**TP Docker**

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

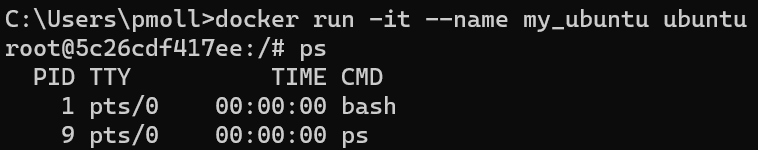
* docker run hello-world:latest

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

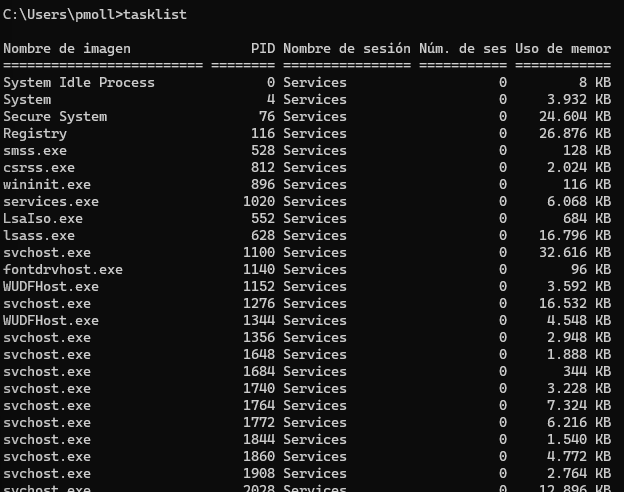
* docker pull ubuntu:latest

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

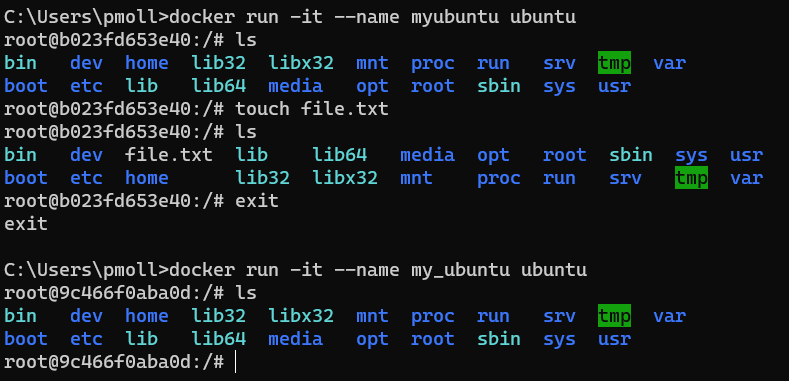
4. [On the bash that runs inside the container] list the processes running inside the container with ps. [Outside] Open a shell outside the container and list processes running on host machine.

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



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

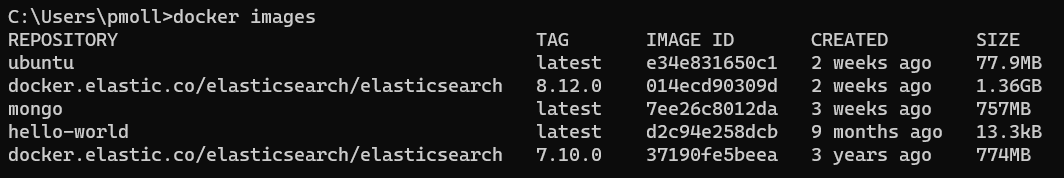
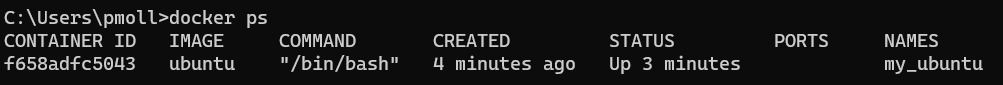
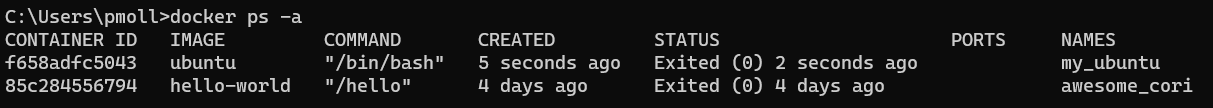
5. [In another terminal] Create another container, also with a bash attached, of the same ubuntu image, and give it an explicit name, ex: myubuntu. Then check that a file created on container myubuntu (touch myfile.txt) does not appear in the first ubuntu container.

