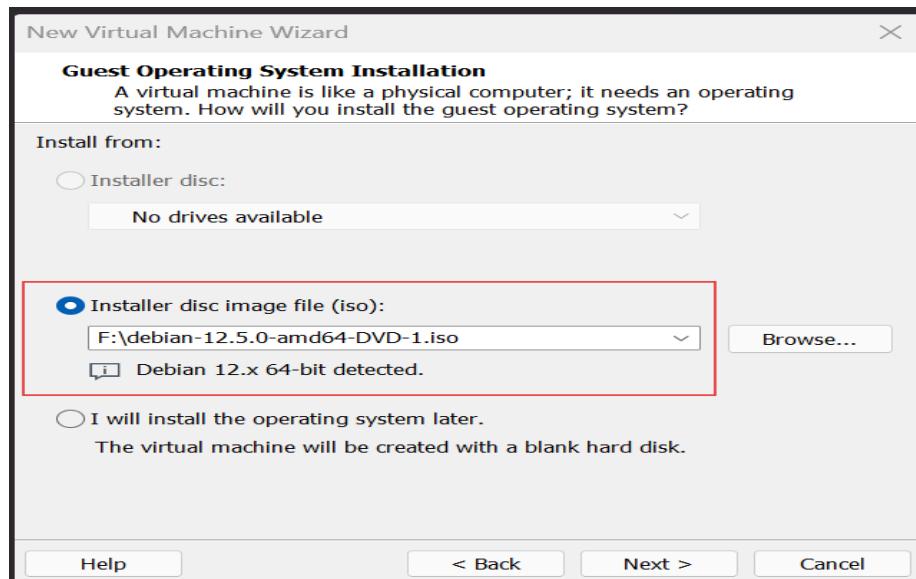


Job 01 :

Création d'une VM Debian minimaliste et y installer Docker CLI uniquement :

Une VM Debian en mode console 8 Go DD / 1 Go RAM / 1 vCPU :

Ajout de l'image ISO



Name the Virtual Machine

What name would you like to use for this virtual machine?

Virtual machine name:

DebianDocker

Location:

C:\Users\reina\OneDrive\Documents\Virtual Machines\DebianDoc

[Browse...](#)

The default location can be changed at [Edit > Preferences](#).

< Back

[Next >](#)

Cancel

1 VCPU

1 Go Ram

8 Go de DD

New Virtual Machine Wizard



Specify Disk Capacity

How large do you want this disk to be?

The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.

Maximum disk size (GB):

Recommended size for Debian 12.x 64-bit: 20 GB

- Store virtual disk as a single file
- Split virtual disk into multiple files

Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.

Help

< Back

Next >

Cancel

New Virtual Machine Wizard

X

Ready to Create Virtual Machine

Click Finish to create the virtual machine and start installing Debian 12.x 64-bit.

The virtual machine will be created with the following settings:

Name:	DebianDocker
Location:	C:\Users\reina\OneDrive\Documents\Virtual Machines\Deb...
Version:	Workstation 17.5 or later
Operating System:	Debian 12.x 64-bit
Hard Disk:	8 GB, Split
Memory:	1024 MB
Network Adapter:	NAT
Other Devices:	CD/DVD, USB Controller, Sound Card

Customize Hardware...

Power on this virtual machine after creation

< Back

Finish

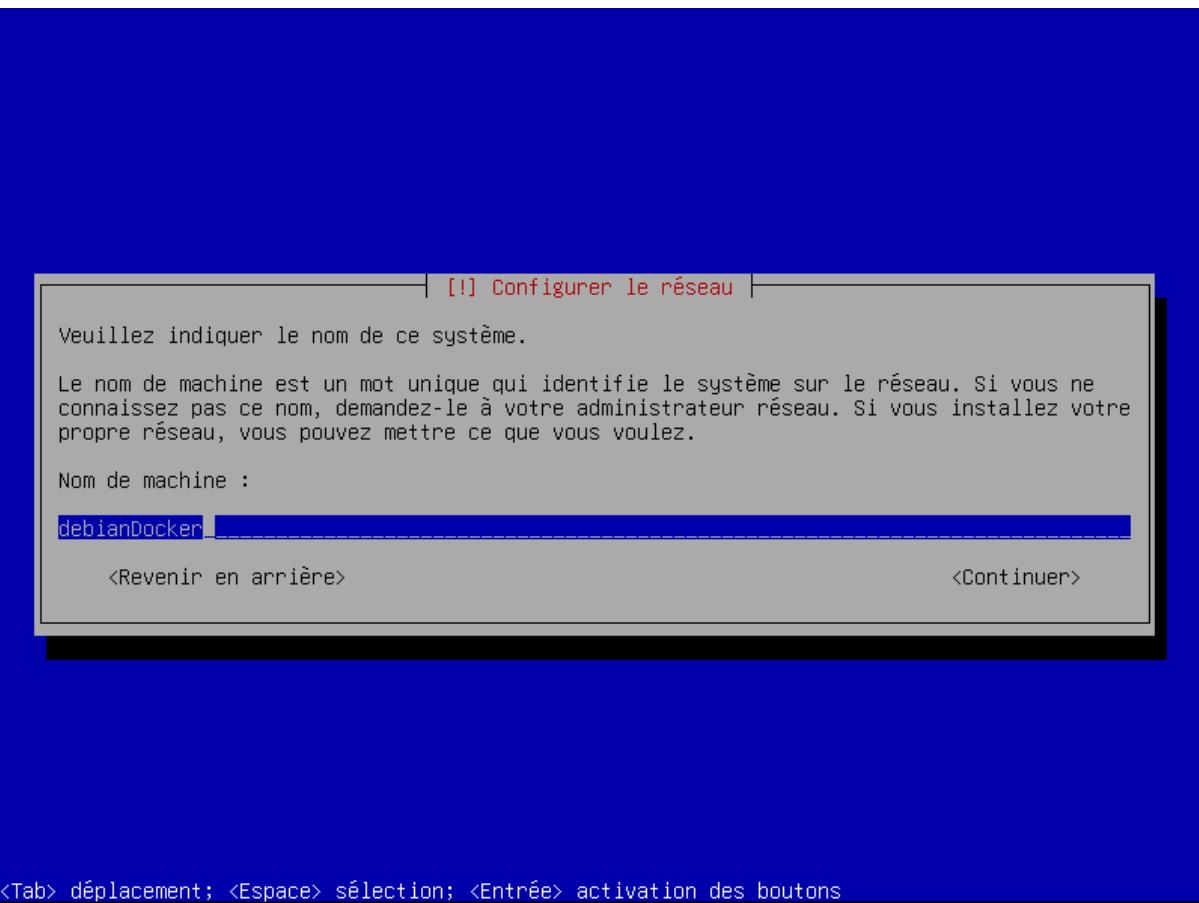
Cancel

© debian 12

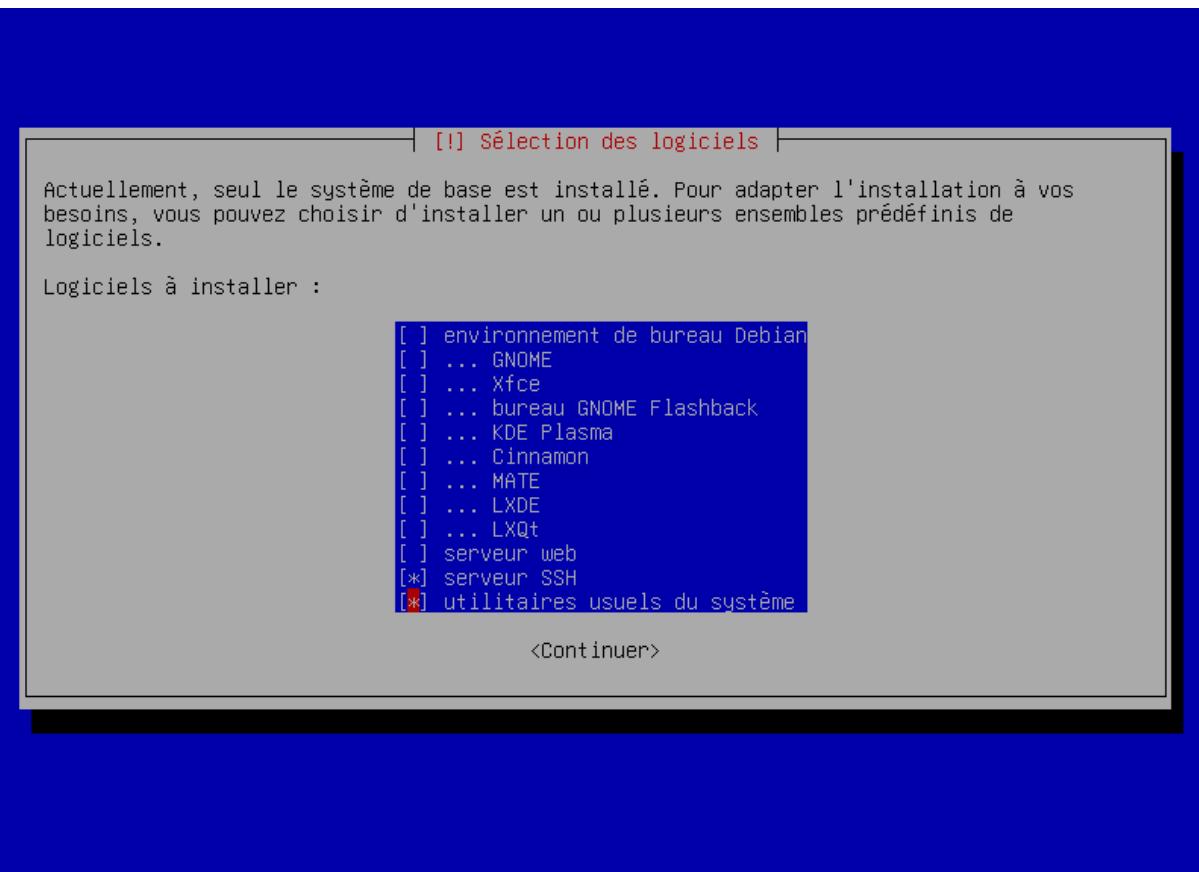
Debian GNU/Linux installer menu (BIOS mode)

- Graphical install
- Install**
- Advanced options >
- Accessible dark contrast installer menu >
- Help
- Install with speech synthesis

Press a key, otherwise speech synthesis will be started in 23 seconds...



