

UNIVERSITE CHEIKH ANTA DIOP DE DAKAR



DEPARTEMENT GENIE INFORMATIQUE

ANNEE 2024/2025

**LICENCE 3 EN SYSTEMES RESEAUX ET TELECOMMUNICATION
(LIC3SRT)**

TECHNOLOGIE DE VIRTUALISATION ET CLOUD

ATELIER 1:
Plateforme de Conteneurisation Docker

REALISE PAR :

Mouhamadou Moustapha LO
Papa Abdoulaye Diop

Dr. Mandicou BA

L'objectif de cet atelier est de vous permettre de se familiariser avec la plateforme de conteneurisation Docker.

À la fin de cet atelier chaque étudiant doit être en mesure de pouvoir :

- ✓ Installer et de configurer correctement Docker
- ✓ Gérer efficacement des conteneurs en mode cli et graphique (**Docker cli, Portainer**).
- ✓ Télécharger ou téléverser des images Docker dans le Hub de Docker (**Docker Hub**)
- ✓ Gérer l'interconnexion entre conteneurs et leurs déploiements (**Docker Compose**)
- ✓ Orchestrer et gérer des clusters avec (**Docker Swarm**)

Pré-requis :

1. Une machine virtuelle ou physique tournant sous l'os Debian 12 (Bookworm).
2. Une connexion internet sur la machine
3. Un compte sur Docker Hub

Ressources :

1. <https://www.mandicouba.net/tools/virtualbox>
2. <https://www.mandicouba.net/os/debian>
3. <https://hub.docker.com/>

Tâche 1 : Mise en place d'une plateforme Docker sous Debian 12

1. Mise à jour de la liste de paquets existante

Avant de procéder à l'installation de Docker, commencer par mettre à jour la liste des paquets de Debian.

```
moustaphalo@moustapha:~$ sudo apt update
[sudo] Mot de passe pour moustaphalo :
Atteint :1 http://deb.debian.org/debian bookworm InRelease
Atteint :2 http://security.debian.org/debian-security bookworm-security InRelease
Atteint :3 http://deb.debian.org/debian bookworm-updates InRelease
Atteint :4 https://download.docker.com/linux/debian bookworm InRelease
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
11 paquets peuvent être mis à jour. Exécutez « apt list --upgradable » pour les voir.
```

2. Permettre à la commande apt d'utiliser les paquets via HTTPS

Installer **Docker** depuis le référentiel officiel afin d'être sûr de disposer la dernière version car les dépôts de **Debian** peuvent ne pas posséder la dernière version. Commencer par installer les prérequis qui vont vous permettre d'utiliser la commande **apt** pour installer des paquets à l'aide du protocole **https**.

```
moustaphalo@moustapha:~$ sudo apt install apt-transport-https ca-certificates curl gnupg2 software-properties-common -y
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
apt-transport-https est déjà la version la plus récente (2.6.1).
ca-certificates est déjà la version la plus récente (20230311).
curl est déjà la version la plus récente (7.88.1-10+deb12u8).
gnupg2 est déjà la version la plus récente (2.2.40-1.1).
software-properties-common est déjà la version la plus récente (0.99.30-4.1~deb12u1).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```

3. Ajout de la clé GPG du référentiel officiel de Docker

La commande **apt** peut utiliser **https** comme mode de transport des paquets. Ajouter la clé **GPG** du répertoire officiel de Docker afin de permettre le chiffrement et la signature des données.

```
moustaphalo@moustapha:~$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
Le fichier « /usr/share/keyrings/docker-archive-keyring.gpg » existe. Faut-il réécrire par-dessus ? (o/N) o
```

Ajouter le dépôt officiel de **Docker** dans la liste des **dépôts** du système.

```
moustaphalo@moustapha:~$ echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

4. Mise à jour de la liste de paquets

Mettre à jour la liste des **paquets**

```
moustaphalo@moustapha:~$ sudo apt update
Atteint :1 http://deb.debian.org/debian bookworm InRelease
Atteint :2 http://security.debian.org/debian-security bookworm-security InRelease
Atteint :3 http://deb.debian.org/debian bookworm-updates InRelease
Atteint :4 https://download.docker.com/linux/debian bookworm InRelease
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Tous les paquets sont à jour.
```

5. Installation de Docker

Tout est prêt, passer maintenant à l'installation de **Docker** en utilisant la commande **apt**

Vu que j'ai déjà installé docker-ce je vais utiliser la commande **apt policy dokcer-ce** pour vérifier si docker est bien installé.

```
moustaphalo@moustapha:~$ sudo apt policy docker-ce
docker-ce:
  Installé : 5:27.4.0-1~debian.12~bookworm
  Candidat : 5:27.4.0-1~debian.12~bookworm
  Table de version :
*** 5:27.4.0-1~debian.12~bookworm 500
      500 https://download.docker.com/linux/debian bookworm/stable amd64 Packages
      100 /var/lib/dpkg/status
  5:27.3.1-1~debian.12~bookworm 500
      500 https://download.docker.com/linux/debian bookworm/stable amd64 Packages
```


6. Vérification de l'état de Docker

Vérifier si tout fonctionne correctement par exemple vérifier que le démon Docker a bien démarré de même si le processus de démarrage automatique est activé afin de permettre à Docker de se lancer directement après le démarrage du Système.

