15.Docker exercise

Lab1: Docker basics

Exercise 1: Install docker

1. Log in to your VM.

2. Start terminal and elevate your privileges to root.

3. Run yum install docker.

4. After installation is finished, start docker by running this command systemctl start docker.

5. Also enable docker service automatic start with command systemctl enable docker.

6. Run docker version to see installed version.

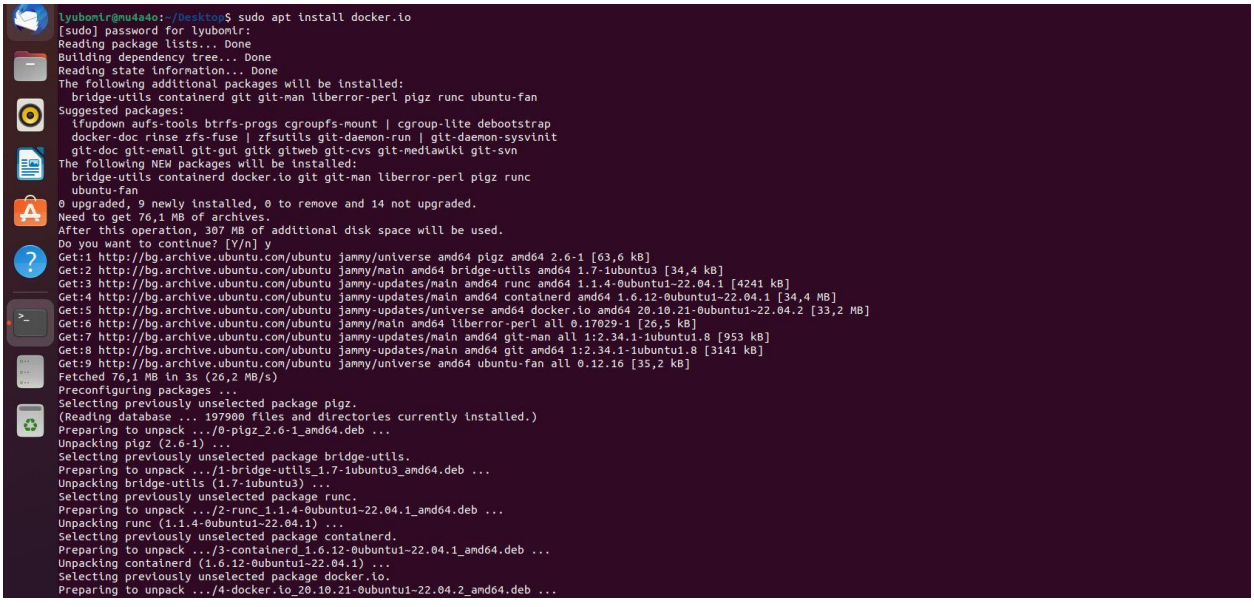
7. Run docker help to see list of available commands.

8. Search for a command (switch) that will show system-wide information for your instance of docker.

9. Test it by running docker <command you have discovered>.

10. From the output try to find where the information of number of containers and images is.

11. Also try to find whether this docker is part of a swarm.



```
Unpacking docker.io (20.10.21-0ubuntu1~22.04.2) ...
Selecting previously unselected package liberror-perl.
Preparing to unpack .../5-liberror-perl_0.17029-1_all.deb ...
Unpacking liberror-perl (0.17029-1) ...
Selecting previously unselected package git-man.
Preparing to unpack .../6-git-man_1%3a2.34.1-1ubuntu1.8_all.deb ...
Unpacking git-man (1:2.34.1-1ubuntu1.8) ...
Selecting previously unselected package git.
Preparing to unpack .../7-git_1%3a2.34.1-1ubuntu1.8_amd64.deb ...
Unpacking git (1:2.34.1-1ubuntu1.8) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../8-ubuntu-fan_0.12.16_all.deb ...
Unpacking ubuntu-fan (0.12.16) ...
Setting up runc (1.1.4-0ubuntu1~22.04.1) ...
Setting up liberror-perl (0.17029-1) ...
Setting up bridge-utils (1.7-1ubuntu3) ...
Setting up pigz (2.6-1) ...
Setting up git-man (1:2.34.1-1ubuntu1.8) ...
Setting up containerd (1.6.12-0ubuntu1~22.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service →
 /lib/systemd/system/containerd.service.
Setting up ubuntu-fan (0.12.16) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service →
 /lib/systemd/system/ubuntu-fan.service.
Setting up docker.io (20.10.21-0ubuntu1~22.04.2) ...
Adding group `docker' (GID 137) ...
Done.
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service →/li
b/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket →/lib/sy
stemd/system/docker.socket.
Setting up git (1:2.34.1-1ubuntu1.8) ...
Processing triggers for man-db (2.10.2-1) ...
lyubomir@mu4a4o:~/Desktop$
```