<Tab> déplacement; <Espace> sélection; <Entrée> activation des boutons



```
Debian GNU/Linux 12 debianDocker tty1

debianDocker login: youcef
Password:
Linux debianDocker 6.1.0-32-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.129-1 (2025-03-06) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
youcef@debianDocker:~$ _
```

apt install -y docker.io

```
root@debdocker:/etc/apt# apt update
Atteint :1 http://security.debian.org/debian-security bookworm-security InRelease
Atteint :2 http://deb.debian.org/debian bookworm InRelease
Atteint :3 http://deb.debian.org/debian bookworm-updates 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.
root@debdocker:/etc/apt# apt update 77 apt upgrade )y
-dash: erreur de syntaxe près du symbole inattendu ` )` »

root@debdocker:/etc/apt#
root@debdocker:/etc/apt# apt update
Atteint :1 http://security.debian.org/debian-security bookworm-security InRelease
Atteint :2 http://deb.debian.org/debian bookworm InRelease
Atteint :3 http://deb.debian.org/debian bookworm-updates 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.
root@debdocker:/etc/apt# apt upgrade
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Calcul de la mise à jour... Fait
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
root@debdocker:/etc/apt# apt install -y docker.io
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  binutils-binutils-x86-64-linux-gnu cgroups-mount containerd criu git git-man iptables libbinutils libctf-nobfd0 libctf0 liberror-perl
  libbpfng0 libini-perl libini-xs-perl libip6tc2 libmodule-find-perl libnetfilter-conntrack3 libnetfilter_conntrack3 libnl3-200 libproc-processstable-perl
  libprotobuf32 libsort-naturally-perl libterm-readkey-perl needrestart patch python3-protobuf runc sgml-base tini
Paquets suggérés :
  binutils-doc containerNetworking-plugins docker-doc aufs-tools btrfs-progs debootstrap rinse rootlesskit xfsprogs zfs-fuse | zfsutils-linux git-daemon-run
  | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn firewalld needrestart-session | libnotify-bin ed difutils-doc
  sgml-base-doc
Les NOUVEAUX paquets suivants seront installés :
  binutils-binutils-common binutils-x86-64-linux-gnu cgroups-mount containerd criu docker.io git git-man iptables libbinutils libctf-nobfd0 libctf0
  liberron-perl libbpfng0 libini-perl libini-xs-perl libip6tc2 libmodule-find-perl libnetfilter-conntrack3 libnetfilter_conntrack3 libnl3-200
  libproc-processstable-perl libprotobuf32 libsort-naturally-perl libterm-readkey-perl needrestart patch python3-protobuf runc sgml-base tini
0 mis à jour, 33 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de prendre 34,3 Mo dans les archives.
Après cette opération, 360 Mo d'espace disque supplémentaires seront utilisés.
Reception de :1 http://deb.debian.org/debian bookworm/main amd64 runc amd64 1.1.5+ds1-1+deb12u1 [2 710 kB]
Reception de :2 http://deb.debian.org/debian bookworm/main amd64 containerd amd64 1.6.20+ds1-1+deb12u1 [25,9 MB]
24% [2 containerd 22,2 MB/25,9 MB 86%]
```

233 kB/s 4min 15s

docker --version

```
root@debdocker:/etc/apt# docker --version
Docker version 20.10.24+dfsg1, build 297e128
root@debdocker:/etc/apt#
```

Job 02

Tester l'installation de docker avec le conteneur « hello-world » et se familiariser avec les commandes.

docker run hello-world

Explication :

docker run :

C'est la commande pour lancer un conteneur Docker à partir d'une image.

hello-world :

C'est le nom de l'image Docker que l'on veut exécuter dans notre exemple. Si elle n'est pas présente localement sur la machine, Docker va télécharger depuis Docker Hub (le registre officiel).

```
root@debdocker:/etc/apt# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:7e1a4e2d11e2ac7a8c3f768d4166c2defeb09d2a750b010412b6ea13de1efb19
Status: Downloaded newer image for hello-world:latest
```

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/
```

```
root@debdocker:/etc/apt# _
```

Commandes gestion d'images :

docker pull <image> : Télécharge une image depuis Docker Hub vers votre machine locale.

docker images : Affiche la liste des images Docker présentes localement.

docker rmi <image> : Supprime une image Docker de votre machine

Commandes gestion des containers :

docker run <image> : Crée et démarre un nouveau conteneur à partir d'une image.

docker run -d <image> : Démarrer un conteneur en arrière-plan (mode détaché).

docker run -it <image> bash : Démarrer un conteneur en mode interactif avec un terminal bash.

docker ps : Affiche les conteneurs en cours d'exécution.

docker ps -a : Affiche tous les conteneurs, y compris ceux arrêtés.

docker stop <nom|id> : Arrête un conteneur en cours d'exécution.

docker start <nom|id> : Démarrer un conteneur arrêté.

docker restart <nom|id> : Redémarre un conteneur.

docker rm <nom|id> : Supprime un conteneur arrêté.

Commandes Clean :

docker system prune : Supprime tous les objets inutilisés (conteneurs arrêtés, images non utilisées, volumes non utilisés, etc.).

docker volume ls : Liste tous les volumes Docker.

docker volume rm <nom> : Supprime un volume Docker.

Commandes utiles diverses :

docker exec -it <nom|id> bash : Ouvre un terminal interactif dans un conteneur en cours d'exécution.

docker logs <nom|id> : Affiche les logs (journaux) d'un conteneur.

docker inspect <nom|id> : Affiche des informations détaillées sur un conteneur ou une image.

Job 03

Utilisation de « Dockerfile » pour recréer le conteneur « helloworld » depuis une image Debian minimum.

Un **Dockerfile** est un **fichier texte contenant un ensemble d'instructions** qui servent à **construire une image Docker**. Ces instructions décrivent **comment configurer l'environnement** dans lequel une application va tourner, de manière automatisée et reproductible

Le Dockerfile sert à :

1. **Créer une image Docker personnalisée** pour ton application.
2. **Définir tout ce qu'il faut pour exécuter ton application** : système de base, bibliothèques, dépendances, fichiers sources, ports à exposer, etc.
3. **Faciliter la portabilité** : tout ce qui est dans le Dockerfile peut être exécuté de la même manière sur n'importe quelle machine ayant installé Docker.

Création du dossier et du Dockerfile :

```
mkdir ~/docker-helloworld
cd ~/docker-helloworld
nano Dockerfile
root@debdocker:/home/jordan/docker-helloworld# ls
Dockerfile
```

```
GNU nano 7.2
#image de base
FROM debian:bookworm-slim
#MAJ
RUN apt-get update && apt-get install -y curl && apt-get clean
#Commande exécutée
CMD echo "Hello from Jordan et Youcef pour le sujet la plateforme Docker"
```

Construction de l'image :

docker build -t helloworld-debian .

```
root@debdocker:/home/Jordan/docker-helloworld# docker build -t helloworld-debian .
```

```
Successfully built 00c24e58b7fe
Successfully tagged helloworld-debian:latest
```

Lancement du container :

docker run helloworld-debian

```
Successfully tagged helloworld-debian:latest
root@debdocker:/home/jordan/docker-helloworld# docker run helloworld-debian
Hello from Jordan et Youcef pour le sujet la plateforme Docker
```

Job 04

Utilisation de Dockerfile pour créer une image SSH (compte : root et mot de passe : root123) sans utiliser une image SSH existante, voir redirection des ports (utiliser un autre port que le 22) , créé et lancez le conteneur et se connecter pour vérifier l'accès SSH.

Création du dossier et du Dockerfile

mkdir ~/docker-ssh

cd ~/docker-ssh

```
[root@debdocker:/home/jordan# mkdir docker-ssh]
```

nano Dockerfile

```
GNU nano 7.2                                            Dockerfile *
FROM debian:bookworm-slim

#MAJ + installation SSH
RUN apt-get update && apt-get install -y openssh-server && apt-get clean

#MDP pour root
RUN echo "root:root123" | chpasswd

#Création repertoire SSH
RUN mkdir /var/run/sshd

#Permettre accès root via SSH
RUN sed -i 's/^#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config

#Changer le port SSH
RUN sed -i 's/#Port 22/Port 2222/' /etc/ssh/sshd_config

#Exposer le port 2222
EXPOSE 2222

#Commande de démarrage du service SSH
CMD ["/usr/sbin/sshd", "-D"]
```

Création de l'image et lancement du container

docker build -t ssh-server .

