TP 1. – INSTALLATION DE DOCKER ET PREMIER PAS

Pour préparer la machine à installer docker, on va d'abord faire ces commandes :

apt update
Puis :
apt install ca-certificates curl gnupg

On va ensuite installer les certificats :
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg

Puis on fait :
chmod a+r /etc/apt/keyrings/docker.gpg
Cette commande sert à donner les permissions au fichier docker.gpg

On va maintenant intégrer le dépôt docker dans le fichier source.list :
echo \
"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
"$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
tee /etc/apt/sources.list.d/docker.list > /dev/null

Malheureusement , il nous dit que il E: Entrée 1 mal formée dans list fichier /etc/apt/sources.list.d/docker.list (URI) E: La liste des sources ne peut être lue.

Le contenu du fichier /etc/apt/sources.list.d/docker.list doit être : deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu jammy stable

On installe docker avec : apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Ensuite pour voir si docker est bien actif, on met systemctl status docker

On ajoute l'utilisateur a docker

```
root@modele:/home/test# gpasswd -a test docker
Ajout de l'utilisateur test au groupe docker
```

La on vérifie la version de docker

```
root@modele:/home/test# docker version
Client: Docker Engine - Community
 Version:           28.0.1
 API version:       1.48
 Go version:        go1.23.6
 Git commit:        068a01e
 Built:             Wed Feb 26 10:41:08 2025
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          28.0.1
  API version:      1.48 (minimum version 1.24)
  Go version:       go1.23.6
  Git commit:       bbd0a17
  Built:            Wed Feb 26 10:41:08 2025
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.7.25
  GitCommit:        bcc810d6b9066471b0b6fa75f557a15a1cbf31bb
 runc:
  Version:          1.2.4
  GitCommit:        v1.2.4-0-g6c52b3f
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
root@modele:/home/test#
```

On va lancer un conteneur test

```
root@modele:/home/test# docker run --rm hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:bfbb0cc14f13f9ed1ae86abc2b9f11181dc50d779807ed3a3c5e55a6936dbdd5
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/
```

On va chercher le mot ubuntu

```
root@modele:/home/test# docker search ubuntu
NAME                              DESCRIPTION                                     STARS     OFFICIAL
ubuntu                            Ubuntu is a Debian-based Linux operating sys…   17496     [OK]
ubuntu/squid                      Squid is a caching proxy for the Web. Long-t…   107
ubuntu/nginx                      Nginx, a high-performance reverse proxy & we…   127
ubuntu/cortex                     Cortex provides storage for Prometheus. Long…   4
ubuntu/apache2                    Apache, a secure & extensible open-source HT…   90
ubuntu/kafka                      Apache Kafka, a distributed event streaming …   53
ubuntu/bind9                      BIND 9 is a very flexible, full-featured DNS…   102
ubuntu/prometheus                 Prometheus is a systems and service monitori…   70
ubuntu/zookeeper                  ZooKeeper maintains configuration informatio…   13
ubuntu/mysql                      MySQL open source fast, stable, multi-thread…   67
ubuntu/postgres                   PostgreSQL is an open source object-relation…   41
ubuntu/redis                      Redis, an open source key-value store. Long-…   23
ubuntu/jre                        Distroless Java runtime based on Ubuntu. Lon…   19
ubuntu/dotnet-aspnet              Chiselled Ubuntu runtime image for ASP.NET a…   25
ubuntu/grafana                    Grafana, a feature rich metrics dashboard & …   12
ubuntu/dotnet-deps                Chiselled Ubuntu for self-contained .NET & A…   16
ubuntu/cassandra                  Cassandra, an open source NoSQL distributed …   2
ubuntu/memcached                  Memcached, in-memory keyvalue store for smal…   5
ubuntu/python                     A chiselled Ubuntu rock with the Python runt…   20
ubuntu/dotnet-runtime             Chiselled Ubuntu runtime image for .NET apps…   20
ubuntu/prometheus-alertmanager    Alertmanager handles client alerts from Prom…   9
ubuntu/mlflow                     MLFlow: for managing the machine learning li…   5
ubuntu/telegraf                   Telegraf collects, processes, aggregates & w…   4
ubuntu/loki                       Grafana Loki, a log aggregation system like …   2
ubuntu/chiselled-jre              [MOVED TO ubuntu/jre] Chiselled JRE: distrol…   3
```

```
root@modele:/home/test# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
5a7813e071bf: Pull complete
Digest: sha256:72297848456d5d37d1262630108ab308d3e9ec7ed1c3286a32fe09856619a782
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

Docker télécharge les différentes couches qui forment l'image.

```
root@modele:/home/test# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    a04dc4851cbc   6 weeks ago    78.1MB
hello-world   latest    74cc54e27dc4   7 weeks ago    10.1kB
```

On va récupérer une autre version de l'image

```
root@modele:/home/test# docker pull ubuntu:lunar
lunar: Pulling from library/ubuntu
6360b3717211: Pull complete
Digest: sha256:5a828e28de105c3d7821c4442f0f5d1c52dc16acf4999d5f31a3bc0f03f06edd
Status: Downloaded newer image for ubuntu:lunar
docker.io/library/ubuntu:lunar
```

```
root@modele:/home/test# docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
ubuntu        latest    a04dc4851cbc   6 weeks ago      78.1MB
hello-world   latest    74cc54e27dc4   7 weeks ago      10.1kB
ubuntu        lunar     f4cdeba72b99   15 months ago    70.3MB
```

docker rmi hello-world

Ensuite, on fait : docker run ubuntu

Ici, on liste les conteneurs actifs

```
root@modele:/home/test# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Ensuite , on va lister les docker actif

```
root@modele:/home/test# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS                      PORTS     NAMES
f329ae601ca2   ubuntu    "/bin/bash"   14 seconds ago   Exited (0) 14 seconds ago             nifty_sutherland
```

docker run ubuntu

docker ps -a

```
root@modele:/home/test# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED         STATUS                     PORTS     NAMES
6144af5359bf   ubuntu    "/bin/bash"   6 seconds ago   Exited (0) 5 seconds ago             angry_clarke
f329ae601ca2   ubuntu    "/bin/bash"   7 minutes ago   Exited (0) 7 minutes ago             nifty sutherland
```

docker run ubuntu cat /etc/lsb-release

```
root@modele:/home/test# docker run ubuntu cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=24.04
DISTRIB_CODENAME=noble
DISTRIB_DESCRIPTION="Ubuntu 24.04.1 LTS"
```

docker ps -a

```
root@modele:/home/test# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                     PORTS     NAMES
e55fc581fc0e   ubuntu    "cat /etc/lsb-release"   3 minutes ago    Exited (0) 3 minutes ago             gracious_hawking
6144af5359bf   ubuntu    "/bin/bash"              4 minutes ago    Exited (0) 4 minutes ago             angry_clarke
f329ae601ca2   ubuntu    "/bin/bash"              11 minutes ago   Exited (0) 11 minutes ago            nifty sutherland
```

docker rm <id conteneur> ou docker rm <nom du conteneur>

```
root@modele:/home/test# docker rm 6144af5359bf
6144af5359bf
root@modele:/home/test# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                      PORTS     NAMES
e55fc581fc0e   ubuntu    "cat /etc/lsb-release"   6 minutes ago    Exited (0) 6 minutes ago              gracious_hawking
f329ae601ca2   ubuntu    "/bin/bash"              14 minutes ago   Exited (0) 14 minutes ago             nifty_sutherland
```

docker run --rm ubuntu cat /etc/lsb-release

Cette commande ci dessus permet de programmer la suppression

docker ps -a

Cette commande permet de voir qu'il n'y a pas de nouveau conteneur

```
test@modele:~$ sudo bash
[sudo] Mot de passe de test :
root@modele:/home/test# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED       STATUS
          PORTS     NAMES
e55fc581fc0e   ubuntu    "cat /etc/lsb-release"   13 days ago   Exited (0) 13 da
ys ago            gracious_hawking
f329ae601ca2   ubuntu    "/bin/bash"              13 days ago   Exited (0) 13 da
ys ago            nifty_sutherland
root@modele:/home/test#
```

Pour relancer un conteneur, on va utiliser cette commande :

docker start <id conteneur> ou docker start <nom du conteneur>

Pour lancer le conteneur au 1er plan, on va utiliser la commande :

```
root@modele:/home/test# docker start -a gracious_hawking
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=24.04
DISTRIB_CODENAME=noble
DISTRIB_DESCRIPTION="Ubuntu 24.04.1 LTS"
```

On va supprimer le conteneur non actif :

```
root@modele:/home/test# docker rm nifty_sutherland
nifty_sutherland
```

On lance un conteneur en interactif :

docker run --name servUbuntu -it ubuntu

On sera connecté au conteneur donc on va faire :

apt update
apt dist-upgrade
apt install openssh-server

Ensuite, on va sortir du conteneur en faisant : exit

On va réactiver le conteneur avec cette commande :

```
root@modele:/home/test# docker start servUbuntu
servUbuntu
```

Et pour accéder au conteneur on va faire :

docker attach servUbuntu

Pour voir les processus lancés dans un conteneur on fait cette commande :

docker top <id conteneur> ou docker top <nom du conteneur>
En l'occurrence on va faire : docker top servUbuntu
Ne pas oublier de lancer le conteneur :

```
root@modele:/home/test# docker start servUbuntu
servUbuntu
root@modele:/home/test# docker top servUbuntu
UID                 PID                 PPID                C
STIME               TTY                 TIME                CMD
root                27122               27103               0
08:50               pts/0               00:00:00            /bin/bash
```

On va activer le service ssh :

docker attach servUbuntu

```
root@b613ba3f3d1c:/# service ssh start
 * Starting OpenBSD Secure Shell server sshd                          [ OK ]
```

Dans un autre terminal :



```
test@modele:~$ docker top servUbuntu
UID                 PID                 PPID                C                   STIME               TTY                 TIME                CMD
root                27122               27103               0                   08:50               pts/0               00:00:00            /bin/bash
root                28437               27122               0                   08:53               ?                   00:00:00            sshd: /usr/sbin/sshd [listener] 0 of 10-100 startups
```

