TP Docker

1. Start a container (no need to first download the image explicitly) that runs the latest version of the helloworld image:

- docker run hello-world:latest

2. Pull the latest version of the ubuntu Docker image.

- docker pull ubuntu:latest

3. Start a container running this ubuntu image; the container must start an interactive bash shell inside (launch container with -it).

- docker run -it –name my-ubuntu ubuntu

4. [On the bash that runs inside the container] list the processes running inside the container with ps. [Outside] Open a shell outside the container and list processes running on host machine.

- On the bash of the container my-ubuntu: ps

```
C:\Users\pmoll>docker run -it --name my_ubuntu ubuntu
root@5c26cdf417ee:/# ps
  PID TTY          TIME CMD
    1 pts/0    00:00:00 bash
    9 pts/0    00:00:00 ps
```

- Outside, on the host system bash (Windows): tasklist

```
C:\Users\pmoll>tasklist

Nombre de imagen                PID Nombre de sesión Núm. de ses Uso de memor
========================= ======== ================ =========== ============
System Idle Process               0 Services                   0          8 KB
System                            4 Services                   0      3.932 KB
Secure System                    76 Services                   0     24.604 KB
Registry                        116 Services                   0     26.876 KB
smss.exe                        528 Services                   0        128 KB
csrss.exe                       812 Services                   0      2.024 KB
wininit.exe                     896 Services                   0        116 KB
services.exe                   1020 Services                   0      6.068 KB
Lsalso.exe                      552 Services                   0        684 KB
lsass.exe                       628 Services                   0     16.796 KB
svchost.exe                    1100 Services                   0     32.616 KB
fontdrvhost.exe                1140 Services                   0         96 KB
WUDFHost.exe                   1152 Services                   0      3.592 KB
svchost.exe                    1276 Services                   0     16.532 KB
WUDFHost.exe                   1344 Services                   0      4.548 KB
svchost.exe                    1356 Services                   0      2.948 KB
svchost.exe                    1648 Services                   0      1.888 KB
svchost.exe                    1684 Services                   0        344 KB
svchost.exe                    1740 Services                   0      3.228 KB
svchost.exe                    1764 Services                   0      7.324 KB
svchost.exe                    1772 Services                   0      6.216 KB
svchost.exe                    1844 Services                   0      1.540 KB
svchost.exe                    1860 Services                   0      4.772 KB
svchost.exe                    1908 Services                   0      2.764 KB
svchost.exe                    2028 Services                   0     12.896 KB
```

5. [In another terminal] Create another container, also with a bash attached, of the same ubuntu image, and give it an explicit name, ex: myubuntu. Then check that a file created on container myubuntu (touch myfile.txt) does not appear in the first ubuntu container.

- docker run -it –name myubuntu ubuntu

- touch myfile.txt – To create files

```
C:\Users\pmoll>docker run -it --name myubuntu ubuntu
root@b023fd653e40:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@b023fd653e40:/# touch file.txt
root@b023fd653e40:/# ls
bin   dev  file.txt  lib    lib64   media  opt   root  sbin  sys  usr
boot  etc  home      lib32  libx32  mnt    proc  run   srv   tmp  var
root@b023fd653e40:/# exit
exit

C:\Users\pmoll>docker run -it --name my_ubuntu ubuntu
root@9c466f0aba0d:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@9c466f0aba0d:/#
```

#Note: When creating a container, always, if need to operate with it, in -it mode.

- docker run -it –name my_container image -> Status: “Running”
- docker run --name my_container_alp -d -it image -> Status: “Running”
- docker run –name my_container image -> Status: “Exited”
- docker run -d –name my_container image -> Status: Exited

6. Find out which instructions can be performed in docker using docker help; deduce how you can list images, and list containers. Obtain more information about the command docker ps and deduce how to list the docker containers present on the machine including the ones that are not presently running.