```
[root@debdocker:/home/jordan/docker-ssh# docker build -t ssh-server .]
```

```
Successfully built a7dfbebd80b0
Successfully tagged ssh-server:latest
```

**docker run -d -p 2222:2222 --name test-ssh ssh-server
port machine hote: port container**

```
[root@debdocker:/home/jordan/docker-ssh# docker run -d -p 2222:2222 --name test-ssh ssh-server
f611f5c46a2f87637b7fchdaf56201cacfecc1b98389c2d0e4df380eahdbbh8e5]
```

Test du ssh avec l'utilisateur root

ssh root@localhost -p 2222

```
root@debdocker:/home/jordan/docker-ssh# ssh root@localhost -p 2222
The authenticity of host '[localhost]:2222 ([::1]:2222)' can't be established.
ED25519 key fingerprint is SHA256:py+xM0wgJcdTynhy1H5Wxlv3tXx1/1OD8P20Tvq8MdU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '[localhost]:2222' (ED25519) to the list of known hosts.
root@localhost's password:
Linux f611f5c46a2f 6.1.0-32-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.129-1 (2025-03-06) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@f611f5c46a2f:~#
```

```
root@f611f5c46a2f:~# hostname
f611f5c46a2f
root@f611f5c46a2f:~# exit
logout
Connection to localhost closed.
root@debdocker:/home/jordan/docker-ssh# docker ps
CONTAINER ID   IMAGE      COMMAND       CREATED     STATUS          PORTS
f611f5c46a2f   ssh-server "/usr/sbin/sshd -D"  3 minutes ago   Up 3 minutes   0.0.0.0:2222->2222/tcp, :::2222->2222/tcp
root@debdocker:/home/jordan/docker-ssh#
```

Job 05

Tout informaticien étant « flemmard », il faut faire des alias pour les commandes docker en CLI , à mettre dans `~/.bashrc` pour manipuler les images / containers.

Modification du `bashrc` pour créer les alias

nano `~/.bashrc`

```
GNU nano 7.2                                .bashrc

esac

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
    alias ls='ls --color=auto'
    alias dir='dir --color=auto'
    alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# colored GCC warnings and errors
#export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01:quote=01'

# some more ls aliases
alias ll='ls -l'
alias la='ls -A'
alias l='ls -CF'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

# Docker alias
alias d='docker'
alias dps='docker ps'
```

Test

Exemple avec la commande `docker ps` et son alias `dps`

```
Jordan@debdocker:~$ docker ps
CONTAINER ID   IMAGE      COMMAND       CREATED      STATUS      PORTS          NAMES
f611f5c46a2f   ssh-server " /usr/sbin/sshd -D"  24 minutes ago   Up 24 minutes  0.0.0.0:2222->2222/tcp, :::2222->2222/tcp   test-ssh
Jordan@debdocker:~$ dps
CONTAINER ID   IMAGE      COMMAND       CREATED      STATUS      PORTS          NAMES
f611f5c46a2f   ssh-server " /usr/sbin/sshd -D"  24 minutes ago   Up 24 minutes  0.0.0.0:2222->2222/tcp, :::2222->2222/tcp   test-ssh
Jordan@debdocker:~$
```

Job 06

Se renseigner sur l'utilisation de volumes entre deux conteneurs et la gestion des volumes.

Volume Docker : En termes simples, un volume Docker est un répertoire ou un espace de stockage dédié qui peut être partagé entre les conteneurs et persiste même lorsque le conteneur associé est arrêté ou supprimé. Cette fonctionnalité fait des volumes un outil essentiel pour la gestion des données dans un environnement Docker.

C'est quoi un volume commun ?

Un volume commun, c'est un volume partagé entre plusieurs conteneurs.
Dans le cadre de l'exercice on veut :

Le conteneur FTP reçoit le fichier index.html via FileZilla.

Le conteneur nginx affiche ce fichier dans le navigateur.

Pour que les deux accèdent au **même fichier**, on doit créer un **volume commun monté dans les deux conteneurs, dans le même dossier** (par exemple /usr/share/nginx/html).

<https://blog.stephane-robert.info/docs/conteneurs/moteurs-conteneurs/volumes/#:~:text=Les%20volumes%20Docker%20standards%20repr%C3%A9sentent,la%20portabilit%C3%A9%20et%20la%20s%C3%A9curit%C3%A9.>

Job 07

À l'aide d'un fichier yml , de docker-compose, faire deux conteneurs : nginx et FTP liés entre eux. Création d'un volume commun pour accéder au dossier web.

Créer sur votre PC un fichier index.html, dans ce fichier faites afficher votre nom/prénom). Installer FileZilla sur votre PC , se connecter en FTP sur le conteneur FTP pour envoyer le fichier index.html, et regarder le résultat.

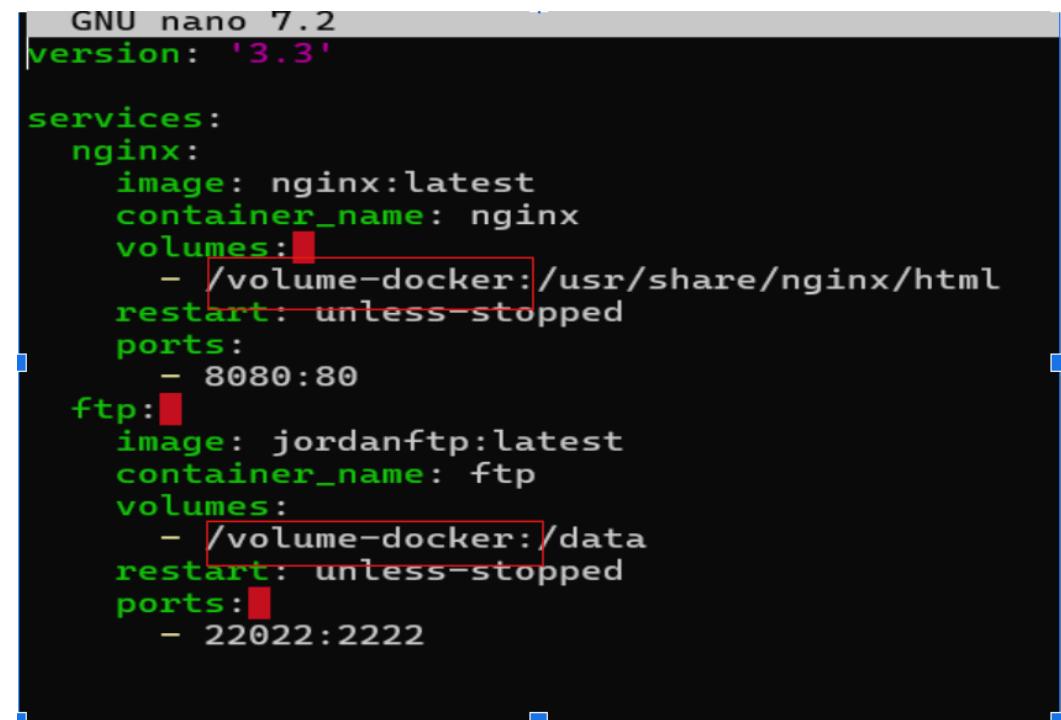
Docker Compose est un outil qui sert à **définir et gérer plusieurs conteneurs Docker en une seule commande** à l'aide d'un fichier de configuration

Installation Docker Compose :

```
sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o
/usr/local/bin/docker-compose
```

```
sudo chmod +x /usr/local/bin/docker-compose
```

Création du fichier yml pour le job07



```
GNU nano 7.2
version: '3.3'

services:
  nginx:
    image: nginx:latest
    container_name: nginx
    volumes:
      - /volume-docker:/usr/share/nginx/html
    restart: unless-stopped
    ports:
      - 8080:80
  ftp:
    image: jordanftp:latest
    container_name: ftp
    volumes:
      - /volume-docker:/data
    restart: unless-stopped
    ports:
      - 22022:2222
```

docker-compose up -d

```
root@debdocker:/home/jordan/job07# docker-compose up -d
Creating network "job07_default" with the default driver
Creating volume "job07_web_data" with default driver
Pulling nginx (nginx:latest)...
latest: Pulling from library/nginx
6e909acdb790: Already exists
5eaa34f5b9c2: Pull complete
417c4bccf534: Pull complete
e7e0ca015e55: Pull complete
373fe654e984: Pull complete
97f5c0f51d43: Pull complete
c22eb46e871a: Pull complete
Digest: sha256:124b44bfc9ccdf3cedf4b592d4d1e8bddb78b51ec2ed5056c52d3692baebc19
Status: Downloaded newer image for nginx:latest
Pulling ftp (fauria/vsftpd)...
latest: Pulling from fauria/vsftpd
2d473b07cdd5: Pull complete
33b95e46f70b: Pull complete
e22029c8d9a7: Pull complete
e1871c5d8fc9: Pull complete
c17c1255c529: Pull complete
ddcbab051542: Pull complete
1c68b0b593f1: Pull complete
dadbb66293c59: Pull complete
99a54b7a405b: Pull complete
200facf93d0a: Pull complete
16ecaf7d0305: Pull complete
Digest: sha256:6d71d7c7f1b0ab2844ec7dc7999a30aef6d758b6d8179cf5967513f87c79c177
Status: Downloaded newer image for fauria/vsftpd:latest
Creating ftp_server ... done
Creating nginx_web ... done
root@debdocker:/home/jordan/job07# |
```

Vérification

docker inspect nginx_web | grep -A 10 Mounts

```
root@debdocker:/home/jordan/job07# docker inspect nginx_web | grep -A 10 Mounts
"Mounts": [
    {
        "Type": "volume",
        "Name": "job07_web_data" [
        "Source": "/var/lib/docker/volumes/job07_web_data/_data",
        "Destination": "/usr/share/nginx/html",
        "Driver": "local",
        "Mode": "rw",
        "RW": true,
        "Propagation": ""
    }
]
```

docker inspect ftp_server | grep -A 10 Mounts

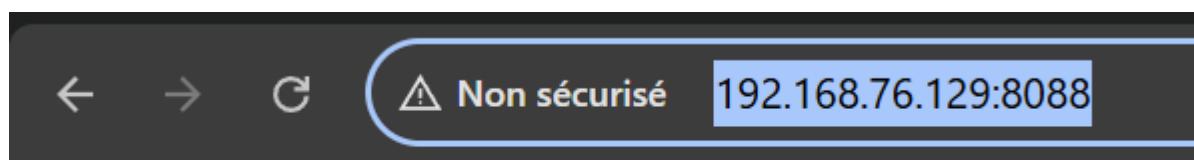
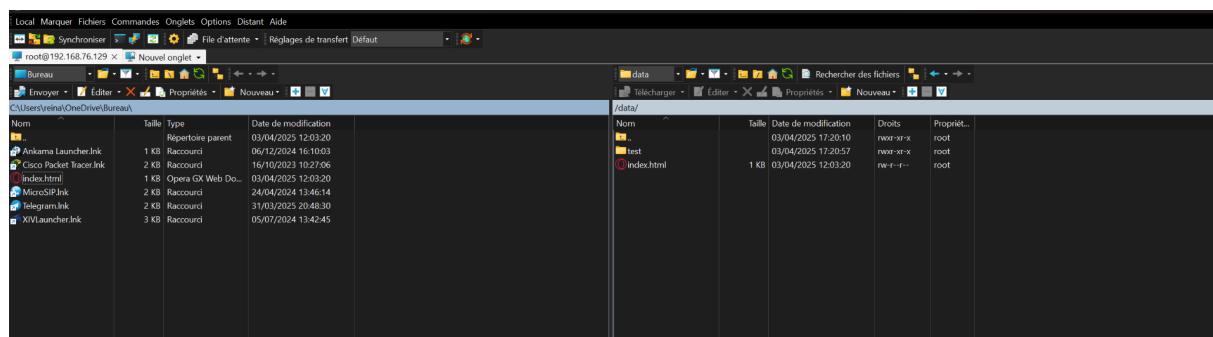
```
root@debdocker:/home/jordan/job07# docker inspect ftp_server | grep -A 10 Mounts
    "Mounts": [
      {
        "Type": "volume",
        "Name": "job07_web_data",
        "Source": "/var/lib/docker/volumes/job07_web_data/_data",
        "Destination": "/home_vsftpd",
        "Driver": "local",
        "Mode": "rw",
        "RW": true,
        "Propagation": ""
      }
    ]
}
```

docker ps