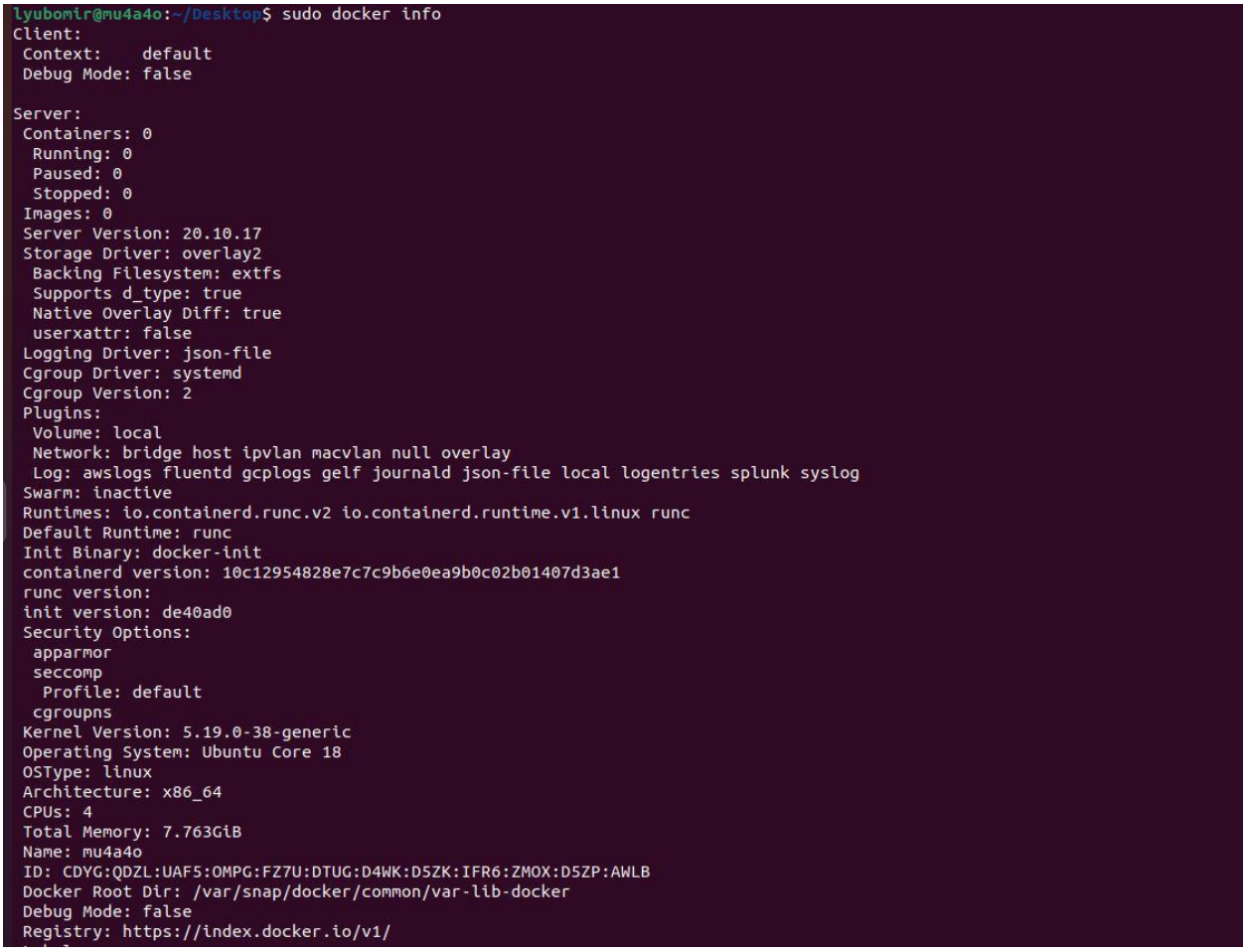
```
lyubomir@mu4a4o:~/Desktop$ docker help

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/home/lyubomir/.docker")
  -c, --context string     Name of the context to use to connect to the daemon (overrides
                           DOCKER_HOST env var and default context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"error"|"fatal")
                           (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
                           "/home/lyubomir/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/home/lyubomir/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/home/lyubomir/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  builder     Manage builds
  config      Manage Docker configs
  container   Manage containers
  context     Manage contexts
  image       Manage images
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes
```