Docker est bien démarré.

```
moustaphalo@moustapha:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
  Loaded: loaded (/lib/systemd/system/docker.service; enabled; preset: enabled)
  Active: active (running) since Mon 2024-12-16 15:08:38 CET; 16min ago
    TriggeredBy: ● docker.socket
    Docs: https://docs.docker.com
   Main PID: 43655 (dockerd)
     Tasks: 17
    Memory: 32.8M
      CPU: 2.659s
     CGroup: /system.slice/docker.service
             └─43655 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

déc. 16 15:08:33 moustapha dockerd[43655]: time="2024-12-16T15:08:33.748136214+01:00" level=info msg="[graphdriver] using prior stor
déc. 16 15:08:34 moustapha dockerd[43655]: time="2024-12-16T15:08:34.711922421+01:00" level=info msg="Loading containers: start."
déc. 16 15:08:36 moustapha dockerd[43655]: time="2024-12-16T15:08:36.765074860+01:00" level=info msg="Default bridge (docker0) is as
déc. 16 15:08:37 moustapha dockerd[43655]: time="2024-12-16T15:08:37.108299785+01:00" level=info msg="Loading containers: done."
déc. 16 15:08:37 moustapha dockerd[43655]: time="2024-12-16T15:08:37.277474687+01:00" level=warning msg="WARNING: bridge-nf-call-ipt
déc. 16 15:08:37 moustapha dockerd[43655]: time="2024-12-16T15:08:37.277558224+01:00" level=warning msg="WARNING: bridge-nf-call-ip6
déc. 16 15:08:37 moustapha dockerd[43655]: time="2024-12-16T15:08:37.277609604+01:00" level=info msg="Docker daemon" commit=92a8393
déc. 16 15:08:37 moustapha dockerd[43655]: time="2024-12-16T15:08:37.279669696+01:00" level=info msg="Daemon has completed initializ
déc. 16 15:08:38 moustapha dockerd[43655]: time="2024-12-16T15:08:38.377081421+01:00" level=info msg="API listen on /run/docker.sock
déc. 16 15:08:38 moustapha systemd[1]: Started docker.service - Docker Application Container Engine.
lines 1-22/22 (END)
```

7. Utiliser Docker sans Sudo

Par défaut, une commande **Docker** est toujours précédée du mot clé **sudo** autrement dit une commande **Docker** ne peut être exécutée que par le super admin **root** ou par un utilisateur du groupe docker. Lever cette contrainte afin d'utiliser **Docker** sans disposer des priviléges du super admin. Par défaut une fois que Docker est installé le groupe docker est automatiquement créé dans le doute créer le groupe.

```
moustaphalo@moustapha:~$ sudo groupadd docker
groupadd : le groupe 'docker' existe déjà
```

De même le socket utilisé par **Docker** doit appartenir au groupe docker. Définir l'appartenance.

```
mandicou@Mandicou :~$ ****
```

Ajouter l'utilisateur courant au groupe **docker** pour lui permettre d'utiliser les commandes docker sans au préalable taper **sudo**

```
moustaphalo@moustapha:~$ sudo usermod -aG docker $USER
```

Appliquer les changements et redémarrer **Docker**

```
moustaphalo@moustapha:~$ sudo systemctl restart docker
```

```
mandicou@Mandicou :~$ sudo systemctl restart docker
```

8. L'aide sous Docker

Afficher la liste des commandes Docker

```
moustaphalo@moustapha:~$ docker --help

Usage: docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
  run      Create and run a new container from an image
  exec     Execute a command in a running container
  ps       List containers
  build    Build an image from a Dockerfile
  pull     Download an image from a registry
  push     Upload an image to a registry
  images   List images
  login    Authenticate to a registry
  logout   Log out from a registry
  search   Search Docker Hub for images
  version  Show the Docker version information
  info     Display system-wide information

Management Commands:
  builder   Manage builds
  buildx*   Docker Buildx
  checkpoint Manage checkpoints

Commands:
  attach    Attach local standard input, output, and error streams to a running container
  commit   Create a new image from a container's changes
  cp       Copy files/folders between a container and the local filesystem
  create   Create a new container
  diff     Inspect changes to files or directories on a container's filesystem
  events   Get real time events from the server
  export   Export a container's filesystem as a tar archive
  history  Show the history of an image
  import   Import the contents from a tarball to create a filesystem image
  inspect  Return low-level information on Docker objects
  kill     Kill one or more running containers
  load    Load an image from a tar archive or STDIN
  logs    Fetch the logs of a container
  pause   Pause all processes within one or more containers
  port    List port mappings or a specific mapping for the container
  rename  Rename a container
  restart Restart one or more containers
  rm     Remove one or more containers
  rmi    Remove one or more images
  save    Save one or more images to a tar archive (streamed to STDOUT by default)
  start   Start one or more stopped containers
  stats   Display a live stream of container(s) resource usage statistics
  stop    Stop one or more running containers
  tag     Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top     Display the running processes of a container
  unpause Unpause all processes within one or more containers
  update  Update configuration of one or more containers
  wait   Block until one or more containers stop, then print their exit codes

Global Options:
  --config string        Location of client config files (default "/home/moustaphalo/.docker")
  -c, --context string   Name of the context to use to connect to the daemon (overrides DOCKER_HOST env var and default context set with "docker context use")
  -D, --debug            Enable debug mode
  -H, --host list         Daemon socket to connect to
  -l, --log-level string Set the logging level ("debug", "info", "warn", "error", "fatal") (default "info")
  --tls                 Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default "/home/moustaphalo/.docker/ca.pem")
  --tlscert string       Path to TLS certificate file (default "/home/moustaphalo/.docker/cert.pem")
  --tlskey string        Path to TLS key file (default "/home/moustaphalo/.docker/key.pem")
  --tlsverify           Use TLS and verify the remote
  -v, --version          Print version information and quit

Run 'docker COMMAND --help' for more information on a command.

For more help on how to use Docker, head to https://docs.docker.com/qa/quides/
```

Pour obtenir de l'aide sur une commande docker il faut utiliser la commande docker help suivis de la commande dont on veut afficher le manuel :

```
moustaphalo@moustapha:~$ docker help run

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

Create and run a new container from an image

Aliases:
  docker container run, docker run

Options:
  --add-host list          Add a custom host-to-IP mapping (host:ip)
  --annotation map         Add an annotation to the container (passed through to the OCI runtime) (default map[])
  -a, --attach list        Attach to STDIN, STDOUT or STDERR
  --blkio-weight uint16    Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
  --blkio-weight-device list Block IO weight (relative device weight) (default [])
  --cap-add list           Add Linux capabilities
  --cap-drop list          Drop Linux capabilities
  --cgroup-parent string   Optional parent cgroup for the container
  --cgroupns string        Cgroup namespace to use (host|private)
  'host':     Run the container in the Docker host's cgroup namespace
  'private':  Run the container in its own private cgroup namespace
  ''         Use the cgroup namespace as configured by the
```