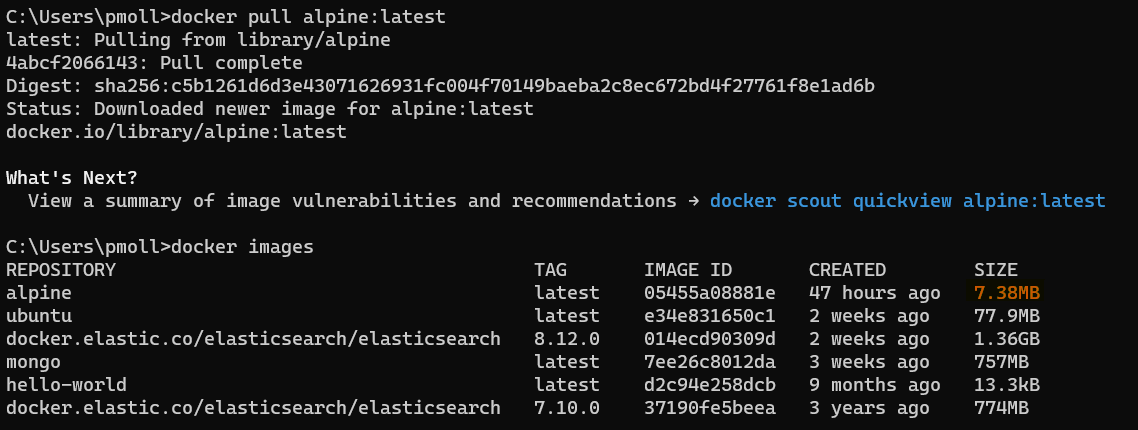
* docker run -it –name myubuntu ubuntu
* touch myfile.txt – To create files

#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

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

* docker run –name my\_alp -d -it alpine

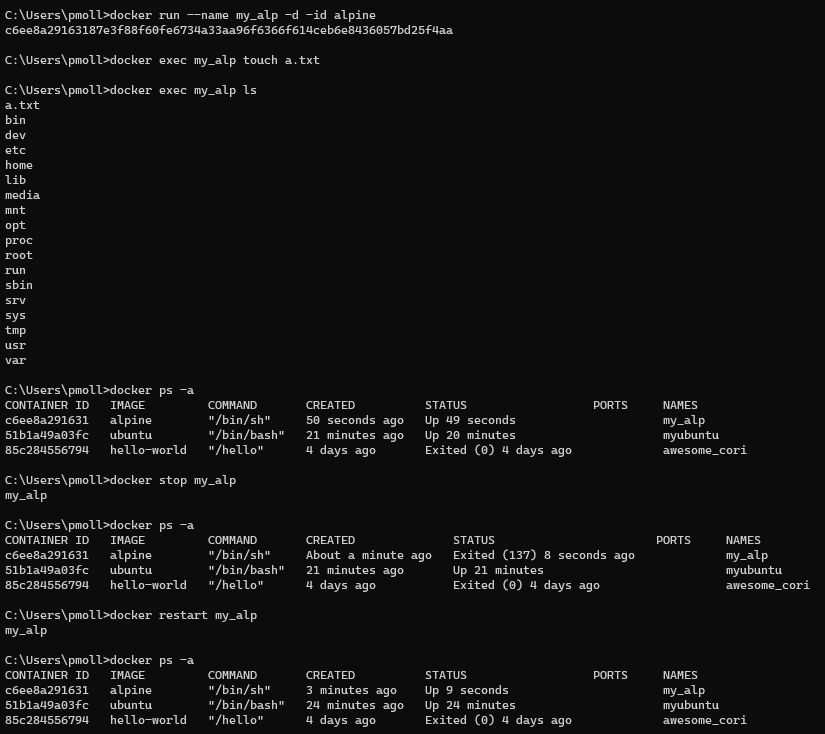
8. [From a terminal outside the container] Execute instructions on your container my\_alp from outside the container: first create a file a.txt on the container. Then list the files at the root.