On peut créer une nouvelle image à partir d'un conteneur à l'aide de cette commande :

docker commit <id conteneur> <nom image> ou docker commit <nom conteneur> <nom image>

On va faire cette commande qui va générer un hash :

```
root@modele:/home/test# docker commit servUbuntu aporaf/ubuntu:ssh
sha256:ae7ab5f059ffccd994a3ae221773ff2203cae84ae167fecfe341e3fc15b1f0d4
```

À l'aide de la commande docker images on peut voir qu'il y a une nouvelle image

```
root@modele:/home/test# docker images
REPOSITORY      TAG       IMAGE ID       CREATED              SIZE
aporaf/ubuntu   ssh       ae7ab5f059ff   About a minute ago   300MB
ubuntu          latest    a04dc4851cbc   8 weeks ago          78.1MB
hello-world     latest    74cc54e27dc4   2 months ago         10.1kB
ubuntu          lunar     f4cdeba72b99   16 months ago        70.3MB
```

Pour sauvegarder une image en local :

docker save <nom image> > <nom_fichier.tar>

Et pour restaurer une image :

docker load -i <nom_fichier.tar>

Docker dispose d'une interface réseau sur la machine hôte :

```
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOW
N group default
    link/ether ca:9c:61:75:38:63 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::c89c:61ff:fe75:3863/64 scope link
       valid_lft forever preferred_lft forever
```

docker run -d -p <IP:port-hôte:port-conteneur> --name <nom conteneur> <nom image> COMMANDE

```
root@modele:/home/test# docker run -d -p 172.17.123.58:999:22 --name servssh apo
raf/ubuntu:ssh /usr/sbin/sshd -D
8b8ebca6273851038108272e048aa035325b2e4eb0097feb56151b0f417abd10
```

```
root@modele:/home/test# ssh user@172.17.123.58 -p 999
The authenticity of host '[172.17.123.58]:999 ([172.17.123.58]:999)' can't be es
tablished.
ED25519 key fingerprint is SHA256:eY3F8UYxtywj5SStKDCG/WPrkYKrZTyQeUPeLJcuOzM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[172.17.123.58]:999' (ED25519) to the list of known
hosts.
```

On fait docker ps :

```
root@modele:/home/test# docker ps
CONTAINER ID   IMAGE              COMMAND               CREATED         STATUS         PORTS                        NAMES
8b8ebca62738   aporaf/ubuntu:ssh  "/usr/sbin/sshd -D"   2 minutes ago   Up 2 minutes   172.17.123.58:999->22/tcp    servssh
```

Et pour afficher les logs du conteneur : docker logs servssh

Fin DU TP

Pour créer un volume, on va faire :

docker volume create <nom_du_volume>
Dans notre cas on va faire cette commande :
docker volume create public-html

Un dossier public-html contenant un sous-dossier vide « _data » a automatiquement été créé dans /var/lib/docker/volumes/.
On va vérifier ça :

```
root@modele:/home/test# cd /var/lib/docker/volumes
root@modele:/var/lib/docker/volumes#
root@modele:/var/lib/docker/volumes# ls
backingFsBlockDev  metadata.db  public-html
root@modele:/var/lib/docker/volumes#
```

On voit bien qu'il a été créé

```
root@modele:/var/lib/docker/volumes# ls -lR /var/lib/docker/volumes/
/var/lib/docker/volumes/:
total 28
brw------- 1 root root  8, 2 mars  26 08:31 backingFsBlockDev
-rw------- 1 root root 32768 mars  26 09:14 metadata.db
drwx-----x 3 root root  4096 mars  26 09:14 public-html

/var/lib/docker/volumes/public-html:
total 4
drwxr-xr-x 2 root root 4096 mars  26 09:14 _data

/var/lib/docker/volumes/public-html/_data:
total 0
```

On va utiliser un volume dans docker :

docker run --name servweb -d -v public-html:/usr/local/apache2/htdocs -p 172.17.123.58:8001:80 httpd

```
root@modele:/var/lib/docker/volumes# docker run --name servweb -d -v public-html
:/usr/local/apache2/htdocs -p 172.17.123.58:8001:80 httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
6e909acdb790: Pull complete
9f9b03a66afb: Pull complete
4f4fb700ef54: Pull complete
09bf08b13dbd: Pull complete
084c58879b9a: Pull complete
c61868f0ad74: Pull complete
Digest: sha256:391a8eb0c1ed464163da46099606a5ec293705118f3054d6c60f5957e2485bd0
Status: Downloaded newer image for httpd:latest
673c611911d587cad293917c94820a3105dc4b85ad02cb916a24dea1f4ef5865
```

ls -l /var/lib/docker/volumes/public-html/_data/

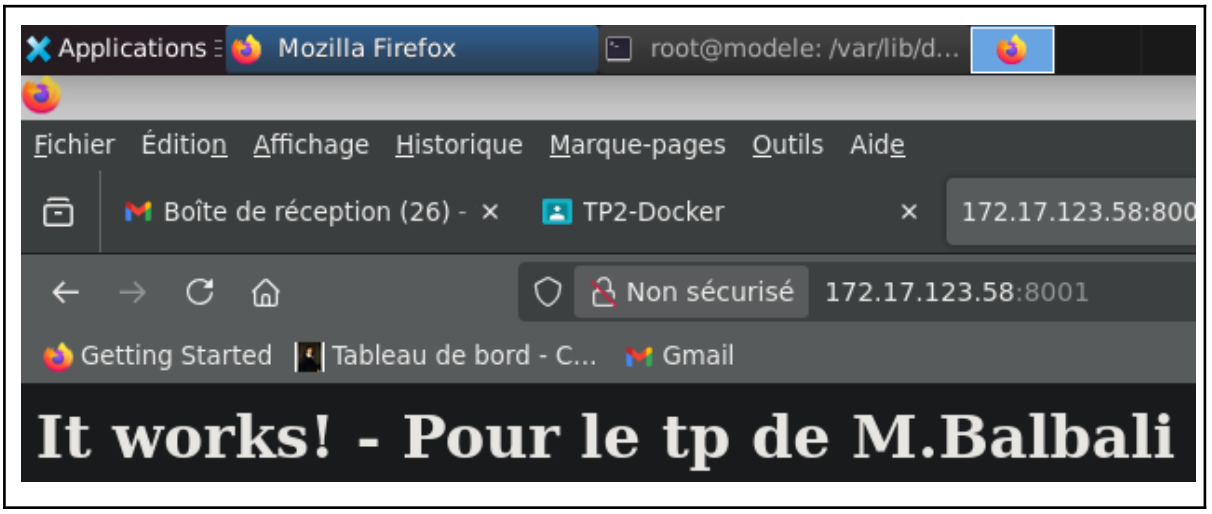
```
root@modele:/var/lib/docker/volumes# ls -l /var/lib/docker/volumes/public-html/_
data/
total 4
-rw-r--r-- 1 501 root 45 juin  11  2007 index.html
```

On va lister le fichier avec la vue des permissions

On va aller modifier l'index.html pour voir si ça marche :

```
root@modele:/var/lib/docker/volumes# cd public-html/
root@modele:/var/lib/docker/volumes/public-html# ls
_data
root@modele:/var/lib/docker/volumes/public-html# cd _data
root@modele:/var/lib/docker/volumes/public-html/_data# ls
index.html
root@modele:/var/lib/docker/volumes/public-html/ data# nano index.html
  GNU nano 6.2                          index.html
<html><body><h1>It works! - Pour ٹe tp de M.Balbali</h1></body></html>
```

On va se connecter à http://adresseIPDocker:8001 (mon ip est : 172.17.123.58:8001)



On va lister les volumes :

```
root@modele:/var/lib/docker/volumes/public-html/_data# docker volume ls
DRIVER    VOLUME NAME
local     public-html
```

Ensuite on va inspecter les volumes :

docker volume inspect public-html

```
root@modele:/var/lib/docker/volumes/public-html/_data# docker volume inspect pub
lic-html
[
    {
        "CreatedAt": "2025-03-26T09:14:46+01:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/public-html/_data",
        "Name": "public-html",
        "Options": null,
        "Scope": "local"
    }
]
```

Pour supprimer un volume : docker volume rm public-html

Installation de Portainer

Tout d'abord, on va créer un volume qui permet de conserver toutes les données de portainer sur

l'espace disque de l'hôte.

```
root@modele:/var/lib/docker/volumes/public-html/_data# docker volume create portainer_data
```

Donner le chemin complet du dossier de l'hôte qui va accueillir les données de Portainer

