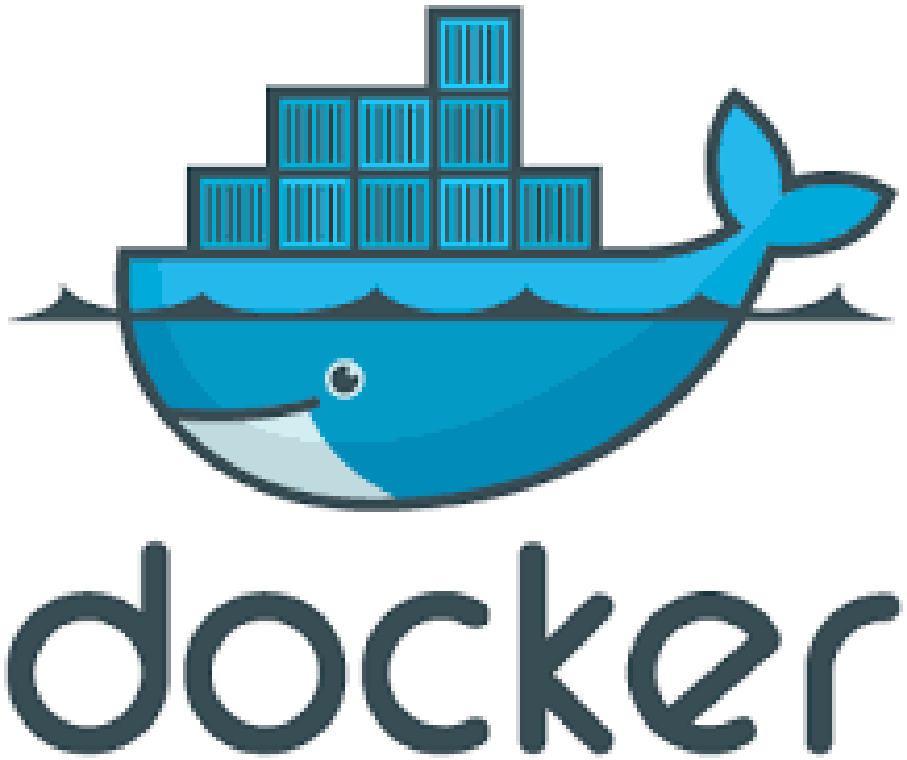
# Docker



# SOMMAIRE

# I.Présentation de docker

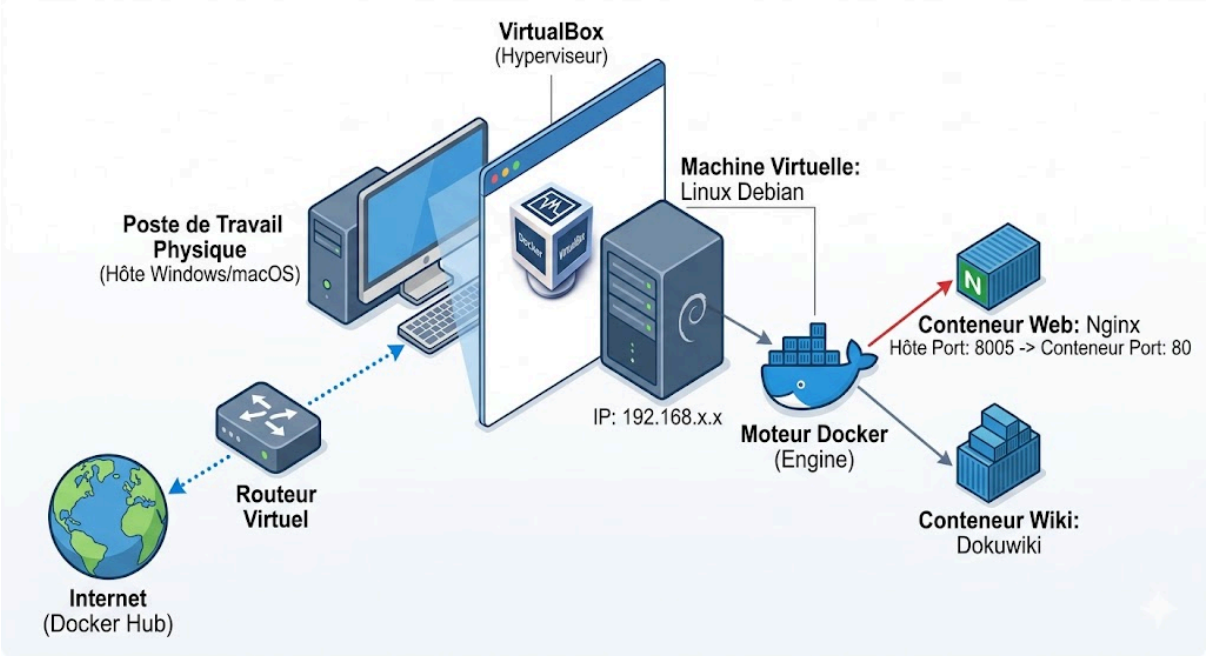

Docker est une plateforme de **contenurisation** qui permet de packager une application et toutes ses dépendances (code, bibliothèques, configurations) dans une unité isolée appelée **conteneur**.

Contrairement à la virtualisation classique qui nécessite un système d'exploitation complet pour chaque machine virtuelle, Docker partage le **légers**, rapides à démarrer et **économes en ressources** (RAM et CPU).

Pour un administrateur ou un développeur, l'intérêt majeur réside dans la **portabilité** : on crée une **image** (un modèle figé) qui garantit que l'application fonctionnera de manière identique sur n'importe quel environnement, qu'il s'agisse d'un poste de développement ou d'un serveur de production.

En résumé, Docker simplifie le **déploiement**, assure l'**isolation des services** et optimise la gestion des infrastructures en permettant de faire tourner de nombreux services différents sur une même machine sans qu'ils ne créent de conflits entre eux.

# II.Architecture et environnement technique de l'infrastructure




Tu as raison, restons-en aux faits pour cette partie. On se concentre uniquement sur la description de la structure technique sans anticiper sur la suite.

Voici le titre et la rédaction pour ton **Grand 2**, de manière simple et précise :

---

Pour la mise en place de cette solution, j'ai utilisé une infrastructure virtualisée organisée en plusieurs couches, comme illustré sur le schéma ci-dessus :

L'élément de base est mon **poste de travail physique**. Pour travailler dans un environnement isolé et stable, j'ai utilisé l'hyperviseur **VirtualBox** afin d'y créer une **machine virtuelle Debian**. C'est cette machine qui sert de serveur hôte pour accueillir l'ensemble des services.

