UNIT 03 & 04 Assignment: DOCKER

The aim of the assignment is to "containerize" Git using Docker. We will do this in two ways.

1. Running a script non-interactively
2. Running a script interactively

Create the required containers to complete Part A |B | C

INSTRUCTIONS

- You must create a document with screenshots to explain step by step that the two parts work. The filename format will be ***Units3-4_Name_Surname.pdf***.
- You must also submit the script required in a and b. The filename format will be ***Units3-4_Name_Surname.sh***.
- So, the submission will be a ZIP file containing: the document with screenshots and the script. The filename format will be ***Units3-4_Name_Surname.zip***.

NOTE: ALL THE EXPLANATIONS AND SCREENSHOTS MUST BE IN ENGLISH.

PART A) RUNNING A SCRIPT NON-INTERACTIVELY

For this part, you have to create a script without parameters which will take the next steps:

1. Install Git.
2. Configure Git with your name and e-mail.
3. It creates a new folder called *sytems_initials* into */root* (if your name is John Doe, the folder will be */root/systems_JD* as a result).
4. The folder should not be empty and will contain something which can be a blank file or subfolder.
5. This new content of */root/systems_initials* will be used later to create a repository (you can create sample files or subfolders from the script with *touch* or *mkdir* respectively).
6. Then, we change the current directory to */root/systems_initials* (remember that you must replace "initials" with the corresponding ones to your name and surnames).
7. Once there, we will create a repository (*git init)* and commit all sample files or subfolders created from the script (remember that we first need to run "*git add*" and then "*git commit*").
8. The script output will be the current state running *git log*.

Finally, the script will be executed in a container based on the Ubuntu image.

1. For this purpose, you must create a Docker volume.
2. The script will be manually copied to the volume folder (run "docker inspect volume-name" to check the real folder).
3. Afterwards, the Ubuntu container will be run with the command bash which executes the script from the volume.
4. You can associate the volume with the Linux folder you want, for example /root (the command would be "bash /root/script.sh") in this case.

What happens?

At this point, we have executed the script and we have the output of git log. But we cannot interact with the container. So, let's move on the next part, which makes more sense.

PART B) RUNNING A SCRIPT INTERACTIVELY

We will use the same script as in the previous part.

1. Although this time we will run an interactive Ubuntu container with a shell bash.
2. The script will be in a volume too.
3. But we must manually run the script from the terminal.

What happens?

This alternative requires to manually execute the script from the container. If you "exit" from the interactive bash, the container will stop.
You can restart the container with docker start. You must also explain how to run the same container again with an interactive terminal.

PART C) REMOVE ALL THE CONTAINERS AND IMAGES CREATED

Explain the commands you need.

STEP BY STEP Part A

Note: Docker has been already installed.

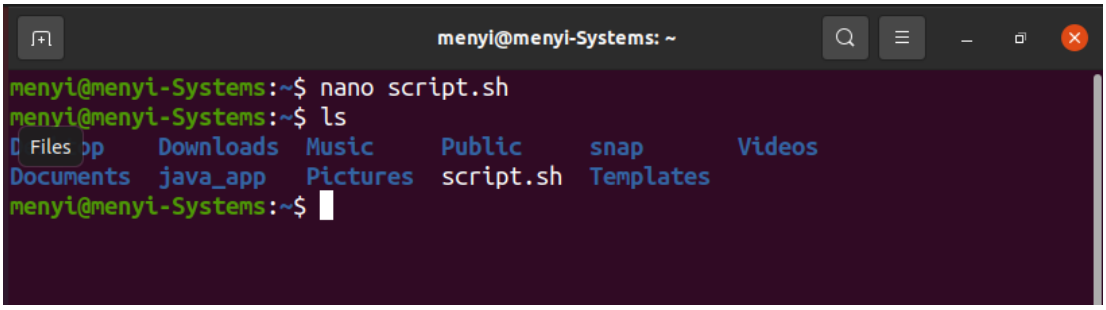
Create and save the script name ***script.sh*** with code according to the instructions.