```
jordan@debdocker:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
9f516b4844be nginx:latest "/docker-entrypoint..." 48 minutes ago Up 48 minutes 0.0.0.0:8080->80/tcp, :::8080->80/tcp nginx
169b4078b7a3 jordanftp:latest "/usr/sbin/sshd -D" 48 minutes ago Up 48 minutes 0.0.0.0:22022->2222/tcp, :::22022->2222/tcp ftp
jordan@debdocker:~$
```

Test du transfert de fichier :

WIN



Bienvenue, Jordan !

Job 08

Sans utiliser une image nginx existante , utilisation de « Dockerfile » pour créer un conteneur nginx, voir redirection des ports, et se connecter.

Création dossier job08 et fichier essentiels

default.conf

```
GNU nano 7.2                                            default.conf
server {
    listen 80;
    server_name localhost;

    location / {
        root /usr/share/nginx/html;
        index index.html;
    }
}
```

Dockerfile

```
GNU nano 7.2                                            Dockerfile
FROM debian:bullseye

RUN apt-get update && \
    apt-get install -y nginx && \
    apt-get clean

# Supprimer la config par défaut
RUN rm /etc/nginx/sites-enabled/default

# Copier ta config à la bonne place
COPY default.conf /etc/nginx/sites-enabled/default
COPY index.html /usr/share/nginx/html/index.html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

docker-compose-yml

```
GNU nano 7.2
version: '3.3'

services:
  nginx_custom:
    build: .
    ports:
      - "8088:80"
    container_name: nginx_custom_job08
```

index.html

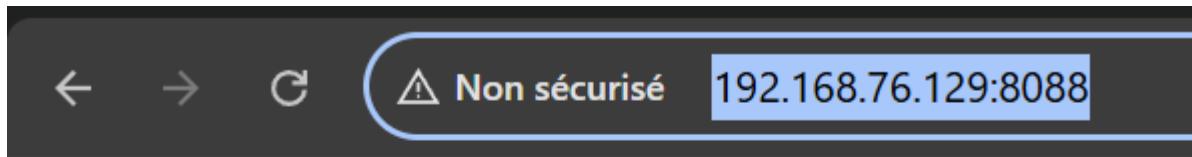
```
GNU nano 7.2                                            index.html
<!DOCTYPE html>
<html>
<head>
    <title>Job 08</title>
</head>
<body>
    <h1>Bienvenue, Jordan !</h1>
</body>
</html>
```

Construction de l'image et création des containers
docker-compose build --no-cache
docker-compose up -d

Vérification

docker ps

Jordan@debdocker:~/job08\$ docker ps							NAMES
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS		
1609792d936	job08_nginx_custom	"nginx -g 'daemon off;'	6 minutes ago	Up 6 minutes	0.0.0.0:8088->80/tcp, :::8088->80/tcp		nginx_custom_job08



Bienvenue, Jordan !

Méthode Dockerfile simple

Création du dossier et du Dockerfile :

```
FROM debian:bookworm-slim

RUN apt-get update && apt-get install -y nginx && apt-get clean

# Modifier la conf nginx pour qu'il écoute sur 7500
RUN sed -i 's/listen 80 default_server;/listen 7500 default_server;/' /etc/nginx/sites-available/default && \
    sed -i 's/listen \[::\]:80 default_server;/listen \[:\]:7500 default_server;/' /etc/nginx/sites-available/default

EXPOSE 7500

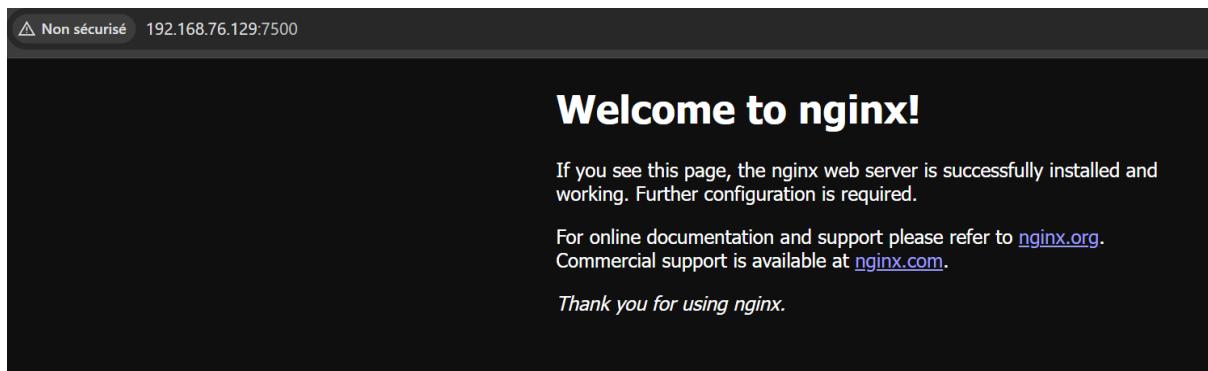
CMD ["nginx", "-g", "daemon off;"]
```

build et lancement de l'image :

```
docker build -t nginx-custom .
docker run -d -p 7500:7500 --name mon-nginx nginx-custom
```

```
jordan@debdocker:~/nginx-custom$ docker build -t nginx-custom .
Sending build context to Docker daemon 2.048kB
Step 1/5 : FROM debian:bookworm-slim
--> 59429810452a
Step 2/5 : RUN apt-get update && apt-get install -y nginx && apt-get clean
--> Running in 018eb1a5137d
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8792 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [512 B]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [258 kB]
Fetched 9305 kB in 2s (5301 kB/s)
Reading package lists...
Reading package lists...
Building dependency tree...
Reading state information...
The following packages were automatically installed and are no longer required:
  libcurl4-openssl-dev libcurl4-openssl-dev:i386 libcurl4-openssl-dev:amd64
Use 'apt autoremove' to remove them.
Successfully built 49a4dfd7c533
Successfully tagged nginx-custom:latest
jordan@debdocker:~/nginx-custom$ docker run -d -p 7500:7500 --name mon-nginx nginx-custom
a2da5773dc0c1b57a198c5bd54f3a787c
```

test :



Job09

Création et utilisation d'un "registry" local. Et ajouter une < UI > pour le gérer depuis une interface web.

Création dossier du job et de l'espace de stockage

mkdir dossier du job

mkdir data (espace stockage pour image du registry qui servira de volume)

Création fichier yml

fichier yml :

```
GNU nano 7.2                                            docker-compose.yml

version: '3.3'

services:
  registry:
    image: registry:2
    container_name: registry-server
    ports:
      - "5000:5000"
    volumes:
      - ./data:/var/lib/registry
    environment:
      REGISTRY_STORAGE_DELETE_ENABLED: "true"
      REGISTRY_HTTP_HEADERS_Access-Control-Allow-Origin: '[["http://192.168.76.129:8081"]'
      REGISTRY_HTTP_HEADERS_Access-Control-Allow-Methods: '[["GET", "OPTIONS", "PUT", "DELETE"]'
      REGISTRY_HTTP_HEADERS_Access-Control-Allow-Headers: '[["Authorization", "Accept"]'

  registry-ui:
    image: joxit/docker-registry-ui:latest
    container_name: registry-ui
    ports:
      - "8081:80"
    depends_on:
      - registry
    environment:
      REGISTRY_TITLE: "Registry Local Privé"
      SINGLE_REGISTRY: "true"
      REGISTRY_NAME: "LocalRegistry"
      DELETE_IMAGES: "true"
      REGISTRY_SECURED: "false"
      REGISTRY_URL: "http://192.168.76.129:5000"
```

explication du fichier yml :

REGISTRY_STORAGE_DELETE_ENABLED : autorise la suppression des images du registre (par défaut, c'est désactivé).

REGISTRY_HTTP_HEADERS_Access-Control-Allow-Origin : autorise l'accès au registre depuis l'interface web à l'adresse spécifiée (nécessaire pour éviter les erreurs CORS).

REGISTRY_HTTP_HEADERS_Access-Control-Allow-Methods : définit les méthodes HTTP autorisées pour l'UI (lecture, envoi, suppression...).

REGISTRY_HTTP_HEADERS_Access-Control-Allow-Headers : indique les en-têtes HTTP acceptés (ex : Authorization, Accept), utilisés dans les requêtes de l'interface.

REGISTRY_TITLE: nom affiché dans l'interface.

SINGLE_REGISTRY: un seul registre géré.

DELETE_IMAGES: autorise la suppression via l'UI.

REGISTRY_SECURED: ici false → pas de HTTPS.