Afin de permettre la communication avec l'extérieur, la machine virtuelle est reliée à un **routeur virtuel**. Ce dernier lui offre un accès à **Internet**, ce qui est indispensable pour récupérer les paquets d'installation et les modèles d'applications sur le **Docker Hub**.

Au sein de ce serveur Debian, le **moteur Docker (Docker Engine)** assure la gestion des **conteneurs**. Contrairement à une installation classique, chaque application est encapsulée dans une unité logicielle indépendante. Cette architecture garantit l'**isolation des services** : chaque conteneur possède son

propre environnement, évitant ainsi tout conflit entre les logiciels installés sur la même machine.

# III.Objectif : Installer Docker et déployer des services Web

L'**objectif** de ce TP est d'apprendre à utiliser la technologie des conteneurs pour mettre en ligne des applications rapidement. Le but est de passer d'une machine Debian vide à un serveur capable d'héberger plusieurs services isolés (comme un Wiki ou un serveur Web) de manière professionnelle.

Pour réussir ce TP, voici ce que je dois réaliser :

- **Installer proprement Docker** sur Linux en utilisant un script automatisé.
- **Vérifier le bon fonctionnement** du moteur Docker (test du service et image "Hello-world").
- **Déployer des applications** concrètes (Dokuwiki et Nginx).
- **Personnaliser un serveur** (Nginx) pour qu'il affiche ma propre page web au lieu de la page d'accueil par défaut.

**Les prérequis pour commencer :**

- Une machine virtuelle **Debian** sur VirtualBox.
- Une connexion **Internet** (via le routeur virtuel).
- Les droits d'administrateur (**root**).

