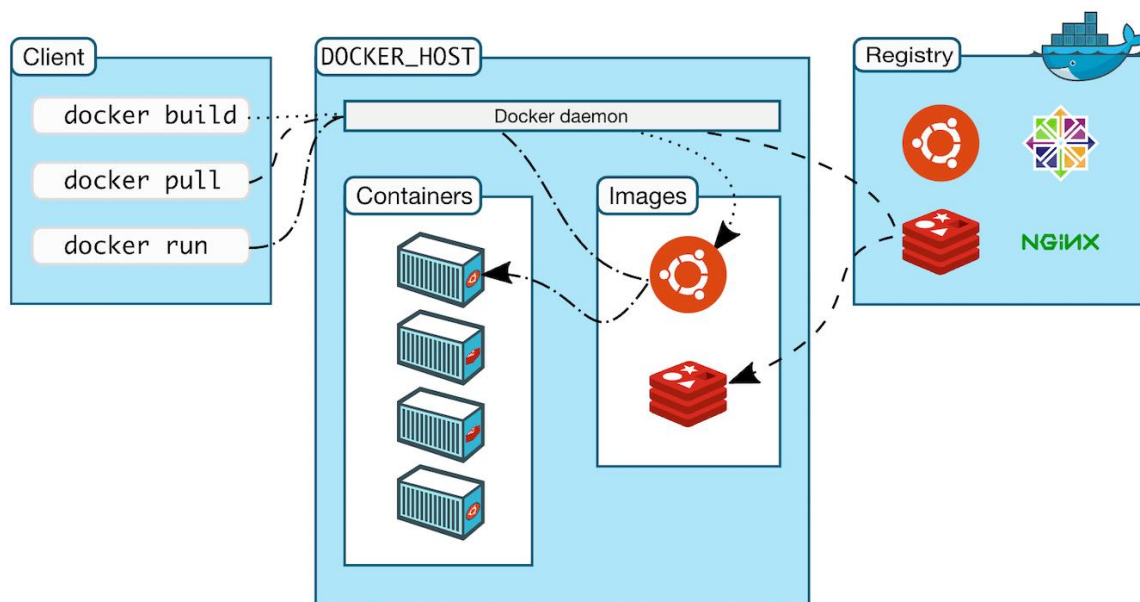# Activity No. 5: Docker practical session [10%]

Docker is a command line-based software allowing users to manipulate images and create application containers.

As presented in the course, Docker consists of two elements:
- a client, to receive commands from the user
- a server, to execute commands and manage images and containers

Docker commands architecture

Typing this command will give the Client and Server versions available on your computer.
Paste a screenshot of result

docker *version*	

Usage, options and a full list of available commands can be accessed through the command line in a terminal.Type the following command

docker *--help*

The general usage of a Docker command line is as follows:

```
docker [OPTIONS] COMMAND [arg...]
```

Questions
1. How many arguments are absolutely required by the command 'docker pull' ? R/N pero admite la imagen.
2. Do you remember what a registry is? R/ Es un sistema de almacenamiento y distribución de imágenes.

Download a predefined image available on the DockerHub

In a web browser, navigate to the DockerHub : https://hub.docker.com/
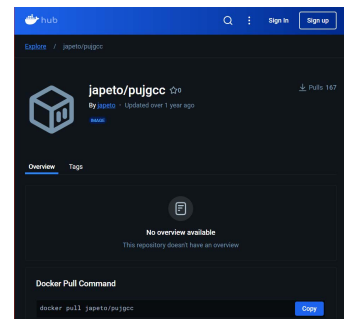In the top search bar, type : japeto/pujgcc and paste a screenshot

```

```

Our course has images available for the development of practical sessions.

Questions
1. How many times was the *japeto* image downloaded ?R/ 167

Execute the command inside a terminal.
Paste a screenshot of result

```
docker pull japeto/pujgcc:v0.12
```

You will get an error as this image has no default tag ("latest"). So we need to specify one in the command line.

Go to the "Tags" tab and copy the pull command of version latest
Paste a screenshot of result


```
docker pull japeto/pujgcc:v0.12
```

Question:
1. How many times do you see 'Pull complete' displayed ? Why ?R/ 11 VECES

Now, to be sure that the image was correctly pulled, let's see the list of all available downloaded images inside our workspace. *Paste a screenshot of result*


```
docker image
```

```
PS C:\Users\usuario> docker image ls
REPOSITORY        TAG        IMAGE ID       CREATED         SIZE
japeto/pujgcc     latest     05141310a94c   19 months ago   1.39GB
```

Question:
1. What is the size of the japeto/pujgcc image ? R/ 1.39GB

Perform a task using a pulled image

Among the Docker commands, we will now use the 'run' command.

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

Question:
1. What are the options and parameters of the 'run' command
[OPTIONS][COMMAND][ARG...]

---
docker run --help

As displayed in the terminal, the description of the command is 'Run a command in a
new container'. Question :

1. What is the difference between an image and a container ?
R/ Imagen: Es un archivo de solo lectura que actúa como una plantilla para crear contenedores.
Contenedor: Es una instancia de la imagen en ejecución, que es mutable y puede realizar tareas
aisladas dentro de su entorno.

