#007 Docker Images and Containers

Introduction

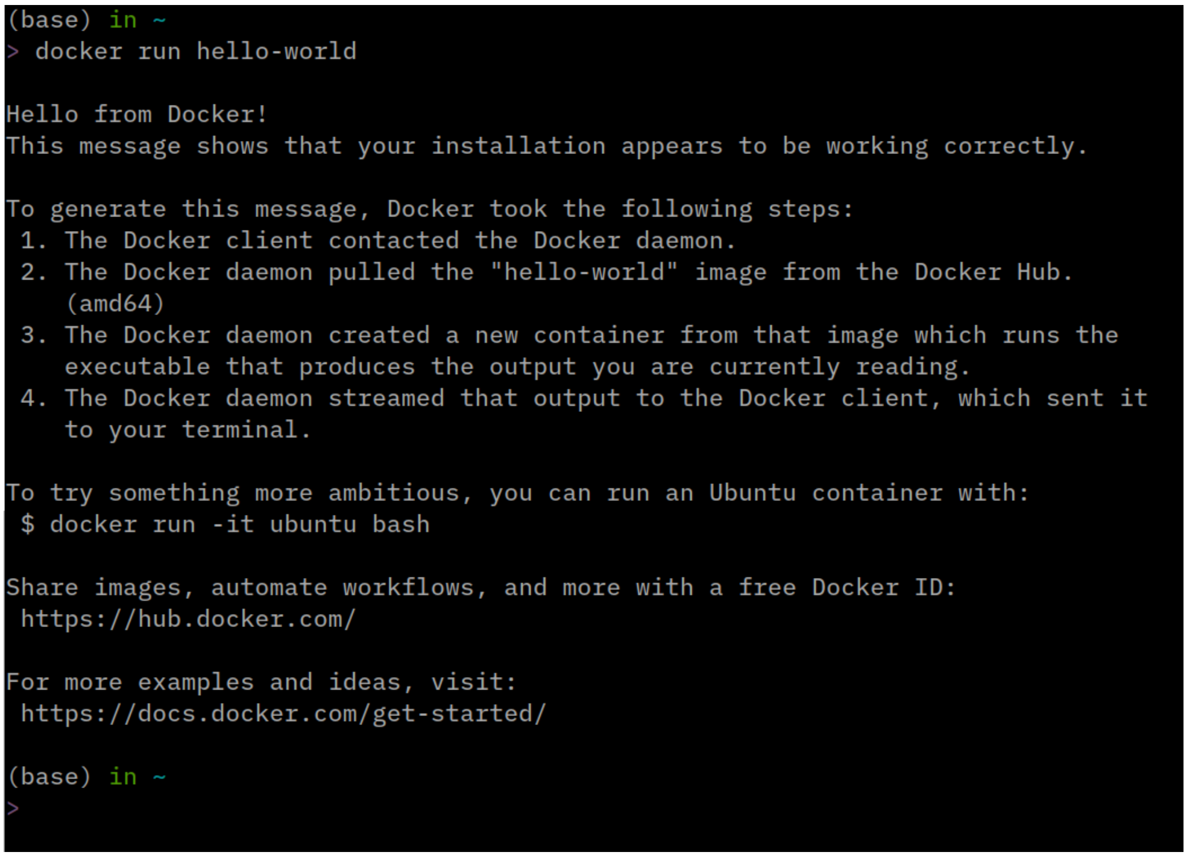
this is part 7 from the journey it's a long journey(360 day) so go   please check previous parts , and if you need to walk in the journey   with me please make sure to follow because I may post more than once   in 1 Day but surely I will post daily at least one 😍.

And I will cover lot of tools as we move on.

Back to Hello World demo

in part 6 we run

```
docker run Hello-World
```



```
(base) in ~
> docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

(base) in ~
>
```

As we see in 1 the docker client contacted the docker daemon as we discussed earlier in step 2 the docker doesn't find the image hello-world locally so he go to the docker hub and downloaded for us!

see how easy it is with one command we got an image!

in step 3 the Docker create a new container using the image we downloaded and run it so we got this output

in step 4 the deamon (server) send back the message to us ! (client)

Docker Image and container

If you are a programmer and you study OOP(Object Oriented Programming) we can think about Image as an Class ,

and the container is an instance of it kind of

```
Car tesla = new Car();
Car toyota = new Car();
```

We can imagine the Image as the Car and the Container as tesla and toyota , see I build on top of 1 Image 2 Containers and there is no need to pull image again from dockerhub so it will skip step 2!