# IV.Installation du moteur Docker et premiers tests de fonctionnement

## 4.1.Installation automatisée de docker

Pour installer Docker proprement, j'ai utilisé le script d'installation automatique fourni par Docker, j’ai donc tapé la commande **wget https://get.docker.com/** puis **bash index.html**.

```
root@buster:~# wget https://get.docker.com/
```

```
root@buster:~# bash index.html
```

Pour m'assurer que le service Docker est correctement démarré et prêt à l'emploi, j'ai utilisé la commande : **systemctl status docker**

```
root@buster:~# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2026-04-21 20:03:42 CEST; 42s ago
     Docs: https://docs.docker.com
 Main PID: 12127 (dockerd)
    Tasks: 8
   Memory: 41.0M
   CGroup: /system.slice/docker.service
           └─12127 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

avril 21 20:03:41 buster systemd[1]: Starting Docker Application Container Engine...
avril 21 20:03:41 buster dockerd[12127]: time="2026-04-21T20:03:41.774819708+02:00" level=info msg=
avril 21 20:03:41 buster dockerd[12127]: time="2026-04-21T20:03:41.971902567+02:00" level=info msg=
avril 21 20:03:42 buster dockerd[12127]: time="2026-04-21T20:03:42.235147513+02:00" level=info msg=
avril 21 20:03:42 buster dockerd[12127]: time="2026-04-21T20:03:42.260310815+02:00" level=warning m
avril 21 20:03:42 buster dockerd[12127]: time="2026-04-21T20:03:42.260555937+02:00" level=info msg=
avril 21 20:03:42 buster dockerd[12127]: time="2026-04-21T20:03:42.260698992+02:00" level=info msg=
avril 21 20:03:42 buster dockerd[12127]: time="2026-04-21T20:03:42.319173003+02:00" level=info msg=
avril 21 20:03:42 buster systemd[1]: Started Docker Application Container Engine.
lines 1-19/19 (END)
```

## 4.2. Vérification du bon fonctionnement avec un conteneur de test

Afin de valider que Docker est capable de récupérer des images et de lancer des conteneurs, j'ai exécuté le conteneur de test officiel "Hello-World".

Commande lancée : **docker run hello-world**

```
root@buster:~# docker run hello-world
```

Après avoir testé le "Hello-World", j'ai vérifié les images présentes sur ma machine ainsi que les conteneurs en cours d'exécution pour m'assurer que le système de nettoyage et de stockage de Docker fonctionne.

Commande pour lister les images : **docker images**

```
root@buster:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
hello-world   latest    e2ac70e7319a   4 weeks ago   10.1kB
root@buster:~#
```

Commande pour lister les conteneurs (actifs et éteints) : **docker ps -a**

```
root@buster:~# docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED         STATUS                     PORTS     NAMES
cc28c6176bd6   hello-world   "/hello"   5 minutes ago   Exited (0) 5 minutes ago             strange
_jemison
root@buster:~# _
```

# V.Déploiement de services et administration

## 5.1. Déploiement de DokuWiki

J'ai procédé au déploiement de plusieurs instances de DokuWiki pour tester la gestion des ports et la persistance.

- **Lancement des conteneurs : docker run -d -p 8001:80 --name wiki01 mprasil/dokuwiki**

```
root@buster:~# docker run -d -p 8001:80 --name wiki01 mprasil/dokuwiki
```

- **Vérification :** L'accès se fait via l'adresse IP de la VM sur le port 8001 (ex:**http://192.168.1.57:8001** ).
- **Résultat :** Le service est fonctionnel et accessible via un navigateur web.

## 5.2.Création d'un serveur Web personnalisé (Apache sur Ubuntu)

Pour aller plus loin que le simple déploiement d'images prêtes à l'emploi, j'ai configuré mon propre serveur Web à l'intérieur d'un conteneur Ubuntu.

```
root@bullseye:~# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
20043066d3d5: Pull complete
06808451f0d6: Download complete
Digest: sha256:c35e29c9450151419d9448b0fd75374fec4fff364a27f176fb458d472dfc9e54
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
root@bullseye:~#
```

```
root@bullseye:~# docker image ls
                                                                    Info →   U  In
IMAGE                     ID             DISK USAGE   CONTENT SIZE   EXTRA
hello-world:latest        f7931603f70e       25.9kB         9.52kB    U
mprasil/dokuwiki:latest   507db13c7bda        511MB          125MB    U
ubuntu:latest             c35e29c94501        119MB         31.7MB
root@bullseye:~#
```

```
root@bullseye:~# docker run ubuntu
```

```
root@bullseye:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@bullseye:~# _
```