REGISTRY_URL: adresse locale du registry (doit correspondre à l'IP et port exposé).

docker tag job08_nginx_custom localhost :5000/job08_nginx_custom :latest

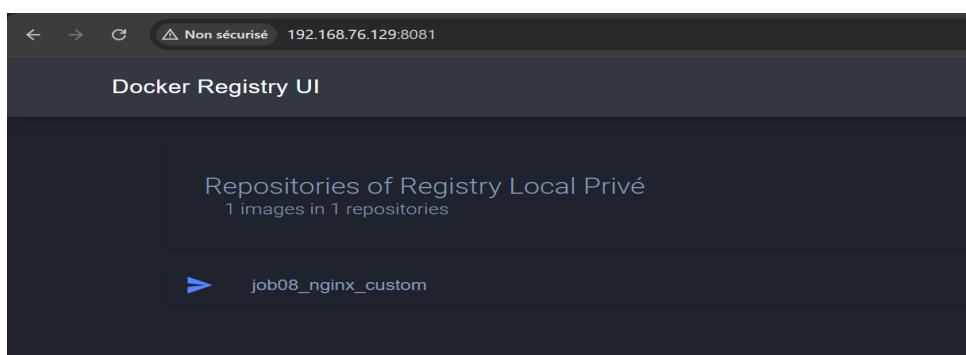
```
jordan@debdocker:~/job09$ docker push localhost:5000/job08_nginx_custom:latest
The push refers to repository [localhost:5000/job08_nginx_custom]
f77aaece06d9: Pushed
14bd1175ce8d: Pushed
afffaf525a517: Pushed
ebb8067cc083: Pushed
3dfea33954f7: Pushed
latest: digest: sha256:28d42840905d0b5757c3bb6049f0397bdac35bd084b59959edd2853ee49e6ec2 size: 1362
```

explication commande :

docker tag : prépare l'image à être poussée vers un registre

docker push : envoie l'image dans le registre docker local

Test :



Job 10

Faire un script bash , pour effacer totalement docker (les images , volumes ,conteneurs, et paquets correspondants) et rendre le système propre.

Et aussi un script pour automatiser l'installation de docker sur une machine Debian.

Fichier de désinstallation

```
GNU nano 7.2
#!/bin/bash

echo "✓ Suppression complète de Docker..."

# 🔴 Arrêt des conteneurs s'ils existent
docker ps -aq >/dev/null && docker stop $(docker ps -aq) 2>/dev/null
docker system prune -a -f --volumes

# 📈 Suppression explicite de docker.io si installé depuis les dépôts Debian
apt-get purge -y docker.io || true

# 📈 Suppression d'autres variantes Docker (au cas où)
apt-get purge -y docker-ce docker-ce-cli containerd containerd.io docker-buildx-plugin docker-compose-plugin || true

# 💡 Nettoyage des dépendances
apt-get autoremove -y
apt-get clean

# 🔴 Suppression manuelle des fichiers liés à Docker
rm -rf /var/lib/docker
rm -rf /var/lib/containerd
rm -rf /etc/docker
rm -rf /etc/systemd/system/docker.service.d
rm -rf /usr/local/bin/docker*
rm -rf /usr/bin/docker*
rm -rf ~/.docker

echo -e "\n✅ Docker et tous ses composants ont été supprimés proprement."
```

sudo sh uninstall.sh

```
ordan@debdocker:~/job10$ which docker
ordan@debdocker:~/job10$ dpkg -l | grep docker
ordan@debdocker:~/job10$
```

Fichier d'installation

```
GNU nano 7.2                                         install_docker_debian.sh

#!/bin/bash

echo "💡 Installation de Docker depuis les dépôts officiels Debian..."

# Mise à jour de la liste des paquets
sudo apt update

# Installation du paquet docker.io fourni par Debian
sudo apt install -y docker.io

# Activation du service Docker au démarrage
sudo systemctl enable docker

# Démarrage immédiat du service
sudo systemctl start docker

# Affichage de la version installée
echo -e "\n✓ Docker a été installé avec succès :"
docker --version

# Test rapide (facultatif)
echo -e "\n● Test rapide : docker run hello-world"
sudo docker run hello-world
```

```
✓ Docker a été installé avec succès :  
Docker version 20.10.24+dfsg1, build 297e128  
-e  
🌐 Test rapide : docker run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
e6590344b1a5: Pull complete  
Digest: sha256:c41088499908a59aae84b0a49c70e86f4731e588a737f1637e73c8c09d995654  
Status: Downloaded newer image for hello-world:latest  
  
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/
```

Job11

Découverte de portainer.

Refaire les Job 2 à 9 , en utilisant l'interface graphique de portainer. Se renseigner sur les alternatives à Portainer.

Création du volume et lancement du container Portainer

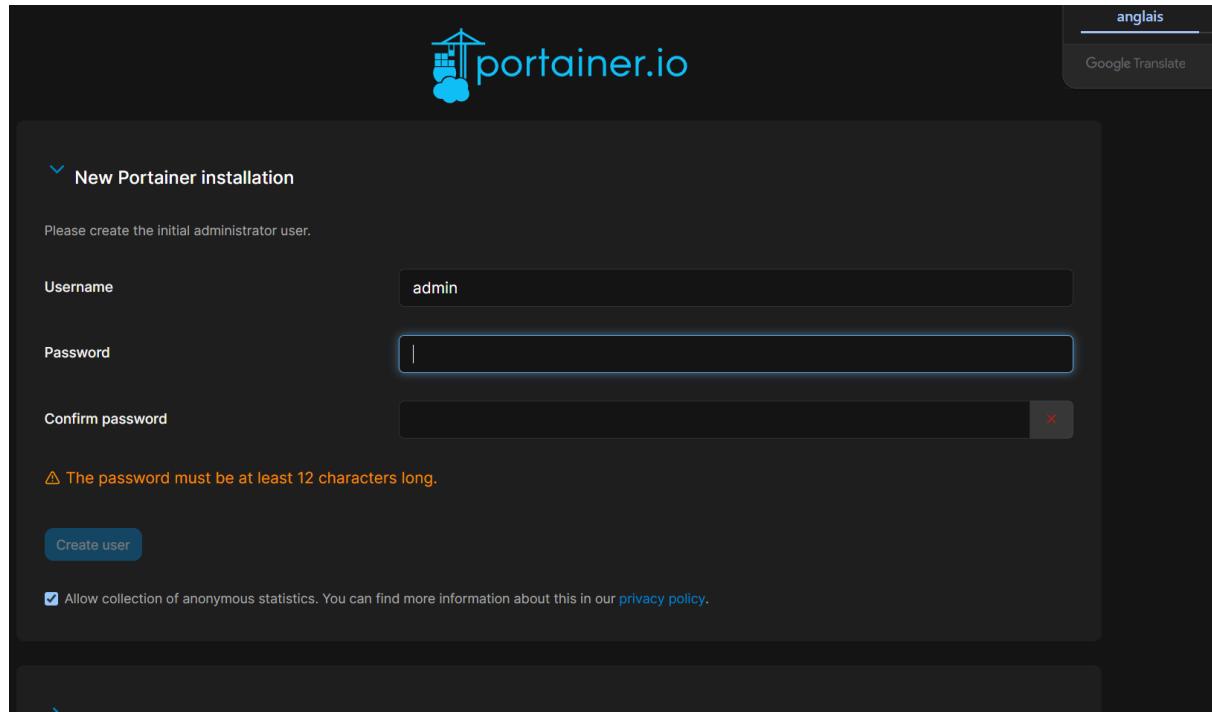
docker volume create portainer_data

```
docker run -d -p 9000:9000 -p 9443:9443 --name portainer  
--restart=always -v /var/run/docker.sock:/var/run/docker.sock -v  
portainer_data:/data portainer/portainer-ce
```

```
jordan@debdocker:~/job10$ docker run -d -p 9000:9000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce  
Unable to find image 'portainer/portainer-ce:latest' locally  
latest: Pulling from portainer/portainer-ce  
6cled96ce20: Pull complete  
c0b075a95f6: Pull complete  
84de093ad5ed: Pull complete  
a9fffa7abff372: Pull complete  
e09d4e560353: Pull complete  
0a930402c29: Pull complete  
8d7f7a6683c39: Pull complete  
df1dbbf4f985: Pull complete  
63bfeef079ce: Pull complete  
454fb7000ff54: Pull complete  
Digest: sha256:5f3f03a7d2e34fe836837f00183aa52ba1e911c9f51b9ef359aa5651a474ff5b  
Status: Downloaded newer image for portainer/portainer-ce:latest  
9a40d0b1cc5eale5a534ede137c4a44ca5a60893b39fa79d8e8c0613dac8b0ab
```

```
jordan@debdocker:~/job10$ docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
9a40d0b1cc5e portainer/portainer-ce "/portainer" 50 seconds ago Up 48 seconds 0.0.0.0:9000->9000/tcp, 8000/tcp, 0.0.0.0:9443->9443/tcp, 0.0.0.0:9443->9443/tcp portainer  
jordan@debdocker:~/job10$ |
```

Création de l'Admin Portainer



 portainer.io
COMMUNITY EDITION

Home

Environment: / None selected

Administration

- User-related
- Environment-related
- Registries
- Logs
- Notifications
- Settings

Environment Wizard

Quick Setup ⚙

Environment Wizard

Welcome to Portainer

We have connected your local environment of Docker to Portainer.

Get started below with your local portainer or connect more container environments.



Get Started

Proceed using the local environment which Portainer is running in



Add Environments

Connect to other environments

Job 2 version portainer

On clique sur local

The screenshot shows the Portainer Environments interface. At the top, there's a search bar and a refresh button. Below that is a filter bar with dropdowns for Platform, Connection Type, Status, Tags, Groups, Agent Version, and a 'Clear all' button. A 'Sort By' dropdown and an upward arrow are also present. The main list contains one item: 'local' (Up, Standalone), created on 2025-05-03 18:20:20, with the path /var/run/docker.sock. It has 0 stacks, 2 containers (1 running, 1 stopped), 1 volume, 2 images, 1 CPU, and 970.8 MB RAM. A 'Connected' status indicator is shown. At the bottom right, there are buttons for 'Disconnect' and 'Edit'. The bottom of the screen shows pagination controls for 'Items per page' (set to 10) and an upward arrow.