- docker –help – To list help options
- docker images – To list all images
- docker ps – To list containers on running mode
- docker ps -a - To list containers on running and exit mode

```
C:\Users\pmoll>docker images
REPOSITORY                                    TAG       IMAGE ID       CREATED        SIZE
ubuntu                                        latest    e34e831650c1   2 weeks ago    77.9MB
docker.elastic.co/elasticsearch/elasticsearch 8.12.0    014ecd90309d   2 weeks ago    1.36GB
mongo                                         latest    7ee26c8012da   3 weeks ago    757MB
hello-world                                   latest    d2c94e258dcb   9 months ago   13.3kB
docker.elastic.co/elasticsearch/elasticsearch 7.10.0    37190fe5beea   3 years ago    774MB
```

```
C:\Users\pmoll>docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
f658adfc5043   ubuntu    "/bin/bash"   4 minutes ago    Up 3 minutes              my_ubuntu
```

```
C:\Users\pmoll>docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED          STATUS                      PORTS     NAMES
f658adfc5043   ubuntu        "/bin/bash"   5 seconds ago    Exited (0) 2 seconds ago              my_ubuntu
85c284556794   hello-world   "/hello"      4 days ago       Exited (0) 4 days ago                 awesome_cori
```

7. Start a container my_alp running the alpine image in the background (-d). Check the image size.

```
C:\Users\pmoll>docker pull alpine:latest
latest: Pulling from library/alpine
4abcf2066143: Pull complete
Digest: sha256:c5b1261d6d3e43071626931fc004f70149baeba2c8ec672bd4f27761f8e1ad6b
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview alpine:latest

C:\Users\pmoll>docker images
REPOSITORY                                    TAG       IMAGE ID       CREATED        SIZE
alpine                                        latest    05455a08881e   47 hours ago   7.38MB
ubuntu                                        latest    e34e831650c1   2 weeks ago    77.9MB
docker.elastic.co/elasticsearch/elasticsearch 8.12.0    014ecd90309d   2 weeks ago    1.36GB
mongo                                         latest    7ee26c8012da   3 weeks ago    757MB
hello-world                                   latest    d2c94e258dcb   9 months ago   13.3kB
docker.elastic.co/elasticsearch/elasticsearch 7.10.0    37190fe5beea   3 years ago    774MB
```

- docker run –name my_alp -d -it alpine

8. [From a terminal outside the container] Execute instructions on your container my_alp from outside the container: first create a file a.txt on the container. Then list the files at the root.

- From inside my_alp:
  - docker run –name my_alp -d -it alpine
  - You can't because the image from the container doesn't have a bash.
- From inside my_ubuntu:
  - docker run –name my_ubuntu -d -it ubuntu
  - docker exec my_ubuntu bash
  - Inside bash: touch a.txt
  - Inside bash: ls
- From outside:
  - docker run –name my_alp -d -it alpine
  - Outside (no bash): docker exec my_alp touch a.txt
  - Outside (no bash): docker exec my_alp ls
- From outside:
  - docker run –name my_ubuntu -d -it ubuntu
  - Outside (no bash): docker exec my_ubuntu touch a.txt
  - Outside (no bash): docker exec my_ubuntu ls

9. (Optional) Execute and understand the following two instructions:

- docker exec -e VAR_A =1 -e VAR_B =2 my_alp env

  - The first instruction defines 2 environment variables on my_alp, then prints all environment variables within the container, including the two variables VAR_A and VAR_B set respectively to 1 and 2. NB: those variables are only set for the instruction; if we afterwards execute another instruction that asks for the environment variables on the same container, it will not report those.

- docker exec -w / usr / bin my_alp sh -c 'printenv '

  - The second instruction specifies the directory where the instruction is executed. Since the instruction then moves to

parent directory and print the resulting working directory, it prints /usr.

