TP : Docker

Étape 1 : Installer Docker sur Ubuntu 22.04

La première étape consiste à installer Docker sur le système Ubuntu. Pour ce faire, les commandes suivantes ont été exécutées dans le terminal :

Mettre à jour le système

```
sirine@sirine:~$ sudo apt update && sudo apt upgrade -y
[sudo] Mot de passe de sirine :
Atteint :1 http://security.ubuntu.com/ubuntu noble-security InRelease
Atteint :2 http://tn.archive.ubuntu.com/ubuntu noble InRelease
Atteint :3 http://tn.archive.ubuntu.com/ubuntu noble-updates InRelease
Atteint :4 http://tn.archive.ubuntu.com/ubuntu noble-backports InRelease
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
208 paquets peuvent être mis à jour. Exécutez « apt list --upgradable » pour les voir.
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
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
  aardvark-dns buildah catatonit conmon containernetworking-plugins
  docker-compose fuse-overlayfs golang-github-containers-common
  golang-github-containers-image libllvm17t64 libslirp0 libsubid4 netavark
  passt podman python3-compose python3-docker python3-dockerpty python3-docopt
  python3-dotenv python3-netifaces python3-packaging python3-texttable
  python3-websocket slirp4netns uidmap
Veuillez utiliser « sudo apt autoremove » pour les supprimer
```

Installer Docker

```
sirine@sirine:~$ sudo apt install -y docker.io
[sudo] Mot de passe de sirine :
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Lecture des informations d'état... Fait
docker.io est déjà la version la plus récente (26.1.3-0ubuntu1~24.04.1).
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
  aardvark-dns buildah catatonit conmon containernetworking-plugins docker-compose fuse-overlayfs
  golang-github-containers-common golang-github-containers-image libllvm17t64 libslirp0 libsubid4 netavark passt
  podman python3-compose python3-docker python3-dockerpty python3-docopt python3-dotenv python3-netifaces
  python3-packaging python3-texttable python3-websocket slirp4netns uidmap
Veuillez utiliser « sudo apt autoremove » pour les supprimer.
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```

Activer et démarrer Docker

Une fois Docker installé, nous avons vérifié son bon fonctionnement avec les commandes suivantes :

```
sirine@sirine:~$ sudo systemctl enable --now docker
sirine@sirine:~$ docker --version
Docker version 26.1.3, build 26.1.3-0ubuntu1~24.04.1
sirine@sirine:~$ docker run hello-word
docker: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http
://%2Fvar%2Frun%2Fdocker.sock/_ping": dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
sirine@sirine:~$
```

Vérifier l'installation

```
sirine@sirine:~$ sudo usermod -aG docker $USER
sirine@sirine:~$ newgrp docker
sirine@sirine:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:e0b569a5163a5e6be84e210a2587e7d447e08f87a0e90798363fa44a0464a1e8
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/
```

Étape 2 : Créer une image Docker avec Apache2

Un dossier de travail a été créé pour stocker le **Dockerfile**, qui est un fichier texte contenant les instructions nécessaires à la création de l'image Docker.

Créer un dossier de travail

```
sirine@sirine:~$ mkdir ~/docker-apache
sirine@sirine:~$ cd ~/docker-apache
```

Créer un fichier Dockerfile

```
sirine@sirine:~/docker-apache$ nano Dockerfile
sirine@sirine:~/docker-apache$
```

Le **Dockerfile** a été rédigé comme suit :

Cet **Dockerfile** définit les étapes suivantes :

1. Utilisation d'une image de base Ubuntu 22.04.
2. Installation d'Apache2.
3. Définition du répertoire de travail dans le conteneur.
4. Exposition du port 80 pour accéder au serveur web.
5. Lancement d'Apache en mode premier plan pour que le conteneur fonctionne en continu.

Étape 3 : Construire et exécuter l’image Docker

L'image Docker a été construite à l'aide de la commande suivante :

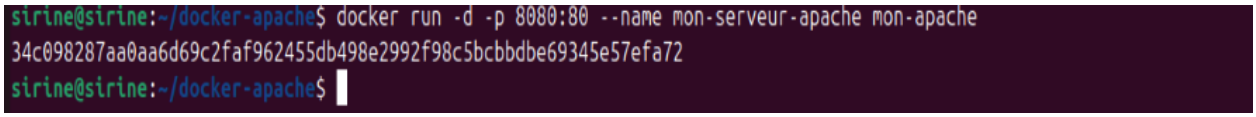
```
sirine@sirine:~/docker-apache$ docker build -t mon-apache .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  2.048kB
Step 1/5 : FROM ubuntu:22.04
22.04: Pulling from library/ubuntu
9cb31e2e37ea: Pull complete
Digest: sha256:ed1544e454989078f5dec1bfdabd8c5cc9c48e0705d07b678ab6ae3fb61952d2
Status: Downloaded newer image for ubuntu:22.04
 ---> a24be041d957
Step 2/5 : RUN apt update && apt install -y apache2
 ---> Running in 4d89c5a71f38

