Lab1: Docker basics

Exercise 1: Install docker:

1. Log in to your VM.

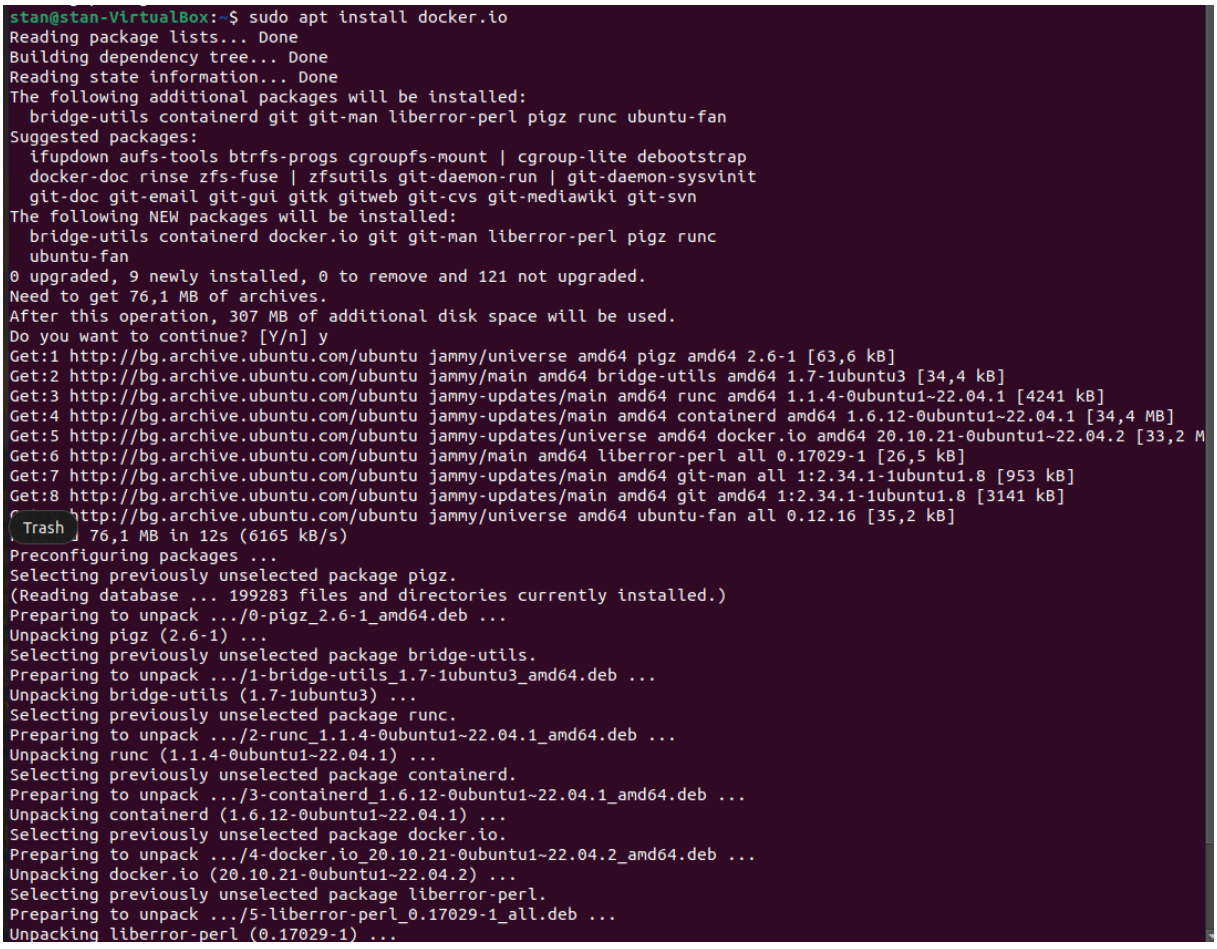
2. Start terminal and elevate your privileges to root.

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.



- downloading docker IO.

```
Fetched 76,1 MB in 12s (6165 kB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 199283 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.6-1_amd64.deb ...
Unpacking pigz (2.6-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7-1ubuntu3_amd64.deb ...
Unpacking bridge-utils (1.7-1ubuntu3) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.1.4-0ubuntu1~22.04.1_amd64.deb ...
Unpacking runc (1.1.4-0ubuntu1~22.04.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.6.12-0ubuntu1~22.04.1_amd64.deb ...
Unpacking containerd (1.6.12-0ubuntu1~22.04.1) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../4-docker.io_20.10.21-0ubuntu1~22.04.2_amd64.deb ...
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) ...
```

- downloading docker IO part2.