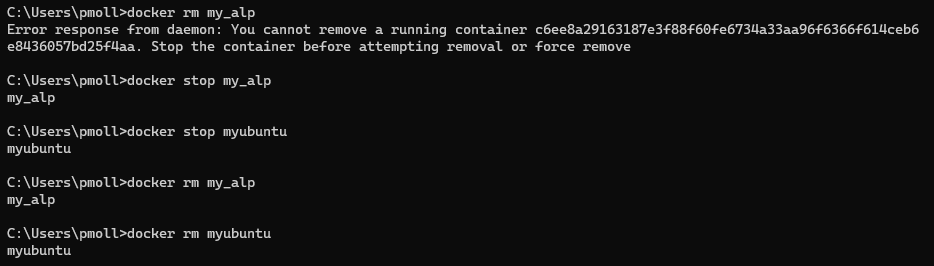
* From inside my\_alp:
  + docker run –name my\_alp -d -it alpine
  + You can’t because the image from the container doesn’t have a bash.
* From inside my\_ubuntu:
  + docker run –name my\_ubuntu -d -it ubuntu
  + docker exec my\_ubuntu bash
  + Inside bash: touch a.txt
  + Inside bash: ls
* From outside:
  + 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

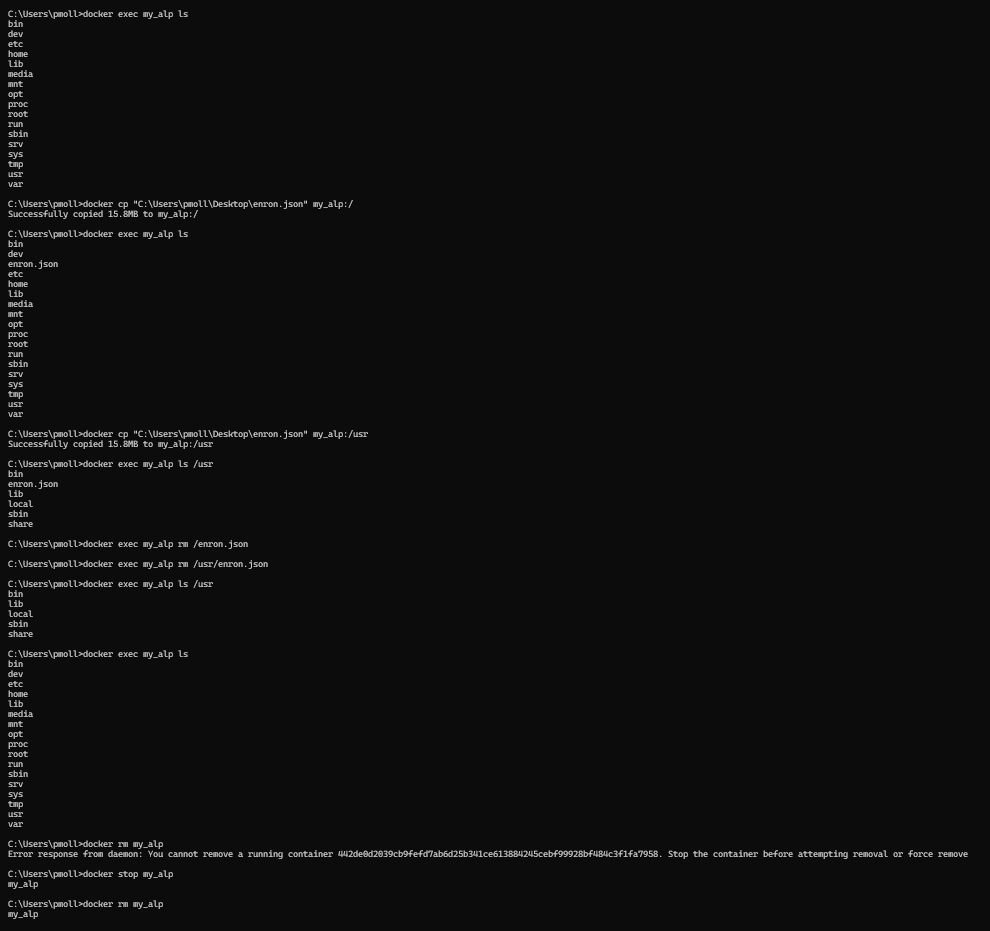
9. (Optional) Execute and understand the following two instructions:

* 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.

  
  
#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.

12. Copy a file from your host machines into the container my\_alp.