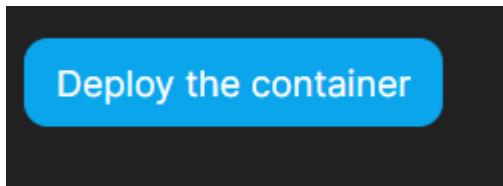
The screenshot shows the Portainer Container dashboard. It features a large blue circle icon with a white cube inside, followed by the number '2' and the text 'Containers'. To the right, it displays metrics: 1 running container (green), 1 stopped container (red), 0 healthy containers (blue), and 0 unhealthy containers (orange). The background is dark blue.

The screenshot shows the Portainer Container list interface. At the top, there's a search bar and various action buttons like Start, Stop, Kill, Restart, Pause, Resume, and Remove. A '+ Add container' button is also visible. The main table lists two containers: 'pedantic_napier' (Status: exited - code 0, Image: hello-world, Created: 2025-05-03 18:02:33, IP Address: -, Published Ports: -, Ownership: administrators) and 'portainer' (Status: running, Image: portainer/portainer-ce, Created: 2025-05-03 18:15:18, IP Address: 172.17.0.2, Published Ports: 9000-9000, 9443-9443, Ownership: administrators). The bottom of the screen shows pagination controls for 'Items per page' (set to 10) and an upward arrow.

On clique sur sur container add container et on cherche l'image officielle de hello-world présente dans dockerhub

The screenshot shows the Portainer Container creation dialog for a container named 'hello-world'. The 'Image Configuration' section includes a 'Registry' dropdown set to 'Docker Hub (anonymous)', an 'Image' dropdown set to 'docker.io/hello-world:latest', and an 'Advanced mode' toggle switch. A note indicates that anonymous pulls are limited to 100 per 6 hours. The 'Webhook' section has a 'Create a container webhook' toggle switch and a 'Business Feature' note. The 'Network ports configuration' section has a 'Publish all exposed ports to random host ports' toggle switch. The 'Port mapping' section has a '+ Map additional port' button. The 'Access control' section has an 'Enable access control' toggle switch. At the bottom, there are two buttons: 'Administrators' (selected) and 'Restricted', each with a note about restricting management to specific users or teams.

On déploie le container



On peut voir qu'il est maintenant bien présent, ce qui n'était pas le cas avant

Name	State	Quick Actions	Stack	Image	Created	IP Address	Published Ports	Ownership
hello-world	exited - code 0	🔍 ⚡	-	hello-world:latest	2025-05-03 18:28:06	-	-	administrators
pedantic_napier	exited - code 0	🔍 ⚡	-	hello-world	2025-05-03 18:02:33	-	-	administrators
portainer	running	🔍 ⚡ ⚡	-	portainer/portainer-ce	2025-05-03 18:15:18	172.17.0.2	9000:9000, 9443:9443	administrators

Vérification que le hello-world s'est bien lancé via les logs du container

Containers > hello-world > Logs

Container logs

Log viewer settings

Auto-refresh logs

Wrap lines

Display timestamps

Fetch: All logs

Search: Filter...

Lines: 100

Actions: Download logs, Copy, Copy selected lines, Unselect

```
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/
```

Job 03 version portainer

Construction de l'image

The screenshot shows the Portainer.io interface. On the left, there's a sidebar with various navigation options like Home, Dashboard, Templates, Stacks, Containers, and Images. The 'Images' option is selected and highlighted with a red box. The main area is titled 'Image list' and contains a 'Pull image' section where you can search for Docker images from Docker Hub. Below this is a table of existing images, also with a red box around the 'Build a new image' button at the top right of the table.

Création de l'image avec le Dockerfile

This screenshot shows the Dockerfile editor within Portainer. It has three tabs: 'Web editor' (which is active and highlighted with a red box), 'Upload', and 'URL'. The 'Web editor' tab displays a code editor with the following Dockerfile content:

```
1 FROM debian:bookworm-slim
2 CMD echo "Hello World from Dockerfile!"
```

Finalisation de la construction de l'image

The screenshot shows the final step in the Portainer interface. It features a large blue button labeled 'Build the image' which is highlighted with a red box. Above it, the word 'Actions' is displayed in a large font.

Build image

Builder

Output

```
Step 1/2 : FROM debian:bookworm-slim
```

```
---> 59429810452a
```

```
Step 2/2 : CMD echo "Hello World from Dockerfile!"
```

```
---> Running in 41324f076047
```

```
Removing intermediate container 41324f076047
```

```
---> b1cd18815591
```

```
Successfully built b1cd18815591
```

```
Successfully tagged hello_world_custom:latest
```

Création du container avec l'image custom build précédemment

Name

Image Configuration

Registry

Image
Search

Advanced mode

Always pull the image (?)

You are currently using an anonymous account to pull images from DockerHub and will be limited to 100 pulls every 6 hours. You can configure DockerHub authentication in the [Registries View](#). Remaining pulls: 96/100

Webhook

Create a container webhook (?) Business Feature

Network ports configuration

Published all exposed ports to random host ports

Port mapping (?)

Access control

Enable access control

image montrant que l'image avait bien été build

The screenshot shows the Portainer.io interface, specifically the 'Images' section. On the left, there's a sidebar with various navigation options like Home, Dashboard, Templates, Stacks, Containers, Images (which is currently selected), Networks, Volumes, Events, and Host. Below that is an 'Administration' section with User-related, Environment-related, and other options. The main area is titled 'Image list' and has a 'Pull image' section where you can select a registry (Docker Hub anonymous) and search for an image. Below this is a table of images:

ID	Tags	Size	Created
sha256:59429810452a41b3c3c45ec3ad270a...	debian:bookworm-slim	74.8 MB	2025-04-28 02:00:00
sha256:74cc54e27dc41bb10dc4b2226072d4...	hello-world:latest	10.1 kB	2025-01-22 00:32:32
sha256:b1cd188155910f1314fa27c9e71eb...	hello_world_custom:latest	74.8 MB	2025-05-03 18:40:06
sha256:d3e9f39af9c561cd919998ef6c7ea1...	portainer/portainer-ce:latest	268.2 MB	2025-05-02 00:50:21

There are buttons for Search, Remove, Import, Export, and Build a new image. The bottom right shows 'Items per page' set to 10.

Job 04 version portainer

On va dans image

The screenshot shows the Portainer.io interface with the 'Images' tab selected in the sidebar. The main area displays a table of images from Docker Hub (anonymous). The table has columns for Id, Tags, Size, and Created. One image is marked as 'Unused'. A search bar and various management buttons like Remove, Import, Export, and Build a new image are visible at the top of the table.

Id	Tags	Size	Created
sha256:59429810452a41b3c3c45ec3ad270e...	debian:bookworm-slim	74.8 MB	2025-04-28 02:00:00
sha256:74cc54e27dc41bb10dc4b2226072d4...	hello-world:latest	101 kB	2025-01-22 00:32:32
sha256:b1cd188155910f1314faf27c9e71eb...	hello_world_custom:latest	74.8 MB	2025-05-03 18:40:06
sha256:d3e9f30af9c561cd919998ef6c7ea1...	portainer/portainer-ce:latest	268.2 MB	2025-05-02 00:50:21

Ecriture du Dockerfile

The screenshot shows the 'Web editor' section of Portainer.io where a Dockerfile is being written. The code is as follows:

```
1 FROM debian:bookworm-slim
2
3 # MAJ + installation SSH
4 RUN apt-get update && apt-get install -y openssh-server && apt-get clean
5
6 # MDP pour root
7 RUN echo "root:root123" | chpasswd
8
9 # Création repertoire SSH
10 RUN mkdir /var/run/sshd
11
12 # Permettre l'accès root via SSH
13 RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
14
15 # Changer le port SSH
16 RUN sed -i 's/#Port 22/Port 2222/' /etc/ssh/sshd_config
17
18 # Exposer le port 2222
19 EXPOSE 2222
20
21 # Commande de démarrage du service SSH
22 CMD ["/usr/sbin/sshd", "-D"]
```

Création du container

The screenshot shows the Portainer.io interface for creating a new Docker container. The left sidebar shows the navigation menu with 'Containers' selected. The main panel is titled 'Create container' and has the following configuration:

- Name:** ssh_test
- Image Configuration:**
 - Registry:** Docker Hub (anonymous)
 - Image:** docker.io/ssh_server_custom:latest
- Always pull the image:** Enabled (blue switch)
- Webhook:** Create a container webhook (disabled)
- Network ports configuration:** Publish all exposed ports to random host ports (Enabled)
- Port mapping:** Host 2222 mapped to Container 2222 (Protocol: TCP)
- Access control:** Enabled access control (disabled)

test de la connexion avec l'utilisateur root

```
Microsoft Windows [version 10.0.26100.3775]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\reina>ssh root@192.168.76.129 -p 2222
ssh: connect to host 192.168.76.129 port 2222: Connection refused

C:\Users\reina>ssh root@192.168.76.129 -p 2222
The authenticity of host '[192.168.76.129]:2222 ([192.168.76.129]:2222)' can't be established.
ED25519 key fingerprint is SHA256:v2Lr9KZ7+e2KW20S8mmnihL0KsefmA0Or8nqf9Kqpc8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? Y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '[192.168.76.129]:2222' (ED25519) to the list of known hosts.
root@192.168.76.129's password:
Linux b7648dbe5c0a 6.1.0-32-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.129-1 (2025-03-06) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@b7648dbe5c0a:~# |
```

Job7 version portainer

Fichier yml dans stacks avec utilisation de l'image ssh du job 04

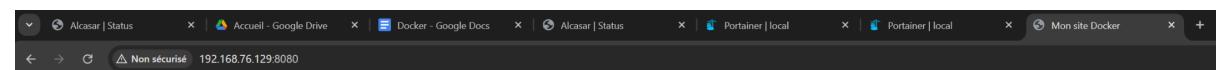
```
① Define or paste the content of your docker compose file here

1  version: "3.3"
2
3  services:
4    nginx:
5      image: nginx:latest
6      container_name: nginx
7      volumes:
8        - /home/jordan/commondir:/usr/share/nginx/html
9      restart: unless-stopped
10     ports:
11       - "8080:80"
12
13    ftp:
14      image: ssh_server_custom:latest
15      container_name: ftpjordan
16      volumes:
17        - /home/jordan/commondir:/data
18      restart: unless-stopped
19      ports:
20        - "2222:2222"
21
```

test avec WinSCP



Vérification que l'index.html de base a bien été remplacé par notre fichier transfert



Jordan / Reinaldo ðŸ˜Ž

Job 8 version portainer

Apache version

fichier github pour le stack de portainer

The screenshot shows a GitHub repository named 'portainerjob08' created by 'jordan-reinaldo'. The repository has 1 branch and 0 tags. It contains the following files:

- index.php (Create)
- www (Create)
- Dockerfile (Add files via upload)
- docker-compose.yml (Add files via upload)

A README file is present but empty. The repository has 2 commits, 0 stars, 1 watching, and 0 forks. It includes sections for About, Releases, Packages, and Languages.