The script.sh code

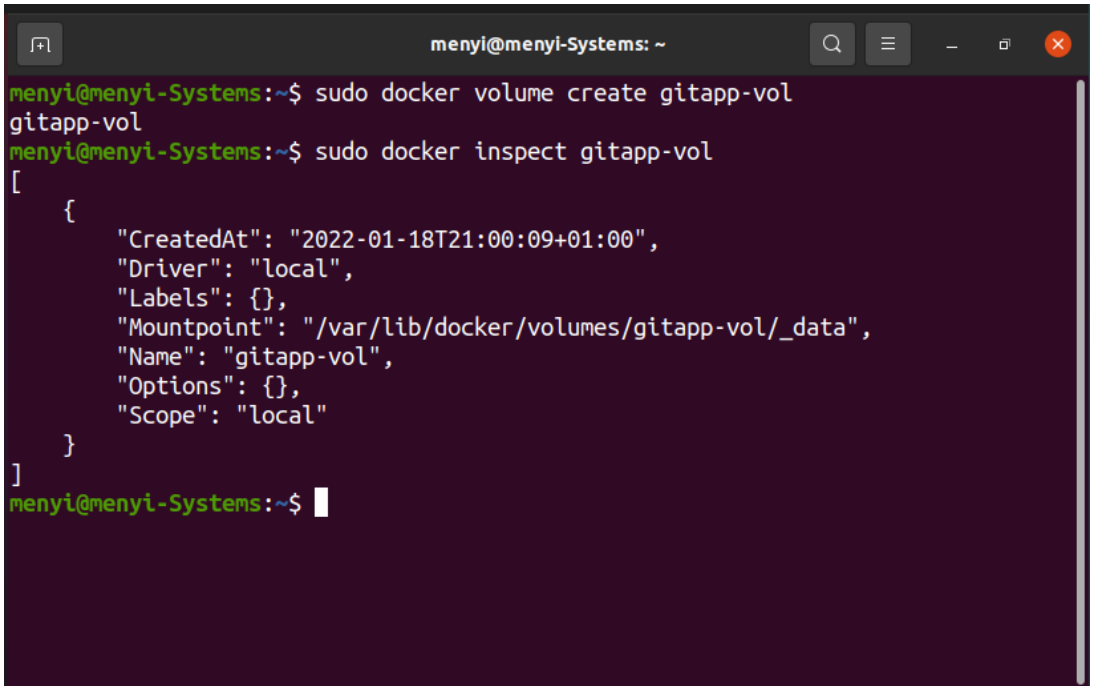
To create the final code, each command was tested until the output was the required.

Confirmation dialogs will appear, and the answers to those must be yes. So, in order to continue the script running, the parameter *–y (yes)* is used.

Also, and *autoremove -y* command has been added to prevent an error from previous Git installations.

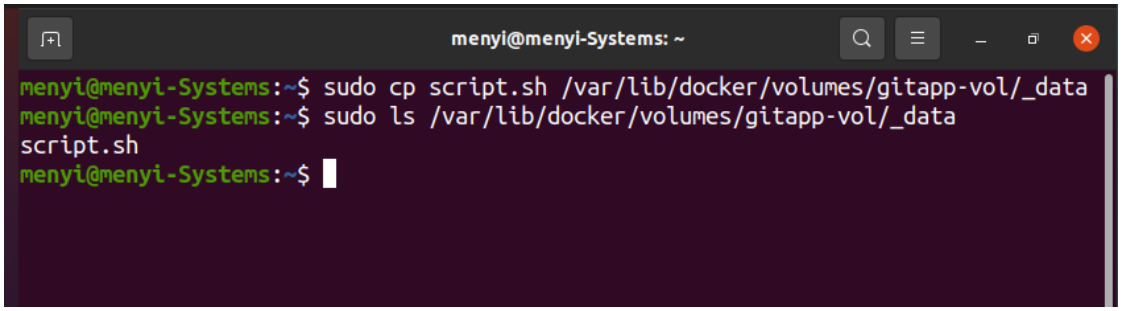


Create a **volume** named ***gitapp-vol*** to run the script already created, and check where it is really located at by using *docker inspect*