PS : Afin de s'assurer que vous avez parfaitement réussi la **Tâche 1**, vous devez lancer l'image prédéfinie hello-world habituellement utilisée pour vérifier que Docker est correctement installé et que tout fonctionne à merveille.

```
moustaphalo@moustapha:~$ 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/
```

Tâche 2 : Gestion des conteneurs Docker

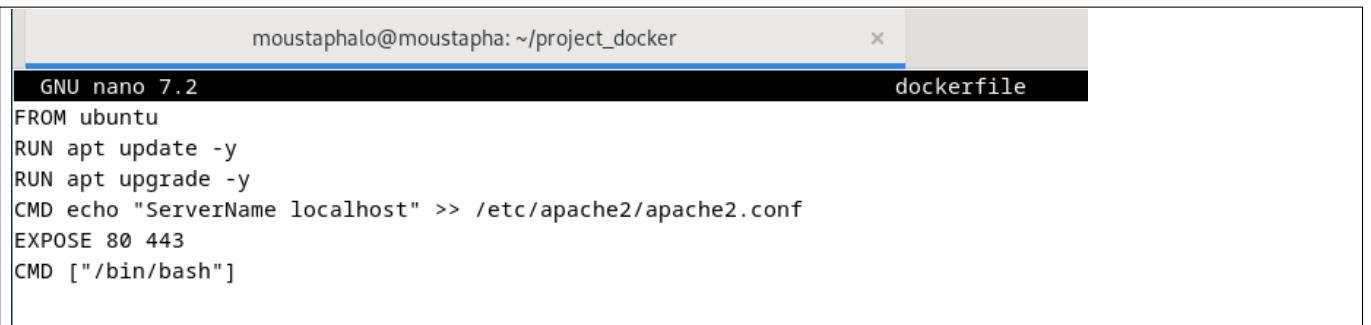
1. Création d'une image

Créer une image docker en commençant par créer un dossier qui va contenir les données de votre image ainsi que le fichier Dockerfile.

Le nom du dossier `project_docker`

```
moustaphalo@moustapha:~$ mkdir project_docker
moustaphalo@moustapha:~$ cd project_docker
moustaphalo@moustapha:~/project_docker$ nano dockerfile
moustaphalo@moustapha:~/project_docker$ ls
dockerfile
```

Contenu du fichier Dockerfile :



```
moustaphalo@moustapha:~/project_docker
GNU nano 7.2
FROM ubuntu
RUN apt update -y
RUN apt upgrade -y
CMD echo "ServerName localhost" >> /etc/apache2/apache2.conf
EXPOSE 80 443
CMD ["/bin/bash"]
```

Une fois le fichier Dockerfile créé, créer l'image intitulée (premiere-image).

```
moustaphalo@moustapha:~/project_docker$ docker build -t premiere-image .
[+] Building 2.2s (7/7) FINISHED
      docker:default
-> [internal] load build definition from dockerfile          0.5s
-> => transferring dockerfile: 185B                         0.0s
-> [internal] load metadata for docker.io/library/ubuntu:latest 0.0s
-> [internal] load .dockerrcignore                           0.4s
-> => transferring context: 2B                            0.0s
-> [1/3] FROM docker.io/library/ubuntu:latest              0.0s
-> CACHED [2/3] RUN apt update -y                          0.0s
-> CACHED [3/3] RUN apt upgrade -y                        0.0s
-> exporting to image                                     0.5s
-> => exporting layers                                    0.0s
-> => writing image sha256:c439c114edc6c301188a960a07cff3c6422adab33e5c5 0.0s
-> => naming to docker.io/library/premiere-image          0.1s

1 warning found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 5)
```

Maintenant que l'image est créée lancer le conteneur

```
moustaphalo@moustapha:~/project_docker$ docker run -it premiere-image
root@4ac4d3c17d82:/# apt install net-tools curl
```

Installer manuellement quelques programmes usuels dans le conteneur exemple (net-tools, curl, apache ...)

```
root@4ac4d3c17d82:/# apt install net-tools curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates krb5-locales libbrotli1 libcurl4t64 libgssapi-krb5-2 libk5crys
  libkrb5-3 libkrb5support0 libldap-common libldap2 libnghhttp2-14 libpsl5t64 lib
  libsasl2-modules libsasl2-modules-db libssh-4 openssl publicsuffix
Suggested packages:
  krb5-doc krb5-user libsasl2-modules-qssapi-mit | libsasl2-modules-qssapi-heimd
```

```

root@4ac4d3c17d82:/# apt policy apache2 curl net-tools
apache2:
  Installed: 2.4.58-1ubuntu8.5
  Candidate: 2.4.58-1ubuntu8.5
  Version table:
*** 2.4.58-1ubuntu8.5 500
      500 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages
        100 /var/lib/dpkg/status
  2.4.58-1ubuntu8.4 500
      500 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages
  2.4.58-1ubuntu8 500
      500 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages
curl:
  Installed: 8.5.0-2ubuntu10.6
  Candidate: 8.5.0-2ubuntu10.6
  Version table:
*** 8.5.0-2ubuntu10.6 500
      500 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages
      500 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages
        100 /var/lib/dpkg/status
  8.5.0-2ubuntu10 500
      500 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages
net-tools:
  Installed: 2.10-0.1ubuntu4
  Candidate: 2.10-0.1ubuntu4

```

2. Lister les conteneurs actifs

docker ps affiche **uniquement** les conteneurs en cours d'exécution.

```
moustaphalo@moustapha:~/project_docker$ docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS        PORTS     NAMES
4ac4d3c17d82   premiere-image  "/bin/bash"   50 minutes ago  Up 38 seconds  80/tcp, 443/tcp  suspicious_wing
```

docker ps -a affiche **tous** les conteneurs, qu'ils soient en cours d'exécution, arrêtés ou terminés.

```
moustaphalo@moustapha:~/project_docker$ docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS        PORTS     NAMES
4ac4d3c17d82   premiere-image  "/bin/bash"   54 minutes ago  Up 4 minutes  80/tcp, 443/tcp  suspicious_wing
db6f64a04420   89dc7ee30ca6   "/bin/sh -c 'echo \"S...\""
enberg        4299a0575be7   89dc7ee30ca6   "/bin/sh -c 'echo \"S...\""
joliot        000373078c20   89dc7ee30ca6   "/bin/sh -c 'echo \"S...\""
liamson        54 minutes ago  About an hour ago  Exited (2) About an hour ago
                                    About an hour ago  Exited (2) About an hour ago
                                    About an hour ago  Exited (2) About an hour ago

