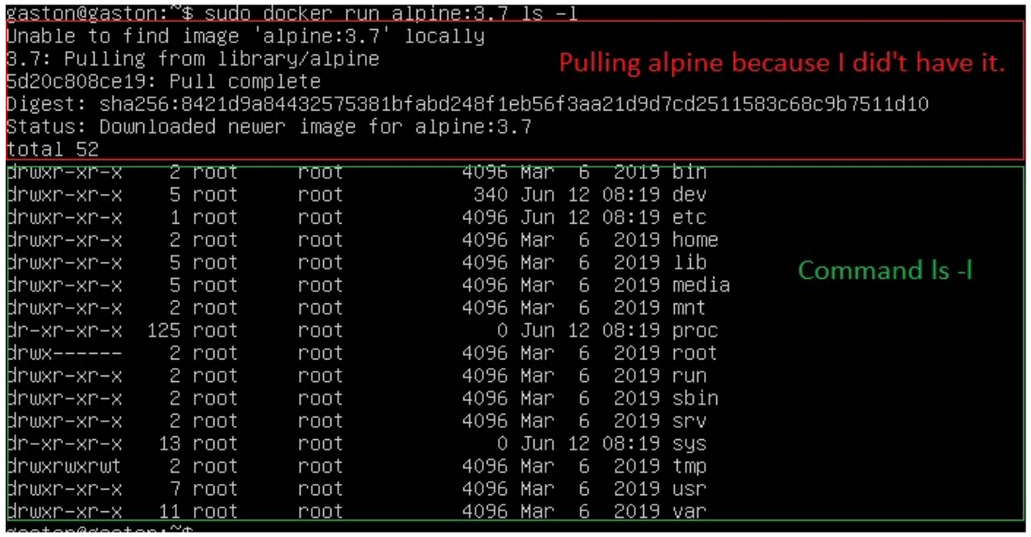
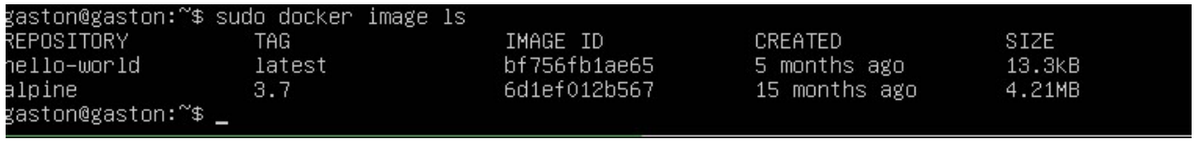
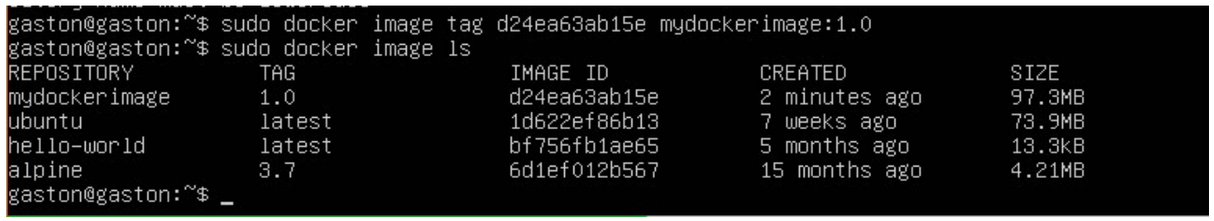
Command	Description	Key word
docker version	To show docker version	Version
docker ps	To show the conteiners that are running, you can run also docker ps -a | head head = the last ten. a = Also the containers that they are not running.	Show
docker run nameOfTheImage	Run the container, example: docker run alpine:3.7 ls -l and I can see all the files  	Run
docker run -d nameOfTheImage	With -d I run the container and it continues running , we use -d when we run for example a nginx service.	Run
docker start idOfCointaner.	To start a container.	Start
docker stop idOfCointaner.	To stop a container.	Stop
docker rm idOfCointaner.	To remove a container.	Remove
docker pull nameOfTheContainerImage	To download a docker image	Pull / Download
docker run -it nameOfTheImage sh docker run -it nameOfTheImage /bin/bash	To run a shell in the container. i = interactive  	Run
docker exec -it IdOfTheContainer sh docker exec -it IdOfTheContainer bin/bash	To execute a container that it is running.	Execute