Notice that **mountpoint** is: */var/lib/docker/volumes/gitapp-vol/_data*

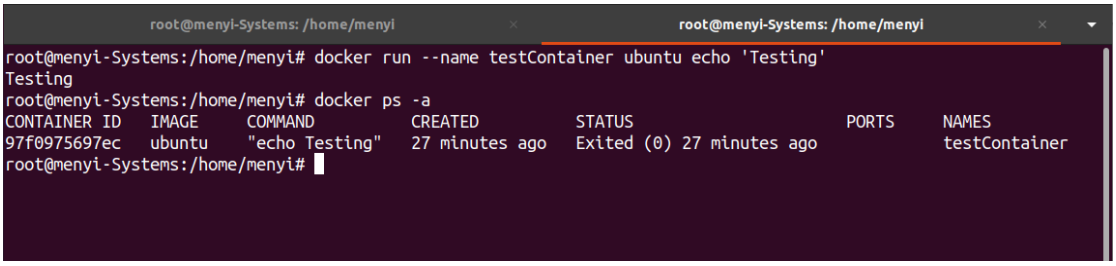
Copy the script file ***script.sh*** into the volume folder (mountpoint) and check it



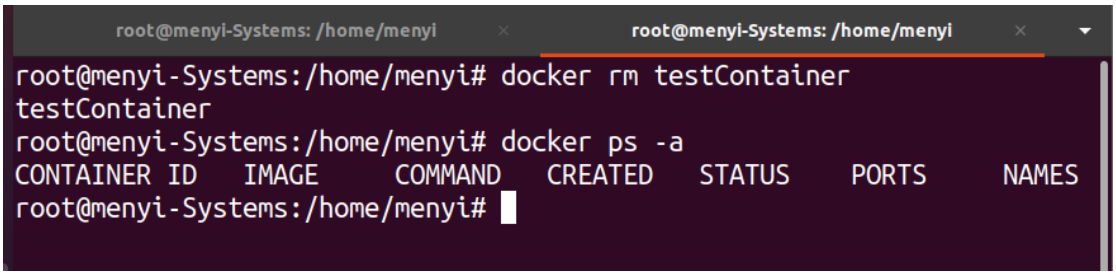
Also, check permissions to the script file, and grant permission to all users to execute the file.

At this point, everything is ready to create the Ubuntu image, and to initialize a non-interactive test container named *testContainer* using the ***--name*** parameter.

Then, check it´s all working properly by using ***docker ps –a*** command.

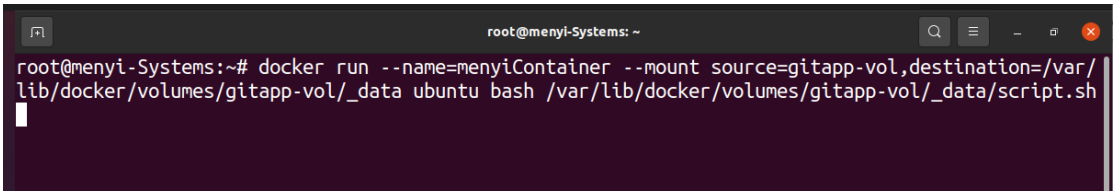


Once the test is done, remove the container and all ready to run the script file created

