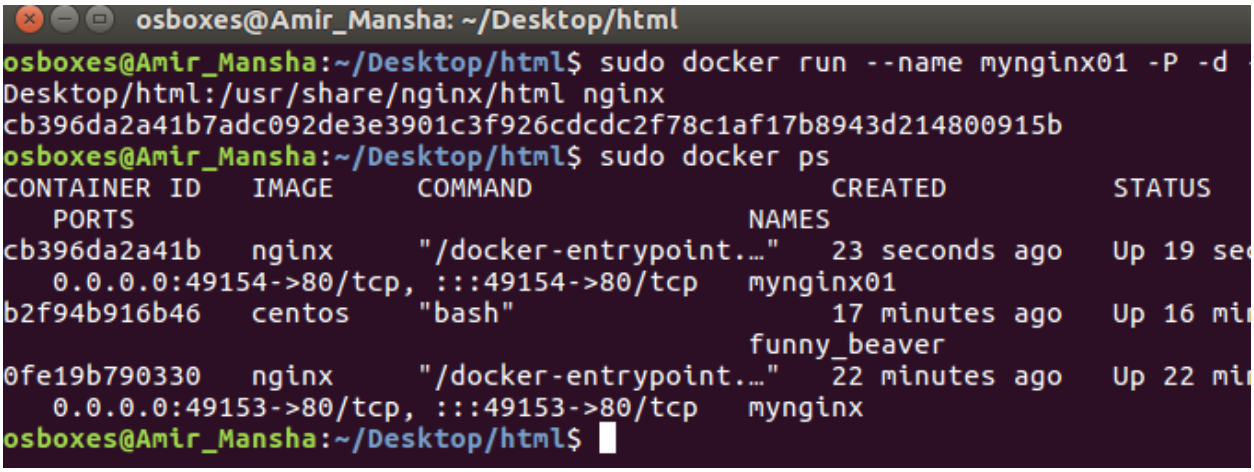


We typed the localhost:49153 to open the nginx webserver and as you can see it was successfully opened using the port# that was given to us in the screenshot above.

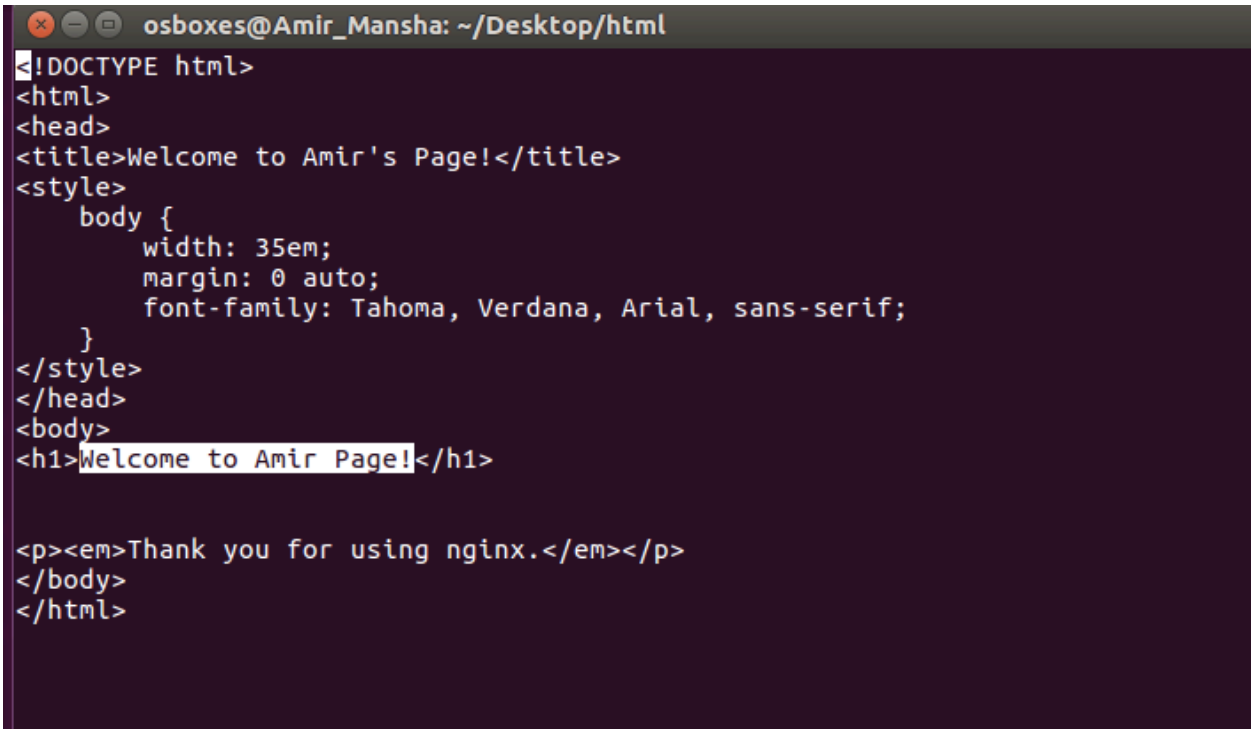
```
osboxes@Amir_Mansha:~$ curl http://localhost:49153
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
osboxes@Amir_Mansha:~$
```

I also used the curl command to see the contents of the html document for the nginx webpage.