docker commit IdOfTheContainer	To create a new image with our changes. Example: I ran an alpine container, I installed new packages and then I close the container session, I can see the container that were running (docker ps -a) and create an image with the command commit. Other example, You can install in ubuntu image figlet and then create yout own image and run it :  In this example the image is taged with a name and version.	commit
docker image ls docker images	To show our images:  gaston@gaston:~$ sudo docker image ls REPOSITORY TAG IMAGE ID CREATED SIZE hello-world latest bf756fb1ae65 5 months ago 13.3kB alpine 3.7 6d1ef012b567 15 months ago 4.21MB gaston@gaston:~$ _	Show
docker rmi idOfTheImage	To remove an image.	Remove
docker image tag IdOfTheImage	To tag the image, you can run docker image tag IdOfTheContainerImage myDockerImage:1.0 or docker image tag IdOfTheContainerImage myDockerImage , if you don't especify the version the default is latest.  gaston@gaston:~$ sudo docker image tag d24ea63ab15e mydockerimage:1.0 gaston@gaston:~$ sudo docker image ls REPOSITORY TAG IMAGE ID CREATED SIZE mydockerimage 1.0 d24ea63ab15e 2 minutes ago 97.3MB ubuntu latest 1d622ef86b13 7 weeks ago 73.9MB hello-world latest bf756fb1ae65 5 months ago 13.3kB alpine 3.7 6d1ef012b567 15 months ago 4.21MB gaston@gaston:~$ _	Tag

<table>
<tr>
<td>Dockerfile</td>
<td>Dockerfile is a set of instuctions to create images.
I create a file with instructions that update ubuntu and the install figlet:


```
gaston@gaston:~/docker$ cat Dockerfile
FROM ubuntu

RUN apt-get update && apt-get install figlet -y
```


Now I build a new image, check if was created and run to check if it is woking:


```
gaston@gaston:~/docker$ sudo docker build -t mydockerimage:1.2 .
Sending build context to Docker daemon  2.048kB
Step 1/2 : FROM ubuntu
 ---> 1d622ef86b13
Step 2/2 : RUN apt-get update && apt-get install figlet -y
 ---> Using cache
 ---> a4aeecb77ad9
Successfully built a4aeecb77ad9
Successfully tagged mydockerimage:1.2
gaston@gaston:~/docker$ sudo docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mydockerimage       1.1                 a4aeecb77ad9        7 minutes ago       97.3MB
mydockerimage       1.2                 a4aeecb77ad9        7 minutes ago       97.3MB
mydockerimage       1.0                 d24ea63ab15e        23 minutes ago      97.3MB
ubuntu              latest              1d622ef86b13        7 weeks ago         73.9MB
hello-world         latest              bf756fb1ae65        5 months ago        13.3kB
alpine              3.7                 6d1ef012b567        15 months ago       4.21MB
gaston@gaston:~/docker$ sudo docker run mydockerimage:1.2 figlet "Hi All! - Gaston"
 _   _ _     _   _ _ _          ____          _
| | | (_)   / \  | | | |        / ___| __ _ ___| |_ ___  _ __
| |_| | |  / _ \ | | | |  _____ | |  _ / _` / __| __/ _ \| '_ \
|  _  | | / ___ \| | |_| |_____| | |_| | (_| \__ \ || (_) | | | |
|_| |_|_|/_/   \_\_|_(_)         \____|\__,_|___/\__\___/|_| |_|

gaston@gaston:~/docker$
```