/var/lib/docker/volumes/portainer_data/_data

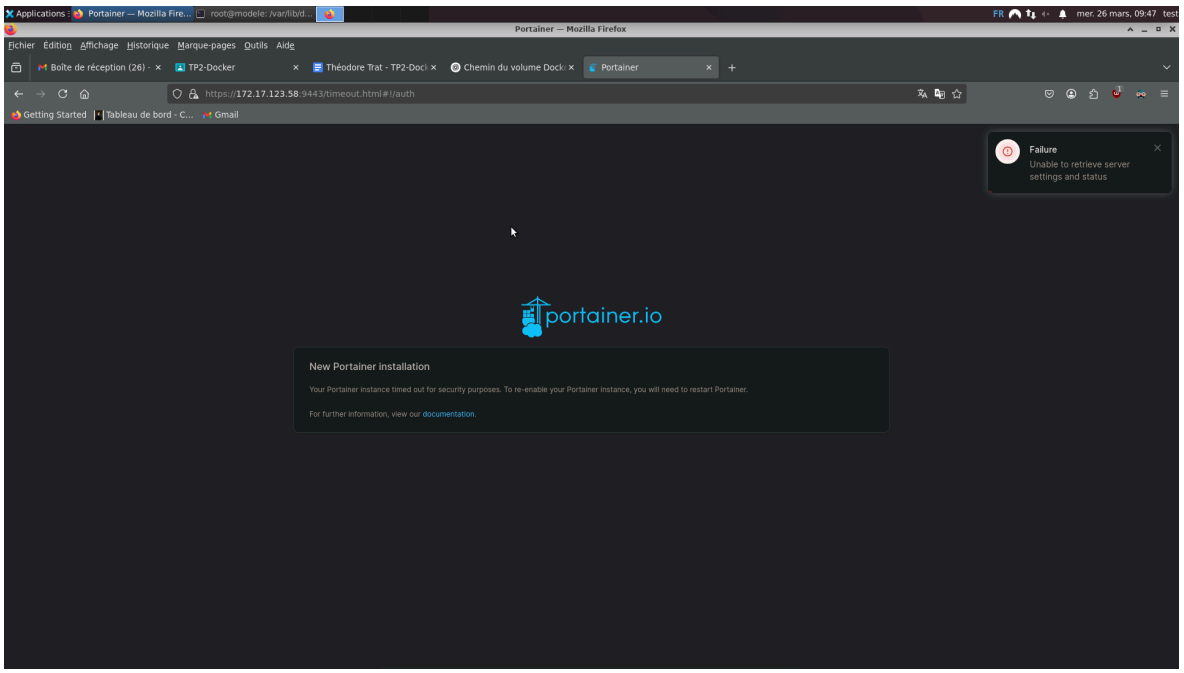
On va lancer le conteneur :

```
root@modele:/var/lib/docker/volumes/public-html/_data# docker run -d -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:lat
est
Unable to find image 'portainer/portainer-ce:latest' locally
latest: Pulling from portainer/portainer-ce
e2e06b27b87e: Pull complete
1fed1531b45b: Pull complete
04de093ad5ed: Pull complete
86a7cce72d42: Pull complete
e09df2601140: Pull complete
eae3ebf29ea8: Pull complete
c12aa3fbd31a: Pull complete
f111bda3f9a6: Pull complete
81021110ed01: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:7f10a26bfdda3fc58295ea09b860117ecd86a642d66fb94ce1f27a4c221d4649
Status: Downloaded newer image for portainer/portainer-ce:latest
6bcb1e66774cc641124de16ea4d36ac9734c4bf43c697a5203f49a8afa8f64c7
root@modele:/var/lib/docker/volumes/public-html/_data# docker run -d -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:lat
```

Expliquer précisément l'option -p 9443:9443

L'option `-p 9443:9443` utilisée dans la commande Docker signifie que tu **rediriges le port 9443 du conteneur** (à droite) vers **9443 de l'hôte** (à gauche).

Donner l'URL pour accéder à l'application Web

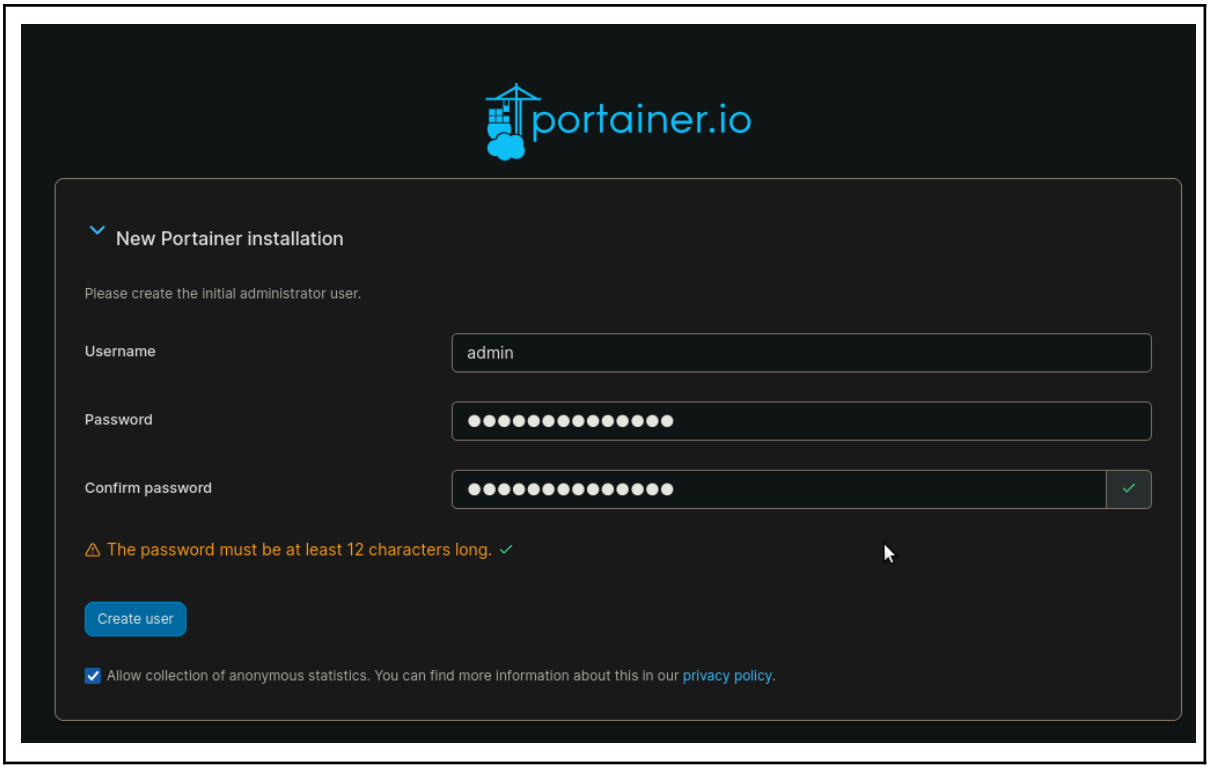


https://172.17.123.58:9443

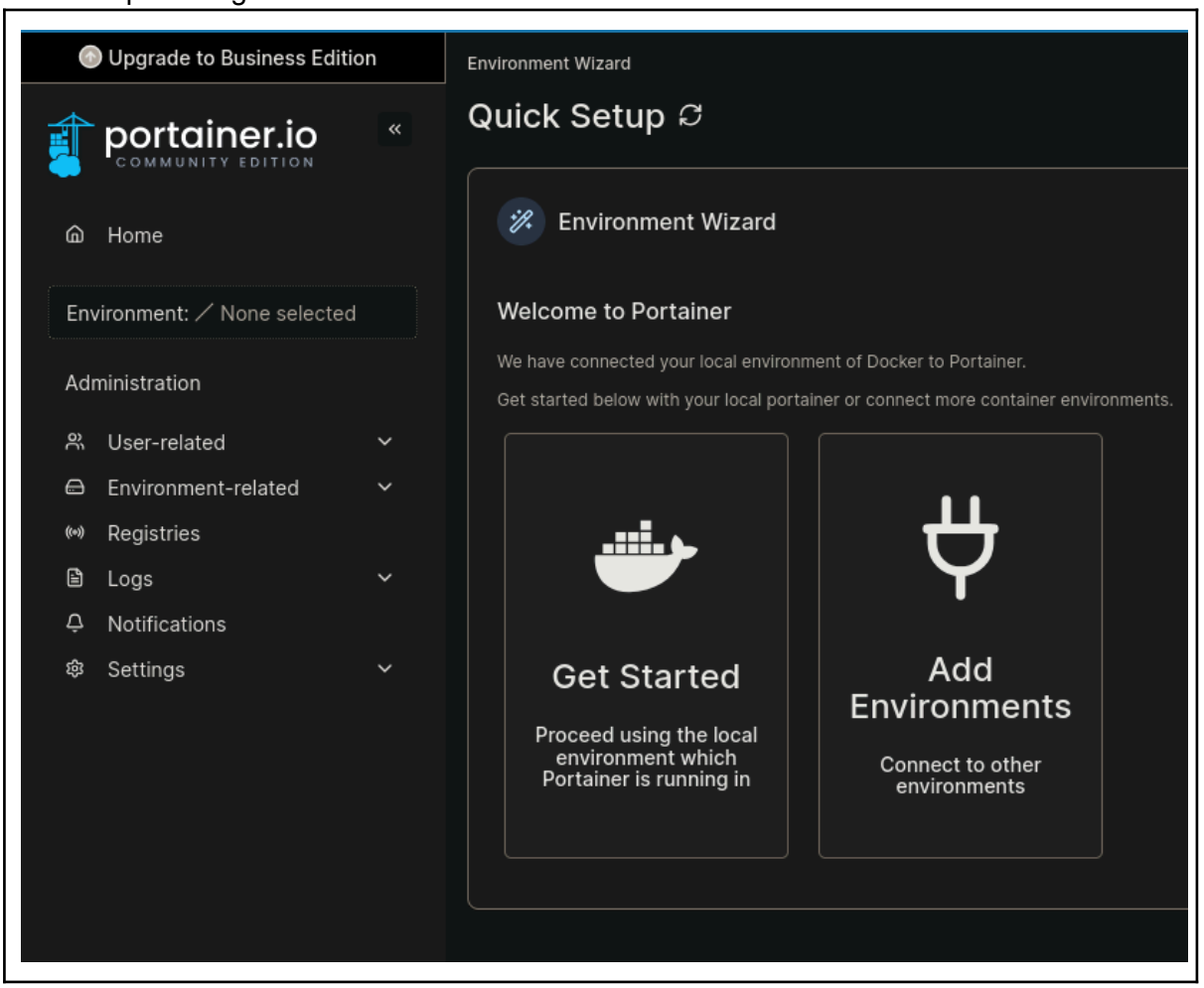
On va donc redémarrer docker avec ces commandes :

```
docker stop portainer
docker start portainer
```