Dockerfile

```
FROM php:8.1-apache
RUN a2enmod rewrite
COPY www/ /var/www/html/
EXPOSE 80
```

docker-compose

```
> Users > reina > Downloads > job08-apache > docker-compose.yml
1  version: '3.3'
2
3  services:
4    apache_web:
5      build: .
6      container_name: apache_job08
7      ports:
8        - "9700:80"
9      volumes:
10     - ./www:/var/www/html"
```

index.php

```
<?php  
phpinfo();  
?>
```

On utilise le repo github où l'on a placé nos fichiers pour faire notre stack

Create stack

Name: e.g. mystack

This stack will be deployed using docker compose.

Build method:

- Web editor
- Upload
- Repository (selected)
- Custom template

Git repository

Authentication

You can use the URL of a git repository.

Repository URL: <https://github.com/jordan-reinaldo/portainerjob01>

Skip TLS Verification

Specify a reference of the repository using the following syntax: branches with `refs/heads/branch_name` or tags with `refs/tags/tag_name`. If not specified, will use the default `HEAD` reference normally the `main` branch.

Repository reference: `refs/heads/main`

Compose path: `docker-compose.yml`

Additional paths

+ Add file

Test

PHP Version 8.1.32

System: Linux 9615b69daef7 6.1.0-32-ams64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.129-1 (2025-03-06) x86_64

Build Date: Apr 28 2025 21:52:20

Build System: Linux - Docker

Build Provider: https://github.com/docker/library/php

Configure Command: ./configure --build=slim-54-armv7p --with-config-file-path=/usr/local/etc/php --with-config-file-scan-dir=/etc/php --enable-option-checking=fatal --with-mhash --with-pcre --enable-mp --enable-mbstring --enable-xml --enable-option-checking=fatal --with-mhash --with-pcre --enable-mp --enable-mbstring --enable-xml --with-password-argon2 --with-sodium=shared --with-pdo-sqlite=usr --with-pdo-sqlite=lib --with-iconv --with-openpfd --with-readline --with-zlib --enable-pprint --with-pear --with-libxml=libxml --with-imap=gn --enable-cgi --with-apc2 --build_alias=x86_64-linux-gnu

Server API: Apache 2.0 Handler

Virtual Directory Support: disabled

Configuration File (php.ini) Path: /usr/local/etc/php

Loaded Configuration File: (none)

Scan this dir for additional .ini files: /usr/local/etc/php/conf.d

Additional .ini files parsed: /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini

PHP API: 20210902

PHP Extension: 20210902

Zend Extension: 420210902

Zend Extension Build: API20210902.NTS

PHP Extension Build: API20210902.NTS

Debug Build: no

Thread Safety: disabled

Zend Signal Handling: enabled

Zend Memory Manager: enabled

Zend MultiByte Support: provided by mbstring

Zend Max Execution Timers: disabled

IPv6 Support: enabled

DTrace Support: disabled

Registered PHP Streams: https, ftps, compress,zlib,php,file,glob,data,http,ftp,phar

Registered Stream Socket Transports: tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3

Registered Stream Filters: zlib*, convert.iconv*, string.rot13, string.toupper, string.tolower, convert*, consumed, dechunk

This program makes use of the Zend Scripting Language Engine:
Zend Engine v4.1.32 Copyright (C) Zend Technologies

zend+engine

Configuration

apache2handler

Apache Version	Apache/2.4.62 (Debian)
Apache API Version	20120211
Server Administrator	webmaster@localhost
Hostname:Port	172.23.0.90:443
User/Group	www-data/www-data
Max Requests	Per Child 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	Yes
Server Root	/etc/apache2
Loaded Modules	core mod_so mod_watchdog mod_http_core mod_log_config mod_logio mod_version mod_mpm_common mod_access_compat mod_alias mod_authn_basic mod_authn_core mod_authz_core mod_authz_host mod_authz_user mod_autoindex mod_deflate mod_dir mod_env mod_filter mod_mime mod_prefork mod_negotiation mod_php mod_reqtimeout mod_rewrite mod_setenvif mod_status

Directive	Local Value	Master Value
allow_url_include	Off	Off

Nginx version :

Création de l'image

The screenshot shows the portainer.io interface. On the left, a sidebar lists various sections: Home, local, Dashboard, Templates, Stacks, Containers, Images (which is selected), Networks, Volumes, Events, Host, Administration, User-related, Environment-related, Registries, Logs, and Notifications. The main area is titled "Names" with a placeholder "e.g. my-image:my-tag". Below it, a note says: "A name must be specified in one of the following formats: `name:tag`, `repository/name:tag` or `registry:port/repository/name:tag` format. If you omit the tag the default `latest` value is assumed." A warning message below states: "The image name must consist of between 2 and 255 lowercase alphanumeric characters, '.', '_', or '-' (e.g., 'my-name', or 'abc-123').". Under "Build method", there are three options: "Web editor" (selected), "Upload", and "URL". The "Web editor" section contains a text input for a Dockerfile. The Dockerfile content is as follows:

```
1 FROM debian:bookworm-slim
2
3 RUN apt-get update && apt-get install -y nginx && apt-get clean
4
5 # Modifier la conf nginx pour qu'il écoute sur le port 7500
6 RUN sed -i '/listen 80 default_server;/ s/listen 80/listen [::]:7500/' /etc/nginx/sites-available/default &&
7     sed -i 's/listen [::]:7500 default_server;/ listen [::]:7500 default_server;/' /etc/nginx/sites-available/default
8
9 EXPOSE 7500
10
11 CMD ["nginx", "-g", "daemon off;"]
```

container créer

The screenshot shows a list of containers. One container is highlighted: "nginx_custom" (running). The status bar indicates: "nginx-custom:latest" (2025-05-05 13:22:33 | 172.17.0.3 | 7500:7500".

test

The screenshot shows a browser window displaying the Nginx welcome page. The address bar shows "192.168.76.129:7500". The page content is as follows:

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

Job 9 version portainer

On va dans stacks puis dans web editor on écrit notre docker-compose.yml

```
1  version: '3.3'
2
3  services:
4    registry:
5      image: registry:2
6      container_name: registry-server
7      ports:
8        - "5000:5000"
9      volumes:
10        - ./data:/var/lib/registry
11      environment:
12        REGISTRY_STORAGE_DELETE_ENABLED: "true"
13        REGISTRY_HTTP_HEADERS_Access-Control-Allow-Origin: '[["http://192.168.76.129:8081"]'
14        REGISTRY_HTTP_HEADERS_Access-Control-Allow-Methods: '[["GET", "OPTIONS", "PUT", "DELETE"]'
15        REGISTRY_HTTP_HEADERS_Access-Control-Allow-Headers: '[["Authorization", "Accept"]'
16
17    registry-ui:
18      image: joxit/docker-registry-ui:latest
19      container_name: registry-ui
20      ports:
21        - "8081:80"
22      depends_on:
23        - registry
24      environment:
25        REGISTRY_TITLE: "Registry Local Privé"
26
27        SINGLE_REGISTRY: "true"
28        REGISTRY_NAME: "LocalRegistry"
29        DELETE_IMAGES: "true"
30        REGISTRY_SECURED: "false"
31        REGISTRY_URL: "http://192.168.76.129:5000"
```

On déploie les containers



les deux containers ont bien été déployés

The screenshot shows the Docker UI interface. At the top, there is a table of deployed stacks:

Name	Status	Image	Created	IP Address	Published Ports	Ownership
registry-server	running	job09stacks registry:2	2025-05-03 22:34:58	172.19.0.2	5000:5000	administrators
registry-ui	running	job09stacks joxit/docker-registry-uisatest	2025-05-03 22:34:58	172.19.0.3	8081:80	administrators

Below this is the "Stack details" section for the "job09stacks" stack. It includes tabs for "Stack" (selected) and "Editor". Under "Stack details", there are buttons for "Stop this stack", "Delete this stack", and "Create template from stack". A "Stack duplication / migration" section allows users to duplicate or migrate the stack, with a dropdown menu for selecting a target stack and buttons for "Migrate" and "Duplicate".

At the bottom is the "Containers" section, which lists the same two containers with their respective details.

on tag puis on push une image en CLI pour voir si notre registry fonctionne

```
jordan@debdocker:~/job11$ docker push localhost:5000/helloworld-local
Using default tag: latest
The push refers to repository [localhost:5000/helloworld-local]
08000c18d16d: Pushed
latest: digest: sha256:a57366d044dd66a398ffb66fe4409205d15ee694f47faa43422c6e3410509f54 size: 528
jordan@debdocker:~/job11$ |
```

test

The screenshot shows the Docker Registry UI. At the top, it displays the URL "192.168.76.129:8081". The main page title is "Docker Registry UI". Below the title, it says "Repositories of Registry Local Privé" and "1 images in 1 repositories". A list item is shown with a blue arrow icon and the text "helloworld-local".