</td>
<td>Dockerfile / Build</td>
</tr>
<tr>
<td>docker image history IdOfTheImage</td>
<td>To check the history of the image:


```
gaston@gaston:~/docker$ sudo docker image history a4aeecb77ad9
IMAGE               CREATED             CREATED BY                                      SIZE
        COMMENT
a4aeecb77ad9        12 minutes ago      /bin/sh -c apt-get update && apt-get install…   23.4MB

1d622ef86b13        7 weeks ago         /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B

<missing>           7 weeks ago         /bin/sh -c mkdir -p /run/systemd && echo 'do…   7B

<missing>           7 weeks ago         /bin/sh -c set -xe   && echo '#!/bin/sh' > /…   811B

<missing>           7 weeks ago         /bin/sh -c [ -z "$(apt-get indextargets)" ]     1.01MB

<missing>           7 weeks ago         /bin/sh -c #(nop) ADD file:a58c8b447951f9e30…   72.8MB
```

</td>
<td>History</td>
</tr>
<tr>
<td>docker logs IdOfTheContainer</td>
<td>To check the logs of the container.</td>
<td>Logs</td>
</tr>
<tr>
<td>docker stats IdOfTheContainer</td>
<td>To check the stats of the container.</td>
<td>Stats</td>
</tr>
</table>

Volume	When you run a container, you can create a volume to save files in your host server, for example: docker run -v home/gaston/docker/index.html:/usr/share/nginx/html/index.html:ro -d nginx:1.19.0 docker run -v path_inside_the_host:path_inside_the_container:xx -d web_server_image:version -v = volume -d = run and continue running. ro = read only	Volume
Ports	To open a port you can add in the docker run a new parameter: docker run -v home/gaston/docker/index.html:/usr/share/nginx/html/index.html:ro -p 8080:80 -d nginx:1.19.0 -p host_port:container_port A now you can check from your host server your site in the container. gaston@gaston:~/docker$ sudo docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 1aebc5f890e7 nginx:1.19.0 "/docker-entrypoint.…" 2 minutes ago Up 2 minutes 0.0.0.0:8080->80/tcp practical_darwin gaston@gaston:~/docker$ _	Ports

Docker compose	You can create a yaml script to work in a more organized way and then run the yaml to create the container:
	
	``
	gaston@gaston:~/docker$ cat docker-compose.yaml
	version: '3.1'
	
	services:
	  wordpress:
	  image: wordpress:5.4.2-php7.2-apache
	  ports:
	  - 8080:80
	  environment:
	  WORDPRESS_DB_HOST: mysql
	  WORDPRESS_DB_USER: root
	  WORDPRESS_DB_PASSWORD: root
	  WORDPRESS_DB_NAME: wordpress
	  links:
	  - mysql:mysql #create a line in the host wordpress container to reference th
	e ip of the mysql container, to avoid issues if the ip changes
	
	  mysql:
	  image: mysql:8.0.20
	  command: --default-authentication-plugin=mysql_native_password
	  environment:
	  MYSQL_DATABASE: wordpress
	  MYSQL_ROOT_PASSWORD: root
	  volumes:
	  - /home/gaston/mysql-data:/var/lib/mysql
	gaston@gaston:~/docker$
	``
	
	And then run your yaml script with the command docker-compose up -d
	
	``
	gaston@gaston:~/docker$ sudo docker-compose up -d
	Starting docker_mysql_1 ... done
	Starting docker_wordpress_1 ... done
	gaston@gaston:~/docker$ sudo docker ps
	CONTAINER ID IMAGE COMMAND CREATED STA
	TUS PORTS NAMES
	d66e08072442 wordpress:5.4.2-php7.2-apache "docker-entrypoint.s…" 6 minutes ago Up
	13 seconds 0.0.0.0:8080->80/tcp docker_wordpress_1
	64c00228d911 mysql:8.0.20 "docker-entrypoint.s…" 6 minutes ago Up
	18 seconds 3306/tcp, 33060/tcp docker_mysql_1
	gaston@gaston:~/docker$
	``
	
	As you can see , you have 2 containers, one with mysql and the other with wordpress.
		Docker compose