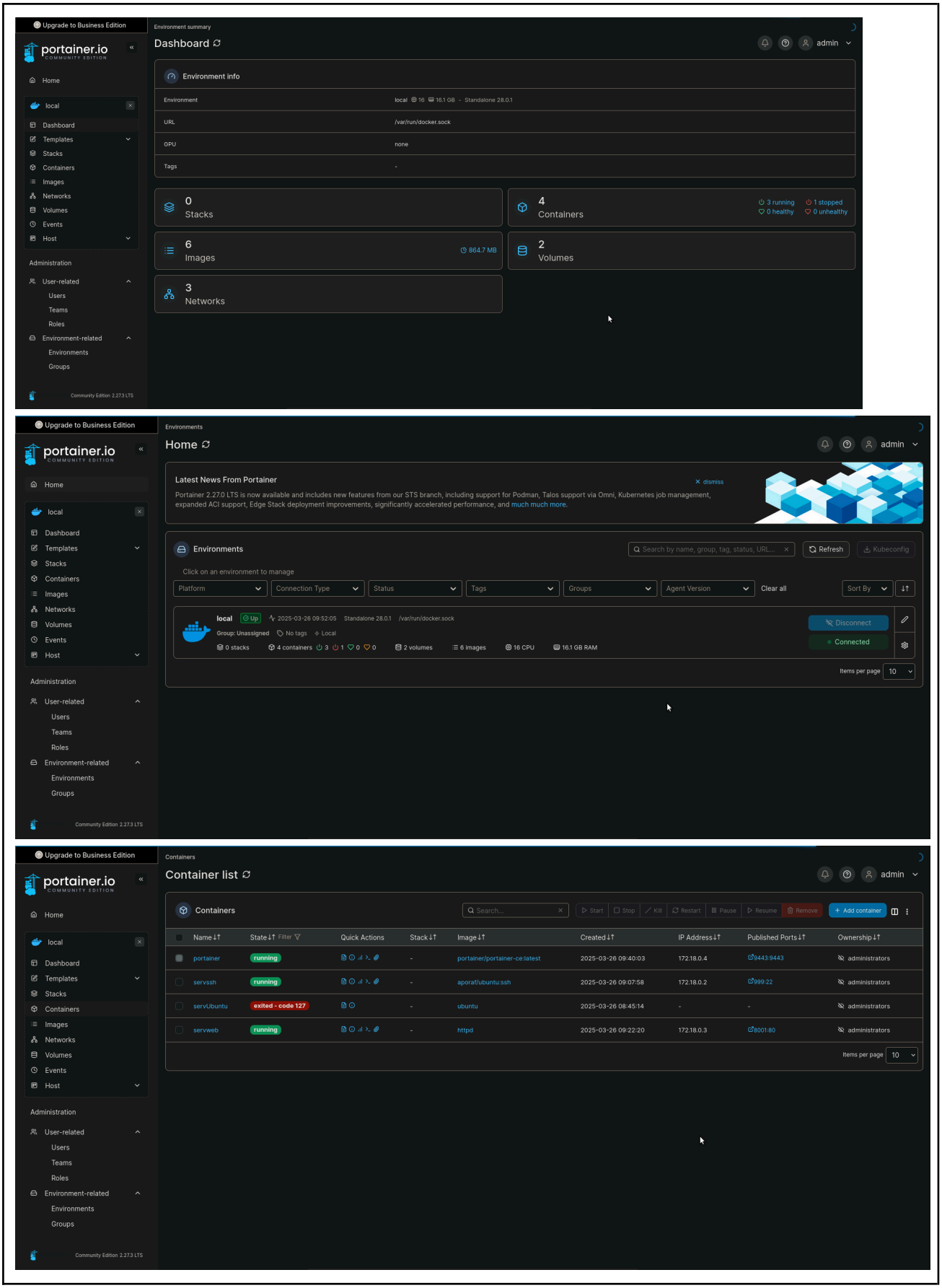
Et on va configurer le login :

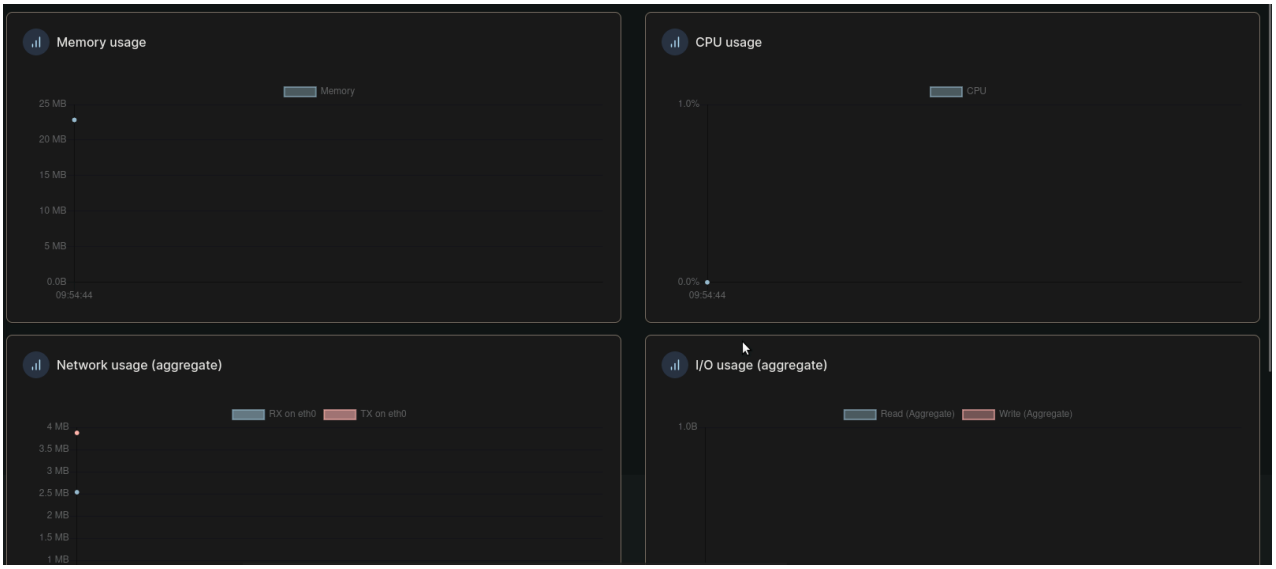


On va cliquer sur get started :



On a plusieurs onglets :

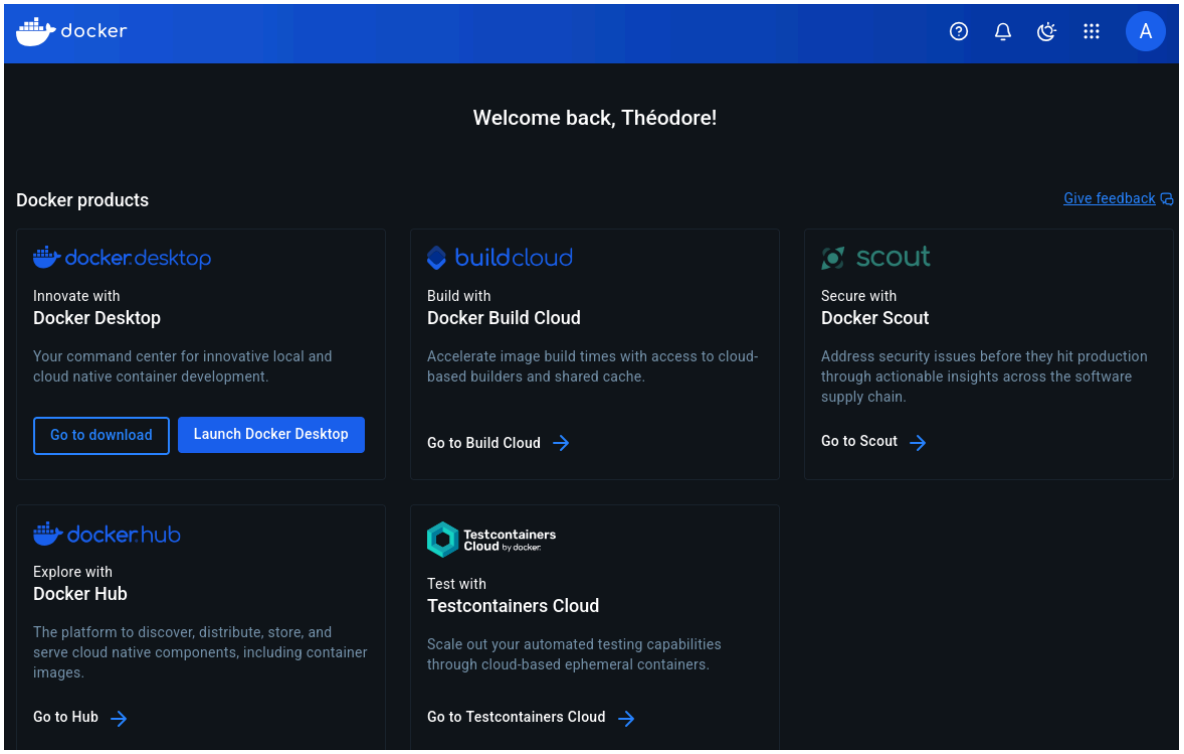

Upgrade to Business Edition
portainer.io
COMMUNITY EDITION
Home
local
Dashboard
Templates
Stacks
Containers
Images
Networks
Volumes
Events
Host
Administration
User-related
Users
Teams
Roles
Environment-related
Environments
Groups
Community Edition 2.27.3 LTS
Environment summary
Dashboard
admin
Environment info
Environment
local 16 16.1 GB - Standalone 28.0.1
URL
/var/run/docker.sock
GPU
none
Tags
-
0
Stacks
4
Containers
3 running
1 stopped
0 healthy
0 unhealthy
6
Images
864.7 MB
2
Volumes
3
Networks
Upgrade to Business Edition
portainer.io
COMMUNITY EDITION
Home
local
Dashboard
Templates
Stacks
Containers
Images
Networks
Volumes
Events
Host
Administration
User-related
Users
Teams
Roles
Environment-related
Environments
Groups
Community Edition 2.27.3 LTS
Environments
Home
admin
Latest News From Portainer
dismiss
Portainer 2.27.0 LTS is now available and includes new features from our STS branch, including support for Podman, Talos support via Omni, Kubernetes job management, expanded ACI support, Edge Stack deployment improvements, significantly accelerated performance, and much much more.
Environments
Search by name, group, tag, status, URL...
Refresh
Kubeconfig
Click on an environment to manage
Platform
Connection Type
Status
Tags
Groups
Agent Version
Clear all
Sort By
local
Up
2025-03-26 09:52:05
Standalone 28.0.1
/var/run/docker.sock
Group: Unassigned
No tags
Local
0 stacks
4 containers
3
1
0
0
2 volumes
6 images
16 CPU
16.1 GB RAM
Disconnect
Connected
Items per page
10
Upgrade to Business Edition
portainer.io
COMMUNITY EDITION
Home
local
Dashboard
Templates
Stacks
Containers
Images
Networks
Volumes
Events
Host
Administration
User-related
Users
Teams
Roles
Environment-related
Environments
Groups
Community Edition 2.27.3 LTS
Containers
Container list
admin
Containers
Search...
Start
Stop
Kill
Restart
Pause
Resume
Remove
Add container
Name
State
Filter
Quick Actions
Stack
Image
Created
IP Address
Published Ports
Ownership
portainer
running
portainer/portainer-ce:latest
2025-03-26 09:40:03
172.18.0.4
9443:9443
administrators
servssh
running
aporaf/ubuntu:ssh
2025-03-26 09:07:58
172.18.0.2
999:22
administrators
servUbuntu
exited - code 127
ubuntu
2025-03-26 08:45:14
-
-
administrators
servweb
running
httpd
2025-03-26 09:22:20
172.18.0.3
8001:80
administrators
Items per page
10