```

3. Status du dernier conteneur créé

La commande **docker ps -l** affiche **le dernier conteneur utilisé ou créé**, qu'il soit en cours d'exécution ou arrêté.

```
moustaphalo@moustapha:~/project_docker$ docker ps -l
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS        PORTS     NAMES
4ac4d3c17d82   premiere-image  "/bin/bash"   56 minutes ago  Up 6 minutes  80/tcp, 443/tcp  suspicious_wing
```

4. Démarrer un conteneur

Je prends comme exemple ce conteneur donc pour le démarrer j'utilise la commande **docker start -i** suivi de l' ID du conteneur

```
98865605b884   ubuntu          "/bin/bash"    13 days ago   Exited (0) 9 minutes ago   jovial_pt
v
moustaphalo@moustapha:~/project_docker$ docker start -i 98865605b884
root@98865605b884:/#
```

5. Arrêter un conteneur

Pour arrêter un conteneur il suffit de faire **docker stop** suivi de l' **ID du conteneur**

```
moustaphalo@moustapha:~/project_docker$ docker stop 98865605b884  
98865605b884
```

6. Supprimer un conteneur

Par exemple je veux supprimer ce conteneur

```
c4997bef7cff  ubuntu          "/bin/bash"           13 days ago      Exited (0) 16 minutes ago      kind banzai
```

Pour cela il faut faire **docker rm** suivi l' **ID du conteneur**

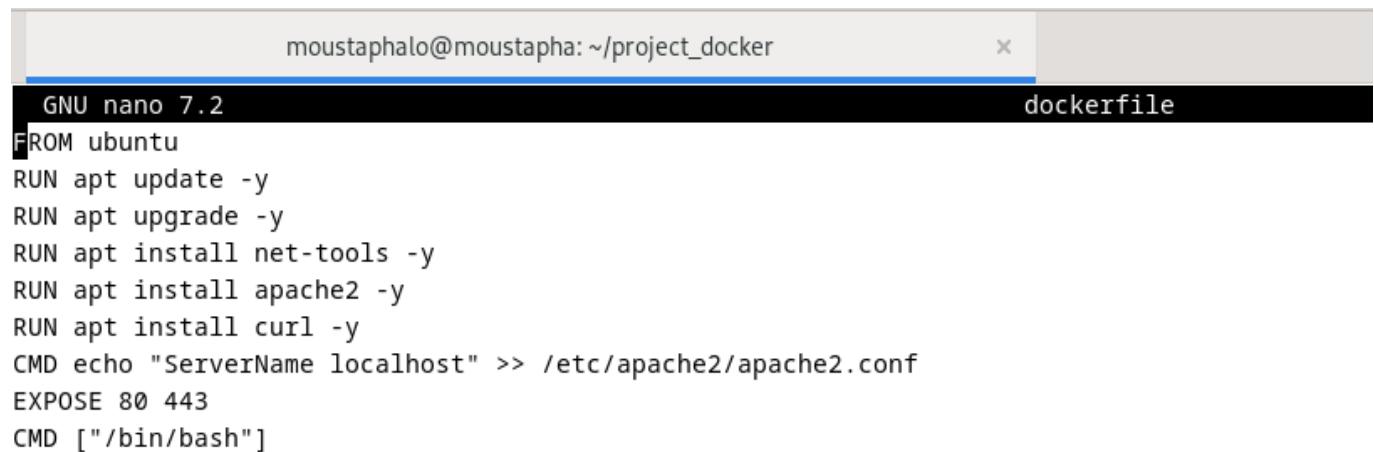
```
moustaphalo@moustapha:~/project_docker$ docker rm c4997bef7cff  
c4997bef7cff
```

7. Installation automatique des programmes précédemment installés

Nous allons éditer à nouveau le dockerfile en y ajoutant

```
RUN apt install net-tools -y  
RUN apt install apache2 -y  
RUN apt install curl -y
```

Contenu du fichier Dockerfile



The screenshot shows a terminal window with the title "moustaphalo@moustapha: ~/project_docker". The file is named "dockerfile". The content of the file is as follows:

```
GNU nano 7.2  
FROM ubuntu  
RUN apt update -y  
RUN apt upgrade -y  
RUN apt install net-tools -y  
RUN apt install apache2 -y  
RUN apt install curl -y  
CMD echo "ServerName localhost" >> /etc/apache2/apache2.conf  
EXPOSE 80 443  
CMD ["/bin/bash"]
```

Donc ici après avoir lancer le conteneur à partir de l'image créé on voit qu'il y'a une installation automatique des programmes précédemment installés

```
moustaphalo@moustapha:~/project_docker$ docker build -t premiere-image .
[+] Building 0.7s (10/10) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 261B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 7)
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 7)
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/6] FROM docker.io/library/ubuntu:latest
=> CACHED [2/6] RUN apt update -y
=> CACHED [3/6] RUN apt upgrade -y
=> CACHED [4/6] RUN apt install net-tools -y
=> CACHED [5/6] RUN apt install apache2 -y
=> CACHED [6/6] RUN apt install curl -y
=> exporting to image
=> => exporting layers
=> => writing image sha256:c846b0a029d0aba5c2fdc8d8c827efa97664ab992e4113d9497d650c99b31338
=> => naming to docker.io/library/premiere-image

2 warnings found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 7)
- MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will be used (line 7)
```

PS : Afin de s'assurer que vous avez réussi parfaitement la **Tâche 2.** vous devez accéder au serveur web depuis le navigateur ou bien depuis le terminal de la machine Docker

Depuis la machine docker

```
root@feccefac1c06:/# systemctl start apache2
bash: systemctl: command not found
root@feccefac1c06:/# curl http://172.17.0.2
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2022-03-22
    See: https://launchpad.net/bugs/1966004
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">
      *
      {
        margin: 0px 0px 0px 0px;
        padding: 0px 0px 0px 0px;
      }

      body, html {
        padding: 3px 3px 3px 3px;
```

Tâche 3 : Docker Hub

1. Recherche d'une image

Docker Hub est le dépôt distant géré par l'organisation Docker, vous pouvez y déposer vos images ou bien même télécharger des images depuis les serveurs Docker Hub. Rechercher une image MySQL dans le Docker Hub.