Pour aller plus loin

On complique un peu le système, il va falloir faire l'équivalent de XAMPP

Avec un fichier Dockerfile et un fichier yml faire :

→ un serveur nginx (ou apache) avec PHP

→ un serveur MariaDB/MySQL (avec phpMyAdmin)

→ un serveur FTP

→ un volume pour stocker l'ensemble des données communes aux différents serveurs

Tester le système.

Fichier docker-compose.yml avec serveur web, mariadb, phpmyadmin et ftp

```
GNU nano 7.2                                            docker-compose.yml

version: '3.3'

services:
  web:
    build: .
    container_name: web_php
    volumes:
      - ./www:/var/www/html
    ports:
      - "9500:80"
    depends_on:
      - db

  db:
    image: mariadb
    container_name: mariadb
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: xampp_db
    volumes:
      - db_data:/var/lib/mysql

  phpmyadmin:
    image: phpmyadmin
    container_name: phpmyadmin
    restart: always
    ports:
      - "9501:80"
    environment:
      PMA_HOST: db

  ftp:
    image: fauria/vsftpd
    container_name: ftp_server
    restart: always
    ports:
      - "9510:21"
      - "21100-21110:21100-21110"
    volumes:
      - ./www:/home/vsftpd
    environment:
```

```
environment:  
  - FTP_USER=user  
  - FTP_PASS=pass  
  - PASV_ENABLE=YES  
  - PASV_MIN_PORT=21100  
  - PASV_MAX_PORT=21110  
  - PASV_ADDRESS=192.168.76.129
```

```
volumes:  
  db_data:
```

Dockerfile

```
GNU nano 7.2  
FROM php:8.1-apache  
  
RUN a2enmod rewrite  
  
COPY www/ /var/www/html/  
  
EXPOSE 80
```

on fait un dossier www puis on crée un fichier index.php

```
jordan@debdocker:~/projet-xampp$ cd www  
jordan@debdocker:~/projet-xampp/www$ ls  
index.php user
```

```
GNU nano 7.2  
?php  
phpinfo();  
>
```

Création de l'image et lancement des containers

Docker-compose build -no-cache

```
jordan@debdocker:~/projet-xampp$ docker-compose build --no-cache
db uses an image, skipping
phpmyadmin uses an image, skipping
ftp uses an image, skipping
Building web
  Sending build context to Docker daemon  6.144kB
Step 1/5 : FROM php:8.1-fpm
8.1-fpm: Pulling from library/php
254e724d7786: Already exists
c65ab0511c0c: Pull complete
396ae0811dc1: Pull complete
08d68830cf18: Pull complete
dbd45828ee1d: Pull complete
66e5f13148ec: Pull complete
6f9c6ce5a07a: Pull complete
49fc91a0979c: Pull complete
f1f4ab0a9742: Pull complete
4f4fb700ef54: Pull complete
5280579f701f: Pull complete
Digest: sha256:10ea2d74309746fe72ded79a994dd28e3f6fc3aecd411b1d7ef2094006ae34
Status: Downloaded newer image for php:8.1-fpm
    ----> d776f258a86d
Step 2/5 : RUN apt-get update && apt-get install -y      nginx      supervisor      && apt-get clean
    ----> Running in ea8fc7fdff09
|
```

docker-compose up -d

```
jordan@debdocker:~/projet-xampp$ docker-compose up -d
Creating network "projet-xampp_default" with the default driver
Creating volume "projet-xampp_db_data" with default driver
Pulling db (mariadb:)... 
latest: Pulling from library/mariadb
2726e237d1a3: Pull complete
0b86886c6aaa: Download complete
2b221cf763a8: Download complete
5e4180757702: Download complete
43028b9f5f8e: Download complete
bbe7eafa75b: Downloading [=====] 13.95MB/69.84MB
ab732728101f: Waiting
0c9f57c1bb30: Waiting
|
```

```
Creating ftp_server ... done
Creating mariadb     ... done
Creating phpmyadmin ... done
Creating web_php     ... done
jordan@debdocker:~/projet-xampp
```

Test

Apache fonctionnel

PHP Version 8.1.32

php

System	Linux ardd532b593a 6.1.0-32-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.129-1 (2025-03-06) x86_64
Build Date	Apr 29 2025 21:52:20
Build System	Linux - Docker
Build Provider	https://github.com/docker-library/php
Configure Command	--prefix=/usr --sysconfdir=/etc/php/8.1 --with-config-file-path=/etc/php/8.1 --with-config-file-scan-dir=/usr/local/etc/php/conf.d --enable-option-checking=fatal --with-mhash --with-pic --enable-fpm --enable-mbstring --enable-mysqlind --with-password-argon2 --with-sodium=shared --with-pdo-sqlite=/usr --with-sqlite3=/usr --with-curl --with-iconv --with-openssl --with-readline --with-zlib --disable-phpdbg --with-pear --with-libsodium=/lib/x86_64-linux-gnu --disable-zgc --with-apr=>2 --build_alias=>x86_64-linux-gnu
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/php-ext-sodium ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902,NTS
PHP Extension Build	API20210902,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
Zend Max Execution Timers	disabled
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, convert.iconv*, string.rot13, string.toupper, string.tolower, convert*, consumed, dechunk

This program makes use of the Zend Scripting Language Engine.
Zend Engine v4.1.32, Copyright (c) Zend Technologies

zend engine

Configuration apache2handler

Apache Version	Apache/2.4.62 (Debian)
Apache API Version	20120211
Server Administrator	webmaster@localhost
Hostname:Port	172.22.0.5:80
User/Group	www-data/33:33
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	Yes
Server Root	/etc/apache2
Loaded Modules	core mod_so mod_watchdog http_core mod_log_config mod_logio mod_version mod_unixd mod_access_compat mod_alias mod_auth_basic mod_authn_core mod_authn_file mod_authz_core mod_authz_host mod_authz_user mod_dav mod_dav_fs mod_deflate mod_dir mod_env mod_mime mod_prefork mod_php mod_negotiation mod_status mod_rewrite mod_setenvif mod_status

PhpMyAdmin fonctionnel

192.168.76.129:9501

phpMyAdmin
Bienvenue dans phpMyAdmin

Langue (Language)
Français - French

Connexion
Utilisateur :
Mot de passe : Connexion

Test de transfert de fichier ftp

Hôte : 192.168.76.129 Nom d'utilisateur : user Mot de passe : **** Port : 9510 Connexion rapide

Status : Serveur non sécurisé, celui-ci ne prend pas en charge FTP sur TLS.
Status : Connecté
Status : Démarrage du téléchargement de /index.html
Status : Transfert de fichier réussi, 147 octets transférés en 5 secondes
Status : Déconnecté du serveur
Status : Connexion interrompue par le serveur

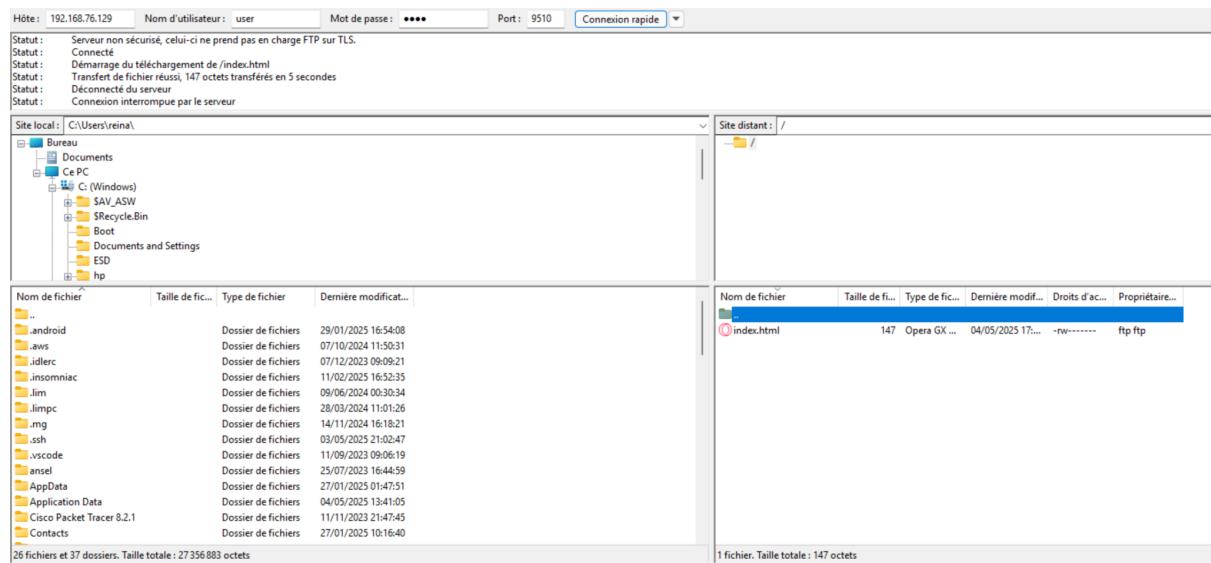
Site local : C:\Users\reina\

Site distant : /

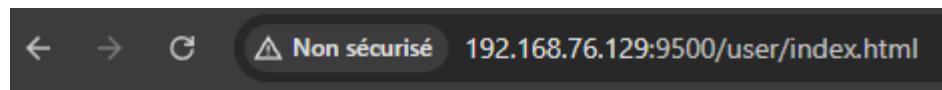
Nom de fichier	Taille de f... 147	Type de f... Opera GX...	Dernière modif... 04/05/2025 17:...	Droits d'ac... -rw-----	Propriétaire... ftp ftp
index.html	147	Opera GX...	04/05/2025 17:...	-rw-----	ftp ftp

26 fichiers et 37 dossiers. Taille totale : 27 356 883 octets.

1 fichier. Taille totale : 147 octets



on peut voir que cela a fonctionné vu que nous sommes sur



Jordan / Reinaldo ḂŹ