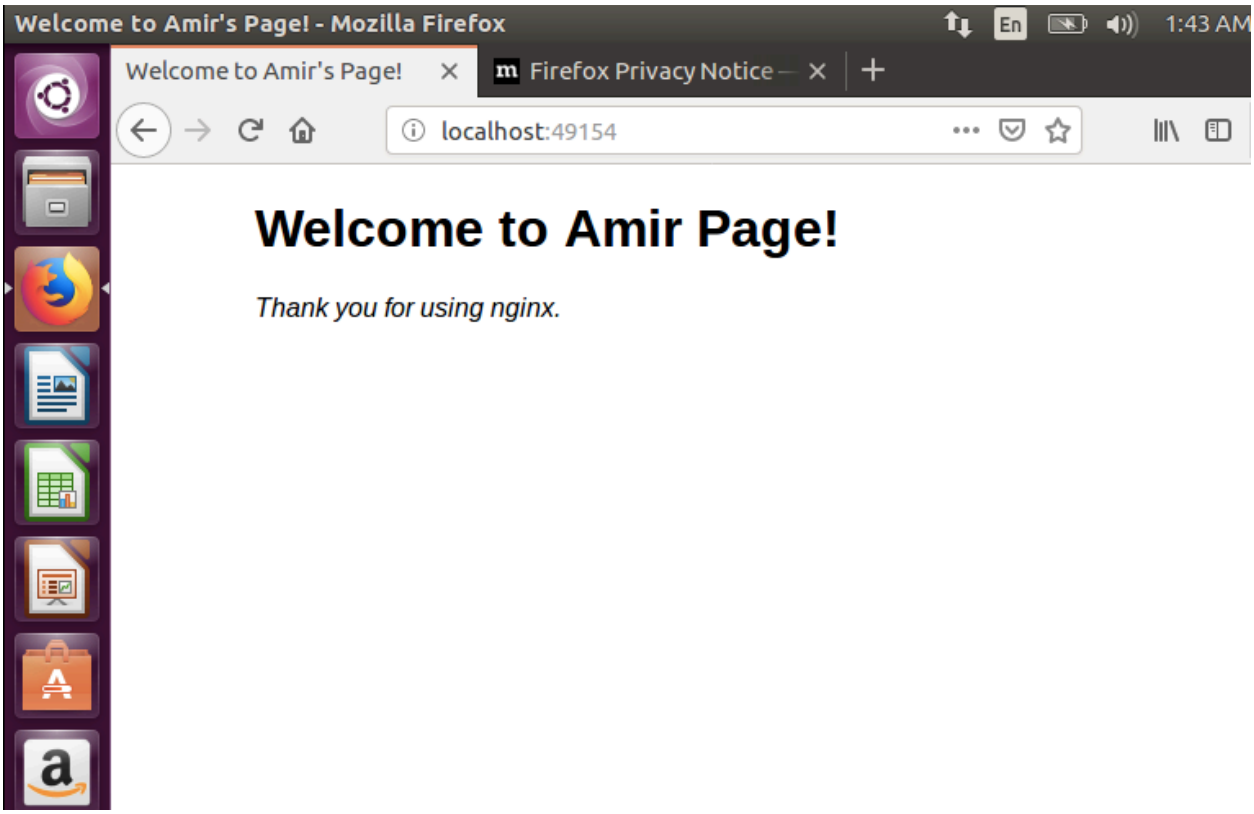
Task 3-Step 2



In this screenshot for task 3- step 2, I created another image using a different name "mynginx01" and used the "ps" command to get the new information for that image such as the port # which is 49154.



I created an index.html file in my desktop and used the contents from the previous webpage and just replaced “welcome to nginx” to “Welcome to Amir’s page!” and saved that html file in my desktop folder. I have to map the html file to the new container I created using the 1st command and provide the path to the desktop directory in where the index.html is in.



On Firefox, I used the port # 49154 that was given in the above screenshot to open the webpage in Firefox and display the html file I created and the title I put in the html contents in the file was displayed on the webpage which was “Welcome to Amir’s Page!”

```
osboxes@Amir_Mansha: ~/Desktop/html
osboxes@Amir_Mansha:~/Desktop/html$ sudo docker stop mynginx01
mynginx01
osboxes@Amir_Mansha:~/Desktop/html$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
 PORTS                                  NAMES
b2f94b916b46   centos    "bash"                   37 minutes ago   Up 36 minut
                                       funny_beaver
0fe19b790330   nginx     "/docker-entrypoint.…"   41 minutes ago   Up 41 minut
 0.0.0.0:49153->80/tcp, :::49153->80/tcp   mynginx
osboxes@Amir_Mansha:~/Desktop/html$
```

In this last screenshot, we stop the run the running container and here I stopped the “mynginx01” container which I used to display my custom webpage. I run the “ps” command and you can see the “mynginx01” is no longer there and stopped. You can only see instances that are running that I created before which are the “centos” and “mynginx” containers.