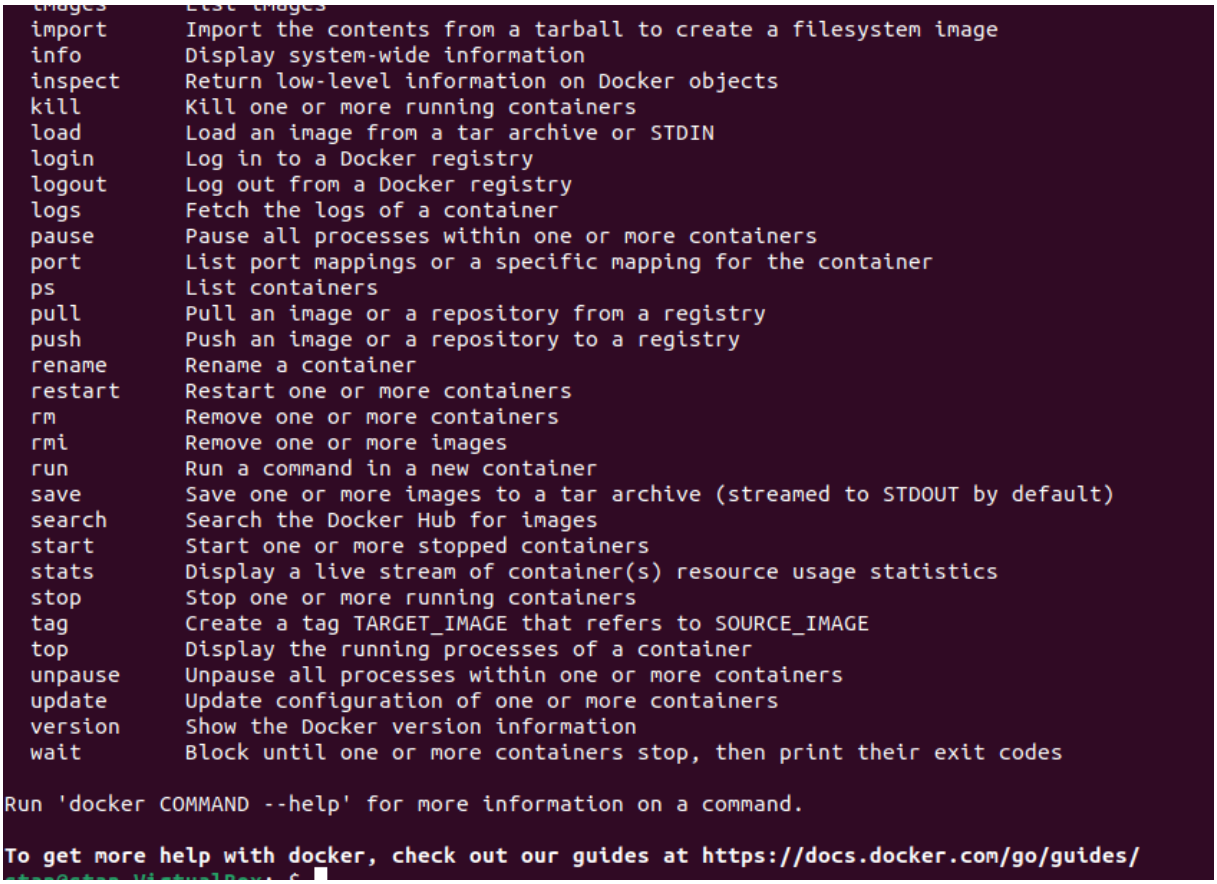
```
stan@stan-VirtualBox:~$ sudo snap install docker
docker 20.10.17 from Canonical✓ installed
stan@stan-VirtualBox:~$ docker --version
Docker version 20.10.21, build 20.10.21-0ubuntu1~22.04.2
stan@stan-VirtualBox:~$
```

- docker version and installing docker.

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

```
stan@stan-VirtualBox:~$ docker help

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default "/home/stan/.docker")
  -c, --context string     Name of the context to use to connect to the daemon (overrides DOCKER_HO
  -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 "
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/home/stan/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/home/stan/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/home/stan/.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

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
```