10. Interrupt the container with docker stop my_alp. Check its status, and check that you cannot perform instructions on the container. Then restart the container with docker restart my_alp. Check that the file you created is still in the container.

```
C:\Users\pmoll>docker run --name my_alp -d -id alpine
c6ee8a29163187e3f88f60fe6734a33aa96f6366f614ceb6e8436057bd25f4aa

C:\Users\pmoll>docker exec my_alp touch a.txt

C:\Users\pmoll>docker exec my_alp ls
a.txt
bin
dev
etc
home
lib
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var

C:\Users\pmoll>docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED          STATUS                     PORTS     NAMES
c6ee8a291631   alpine        "/bin/sh"     50 seconds ago   Up 49 seconds                        my_alp
51b1a49a03fc   ubuntu        "/bin/bash"   21 minutes ago   Up 20 minutes                        myubuntu
85c284556794   hello-world   "/hello"      4 days ago       Exited (0) 4 days ago                awesome_cori

C:\Users\pmoll>docker stop my_alp
my_alp

C:\Users\pmoll>docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED              STATUS                          PORTS     NAMES
c6ee8a291631   alpine        "/bin/sh"     About a minute ago   Exited (137) 8 seconds ago                my_alp
51b1a49a03fc   ubuntu        "/bin/bash"   21 minutes ago       Up 21 minutes                             myubuntu
85c284556794   hello-world   "/hello"      4 days ago           Exited (0) 4 days ago                     awesome_cori

C:\Users\pmoll>docker restart my_alp
my_alp

C:\Users\pmoll>docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED          STATUS                     PORTS     NAMES
c6ee8a291631   alpine        "/bin/sh"     3 minutes ago    Up 9 seconds                         my_alp
51b1a49a03fc   ubuntu        "/bin/bash"   24 minutes ago   Up 24 minutes                        myubuntu
85c284556794   hello-world   "/hello"      4 days ago       Exited (0) 4 days ago                awesome_cori
```

#Note: Use docker start to start a container that is not currently running and use docker restart to stop and then start a container that is already running (or to start a stopped container). The docker restart command is essentially a convenient way to perform a stop and start operation in one step.

11. Remove the hello-world container you created, with docker rm. and remove the running ubuntu containers.

```
C:\Users\pmoll>docker rm my_alp
Error response from daemon: You cannot remove a running container c6ee8a29163187e3f88f60fe6734a33aa96f6366f614ceb6
e8436057bd25f4aa. Stop the container before attempting removal or force remove

C:\Users\pmoll>docker stop my_alp
my_alp

C:\Users\pmoll>docker stop myubuntu
myubuntu

C:\Users\pmoll>docker rm my_alp
my_alp

C:\Users\pmoll>docker rm myubuntu
myubuntu
```

12. Copy a file from your host machines into the container my_alp.

```
C:\Users\pmoll>docker exec my_alp ls
bin
dev
etc
home
lib
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var

C:\Users\pmoll>docker cp "C:\Users\pmoll\Desktop\enron.json" my_alp:/
Successfully copied 15.8MB to my_alp:/

C:\Users\pmoll>docker exec my_alp ls
bin
dev
enron.json
etc
home
lib
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var

C:\Users\pmoll>docker cp "C:\Users\pmoll\Desktop\enron.json" my_alp:/usr
Successfully copied 15.8MB to my_alp:/usr

C:\Users\pmoll>docker exec my_alp ls /usr
bin
enron.json
lib
local
sbin
share

C:\Users\pmoll>docker exec my_alp rm /enron.json

C:\Users\pmoll>docker exec my_alp rm /usr/enron.json

C:\Users\pmoll>docker exec my_alp ls /usr
bin
lib
local
sbin
share

C:\Users\pmoll>docker exec my_alp ls
bin
dev
etc
home
lib
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var

C:\Users\pmoll>docker rm my_alp
Error response from daemon: You cannot remove a running container 442de0d2039cb9fefd7ab6d25b341ce613884245cebf99928bf484c3f1fa7958. Stop the container before attempting removal or force remove

C:\Users\pmoll>docker stop my_alp
my_alp

C:\Users\pmoll>docker rm my_alp
my_alp
```