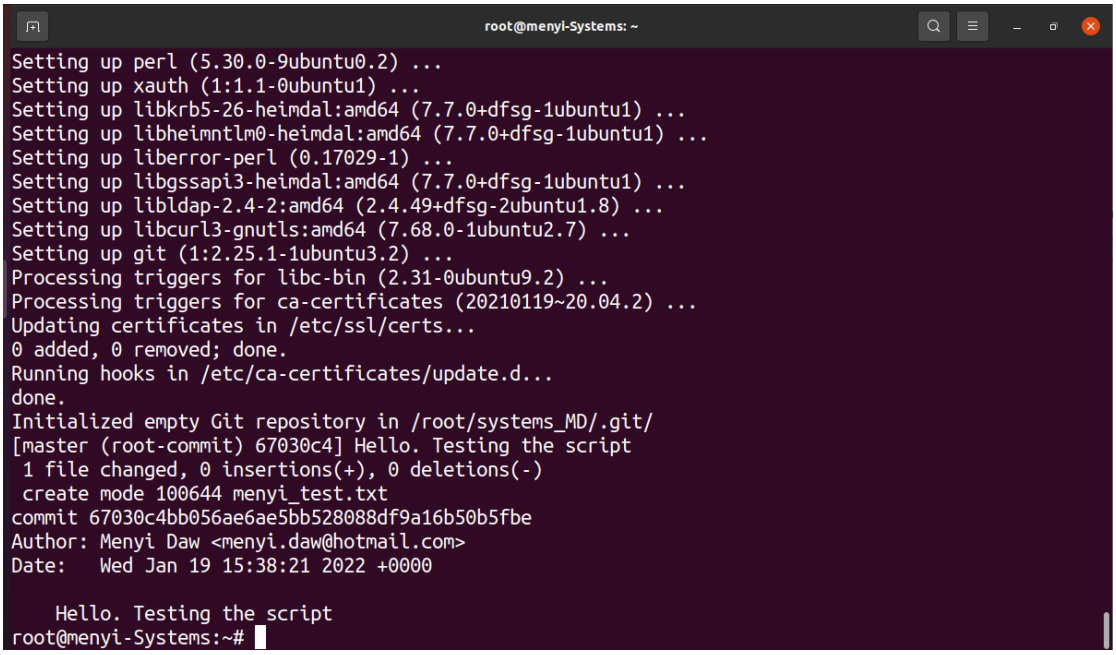


Initialise a new non-interactive container based on Ubuntu image,

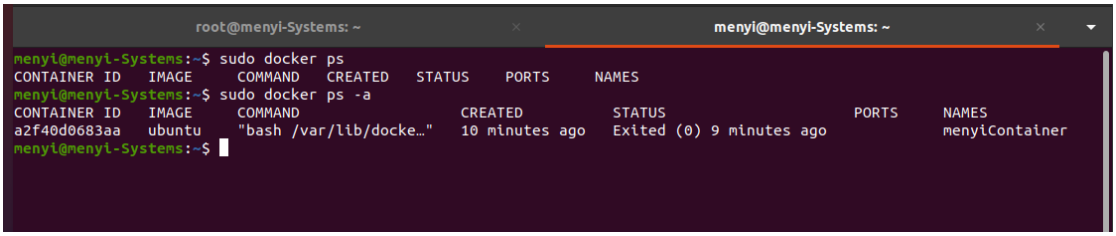
named ***menyiContainer*** and run the script file named *script.sh*



The script has been correctly executed, and the output shows the *git log*



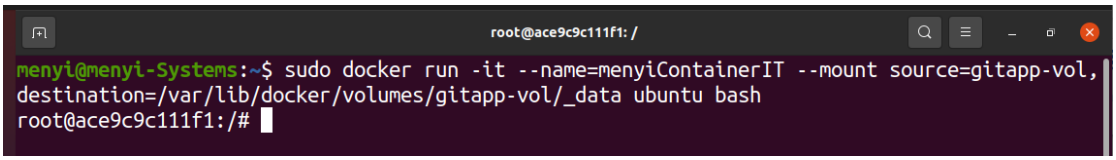
But interact with the container is not possible, check that status is *exited*