---

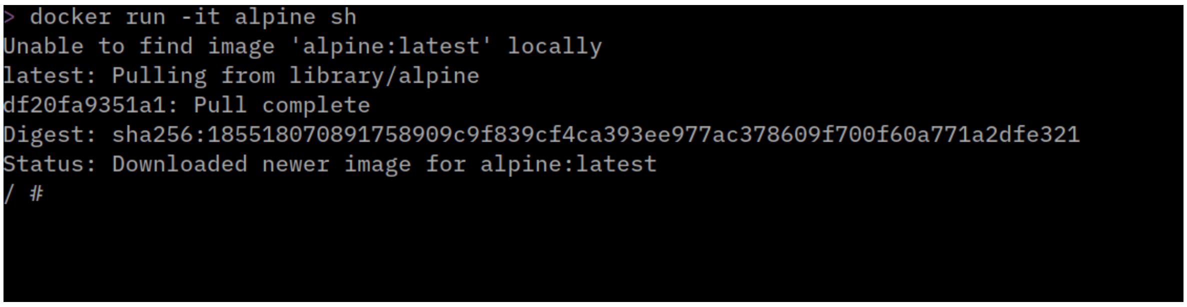
let's run an light linux system (alpine)

```
docker run -it alpine sh
```

-it stands for **i** is interactive and **t** is an pseudo-TTY (pseudo terminal to communicate with bash)

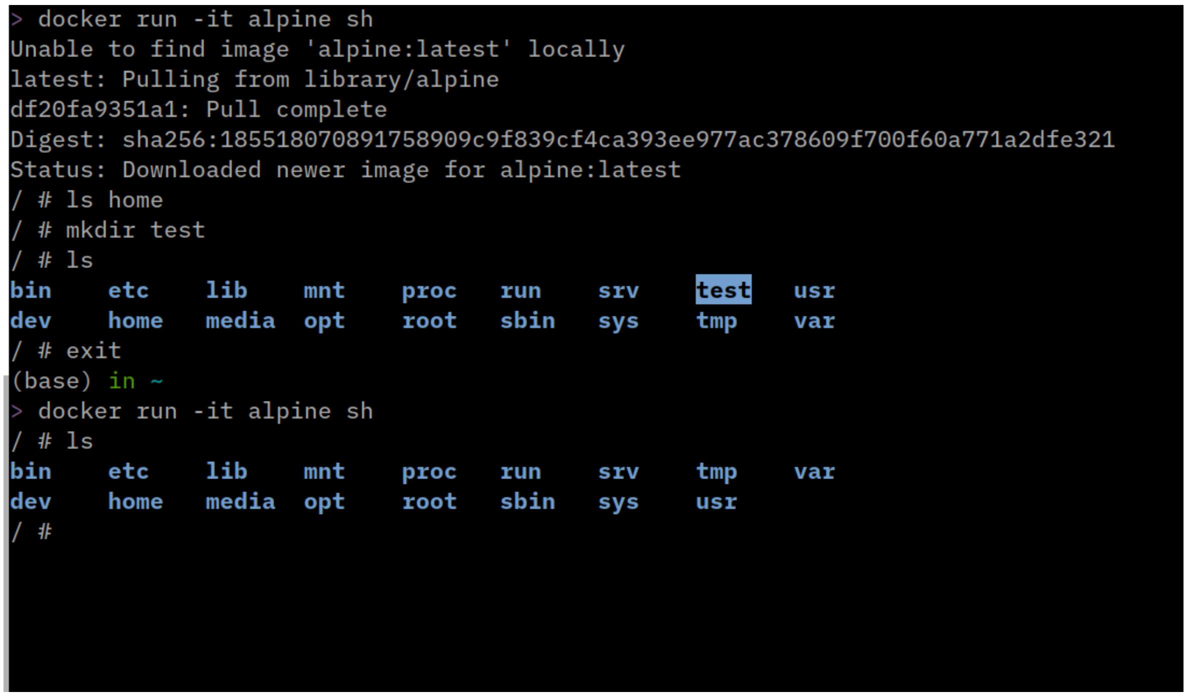
alpine is a light linux distro which contain basic files and kernel to run linux

sh is kind of a bash



```
> docker run -it alpine sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
df20fa9351a1: Pull complete
Digest: sha256:185518070891758909c9f839cf4ca393ee977ac378609f700f60a771a2dfe321
Status: Downloaded newer image for alpine:latest
/ #
```

as we see , docker doesn't find the alpine locally , so he pull it from docker hub



```
> docker run -it alpine sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
df20fa9351a1: Pull complete
Digest: sha256:185518070891758909c9f839cf4ca393ee977ac378609f700f60a771a2dfe321
Status: Downloaded newer image for alpine:latest
/ # ls home
/ # mkdir test
/ # ls
bin    etc    lib    mnt    proc   run    srv    test   usr
dev    home   media  opt    root   sbin   sys    tmp    var
/ # exit
(base) in ~
> docker run -it alpine sh
/ # ls
bin    etc    lib    mnt    proc   run    srv    tmp    var
dev    home   media  opt    root   sbin   sys    usr
/ #
```

as we see home directory is empty (home directory is linux folder like D: in windows you can locate personal files not system files)

mkdir (make directory is a linux also windows command to make a directory) and test is folder name

as we see when I restart the docker , I lost the changes I made !

this will be covered later.