- Docker help command



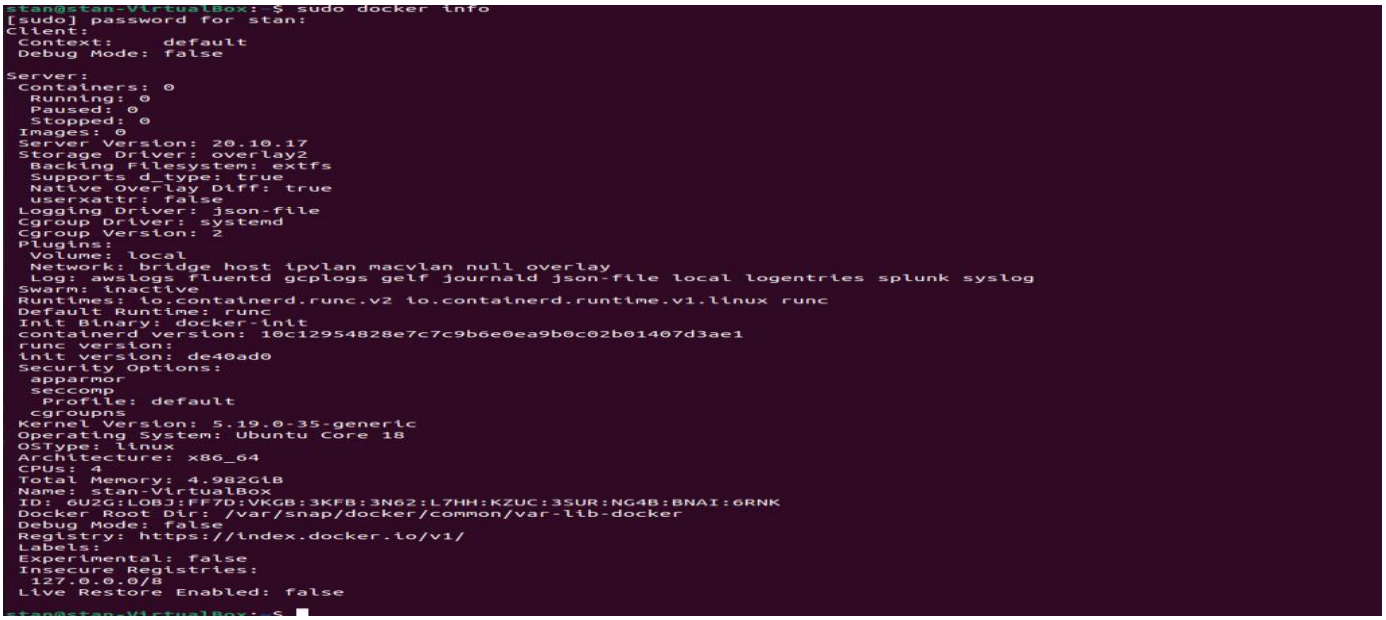
- Docker help command part 2.

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



- Swarm:inactive.

- docker info command

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

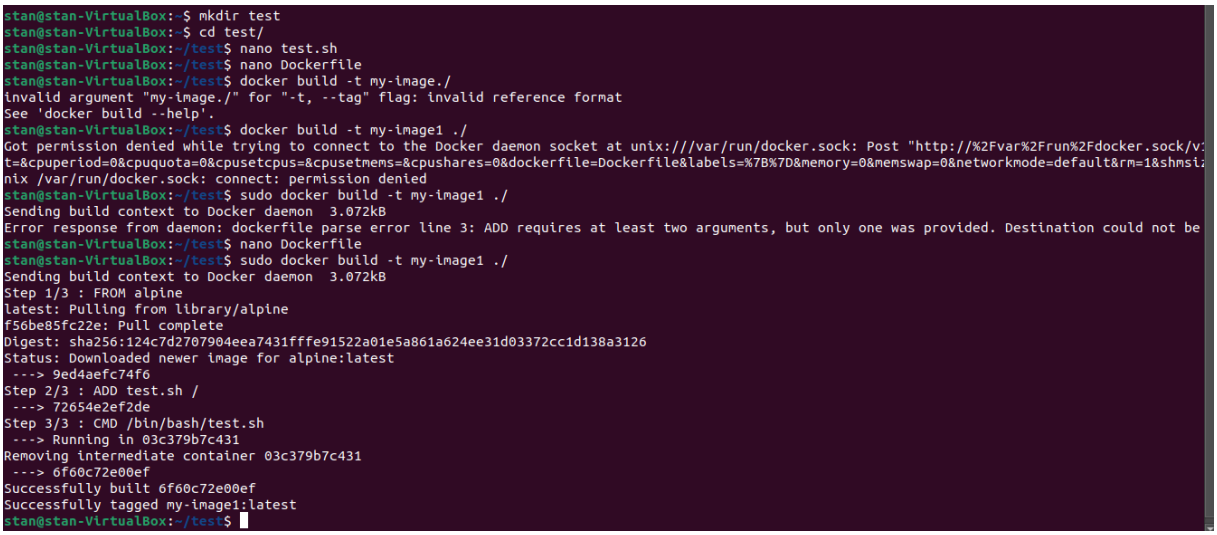
sleep 30

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:



- Creating directory and the 2 files and editing them, running the image

FROM alpine

ADD test.sh /

CMD /bin/bash /test.sh

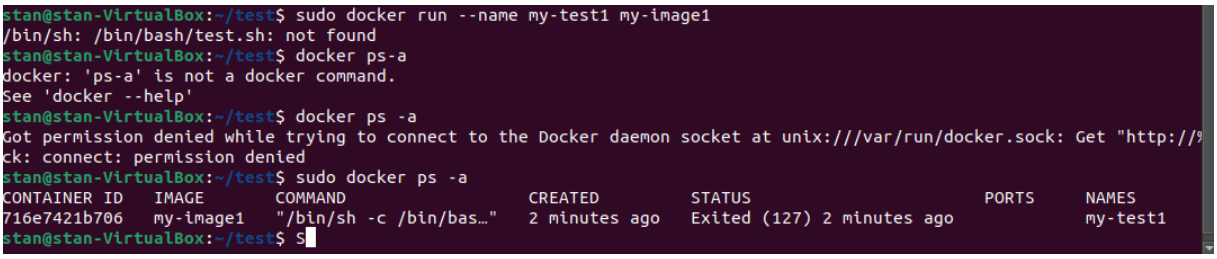
7. Save your Dockerfile.

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?



- It created the container, but the image does not have bash binaries

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.

```
stan@stan-VirtualBox:~/test$ sudo docker logs my-test1
/bin/sh: /bin/bash/test.sh: not found
stan@stan-VirtualBox:~/test$ sudo docker rm -f my-test1
my-test1
stan@stan-VirtualBox:~/test$ sudo docker rmi -f my-image1
Untagged: my-image1:latest
Deleted: sha256:6f60c72e00ef0851b93e5ef50b702220c31d5f274fb1fefb60e865235ba9c356
Deleted: sha256:72654e2ef2de3b96a3424e6e122d9427a62cc47bc838bf41fa40566316da4360
Deleted: sha256:3daa008ccf2acf5fc2598779462f3ed615223086fe4d0545e844b763c0476a35
stan@stan-VirtualBox:~/test$
```

- deleting the image and the container.

Now correct your Dockerfile. In the last line replace CMD /bin/bash /test.sh with CMD

/bin/sh /test.sh.

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

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

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

```
stan@stan-VirtualBox:~/test$ sudo docker rmi -f my-image1
Untagged: my-image1:latest
Deleted: sha256:6f60c72e00ef0851b93e5ef50b702220c31d5f274fb1fefb60e865235ba9c356
Deleted: sha256:72654e2ef2de3b96a3424e6e122d9427a62cc47bc838bf41fa40566316da4360
Deleted: sha256:3daa008ccf2acf5fc2598779462f3ed615223086fe4d0545e844b763c0476a35
stan@stan-VirtualBox:~/test$ ll
total 16
drwxrwxr-x  2 stan stan 4096 мар 29 23:48 ./
drwxr-x--x 17 stan stan 4096 мар 29 23:42 ../
-rw-rw-r--  1 stan stan   50 мар 29 23:48 Dockerfile
-rw-rw-r--  1 stan stan   30 мар 29 23:43 test.sh
stan@stan-VirtualBox:~/test$ nano Dockerfile
stan@stan-VirtualBox:~/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 /
 ---> e5ad39a80986
Step 3/3 : CMD /bin/sh/test.sh
 ---> Running in 454839f0ea56
Removing intermediate container 454839f0ea56
 ---> 202af0d7f78e
Successfully built 202af0d7f78e
Successfully tagged my-image1:latest
stan@stan-VirtualBox:~/test$ docker run --name my-test-image1
"docker run" requires at least 1 argument.
See 'docker run --help'.

Usage:  docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container
stan@stan-VirtualBox:~/test$ sudo docker run --name my-test-image1 my image1
Unable to find image 'my:latest' locally
docker: Error response from daemon: pull access denied for my, repository does not exist or may require 'docker login
See 'docker run --help'.
stan@stan-VirtualBox:~/test$ sudo docker run --name my-test1 my-image1
/bin/sh: /bin/sh/test.sh: not found
stan@stan-VirtualBox:~/test$ sudo docker ps -a
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS                       PORTS     NAMES
c2e4484abcca   my-image1   "/bin/sh -c /bin/sh/…"   25 seconds ago   Exited (127) 23 seconds ago            my-test1
stan@stan-VirtualBox:~/test$ S
```

- Changing the Dockerfile and creating the image and conainer again.

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

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

```
stan@stan-VirtualBox:~/test$ sudo docker rm -f my-test1
my-test1
stan@stan-VirtualBox:~/test$ sudo docker rmi -f my-image1
Untagged: my-image1:latest
Deleted: sha256:202af0d7f78ecf4654d873c7c83cd0b62674dcff6eebe5b831da1d7ac6417255
Deleted: sha256:e5ad39a80986823120aa0bae1eb914d46053c3335ed8a3797c8637a651818c78
Deleted: sha256:3daa008ccf2acf5fc2598779462f3ed615223086fe4d0545e844b763c0476a35
stan@stan-VirtualBox:~/test$
```

- Deleting the test and image(2-nd time).