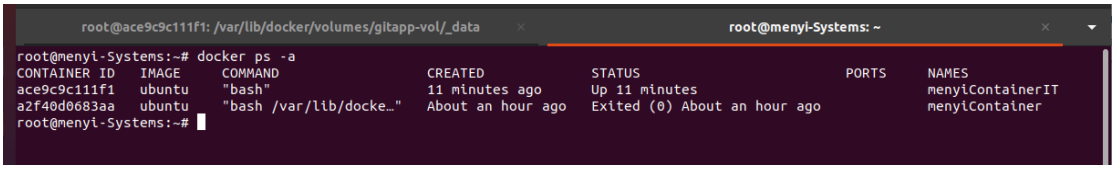
STEP BY STEP Part B

Using the volume created before *gitapp-vol,*

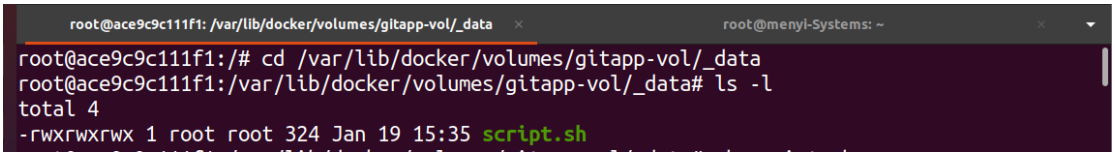
run an interactive Ubuntu container (name *menyiContainerIT*) with a shell bash



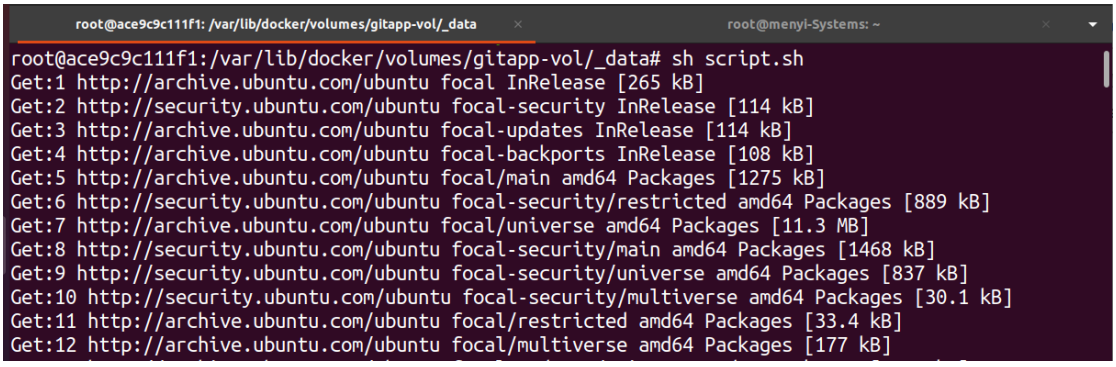
Notice: Log-in as a new user from the terminal, which id matches the id container *menyiContainerIT*



Change directory to script directory and check file permissions.



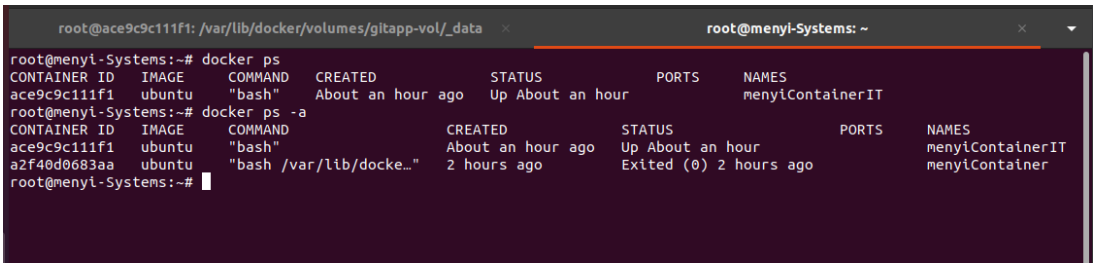
All ready to manually run the script from the terminal



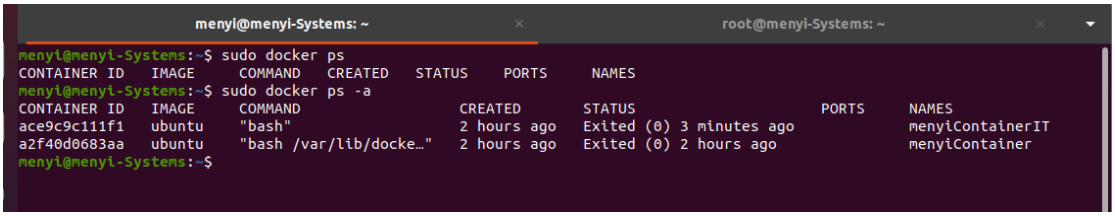
The script has worked, *git log* is output, and the container is still running (notice user id)