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [1230 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2610 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [3612 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [45.2 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [53.3 kB]
```

Le -t mon-apache indique que l'image sera nommée mon-apache. Après la construction, l'image a été vérifiée avec :

```
sirine@sirine:~/docker-apache$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
mon-apache    latest    133ef6cfede4   21 seconds ago   245MB
mon-image     latest    03a4b6b4003d   7 days ago       129MB
ubuntu        22.04     a24be041d957   3 weeks ago      77.9MB
hello-world   latest    74cc54e27dc4   4 weeks ago      10.1kB
ubuntu        20.04     6013ae1a63c2   4 months ago     72.8MB
sirine@sirine:~/docker-apache$
```

Étape 4 : Lancer un conteneur Apache2

Le conteneur a été lancé en utilisant l'image Docker nouvellement créée. La commande suivante a été exécutée :



Cette commande crée un conteneur nommé mon-serveur-apache, lie le port 80 du conteneur au port 8080 de la machine hôte, et le démarre en arrière-plan (-d).

Le bon fonctionnement du conteneur a été vérifié avec :

```
sirine@sirine:~/docker-apache$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS
     NAMES
34c098287aa0   mon-apache     "apachectl -D FOREGR…"   41 seconds ago   Up 40 seconds   0.0.0.0:8080->80/tcp, :::8080->80/
tcp   mon-serveur-apache
sirine@sirine:~/docker-apache$
```

Ensuite, l'interface Apache2 a été accessible via l'URL **http://localhost:8080** sur le navigateur.

Étape 5 : Personnaliser la page web

Le serveur Apache a été personnalisé en modifiant le fichier index.html du serveur web Apache. Un message personnalisé a été ajouté à la page d'accueil :

Accéder au conteneur en ligne de commande

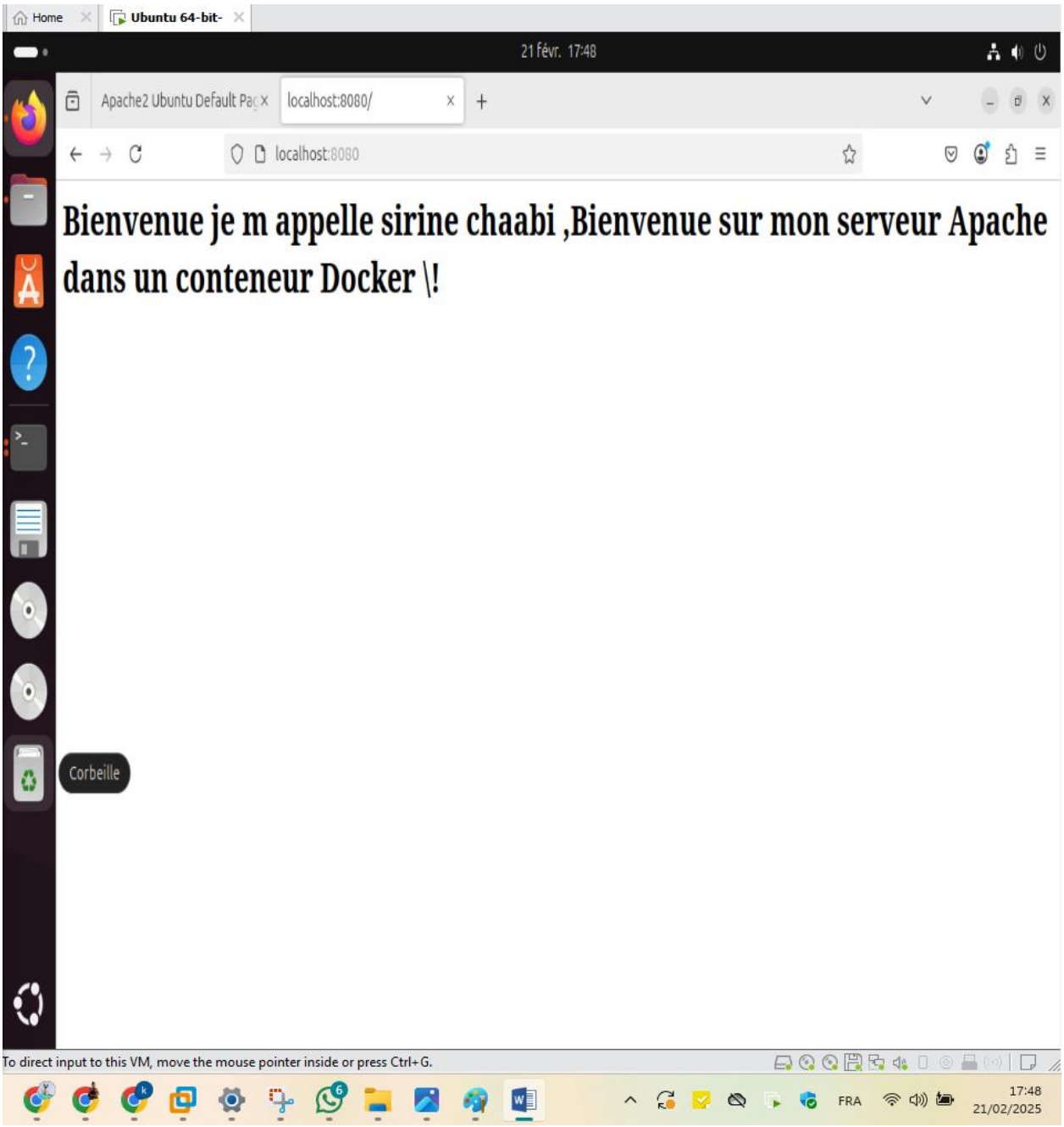
```
root@34c098287aa0:/var/www/html# echo "<h1>Bienvenue je m appelle sirine chaabi ,Bienvenue sur mon serveur Apache dans u
n conteneur Docker \!</h1>" > /var/www/html/index.html
```

Modifier la page d'accueil : echo "<h1>Bienvenue sur mon serveur Apache dans un conteneur Docker !</h1>" > /var/www/html/index.html

Vérifier le fichier : cat /var/www/html/index.html