Lab2: Creating images

Excercise1: Build a simple image

1. Create a Docker container that executes a simple bash script. Go to your home directory and run mkdir test.

Run cd test.

2. Create a simple script. Run vi test.sh.

3. Write the following in your script file:

#!/bin/bash

sleep 30

exit 1

4. Save the file. In vi editor press :wq.

5. Create a docker file. Run vi Dockerfile.

6. Write the following in our Dockerfile:

FROM alpine

ADD test.sh /

CMD /bin/bash /test.sh

7. Save your Dockerfile.

8. Build your image. Run docker build –t my-image1 ./

9. Now spawn a container. Run docker run - -name my-test1 my-image1.

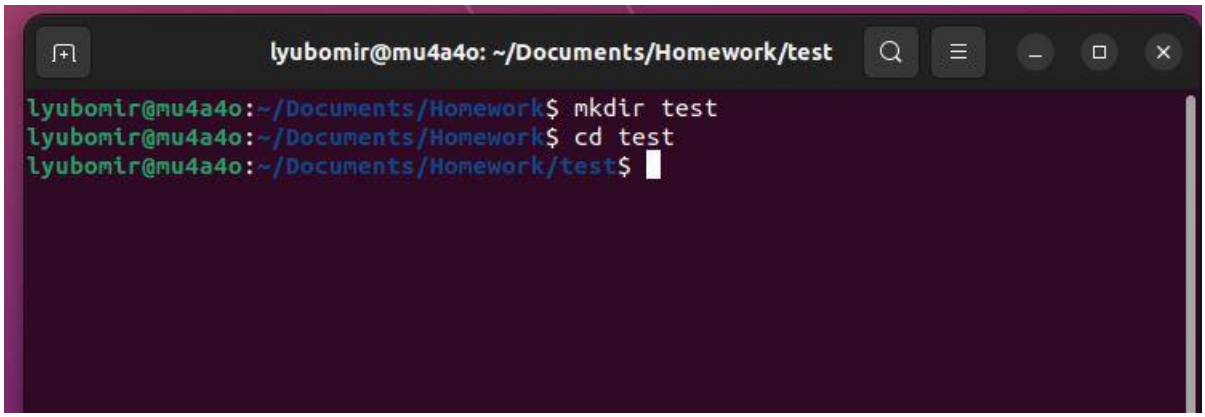
10. Do a docker ps –a. Do you see your container running? _____

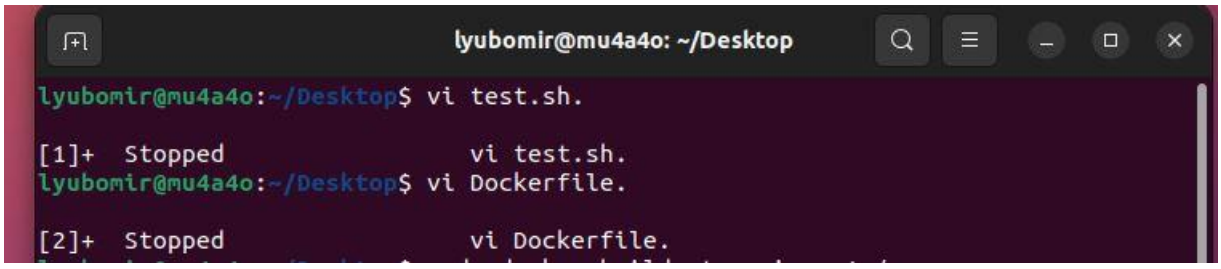
11. Do a docker logs my-test1. What is the output of the log?