By using *docker ps* and *docker ps –a,* check the difference

between the interactive and non-interactive container status

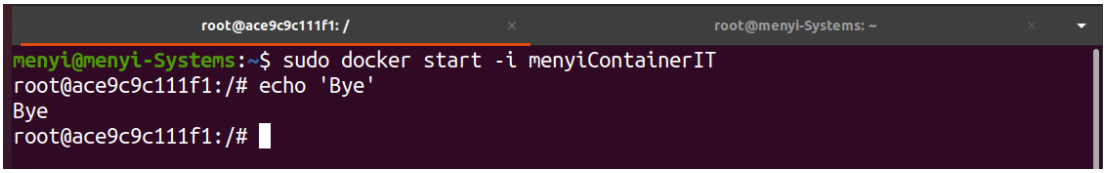


To exit from interactive bash, just type *exit* and also the container will stop.



Easily, restart the container with *docker start*, however the terminal will run on the background,

We can use restart and then attach or directly the *–i* parameter when restarting the container



STEP BY STEP Part C

Remove all the containers and images created.

There are different ways to remove containers either interactive or non-interactive

but first list all the containers created

```
root@menyi-Systems: /home/menyi                    root@menyi-Systems: ~
root@menyi-Systems:/home/menyi# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED       STATUS                     PORTS     NAMES
ace9c9c111f1   ubuntu    "bash"                   2 hours ago   Exited (0) 15 seconds ago            menyiContainerIT
a2f40d0683aa   ubuntu    "bash /var/lib/docke…"   3 hours ago   Exited (0) 3 hours ago               menyiContainer
root@menyi-Systems:/home/menyi#
```

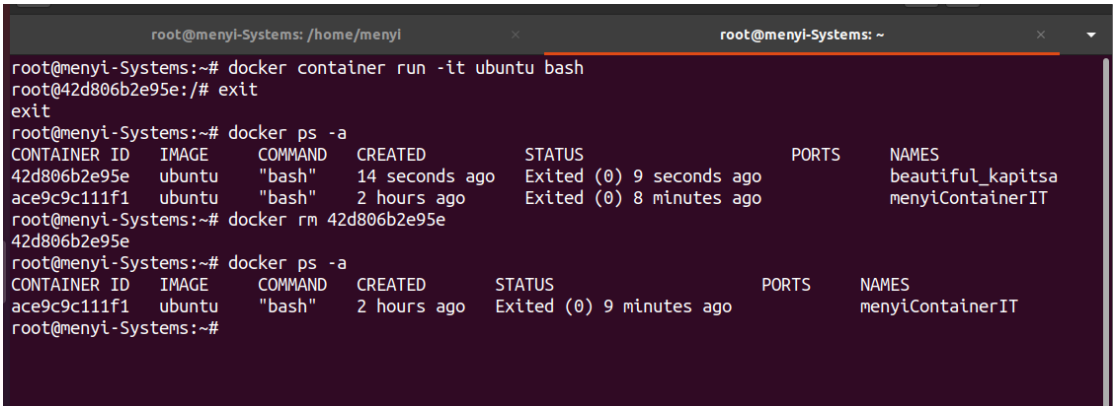
To remove a container directly after finished we need the *--rm* parameter when creating the container

```
root@menyi-Systems: /home/menyi                    root@menyi-Systems: ~
root@menyi-Systems:~# docker container run --rm -it ubuntu bash
root@bd8707b19eb8:/# exit
exit
root@menyi-Systems:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED       STATUS                     PORTS     NAMES
ace9c9c111f1   ubuntu    "bash"                   2 hours ago   Exited (0) 6 minutes ago             menyiContainerIT
a2f40d0683aa   ubuntu    "bash /var/lib/docke…"   3 hours ago   Exited (0) 3 hours ago               menyiContainer
root@menyi-Systems:~#
```