Pour rechercher une image MySQL dans le Docker Hub, on utilise la commande **docker search mysql**

```
moustaphalo@moustapha:~$ docker search mysql
NAME                           DESCRIPTION                               STARS      OFFICIAL
mysql                          MySQL is a widely used, open-source relation... 15542      [OK]
bitnami/mysql                   Bitnami container image for MySQL                    122
circleci/mysql                  MySQL is a widely used, open-source relation... 30
cimg/mysql                      MySQL client                                         3
bitnamicharts/mysql            Bitnami Helm chart for MySQL                         0
ubuntu/mysql                     MySQL open source fast, stable, multi-thread... 66
rapidfort/mysql                 RapidFort optimized, hardened image for MySQL       26
elestio/mysql                    Mysql, verified and packaged by Elestio             1
google/mysql                     MySQL server for Google Compute Engine           25
docksal/mysql                   MySQL service images for Docksal - https://d... 0
alpine/mysql                     mysql client                                         3
mysql/mysql-server              Optimized MySQL Server Docker images. Create... 1023
jumpserver/mysql                MySQL image pre-configured to work smoothly ... 1
datajoint/mysql                 MySQL Router provides transparent routing be... 28
mysql/mysql-router              MySQL Router provides transparent routing be... 1
ddev/mysql                       ARM64 base images for ddev-dbserver-mysql-8... 0
mirantis/mysql                  Mysql configured for running Ilios                  1
corpusops/mysql                 https://github.com/corpusops/docker-images/        0
mysql/mysql-cluster             Experimental MySQL Cluster Docker images. Cr... 100
javanile/mysql                  MySQL for development                         0
vulhub/mysql                     MySQL Operator for Kubernetes                   1
mysql/mysql-operator            MySQL Operator for Kubernetes                   1
vitess/mysql                     Lightweight image to run MySQL with Vitess        1
soccerbox/mysql                 MySQL image for soccerbox                         1
```

2. Téléchargement du conteneur

Télécharger une image MySQL depuis Docker Hub.

Pour télécharger l'image MySQL depuis Docker Hub, on utilise la commande **docker pull mysql**

```
moustaphalo@moustapha:~$ docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
2c0a233485c3: Pull complete
cb5a6a8519b2: Pull complete
570d30cf82c5: Pull complete
a841bff36f3c: Pull complete
80ba30c57782: Pull complete
5e49e1f26961: Pull complete
ced670fc7f1c: Pull complete
0b9dc7ad7f03: Pull complete
cd0d5df9937b: Pull complete
1f87d67b89c6: Pull complete
Digest: sha256:0255b469f0135a0236d672d60e3154ae2f4538b146744966d96440318cc822c6
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
```

3. Téléverser une image dans le docker Hub

Docker Hub ne permet pas uniquement de télécharger des images mais permet aussi de sauvegarder en ligne nos propres images. Pour cela il faut disposer d'un compte docker hub. Créer un compte Docker Hub depuis le site officiel de docker.

Compte déjà créé

A screenshot of a web browser displaying the Docker Hub website at https://hub.docker.com/explore. The page features a large blue header with the text "Docker's curated GenAI catalog" and "Everything you need to build, scale, and deploy AI with ease." Below this is a "View catalog now" button. At the bottom of the main content area, there are tabs labeled "Catalogs" and "Spotlight".

Une fois le compte créé, se connecter à votre dépôt distant docker

Pour se connecter, il faut saisir la commande **docker login -u** suivi du **nom de l'utilisateur**.

Ensuite on saisit le mot de passe qu'on a mis lors de la création du compte

```
moustaphalo@moustapha:~$ docker login -u chacker427
```

Password:

```
WARNING! Your password will be stored unencrypted in /home/moustaphalo/.docker/config.json.
```

Configure a credential helper to remove this warning. See

```
https://docs.docker.com/engine/reference/commandline/login/#credential-stores
```

```
Login Succeeded
```

```
moustaphalo@moustapha:~$
```

Apres avoir connecter à votre dépôt distant, il faut y déposer une image pour cet exemple il faut sauvegarder sur Docker Hub votre image **premiere-image**.

Nous allons sauvegarder sur mon dépôt Docker hub mon image **premier-image**

```
moustaphalo@moustapha:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
apacheready	latest	84551ab5eb6e	4 hours ago	231MB
chacker427/apacheready	version1.0	84551ab5eb6e	4 hours ago	231MB
premiere-image	latest	c846b0a029d0	2 days ago	232MB

Pour cela il faut d'abord taguer l'image avec mon nom d'utilisateur Docker Hub avec la commande docker tag suivi de l'ID de l'image nom-utilisateur-docker/premiere-image :version1.0

```
moustaphalo@moustapha:~$ docker tag c846b0a029d0 chacker427/premiere-image:version1.0
```

Ensute on téléverse l'image sur Docker Hub avec la commande docker push nom-utilisateur r-docker/première-image :version1.0

```
moustaphalo@moustapha:~$ docker push chacker427/premiere-image:version1.0
The push refers to repository [docker.io/chacker427/premiere-image]
9405930ed31d: Pushed
afe5916c191f: Pushed
39b67b154abe: Pushed
178df4937b80: Pushed
13f4f47891a4: Pushed
687d50f2f6a6: Mounted from chacker427/apacheready
version1.0: digest: sha256:1535ba25b4e6b18a316fed7fcbdfe173c75c34f80bfabb1fbbe57a64a9a30e38
size: 1580
```

```
moustaphalo@moustapha:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
chacker427/apacheready	version1.0	84551ab5eb6e	4 hours ago	231MB
apacheready	latest	84551ab5eb6e	4 hours ago	231MB
chacker427/premiere-image	version1.0	c846b0a029d0	2 days ago	232MB

Si on vérifie dans notre dépôt Docker hub on voit bien que l'image est sauvegardée.