Pour lancer un container avec l'image Ubuntu et pouvoir l'utiliser, j'ai utiliser le mode interactif via les options -ti (- va créer un terminal dans le container et -i va permettre de se connecter à celui-ci.

```
root@bullseye:~# docker run -ti ubuntu /bin/bash
root@b18b69505233:/# _
```

```
root@b18b69505233:/# ls -l
total 48
lrwxrwxrwx   1 root root    7 Apr 22  2024 bin -> usr/bin
drwxr-xr-x   2 root root 4096 Apr 22  2024 boot
drwxr-xr-x   5 root root  360 Dec  4 12:49 dev
drwxr-xr-x   1 root root 4096 Dec  4 12:49 etc
drwxr-xr-x   3 root root 4096 Oct 13 14:09 home
lrwxrwxrwx   1 root root    7 Apr 22  2024 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Apr 22  2024 lib64 -> usr/lib64
drwxr-xr-x   2 root root 4096 Oct 13 14:02 media
drwxr-xr-x   2 root root 4096 Oct 13 14:02 mnt
drwxr-xr-x   2 root root 4096 Oct 13 14:02 opt
dr-xr-xr-x 140 root root    0 Dec  4 12:49 proc
drwx------   2 root root 4096 Oct 13 14:09 root
drwxr-xr-x   4 root root 4096 Oct 13 14:09 run
lrwxrwxrwx   1 root root    8 Apr 22  2024 sbin -> usr/sbin
drwxr-xr-x   2 root root 4096 Oct 13 14:02 srv
dr-xr-xr-x  13 root root    0 Dec  4 12:49 sys
drwxrwxrwt   2 root root 4096 Oct 13 14:09 tmp
drwxr-xr-x  12 root root 4096 Oct 13 14:02 usr
drwxr-xr-x  11 root root 4096 Oct 13 14:09 var
root@b18b69505233:/# _
```

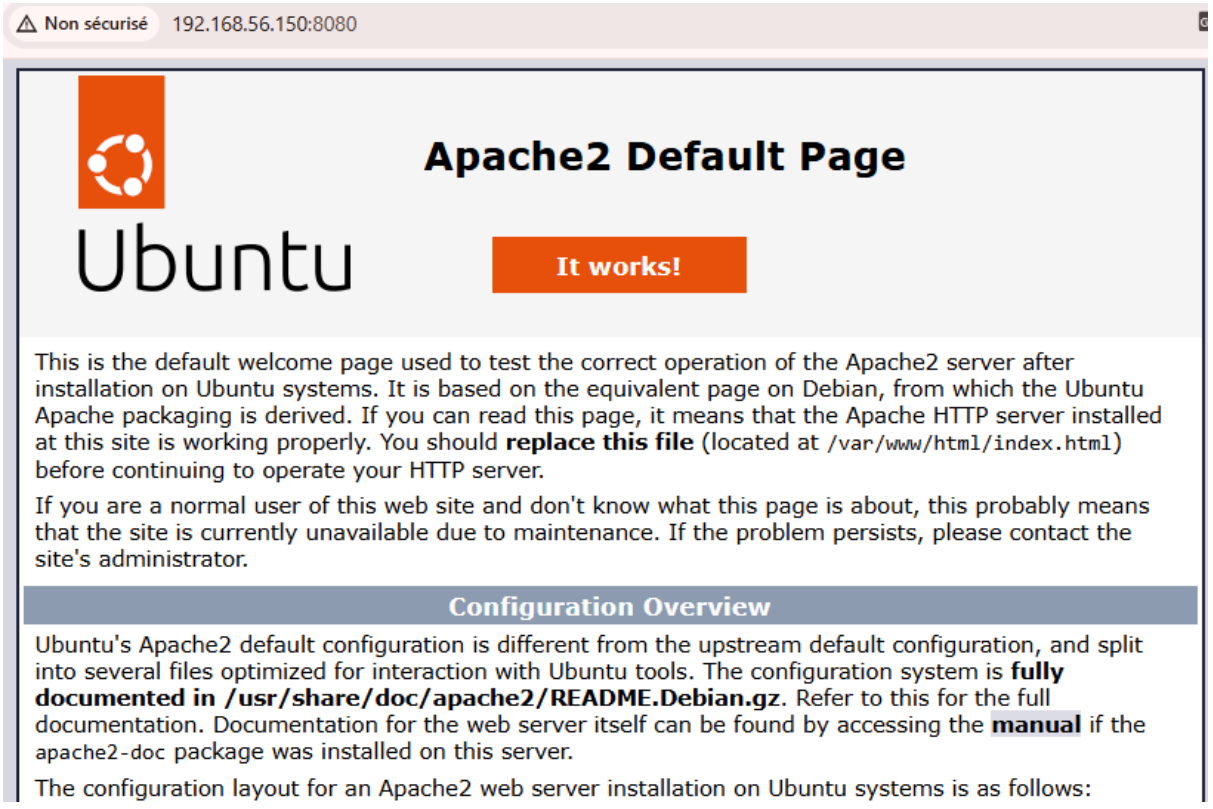
```
root@b18b69505233:/# apt update
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [33.1 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1700 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1182 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [2796 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Get:10 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1808 kB]
Get:11 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [331 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [35.9 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1943 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [2059 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2929 kB]
Get:17 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [34.3 kB]
Get:18 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [49.4 kB]
Fetched 35.0 MB in 8s (4571 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
root@b18b69505233:/#
```

```
root@a4e02904d5e2:/# apt update
Get:1 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [331 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1700 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1182 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [2796 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [33.1 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1808 kB]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1943 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [35.9 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [2059 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2929 kB]
Get:17 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [34.3 kB]
Get:18 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [49.4 kB]
Fetched 35.0 MB in 8s (4438 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```

Ensuite j’ai installer apache en tapant la commande suivante:

```
root@b18b69505233:/# apt install apache2
```

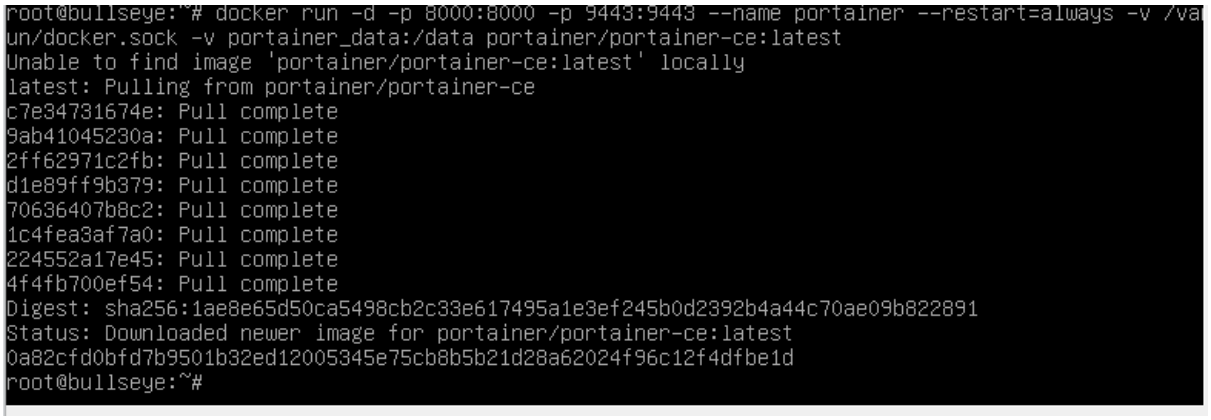
Puis j’ai lancer le conteneur Ubuntu



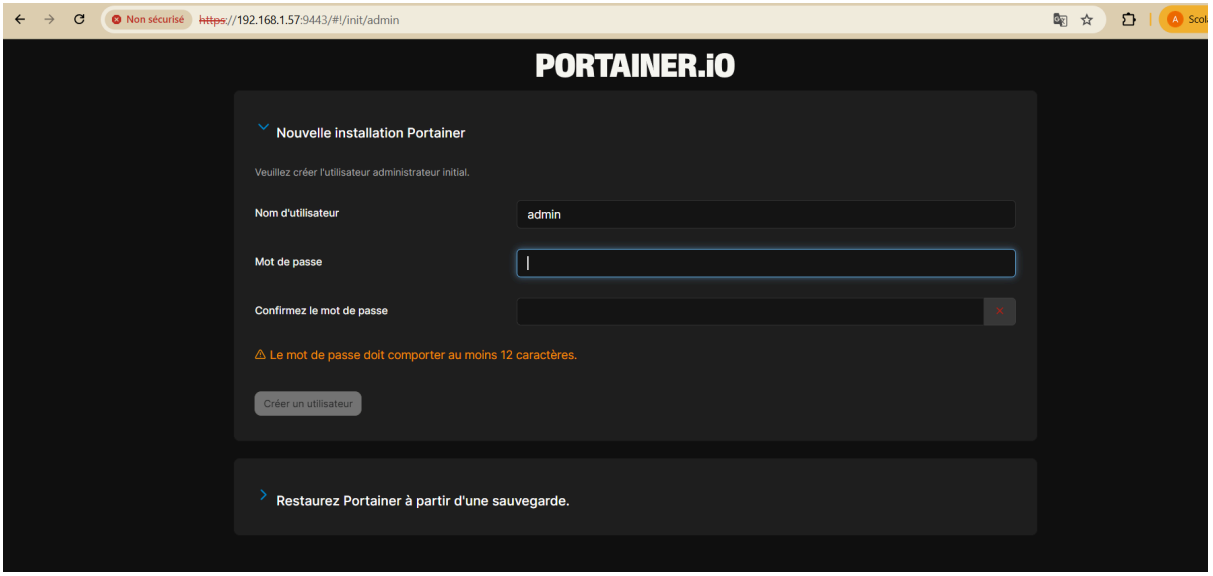
# VI.Installation et gestion de portainer

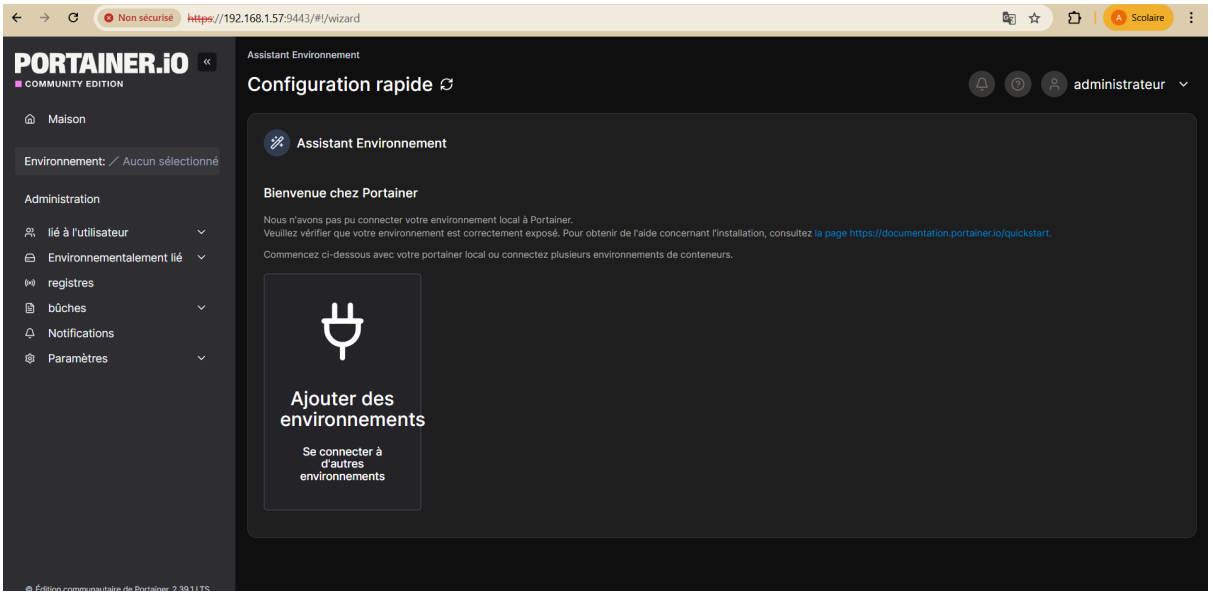
Pour centraliser la gestion de mes conteneurs et répondre aux besoins d'architectures plus complexes, j'ai déployé **Portainer**, une interface graphique intuitive pour Docker.

J'ai créé un volume pour la persistance des données et lancé le conteneur Portainer : **docker volume create portainer_data**
**docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest**


Une fois le conteneur lancé avec succès, Portainer est accessible via une interface web sécurisée.




## VII.Création de deux stacks différents qui communiquent

Dans un premier temps, j'ai créé une stack nommée **"web-stack"**. Je suis allé dans l'onglet **Stacks**, j'ai cliqué sur **Add stack**, puis j'ai utilisé l'éditeur web pour renseigner la configuration suivante : version: '3'
services:
  serveur-web:
    image: nginx:latest
    ports:
      - "8080:80"
    networks:

```
      - reseau-web
  ma-page:
    image: tutum/hello-world
    networks:
      - reseau-web
networks:
  reseau-web:
```

Dans un second temps, j'ai créé une stack nommée **"db-stack"** dédiée à la gestion des données. Pour cela, j'ai cliqué à nouveau sur **Add stack** et j'ai renseigné la configuration YAML suivante :

YAML

```
version: '3.8'
services:
  base-de-donnees:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: mon-super-password
    networks:
      - reseau-db
  gestionnaire-db:
    image: adminer:latest
    ports:
      - "8081:8080"
    networks:
      - reseau-db
networks:
  reseau-db:
```

Pour faire simple, ces deux stacks me permettent de séparer proprement mon infrastructure en deux blocs : d'un côté la "web-stack" qui gère l'affichage avec un serveur Nginx et une page de test accessibles sur le port 8080, et de l'autre la "db-stack" qui s'occupe du stockage avec une base de données MySQL sécurisée et une interface de gestion Adminer sur le port 8081. En utilisant des réseaux isolés pour chaque groupe, j'ai créé un environnement organisé et sécurisé où chaque service reste à sa place tout en étant facile à piloter depuis l'interface, ce qui transforme mon serveur en une véritable plateforme d'hébergement professionnelle.

# VIII.Problèmes rencontrés

Durant la mise en place de l'infrastructure, plusieurs obstacles techniques ont été identifiés et corrigés :

- **Défaut de liaison avec le moteur Docker :** Initialement, l'onglet **Stacks** était invisible dans l'interface Portainer. Ce problème était dû à une mauvaise configuration de l'environnement, celui-ci ayant été configuré par erreur en mode "Agent distant" au lieu d'utiliser le socket local.
    - **Solution :** Suppression des environnements erronés et création d'une connexion via **Socket (Douille)** pointant vers /var/run/docker.sock.
- **Erreur de permission sur le socket Unix :** Lors de la tentative de connexion au démon Docker, Portainer affichait une erreur "Échec : Impossible de se connecter". Le fichier système du socket n'autorisait pas le conteneur Portainer à lire les informations du moteur Docker.
    - **Solution :** Utilisation de la commande sudo chmod 666 /var/run/docker.sock sur le terminal de la machine hôte pour ouvrir les droits d'accès.
- **Montage de volume manquant :** Le conteneur Portainer ne parvenait pas à persister ses données ou à voir les conteneurs existants.
    - **Solution :** Recréation du conteneur Portainer avec l'argument -v /var/run/docker.sock:/var/run/docker.sock, assurant ainsi le lien physique entre l'hôte et l'interface de gestion.

# IX.Conclusion

Ce projet a permis de mettre en place une solution complète de gestion de conteneurs via Portainer. Malgré les difficultés initiales liées aux droits d'accès du système Linux et à la configuration des environnements, l'infrastructure est désormais parfaitement opérationnelle.

Le déploiement des deux stacks (web-stack et db-stack) démontre la puissance de l'outil pour orchestrer des services complexes de manière isolée et sécurisée. Cette installation constitue une base solide pour l'hébergement d'applications web modernes, offrant à la fois une interface visuelle intuitive et la flexibilité de Docker Compose pour la gestion du cycle de vie des conteneurs.