Les statistiques sont en temps réel et permettent de suivre le CPU, la mémoire, l’utilisation du réseau ainsi que les processus en cours d’exécution

Fin du tp 2 docker

On va donc aller sur ce lien : https://hub.docker.com/ et on va créer un compte



username : archanight
password : mon jeu préféré

La commande qui permet de construire l'image a la structure suivante :

```
docker build [-t <nom_image>[:<tag>]] <dossier_dockerfile>
```

On va créer un répertoire de construction :

```
mkdir -p ~/contexte/ubuntu-ssh
```

ensuite on se place dans le répertoire avec :

```
cd ~/contexte/ubuntu-ssh
```

Puis on va faire : nano dockerfile

Cela nous ramène dans ce menu :

```
  GNU nano 6.2
FROM ubuntu:24.04
LABEL Mainteneur="Théodore"
LABEL description="Ubuntu TLS avec le serveur SSH activé"
# Installation des logiciels
RUN apt update \
&& apt -y --no-install-recommends install \
openssh-server \
tzdata \
locales \
&& rm -rf /var/lib/apt/lists/* \
&& apt clean
# Configuration de la Timezone Europe/Paris et des locales
ENV LANG=fr_FR.UTF-8
```

```
# Configuration de la Timezone Europe/Paris et des locales
ENV LANG=fr_FR.UTF-8

RUN rm /etc/localtime \
&& ln -s /usr/share/zoneinfo/Europe/Paris /etc/localtime \
&& dpkg-reconfigure --frontend noninteractive tzdata \
&& sed -i 's/# fr_FR.UTF-8 UTF-8/fr_FR.UTF-8 UTF-8/' /etc/locale.gen \
&& echo ’LANG=${LANG}’>/etc/default/locale \
&& dpkg-reconfigure --frontend=noninteractive locales \
&& update-locale LANG=${LANG}
# Création de l’utilisateur qui pourra se connecter au serveur SSH
RUN useradd --create-home --shell /bin/bash userssh \
&& echo "userssh:mdpuser" | chpasswd
RUN service ssh start
EXPOSE 22
CMD ["/usr/sbin/sshd","-D"]
```

Faire attention aux mots mal copiés ‘

La commande qui permet de construire une image est la suivante :
docker build -t <docker-id>/ubuntu-ssh:1.0 ~/contexte/ubuntu-ssh
On va build l’image avec cette commande :
docker build -t archanight/ubuntu-ssh:1.0 ~/contexte/ubuntu-ssh

```
root@modele:~/contexte/ubuntu-ssh# docker build -t archanight/ubuntu-ssh:1.0 ~/contexte/ubuntu-ssh
[+] Building 3.1s (10/10) FINISHED                                                          docker:default
 => [internal] load build definition from dockerfile                                                  0.0s
 => => transferring dockerfile: 963B                                                                  0.0s
 => [internal] load metadata for docker.io/library/ubuntu:24.04                                       0.8s
 => [auth] library/ubuntu:pull token for registry-1.docker.io                                         0.0s
 => [internal] load .dockerignore                                                                     0.0s
 => => transferring context: 2B                                                                       0.0s
 => [1/5] FROM docker.io/library/ubuntu:24.04@sha256:72297848456d5d37d1262630108ab308d3e9ec7ed1c3286a32fe09856619a782  0.0s
 => CACHED [2/5] RUN apt update && apt -y --no-install-recommends install openssh-server tzdata locales && rm -rf /var/lib/apt/lists/* && apt clean  0.0s
 => [3/5] RUN rm /etc/localtime && ln -s /usr/share/zoneinfo/Europe/Paris /etc/localtime && dpkg-reconfigure --frontend noninteractive tzdata && sed -i 's/# fr_FR.UTF-8 UTF-8/fr_FR.UTF-8 UTF-8/' /etc/loca  1.6s
 => [4/5] RUN useradd --create-home --shell /bin/bash userssh && echo "userssh:mdpuser" | chpasswd   0.3s
 => [5/5] RUN service ssh start                                                                       0.2s
 => exporting to image                                                                                0.2s
 => => exporting layers                                                                               0.2s
 => => writing image sha256:77650086bcfead6b66a4a1099405fdc4a957074e06747d588fb95e26a8ba6457          0.0s
 => => naming to docker.io/archanight/ubuntu-ssh:1.0                                                  0.0s
```

On va faire docker images :

```
root@modele:~/contexte/ubuntu-ssh# docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
archanight/ubuntu-ssh     1.0       77650086bcfe   12 minutes ago   126MB
aporaf/ubuntu             ssh       ae7ab5f059ff   2 hours ago      300MB
portainer/portainer-ce    latest    fab249e1b82c   33 hours ago     268MB
ubuntu                    latest    a04dc4851cbc   8 weeks ago      78.1MB
httpd                     latest    83d938198316   2 months ago     148MB
hello-world               latest    74cc54e27dc4   2 months ago     10.1kB
ubuntu                    lunar     f4cdeba72b99   16 months ago    70.3MB
```

On va lancer le conteneur

```
root@modele:~/contexte/ubuntu-ssh# docker run -d -p 172.17.123.58:2222:22 --name servssh aporaf/ubuntu-ssh:1.0
```

Puis on va refaire docker images :

```
root@modele:~/contexte/ubuntu-ssh# docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
archanight/ubuntu-ssh     1.0       77650086bcfe   25 minutes ago   126MB
aporaf/ubuntu             ssh       ae7ab5f059ff   2 hours ago      300MB
portainer/portainer-ce    latest    fab249e1b82c   33 hours ago     268MB
ubuntu                    latest    a04dc4851cbc   8 weeks ago      78.1MB
httpd                     latest    83d938198316   2 months ago     148MB
hello-world               latest    74cc54e27dc4   2 months ago     10.1kB
ubuntu                    lunar     f4cdeba72b99   16 months ago    70.3MB
aporaf/ubuntu-ssh         1.0       c008dc3c19bd   19 months ago    107MB
```

On va faire cette commande : docker tag c00 aporaf/serveurssh:1.1

Puis on refait docker images :

```
root@modele:~/contexte/ubuntu-ssh# docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
archanight/ubuntu-ssh     1.0       77650086bcfe   28 minutes ago   126MB
aporaf/ubuntu             ssh       ae7ab5f059ff   2 hours ago      300MB
portainer/portainer-ce    latest    fab249e1b82c   33 hours ago     268MB
ubuntu                    latest    a04dc4851cbc   8 weeks ago      78.1MB
httpd                     latest    83d938198316   2 months ago     148MB
hello-world               latest    74cc54e27dc4   2 months ago     10.1kB
ubuntu                    lunar     f4cdeba72b99   16 months ago    70.3MB
aporaf/serveurssh         1.1       c008dc3c19bd   19 months ago    107MB
aporaf/ubuntu-ssh         1.0       c008dc3c19bd   19 months ago    107MB
```

On va supprimer l'image avec le nom qui correspond pas :

```
root@modele:~/contexte/ubuntu-ssh# docker rmi aporaf/serveurssh:1.1
Untagged: aporaf/serveurssh:1.1
```

Puis on va faire docker login pour se connecter :

```
root@modele:~/contexte/ubuntu-ssh# docker login
Authenticating with existing credentials... [Username: archanight]

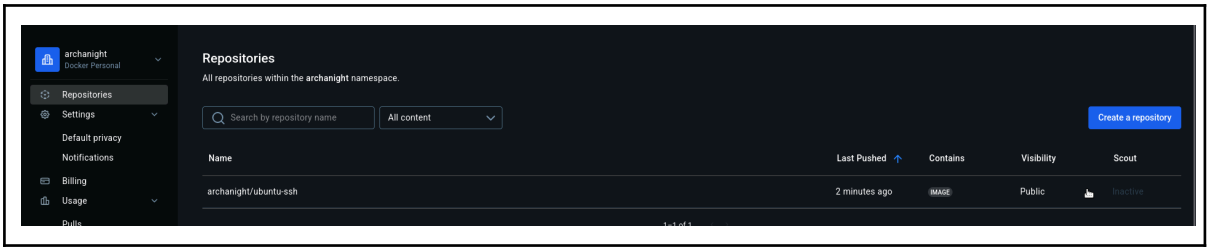
i Info → To login with a different account, run 'docker logout' followed by 'docker login'


Login Succeeded
```