Name	Last Pushed	Contains	Visibility	Scout
chacker427/premiere-image	14 minutes ago	[IMAGE]	Public	Inactive
chacker427/apacheready	about 1 hour ago	[IMAGE]	Public	Inactive

Tâche 4 : Docker Compose

Docker compose est un outil qui permet de gérer les conteneurs comme un ensemble de services interconnectés à l'aide d'un fichier YAML. Pour cet exemple il faut gérer l'interconnexion des deux connecteurs précédemment créés à l'aide du fichier **Dockerfile** et de l'image **MYSQL** téléchargée depuis Docker Hub.

1. Installation de docker compose

Docker compose ne faisant pas parti des outils de base l'or de l'installation de Docker, il faut l'installer.

Installation de docker compose avec la commande **apt install docker-compose -y**

```
moustaphalo@moustapha:~$ sudo apt install docker-compose -y
```

Lecture des listes de paquets... Fait

Construction de l'arbre des dépendances... Fait

Lecture des informations d'état... Fait

Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires

```
    docker-buildx-plugin docker-compose-plugin libslirp0 pigz slirp4netns
```

Veuillez utiliser « sudo apt autoremove » pour les supprimer.

Les paquets supplémentaires suivants seront installés :

```
    cgroupfs-mount containerd criu docker.io libintl-perl libintl-xs-perl
```

Vérifier que docker compose a bien été installé.

La commande **docker-compose --version** nous confirme que docker compose a bien été installé.

```
moustaphalo@moustapha:~$ docker-compose --version
```

```
docker-compose version 1.29.2, build unknown
```

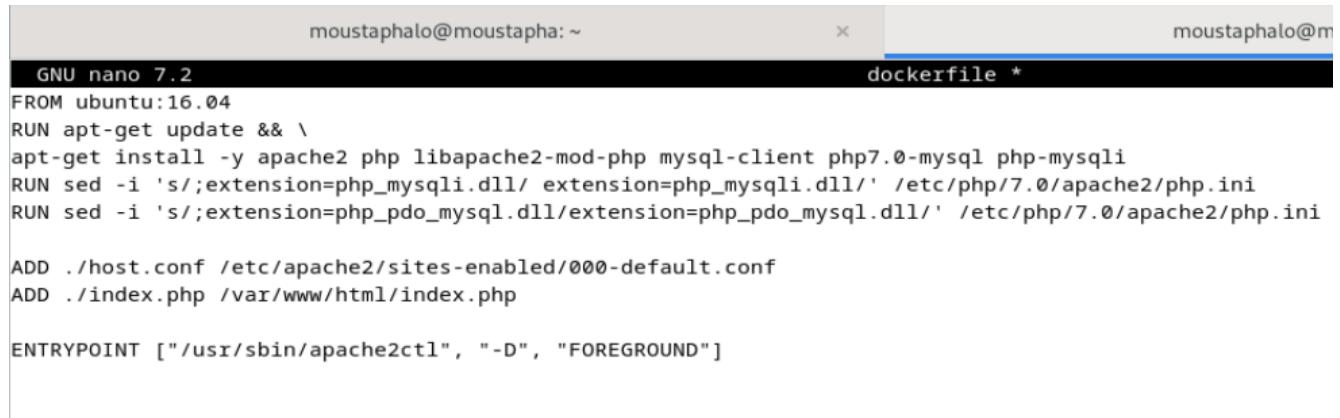
```
moustaphalo@moustapha:~$ █
```

2. création du fichier docker-compose.yml

Docker compose permet de gérer en même temps plusieurs images Docker à l'aide du fichier **docker-compose.yml**. Utiliser les deux images précédemment créer à l'aide du fichier **Dockerfile** et l'image **MySQL** obtenu à l'aide de Docker Hub. Une fois le fichier vous pouvez démarrer et interconnecter les conteneurs en une seule commande. Créer le fichier **docker-compose.yml**

```
moustaphalo@moustapha:~$ nano docker-compose.yml
```

Contenu du fichier dockerfile

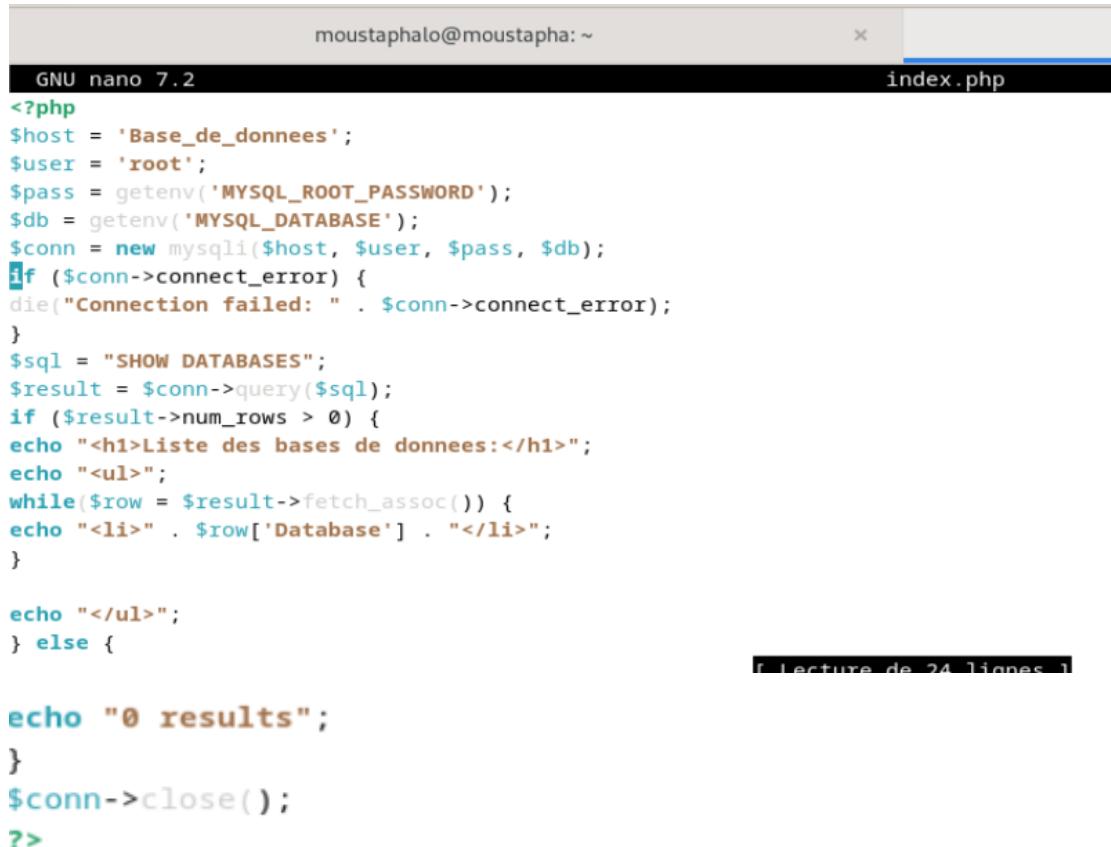


```
moustaphalo@moustapha: ~
GNU nano 7.2
FROM ubuntu:16.04
RUN apt-get update && \
apt-get install -y apache2 php libapache2-mod-php mysql-client php7.0-mysql php-mysqli
RUN sed -i 's/extension=php_mysqli.dll/ extension=php_mysqli.dll/' /etc/php/7.0/apache2/php.ini
RUN sed -i 's/extension=php_pdo_mysql.dll/extension=php_pdo_mysql.dll/' /etc/php/7.0/apache2/php.ini

ADD ./host.conf /etc/apache2/sites-enabled/000-default.conf
ADD ./index.php /var/www/html/index.php

ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Contenu du fichier Index.php