Note: Because alpine is very light Image it does not have bash binaries.

12. Delete my-test. Run docker rm –f my-test1.

13. Delete my–image. Run docker rmi –f my-image1.





```
lyubomir@mu4a4o:/test$ sudo nano test.sh
lyubomir@mu4a4o:/test$ sudo nano Dockerfile
lyubomir@mu4a4o:/test$ sudo docker build -t my-image1 ./
Sending build context to Docker daemon  3.072kB
Step 1/3 : FROM alpine
latest: Pulling from library/alpine
f56be85fc22e: Pull complete
Digest: sha256:124c7d2707904eea7431fffe91522a01e5a861a624ee31d03372cc1d138a3126
Status: Downloaded newer image for alpine:latest
 ---> 9ed4aefc74f6
Step 2/3 : ADD test.sh /
 ---> 8b7a4ab34b9e
Step 3/3 : CMD /bin/bash /test.sh
 ---> Running in 5e5e0c6a51c7
Removing intermediate container 5e5e0c6a51c7
 ---> 08f1b1e9e095
Successfully built 08f1b1e9e095
Successfully tagged my-image1:latest
lyubomir@mu4a4o:/test$ sudo docker run --name my-test1 my-image1
/bin/sh: /bin/bash: not found
lyubomir@mu4a4o:/test$ sudo docker ps -a
CONTAINER ID   IMAGE       COMMAND                  CREATED              STATUS                          PORTS     NAMES
047fdea80a39   my-image1   "/bin/sh -c '/bin/ba…"   About a minute ago   Exited (127) About a minute ago             my-test1
lyubomir@mu4a4o:/test$ sudo docker logs my-test1
/bin/sh: /bin/bash: not found
lyubomir@mu4a4o:/test$ sudo docker rm -f my.test1
Error: No such container: my.test1
lyubomir@mu4a4o:/test$ sudo docker rm -f my-test1
my-test1
lyubomir@mu4a4o:/test$ sudo docker rmi -f my-image1
Untagged: my-image1:latest
Deleted: sha256:08f1b1e9e0955c046766d6c80e54a99a1cebaf5930f3d39c6107acf05dc9d509
Deleted: sha256:8b7a4ab34b9e1537fbf2ab9a9b4296ed2937681ba71a45be270257830031975f
Deleted: sha256:a736a283837e1055ecea881f6c5a5ee342bb79a4b0664105117b66cb157a4d70
lyubomir@mu4a4o:/test$
```

```
lyubomir@mu4a4o:/test$ sudo nano Dockerfile
lyubomir@mu4a4o:/test$ sudo docker build -t my-image1 ./
Sending build context to Docker daemon  3.072kB
Step 1/3 : FROM alpine
 ---> 9ed4aefc74f6
Step 2/3 : ADD test.sh /
 ---> e3631c394f7a
Step 3/3 : CMD /bin/sh /test.sh
 ---> Running in 04aa74dd10e1
Removing intermediate container 04aa74dd10e1
 ---> f2e775e17e21
Successfully built f2e775e17e21
Successfully tagged my-image1:latest
lyubomir@mu4a4o:/test$ sudo docker run --name my-test1 my-image1
lyubomir@mu4a4o:/test$ sudo docker ps -a
CONTAINER ID   IMAGE       COMMAND                  CREATED              STATUS                      PORTS     NAMES
90d20b029658   my-image1   "/bin/sh -c '/bin/sh…"   About a minute ago   Exited (1) 38 seconds ago             my-test1
```

```
lyubomir@mu4a4o:/test$ sudo docker rm -f my-test1
my-test1
lyubomir@mu4a4o:/test$ sudo docker rmi -f my-image1
Untagged: my-image1:latest
Deleted: sha256:f2e775e17e217965a21d34456c20a4ee1cb4950c358cf75e2bb289ea9be6a70a
Deleted: sha256:e3631c394f7acce2d3ffc8a6f7d9ba823d92aab63ca561287493d68761f7871f
Deleted: sha256:a736a283837e1055ecea881f6c5a5ee342bb79a4b0664105117b66cb157a4d70
lyubomir@mu4a4o:/test$
```