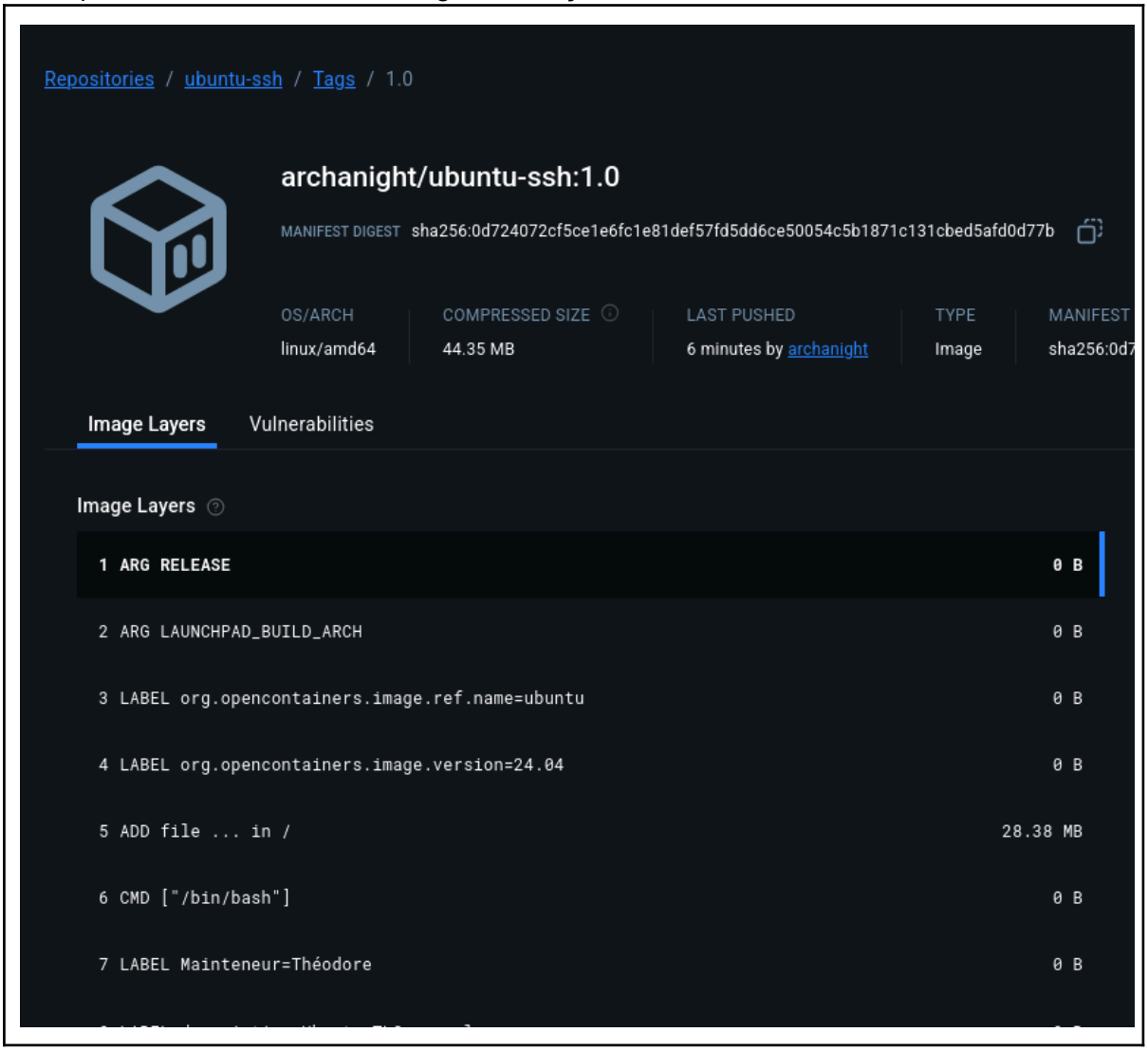
On va envoyer l'image sur docker hub public :

```
test@modele:~$ docker push archanight/ubuntu-ssh:1.0
The push refers to repository [docker.io/archanight/ubuntu-ssh]
89e00d4c3e55: Pushed
2771ca2bd123: Pushed
10b47555075a: Pushed
703099697a6b: Pushed
4b7c01ed0534: Mounted from library/ubuntu
1.0: digest: sha256:0d724072cf5ce1e6fc1e81def57fd5dd6ce50054c5b1871c131cbed5afd0d77b size: 1367
```

On va vérifier sur le site dockerhub si le dépôt a été bien crée :



En cliquant sur 1.0, le nom de l'image on voit ça :



On va créer un fichier nommé : script_entry.sh dans le répertoire : ~/contexte/ubuntu-ssh

```
root@modele:~/contexte/ubuntu-ssh# nano script_entry.sh
```

```
  GNU nano 6.2
#!/bin/bash
# Attribution du mot de passe de root (rootAdmin par défaut défini dans le dockerfile)
echo root:$MDP | chpasswd
# Démarrage au premier plan du service
/usr/sbin/sshd -D
```

On va modifier après le dockerfile :

```
# Pour permettre également à root de se connecter
RUN sed -ri ’s/^#?PermitRootLogin\s+.*/PermitRootLogin yes/’ /etc/ssh/sshd_config
ENV MDP=rootAdmin
COPY script_entry.sh /script_entry.sh
RUN chmod +x /script_entry.sh
```

faire attention au caractère mal copié

On va rebuild l’image :

```
root@modele:~/contexte/ubuntu-ssh# docker build -t aporaf/ubuntu-ssh:1.1 ~/contexte/ubuntu-ssh/
[+] Building 1.1s (12/12) FINISHED                                                                docker:default
 => [internal] load build definition from dockerfile                                                        0.0s
 => => transferring dockerfile: 1.09kB                                                                      0.0s
 => [internal] load metadata for docker.io/library/ubuntu:24.04                                             0.4s
 => [internal] load .dockerignore                                                                           0.0s
 => => transferring context: 2B                                                                             0.0s
 => [1/7] FROM docker.io/library/ubuntu:24.04@sha256:72297848456d5d37d1262630108ab308d3e9ec7ed1c3286a32fe09856619a782   0.0s
 => [internal] load build context                                                                           0.0s
 => => transferring context: 37B                                                                            0.0s
 => CACHED [2/7] RUN apt update && apt -y --no-install-recommends install openssh-server tzdata locales && rm -rf /var/lib/apt/lists/* && apt clean   0.0s
 => CACHED [3/7] RUN rm /etc/localtime && ln -s /usr/share/zoneinfo/Europe/Paris /etc/localtime && dpkg-reconfigure --frontend noninteractive tzdata && sed -i 's/# fr_FR.UTF-8 UTF-8/fr_FR.UTF-8 UTF-8/' /e  0.0s
 => [4/7] RUN sed -ri 's/^#?PermitRootLogin\s+.*/PermitRootLogin yes/' /etc/ssh/sshd_config                 0.2s
 => [5/7] COPY script_entry.sh /script_entry.sh                                                             0.0s
 => [6/7] RUN chmod +x /script_entry.sh                                                                     0.2s
 => [7/7] RUN service ssh start                                                                             0.2s
 => exporting to image                                                                                      0.0s
 => => exporting layers                                                                                     0.0s
 => => writing image sha256:b3edf6c334d58437145b910573f95ff67ef28fa3483d409c154dbab73cf565cf                0.0s
 => => naming to docker.io/aporaf/ubuntu-ssh:1.1                                                            0.0s
```

On va supprimer le conteneur :

```
docker rm -f servssh
```

On va envoyer l’image sur docker hub :

```
root@modele:~/contexte/ubuntu-ssh# docker push archanight/ubuntu-ssh:1.0
The push refers to repository [docker.io/archanight/ubuntu-ssh]
89e00d4c3e55: Layer already exists
2771ca2bd123: Layer already exists
10b47555075a: Layer already exists
703099697a6b: Layer already exists
4b7c01ed0534: Layer already exists
1.0: digest: sha256:0d724072cf5ce1e6fc1e81def57fd5dd6ce50054c5b1871c131cbed5afd0d77b size: 1367
```

On va créer une image en autonomie

Créer le fichier dockerfile

On se place dans le répertoire : cd ~/contexte/ubuntu-ssh
Et on fait nano dockerfile :
Et on met ceci dans le fichier :

```
FROM debian:12

RUN apt-get update && \
        apt-get install -y apache2 locales && \
        apt-get clean

RUN echo "Europe/Paris" > /etc/timezone && \
        ln -fs /usr/share/zoneinfo/Europe/Paris /etc/localtime && \
        dpkg-reconfigure -f noninteractive tzdata && \
        sed -i 's/# fr_FR.UTF-8 UTF-8/fr_FR.UTF-8 UTF-8/' /etc/locale.gen && \
```

```
        locale-gen fr_FR.UTF-8 && \
        update-locale LANG=fr_FR.UTF-8

ENV LANG=fr_FR.UTF-8 \
        LANGUAGE=fr_FR:fr \
        LC_ALL=fr_FR.UTF-8

COPY index.html /var/www/html/index.html

EXPOSE 80

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

Une fois cela fait on veut créer l'image :

On va la créer à l'aide de cette commande :

docker build -t archanight/debian12-apache2:1.0 .

Pour lancer le conteneur docker on va faire cette commande :

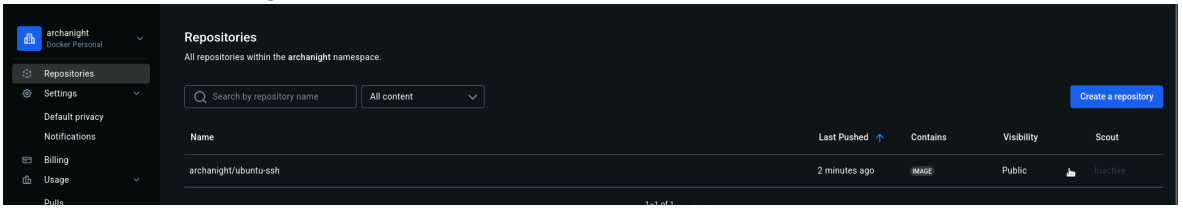
```
docker run -d -p 8008:80 --name autonomie archanight/debian12-apache2:1.0
```

Dans le navigateur, je lance cet url :

http://localhost:8008
Je vois ma page html personnalisé donc c'est bon

Pour publier mon image sur docker hub je vais devoir faire ces commandes suivantes :

```
docker login
docker push archanight/debian12-apache2:1.0
```



L'image est disponible en ligne

On va se mettre dans le bon répertoire avec la commande :

```
cd contexte
```

Puis on va récupérer les fichiers depuis le git :

```
git clone https://github.com/onemarcfifty/kali-linux-docker.git
```

Une fois les fichiers récupérés, on va se mettre dans le dossier suivant :

```
cd kali-linux-docker
```

Et on va copier :

```
cp env_template env
```

Remplacer les deux lignes 83 et 84 par celles qui le permettent.