```
moustaphalo@moustapha: ~
GNU nano 7.2
<?php
$host = 'Base_de_donnees';
$user = 'root';
$pass = getenv('MYSQL_ROOT_PASSWORD');
$db = getenv('MYSQL_DATABASE');
$conn = new mysqli($host, $user, $pass, $db);
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
$sql = "SHOW DATABASES";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
echo "<h1>Liste des bases de données:</h1>";
echo "<ul>";
while($row = $result->fetch_assoc()) {
echo "<li>" . $row['Database'] . "</li>";
}
echo "</ul>";
} else {
}
echo "<0 results";
}
$conn->close();
?>
```

3. lancement du docker compose

Lancer Docker Compose une fois lancé les deux conteneur seront automatiquement lancés.

```
moustaphalo@moustapha:~$ docker-compose up -d
```

```
mandicou@Mandicou :~$ ****
```

PS : Afin de s'assurer que vous avez réussi parfaitement la **Tâche 4.** vous devez accéder au serveur web depuis le navigateur ou bien depuis le terminal de la machine Docker et voir le contenu de la base de données MYSQL.

Tâche 5 : Docker Swarm

L'objectif est de créer un cluster Docker et y déployer un service avec le système d'orchestration de conteneurs natif de Docker(Docker Swarm).

Prérequis : En plus de la machine Debian 12 où Docker est installé qui sera notre noeud principal (Master), il nous faut également une deuxième machine avec un OS de votre choix où Docker est correctement installé. Cette deuxième machine sera notre noeud de travail (Worker).

PS : Pour gagner du temps, vous pouvez utiliser **Docker Machine** pour créer le noeud de travail (Worker)

1. Initialiser Docker Swarm

La phase d'initialisation se fait sur le noeud principal et va vous permettre de récupérer le token utilisé pour ajouter des noeuds de travail."

```
docker swarm init --advertise-addr [ip-du-manager]
```

```
moustaphal0@moustapha:~$ docker swarm init --advertise-addr 192.168.1.239
Swarm initialized: current node (qfjv1mgtqnqe1s8ctdd8v0k5ay) is now a manager.
```

To add a worker to this swarm, run the following command:

```
docker swarm join --token SWMTKN-1-35ob6zh7hjqvtrg96bte826keyxfwu01n6b9jygh6qmnirx3r8-amvjv1rqdtslvtlrov3gzi6lu 192.168.1.239:2377
```

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

2. Ajouter un noeud worker au Swarm

Utiliser le token fourni par la commande précédente et ajouter la deuxième machine en tant que noeud de travail.

```
docker swarm join --token SWMTKN-1-35ob6zh7hjqvtrg96bte826keyxfwu01n6b9jygh6qmnirx3r8-amvjv1rqdtslvtlrov3gzi6lu 192.168.1.239:2377
```

Machine 2 :

```
root@chacker:/home/chackerlo# docker swarm join --token SWMTKN-1-35ob6zh7hjqvtrg96bte826keyxfwu01n6b9jygh6qmnirx3r8-amvjv1rqdtslvtlrov3gzi6lu 192.168.1.239:2377
This node joined a swarm as a worker.
root@chacker:/home/chackerlo#
```

3. Afficher les informations du clusters

```
moustaphalo@moustapha:~$ docker node ls
ID                  HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS   ENGINE VERSION
rdh9iwapxn8urq488rctlvwyu  chacker  Ready   Active
qfjv1mgtnqeis8ctdd8v0k5ay *  moustapha  Ready   Active        Leader
moustaphalo@moustapha:~$ 

moustaphalo@moustapha:~$ docker info
Client:
  Context:    default
  Debug Mode: false
  Plugins:
    buildx: Docker Buildx (Docker Inc., v0.20.0)
    compose: Docker Compose (Docker Inc., v2.32.4)

Server:
  Containers: 28
    Running: 1
    Paused: 0
    Stopped: 27
  Images: 19
  Server Version: 20.10.24+dfsg1
  Storage Driver: overlay2
    Backing Filesystem: extfs
    Supports d_type: true
    Native Overlay Diff: true
    userxattr: false
  Logging Driver: json-file
  Cgroup Driver: systemd
  Cgroup Version: 2