```
root@34c098287aa0:/var/www/html# cat /var/www/html/index.html
<h1>Bienvenue je m appelle sirine chaabi ,Bienvenue sur mon serveur Apache dans un conteneur Docker \!</h1>
root@34c098287aa0:/var/www/html#
```

Cela a permis de valider que la personnalisation du serveur fonctionnait correctement. Le message a été affiché après avoir rafraîchi la page http://localhost:8080.



Étape 6 : Gérer les Conteneurs et Images

Arrêter un conteneur : docker stop mon-serveur-apache

Redémarrer un conteneur : docker start mon-serveur-apache

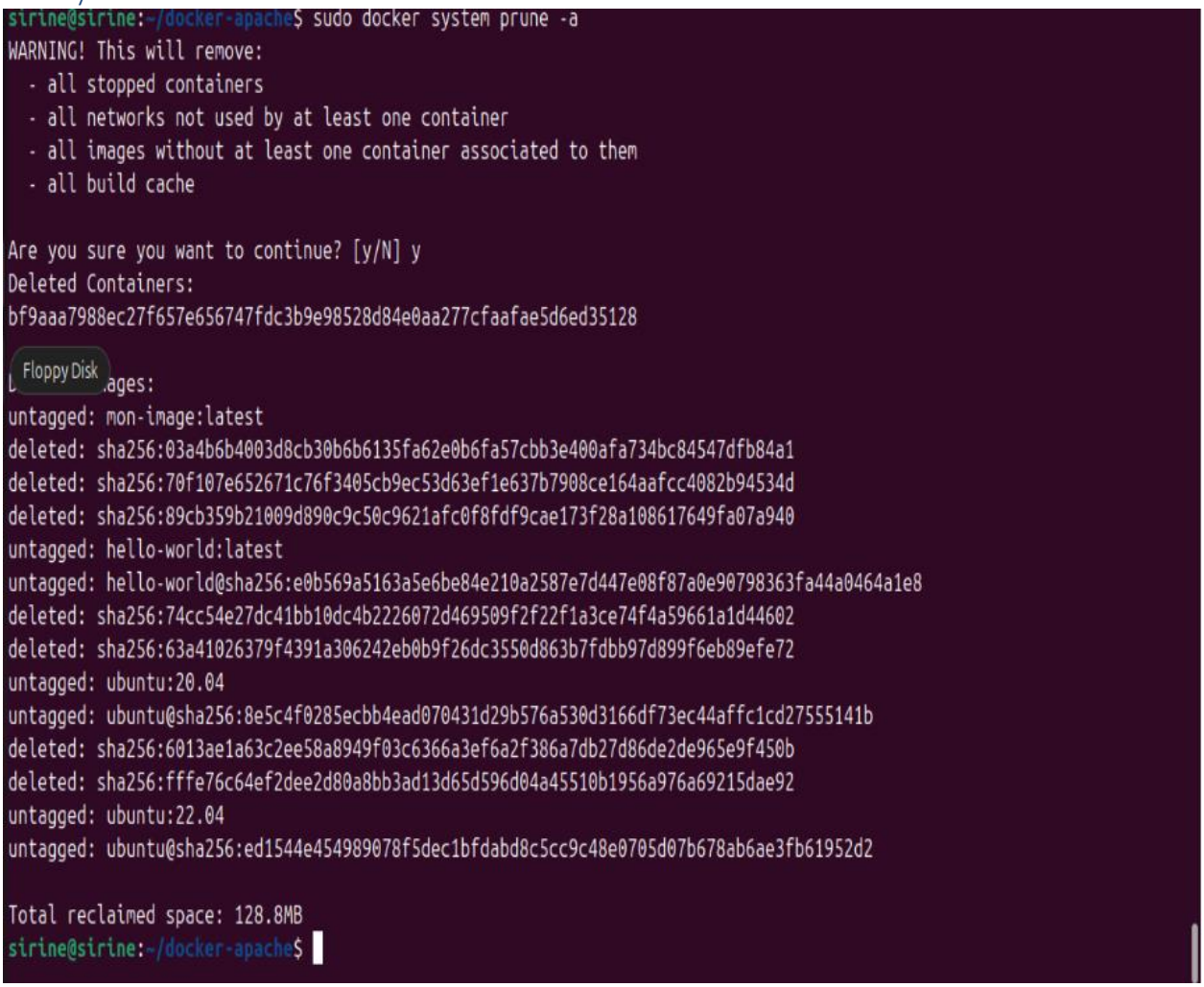
```
sirine@sirine:~/docker-apache$ sudo docker stop mon-serveur-apache
Error response from daemon: cannot stop container: mon-serveur-apache: permission denied
sirine@sirine:~/docker-apache$ sudo docker start mon-serveur-apache
mon-serveur-apache
sirine@sirine:~/docker-apache$
```

Supprimer un conteneur : docker rm mon-serveur-apache

Supprimer une image : docker rmi mon-apache

```
-virtual-machine:~/docker-apache$ docker rm mon-serveur-apache
veur-apache
-virtual-machine:~/docker-apache$ docker rmi mon-apache
d: mon-apache:latest
: sha256:a45ab5695a6b4d6d652eb309463543cbf32a273778cfa0c13fe96d3294

: sha256:bf225ee723053f42f19172b0638b3de2f40fdf53a23a8b742a76e750b5

: sha256:d8832f155d52fcc61dc6381f22d9467bff36672a6944b17f5f8a023aa6

: sha256:0ef4ebec542919adab506f82f05cebcbf5a8c0f609ce5c9f6af37eec21

: sha256:402dae0a903a0de877b20576cc24797fd394bb7c5de78a79bc0c009bc0

-virtual-machine:~/docker-apache$
```

Nettoyer Docker



Conclusion

Ce projet a démontré la puissance et la flexibilité de Docker pour déployer rapidement des applications dans des environnements isolés. En utilisant Docker, nous avons pu créer un serveur web Apache2 fonctionnel en quelques étapes simples. Cette méthode peut être étendue pour déployer d'autres applications dans des conteneurs et gérer des environnements complexes de manière efficace.