Now, to run the application, execute the following command:
Paste a screenshot of result

```
PS C:\Users\usuario> docker run japeto/pujgcc bash --help
GNU bash, version 4.3.30(1)-release-(x86_64-pc-linux-gnu)
Usage:  bash [GNU long option] [option] ...
        bash [GNU long option] [option] script-file ...
GNU long options:
        --debug
        --debugger
        --dump-po-strings
        --dump-strings
        --help
        --init-file
        --login
        --noediting
        --noprofile
        --norc
        --posix
        --rcfile
        --restricted
        --verbose
        --version
Shell options:
        -ilrsD or -c command or -O shopt_option      (invocation only)
        -abefhkmnptuvxBCHP or -o option
Type `bash -c "help set"' for more information about shell options.
Type `bash -c help' for more information about shell builtin commands.
Use the `bashbug' command to report bugs.
```

---
docker run japeto/pujgcc bash --help

Congratulations!
You just successfully downloaded and used your first Docker image!

Running *PUJGCC* without parameters was interesting as a demonstration of Docker's features.
But if we want to really run *PUJGCC*, we also need to provide parameters and, most
importantly, input files.

Find the paths to bind
To bind our current folder to the /data/ folder located inside a container, we first need the
absolute path of the current folder, obtained through the unix pwd command.
Paste a screenshot of result

```
PS C:\Users\usuario> pwd

Path
----
C:\Users\usuario
```

---
pwd

This path will be used in further commands through ${PWD}.

Instead of running ls command to /home/ files, we will now just list the content of the /data/ folder inside the container but bind with the host.
Paste a screenshot of result

docker run japeto/pujgcc:latest ls /data


```
PS C:\Users\usuario> pwd

Path
----
C:\Users\usuario

PS C:\Users\usuario> docker run japeto/pujgcc:latest ls /data
ls: cannot access /data: No such file or directory
PS C:\Users\usuario>
```

If nothing appears, it is normal: the folder is empty and only serves as a "*branching point*".

We now have the paths of the two folders we want to bind together.

Bind a local folder into a container

To perform the folder mapping between the current folder and /data inside the image, the syntax is simple. *Paste a screenshot of result*

docker run -v ${PWD}:/data/ japeto/pujgcc:latest ls /data/

Question:
1. Is the displayed list the same as what is in your current folder?
R/ NO

Finally, we can run C on a C or C++ file located in the Data folder. Change the name of the file to any of the provided files.
Paste a screenshot of result

docker run -v ${PWD}:/data/ japeto/pujgcc:latest gcc /data/helloworld.c


```
PS C:\Users\usuario> docker run -it --name temp-container japeto/pujgc
c:latest /bin/bash
root@9c6fc29c437b:/# cd /data
bash: cd: /data: No such file or directory
root@9c6fc29c437b:/# nano helloworld.c
root@9c6fc29c437b:/# docker run -v ${PWD}:/data japeto/pujgcc:latest g
cc /data/helloworld.c -o /data/helloworld
bash: docker: command not found
root@9c6fc29c437b:/# docker run -v ${PWD}:/data japeto/pujgcc:latest g
cc /data/helloworld.c
bash: docker: command not found
root@9c6fc29c437b:/# nano helloworld.c
root@9c6fc29c437b:/# gcc helloworld.c -o helloworld
root@9c6fc29c437b:/# ./helloworld
Hello, world!
root@9c6fc29c437b:/#
```

Restart and detach a container

Learn how to re-use a container where you installed something

Use the start command to restart the container created in the last exercise
Paste a screenshot of result

docker start -ti mycontainer /bin/bash

Go back to the container using the exec command instead of the run command.
Paste a screenshot of result

docker exec -ti mycontainer /bin/bash

Question :

1. *What happens now when you exit the container? Is it stopped?*
   R/ No se detiene por default

Check with and **Paste a screenshot of result**

```
PS C:\Users\usuario> docker start mycontainer
mycontainer
PS C:\Users\usuario> docker exec -it mycontainer /bin/bash
root@ed9c9b08f80b:/# exit
exit

What's next:
    Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug mycontainer
    Learn more at https://docs.docker.com/go/debug-cli/
PS C:\Users\usuario> docker ps -l
CONTAINER ID   IMAGE               COMMAND              CREATED        STATUS              PORTS      NAMES
4812c3aac174   japeto/pujgcc:latest "/data/helloworld"   4 minutes ago  Exited (0) 4 minutes ago       optimist
ic_taussig
PS C:\Users\usuario>
```

docker ps -l

In fact, the container keeps on running. This is because re-starting a container turns it into a "detached process" running in the background. Alternatively, we could have added the -d option to the first docker run command, creating directly a detached container.

Finally, you can stop the container.

```
PS C:\Users\usuario> docker stop mycontainer
mycontainer
PS C:\Users\usuario>
```

docker stop mycontainer

This is the end of the practical session. We hope you enjoyed it. Don't hesitate to ask any questions and feel free to contact us any time after the session!

List of commands

Search the available versions of an image in the Docker registry:

docker search

Pulling an image:

docker pull

Starting a container on a given image running a single command:

docker run -ti

Starting a container on a given image running a single command (detached):

docker run -d

List all containers and their status

```
docker ps -l
```

List all pulled images

```
docker images
```

Removing one local container

```
docker rm
```

Removing one local image

```
docker rmi
```

Clean all containers

```
docker rm $(docker ps -aq)
```

Clean all images (after cleaning the containers)

```
docker rmi $(docker images -aq)
```

Observations

- Deliveries must be made in teams of 4. Using a public repository on github and a pdf report
- If you do not understand the instructions for any of the activities, do not hesitate to write to jeffersonamado.pena@javerianacali.edu.co.