  Plugins:
    Volume: local
    Network: bridge host ipvlan macvlan null overlay
    Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
  Swarm: active
    NodeID: qfjv1mgtnqeis8ctdd8v0k5ay
    Is Manager: true
    ClusterID: vaw2rkokvuy2bynf8apsgbk05
    Managers: 1
    Nodes: 2
    Default Address Pool: 10.0.0.0/8
```

4. Déploiement d'un service sur le cluster

Déployer un service simple basé sur l'image nginx qui sera accessible depuis le port 80 :

```
docker service create --name mon-service-nginx-lic3srt --publish 80:80 nginx
```

```
moustaphalo@moustapha:~$ docker service create --name mon-service-nginx-lic3srt --publish 80:80 nginx
oryjixeay3h1i396qtqnolxzy
overall progress: 1 out of 1 tasks
1/1: running  [=====]
verify: Service converged
```

5. Vérifier l'état du service

```
docker service ls
```

```
moustaphalo@moustapha:~$ docker service ls
ID          NAME          MODE        REPLICAS   IMAGE          PORTS
oryjixeay3h1  mon-service-nginx-lic3srt  replicated  1/1      nginx:latest  *:85->80/tcp
moustaphalo@moustapha:~$
```

6. Mise à l'échelle du service

Augmenter ou diminuer le nombre de réplicas de votre service en utilisant la commande scale.

```
docker service scale mon-service-nginx-lic3srt=3
```

```
moustaphalo@moustapha:~$ docker service scale mon-service-nginx-lic3srt=3
mon-service-nginx-lic3srt scaled to 3
overall progress: 3 out of 3 tasks
1/3: running  [=====>]
2/3: running  [=====>]
3/3: running  [=====>]
verify: Service converged
moustaphalo@moustapha:~$
```

```
moustaphalo@moustapha:~$ docker service ls
ID          NAME          MODE        REPLICAS   IMAGE          PORTS
oryjixeay3h1  mon-service-nginx-lic3srt  replicated  3/3      nginx:latest  *:85->80/tcp
```

PS : Afin de s'assurer que vous avez réussi parfaitement la **Tâche 5.** vous devez accéder au serveur Nginx depuis un navigateur ou bien depuis le terminal d'une des machine du cluster et vérifier la tolérance aux pannes.

Depuis le navigateur



Depuis le terminal

```
moustaphalo@moustapha:~$ curl http://192.168.1.239:85
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
```

Tâche 6 : Portainer

Il est possible de gérer les conteneurs docker depuis un interface graphique, plusieurs outils permettent de le faire nous allons prendre un des outils qui s'appelle **Portainer** qui va permettre depuis son interface web de gerer les images de Docker.

1. installation de Portainer

Commencer par créer un volume qui va accueillir les données de **Portainer**.

```
moustaphalo@moustapha:~$ docker volume create portainer_data-li3csrt
portainer_data-li3csrt
moustaphalo@moustapha:~$ █
```

Après avoir créer le volume, démarrer Portainer depuis le port 9000

```
moustaphalo@moustapha:~$ docker run -d --name=portainer -p 9000:9000 -p 8000:8000 --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data-li3csrt:/data portainer/portainer-ce
409ce11261136b2ca22a2381840b40cf163d91ebd57a4ab59b834d11cae6883d
```

2. Utilisation de Portainer

Une fois que Portainer est lancé il faut ouvrir votre navigateur et saisir l'adresse suivante

On ouvre le navigateur et on tape localhost:9000

Installation de la nouvelle installation de Portainer

Veuillez créer l'utilisateur initial de l'administrateur.

Nom d'utilisateur	moustapha
Mot de passe	████████████████
Confirmer le mot de passe	████████████████ ✓

⚠ Le mot de passe doit comporter au moins 12 caractères. ✓

Créer un utilisateur

Autoriser la collecte de statistiques anonymes. Vous trouverez plus d'informations à ce sujet dans notre [politique de la vie privée](#).

Home

Environments

Click on an environment to manage

Platf...	Connecti...	Status	Tags	Grou...	Agent ...	Clear all	So...	↓↑	
 local	Up	2025-02-14 17:10:47	Swarm 20.10.24+dfsg1	/var/run/docker.sock		Group: Unassigned	No tags	Local	
0 stacks	1 service	31 containers	4 green	26 orange	0 red	5 volumes	Disconnected		
18 images	4 CPU	6.2 GB RAM	2 nodes						

Items per page 10

Dashboard

Information

Portainer is connected to a node that is part of a Swarm cluster. Some resources located on other nodes in the cluster might not be available for management, have a look at [our agent setup](#) for more details.

Environment info

Environment	local	4	6.2 GB	Swarm 20.10.24+dfsg1
URL	/var/run/docker.sock			
GPU	none			
Tags	-			

[Go to cluster visualizer](#)

PS : Afin de s'assurer que vous avez parfaitement réussi la **Tâche 6**, vous devez tester certaines fonctionnalités de Portainer, par exemple créer une image, lancer un conteneur, effectuer un déploiement rapide d'application à l'aide de l'interface graphique.

Pour créer une image dans portainer

The screenshot shows the Portainer interface. At the top, a header bar displays a container ID (98721493a364389947cd6032222a...), status (Unused), image (python:latest), memory (1 GB), and created date (2025-02-05 00:51:20). Below this is a section titled "Lancer un conteneur" (Launch a container) with a "Create container" button. The "Create container" form is filled with the following values:

- Name: monconteneur
- Registry: Docker Hub (anonymous)
- Image: docker.io python:latest

Below the form, there is an "Advanced mode" link and a "Always pull the image" toggle switch, which is turned on. A note indicates that the user is using an anonymous account and will be limited to 100 pulls every 6 hours. The "Webhook" section shows a "Create a container webhook" button with a toggle switch and a "Business Feature" link. A success message box is displayed, stating "Success Container successfully created". At the bottom, the "Container list" shows one container named "monconteneur" with the status "exited - code 0".

Effectuer un déploiement rapide d'application à l'aide de l'interface graphique.

Stacks > Add stack

Create stack

Name mystack

This stack will be deployed using the equivalent of the `docker stack deploy` command.

Build method

- Web editor
- Upload
- Repository
- Custom template

Upload

You can upload a Compose file from your computer.

Select file docker-compose.yml