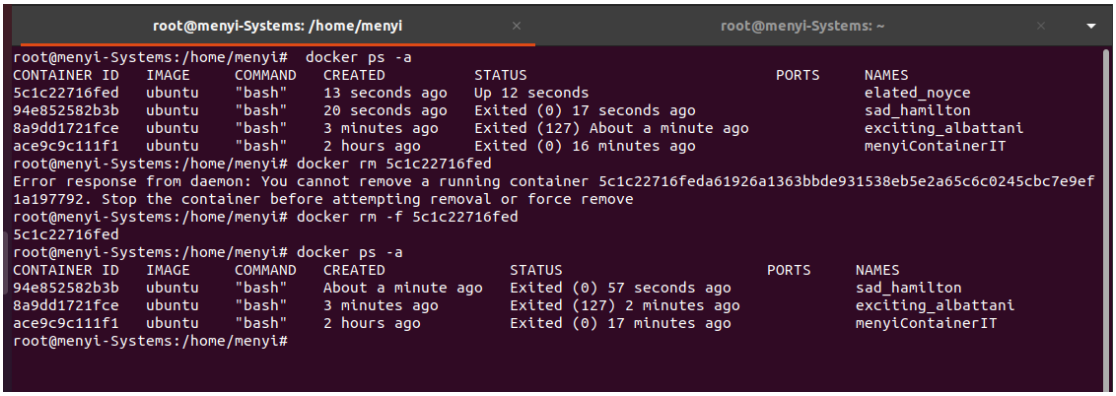
Remove one container by its name

```
root@menyi-Systems: /home/menyi                    root@menyi-Systems: ~
root@menyi-Systems:~# docker rm menyiContainer
menyiContainer
root@menyi-Systems:~# docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED       STATUS                     PORTS     NAMES
ace9c9c111f1   ubuntu    "bash"    2 hours ago   Exited (0) 7 minutes ago             menyiContainerIT
root@menyi-Systems:~#
```

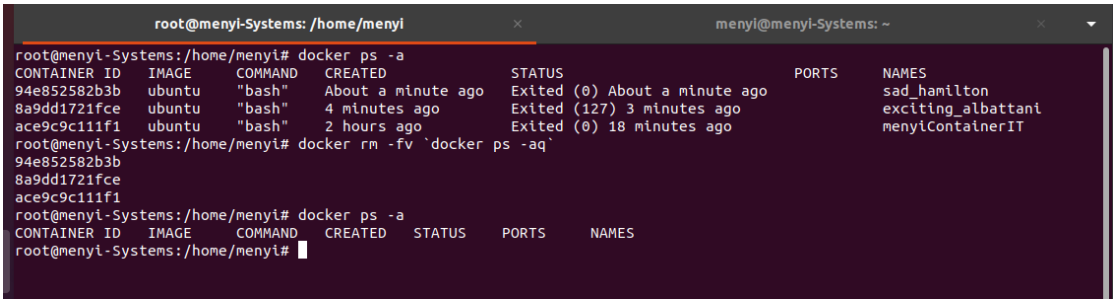
Remove one container by its id



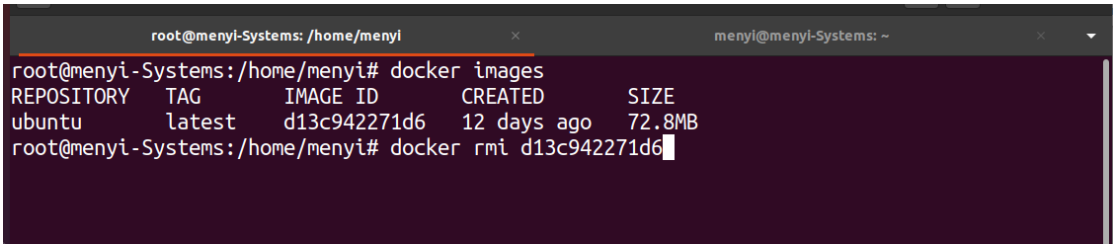
Remove and stop it if it is running using *–f* parameter



Remove all containers at the same time with just one command



There are also different ways to **remove an image** from Docker: remove one image by its id



Remove all images at the same time with one command, then check it with *docker images*