```
RUN apt update -q --fix-missing
RUN apt upgrade -y
RUN apt -y install --no-install-recommends sudo wget curl dbus-x11 xinit openssh-server ${DESKTOP_PKG
RUN apt -y install locales
RUN sed -i 's/^# en_US.UTF-8 UTF-8/en_US.UTF-8 UTF-8/' /etc/locale.gen
RUN locale-gen
```

Modifier, dans le fichier « env » les variables d'environnement correspondantes.

```
# Remote access metohod for desktop
# Options: "vnc" "rdp" "x2go"
# Once rdp is up, it may take a moment to load
# Note: Only xfce works with rdp right now
XREMOTE_ACCESS="rdp"

# Kali packages
# Options: "arm" "core" "default" "everything" "firmware" "headless" "labs" "large" "nethunter"
XKALI_PKG="core"

# Network configuration
# Options: "bridge" "host"
XNETWORK="bridge"

# Target build architecture
# Options: "amd64" "arm64"
# Other untested options: https://docs.docker.com/build/building/multi-platform/
XBUILD_PLATFORM="amd64"

# Name for local custom Docker image
DOCKERIMG="archanight/kali-linux:core"

# Name for Docker container generated from custom image
CONTAINER="kali-linux-core"

# Host directory to mount to container
HOSTDIR="archanight/kali-linux-core"

# Container directory to mount host directory
CONTAINERDIR="/home/userkali"

# Username for container user
USERNAME="userkali"

# Password for container user
PASSWORD="password123"
```

Supprimer l'exposition du port VNC dans le Dockerfile.

```
ARG VNC_PORT
```

On supprime ça dans le fichier Dockerfile

Modifier la création du conteneur dans le script « build » pour supprimer le port d'écoute de

VNC

```
if [ -z $XVNC_PORT ];
then
    XVNC_PORT=590$XVNC_DISPLAY
fi
```

On supprime ça dans le fichier build

Créer le $HOSTDIR (si vous ne le créez pas, la commande de création de conteneur va le faire mais attribuera ce dossier à l'utilisateur root).

mkdir HOSTDIR

On va se positionner dans le répertoire du contexte :

cd ~/contexte/kali-linux

Puis on va faire cette commande ./build

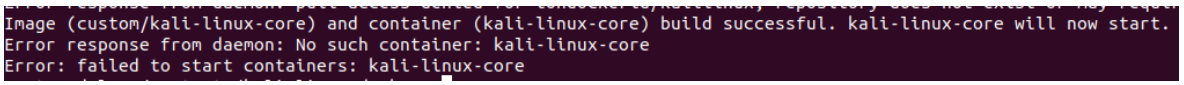
On va faire ./build
Cela nous amène ici :

```
Configuration:

Desktop environment:    xfce
Remote Access:          rdp
Kali packages:          core
Network:                bridge
Build platform:         amd64
Image name:             archanight/kali-linux:core
Conatiner name:         kali-linux-core
Host dir mount:         archanight/kali-linux-core
Container dir mount:    /home/test/userkali
Username:               userkali
Password                [redacted]

Hit enter to start building the container
```

On fait entrée
Le conteneur est crée si ceci est écrit :
Image (archanight/kalilinux:core) and container (kali-linux) build successful. kali-linux will now start.



Malgré mes tentatives, je n'arrive pas à fabriquer le build

Écrire la commande de création du conteneur en remplaçant chaque variable par sa valeur.

```
docker create \
  --name kali-linux-core \
  --network host \
  -v home_kali-linux-core:/home/userkali \
  custom/kalilinux:core
```

Écrire la commande de démarrage du conteneur.

docker start kali-linux-core

Écrire la commande de démarrage du conteneur.

```
ssh userkali@localhost
```

Configurer le client bureau à distance pour accéder à l'interface graphique de Kali.

```
remmina
Comme je n'arrive pas à faire le conteneur, je ne peux pas accéder à l'interface graphique de Kali.
```

Publier l'image sur le registre officiel.

```
docker tag custom/kalilinux:core archanight/kalilinux:core
docker push archanight/kalilinux:core
```

Donner la taille de votre image.

```
docker images
Et on regarde la taille, je n'ai pas réussi à la créer donc je ne peux pas le dire
```

Vérifier que le conteneur se soit bien créé.

```
docker images
On regarde si il y a un repository
```

Supprimer le conteneur créé à l'étape précédente.

```
docker rm -f kali-linux-core
```

Supprimer l'espace disque actuellement utilisé (/home/$USERNAME/kali-linux-core)

```
rm -rf ~/kali-linux-core
```

Modifier la création du conteneur dans le script « build » de manière à intégrer cette nouvelle gestion de volumes (le volume associé à un conteneur se nommera « home_<nom du conteneur> »).

```
docker create \
  --name "$XKALI_CONTAINER" \
  --network host \
  -v "home_${XKALI_CONTAINER}:${XKALI_HOME}" \
  "$XKALI_IMAGE:$XKALI_TAG"
```

Supprimer le conteneur

```
docker rm -f kali-linux-core
```

Modifier la création du conteneur dans le script « build » de manière à intégrer cette nouvelle gestion du réseau

```
docker create \
  --name "$XKALI_CONTAINER" \
  --network bridge \
  -p "$XKALI_SSHPORT":22 \
  -p "$XKALI_RDPPORT":3389 \
  -v "home_${XKALI_CONTAINER}:${XKALI_HOME}" \
  "$XKALI_IMAGE:$XKALI_TAG"
```

Fin du tp

Créer dans « contexte » un dossier « dc_phpipam ».

```
root@modele:~/contexte# mkdir dc_phpipam
```

Créer dans le dossier « php_ipam » le fichier compose.yaml suivant élaboré à partir de https://hub.docker.com/r/phpipam/phpipam-www.

home > test > compose.yaml

```yaml
services:
  phpipam-web:
    image: phpipam/phpipam-www:latest
    container_name: phpipam-web
    ports:
      - "8099:80"
      - "44399:443"
    environment:
    - TZ=Europe/Paris
    - IPAM_DATABASE_HOST=phpipam-mariadb
    - IPAM_DATABASE_PASS=mysql&Ipam0
    - IPAM_DATABASE_WEBHOST=%
    restart: unless-stopped
    volumes:
    - phpipam-logo:/phpipam/css/images/logo
    - phpipam-ca:/usr/local/share/ca-certificates:ro
    depends_on:
    - phpipam-mariadb

  phpipam-cron:
    image: phpipam/phpipam-cron:latest
    container_name: phpipam-cron
    environment:
    - TZ=Europe/Paris
    - IPAM_DATABASE_HOST=phpipam-mariadb
    - IPAM_DATABASE_PASS=mysql&Ipam0
    - SCAN_INTERVAL=1h
    restart: unless-stopped
    volumes:
    - phpipam-ca:/usr/local/share/ca-certificates:ro
    depends_on:
    - phpipam-mariadb

  phpipam-mariadb:
    image: mariadb:latest
    container_name: phpipam-db
    environment:
    - MYSQL_ROOT_PASSWORD=mysql&Root0
    restart: unless-stopped
    volumes:
    - phpipam-db-data:/var/lib/mysql

volumes:
  phpipam-db-data:
  phpipam-logo:
  phpipam-ca:
```

```
root@modele:~/contexte/dc_phpipam/php_ipam# mv /home/test/compose.yaml ~/contexte/dc_phpipam/php_ipam/
```

```
root@modele:~/contexte/dc_phpipam/php_ipam# ls
compose.yaml
```

A partir du répertoire qui contient le fichier compose.yaml on fait cette commande :

docker compose up -d

```
root@modele:~/contexte/dc_phpipam/php_ipam# docker compose up -d
[+] Running 16/16
 ✔ phpipam-web Pulled
   ✔ 0a9a5dfd008f Pull complete
   ✔ cb85e684f0cc Pull complete
   ✔ 249b3c2b26c5 Pull complete
   ✔ 982c6b657c2d Pull complete
 ✔ phpipam-mariadb Pulled
   ✔ 2726e237d1a3 Pull complete
   ✔ 0b86886c6aaa Pull complete
   ✔ 2b221cf763a8 Pull complete
   ✔ 5e4180757702 Pull complete
   ✔ 43028b9f5f8e Pull complete
   ✔ bbef7eafa75b Pull complete
   ✔ ab732728101f Pull complete
   ✔ 0c9f57c1bb30 Pull complete
 ✔ phpipam-cron Pulled
   ✔ 4f4fb700ef54 Pull complete
[+] Running 7/7
 ✔ Network php_ipam_default            Created
 ✔ Volume "php_ipam_phpipam-logo"      Created
 ✔ Volume "php_ipam_phpipam-ca"        Created
 ✔ Volume "php_ipam_phpipam-db-data"   Created
 ✔ Container phpipam-db                Started
 ✔ Container phpipam-web               Started
 ✔ Container phpipam-cron              Started
root@modele:~/contexte/dc_phpipam/php_ipam#
```

Écrire ci-dessous les URL permettant d'accéder en HTTP et en HTTPS à l'application phpIPAM.

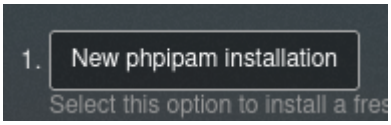
Dans le fichier compose.yaml, on retrouve :
ports:
- "8099:80"
- "44399:443"

Donc les urls pour accéder à phpIPAM sont :
http://localhost:8099 et https://localhost:44399

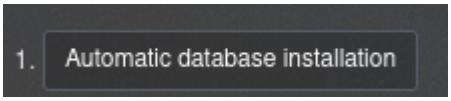
Une fois les containers lancés, se rendre en HTTPS sur l'interface Web.



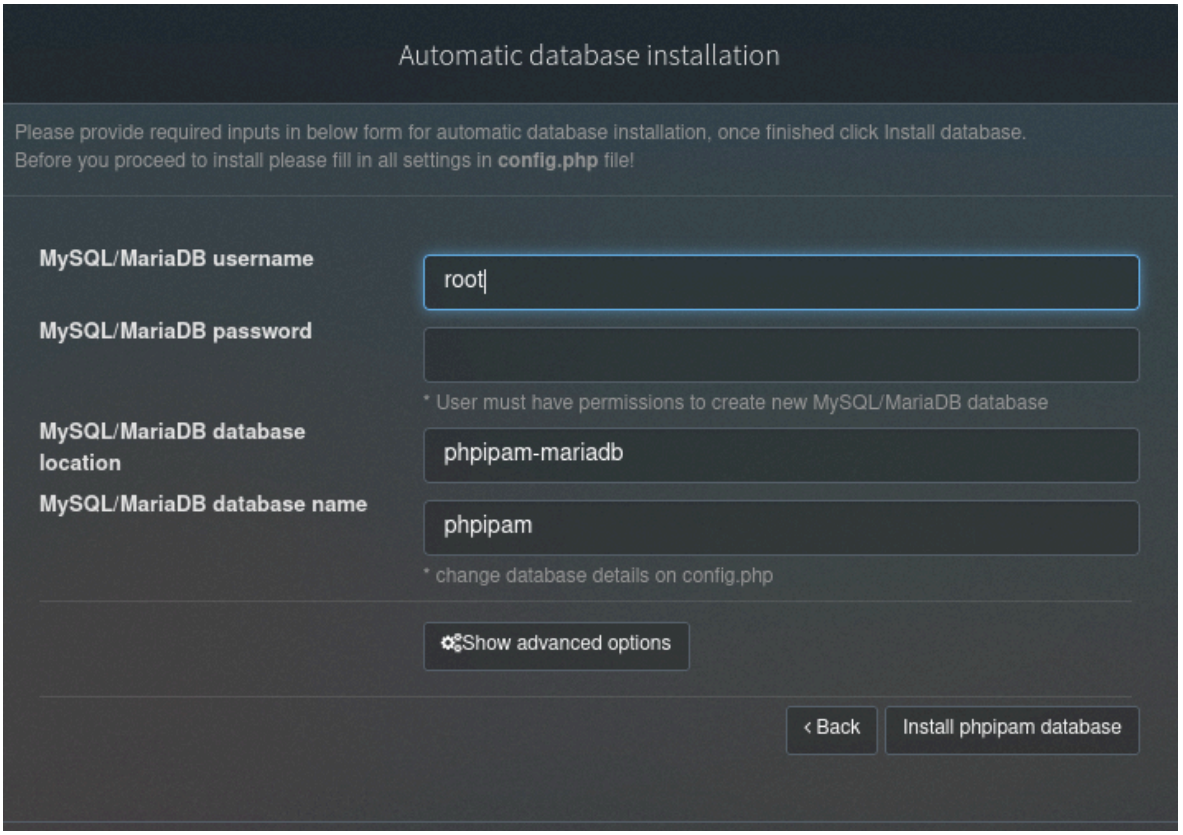
Cliquer sur « New phpipam installation »



Cliquer sur « automatic database installation » et saisir les valeurs demandées.

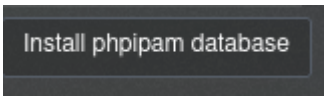


On arrive sur cette page :



le login est : root
le mot de passe est : mysql&Root0

Ensuite, on clique sur :

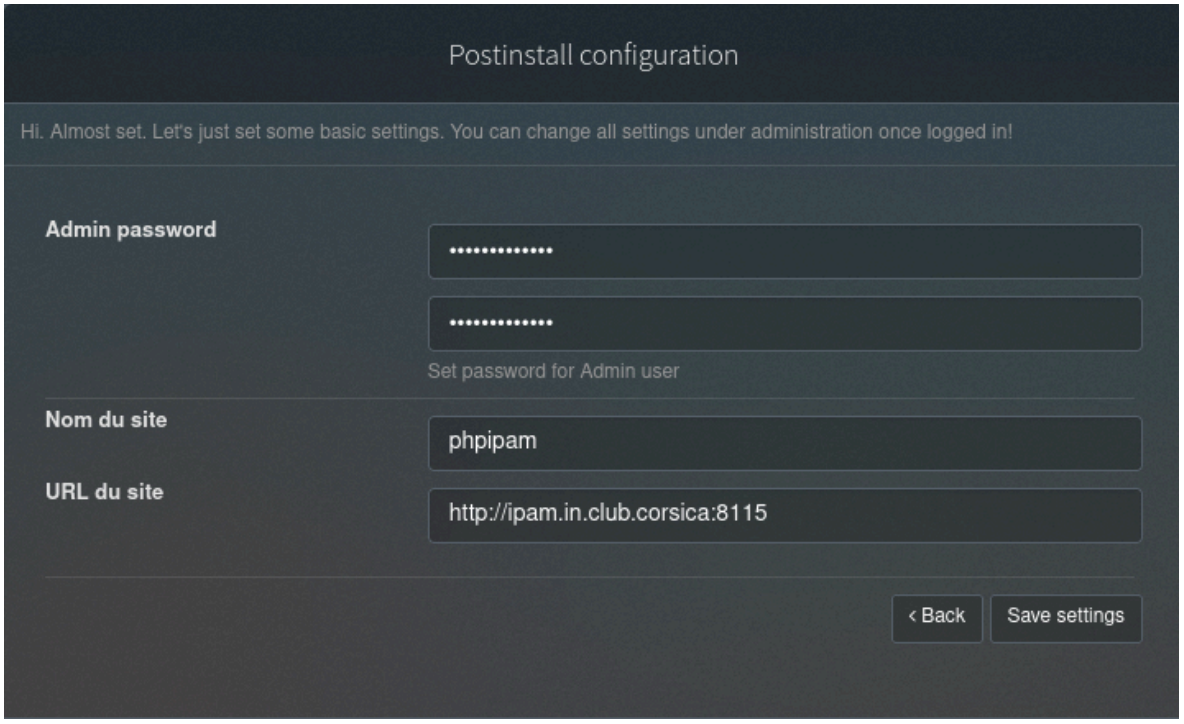


Puis après sur :

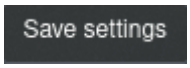


On arrive sur cette page :

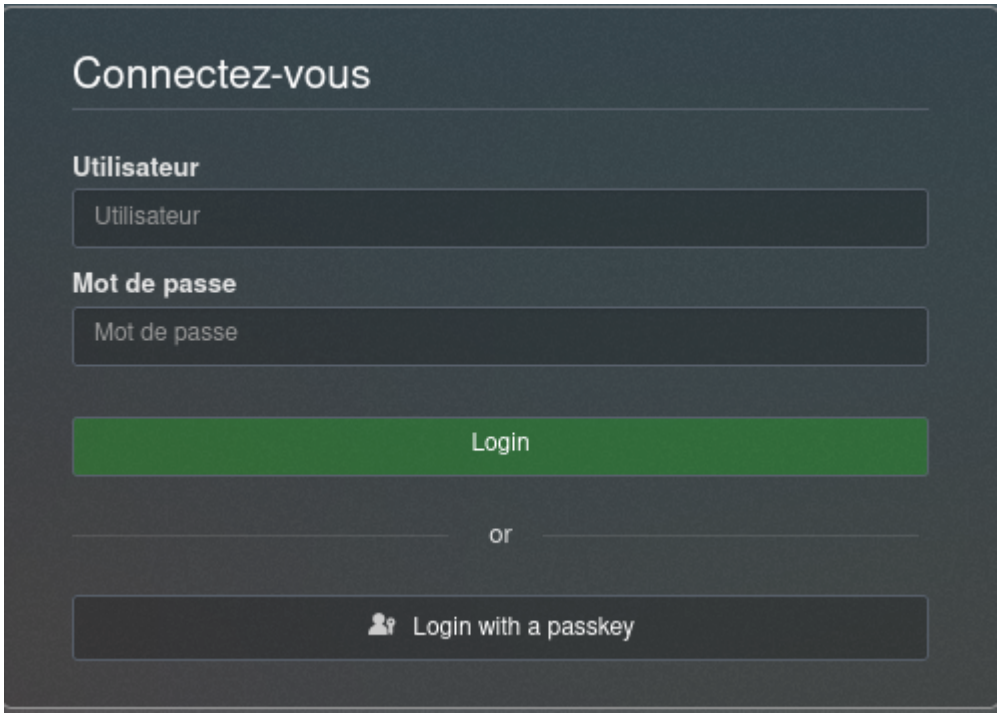
Le mot de passe de l'admin est : phpipamAdmin0



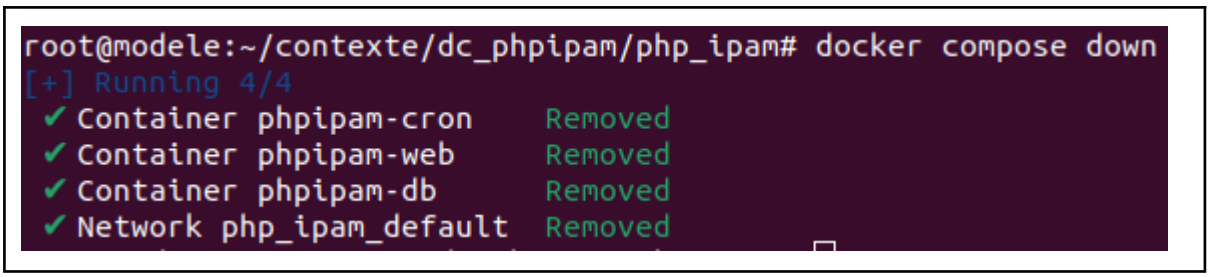
Ensuite, on fait :



On peut voir qu’on peut enfin se connecter :



Pour supprimer les conteneurs et le réseau associé on fait : docker compose down


root@modele:~/contexte/dc_phpipam/php_ipam# docker compose down
[+] Running 4/4
✔ Container phpipam-cron Removed
✔ Container phpipam-web Removed
✔ Container phpipam-db Removed
✔ Network php_ipam_default Removed