

# Fiche de Procédure

**DOCKER => Linux**

## **Sommaire**

### **I. Introduction à Docker**

- Qu'est-ce que Docker ?
- Avantages de Docker

### **II. Installation de Docker**

- Prérequis
- Installation de Docker (LINUX)

### **III. Utilisation de Docker**

- Gestion des conteneurs (démarrage, arrêt, suppression)
- Commandes de base de Docker + Création d'un conteneur Docker

### **IV. Gestion des images Docker**

- Gestion des images (démarrage, arrêt, suppression)
- Création d'une image Docker

### **V. Gestion de Docker Compose**

- Utilisation de Docker Compose pour déployer un conteneur NGINX
- Commandes de conteneurs Docker via Docker Compose
- Configuration d'un environnement WordPress + MySQL avec Docker Compose

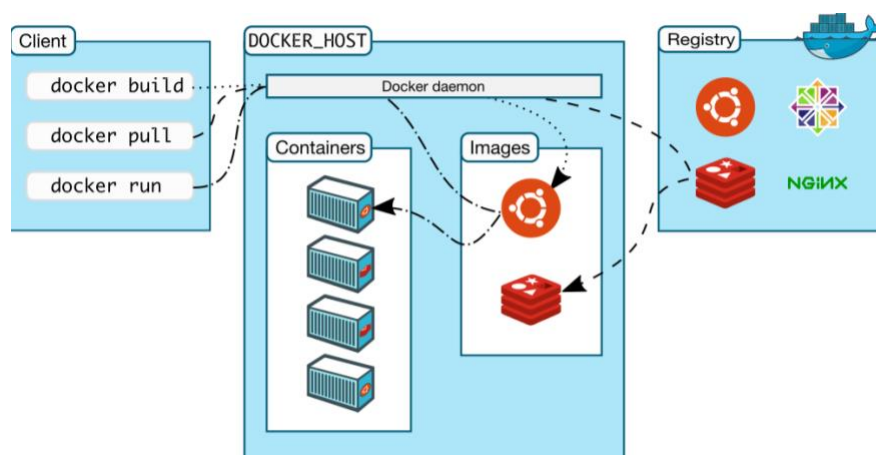
## I. Introduction à Docker

### Qu'est-ce que Docker ?

Docker est une plateforme ouverte pour le développement, le déploiement et l'exécution d'applications. Il permet d'embarquer et d'exécuter une application dans un environnement cloisonné appelé conteneur. Ce cloisonnement permet d'exécuter plusieurs conteneurs simultanément sur un hôte donné. À la différence de la virtualisation où la machine virtuelle contient un système d'exploitation complet, les outils associés et l'application hébergée, le conteneur ne contient que les bibliothèques et les outils nécessaires à l'exécution de l'application, et il fonctionne directement dans le noyau de la machine hôte.

Un conteneur est une unité standard de logiciel, qui embarque le code et toutes ses dépendances, afin que l'application « conteneurisée » fonctionne normalement et de façon fiable, quelle que soit la machine hôte. L'image d'un conteneur, présente sur le système, devient un conteneur au moment de l'exécution. Dans le cas d'un conteneur Docker, l'image devient un conteneur lorsqu'elle fonctionne sur le Docker daemon, ou Docker Engine. L'application « conteneurisée » fonctionnera toujours de la même manière, quel que soit l'hôte. Docker peut s'exécuter soit sur un serveur, soit dans un Cloud privé, ou public, apportant une capacité de portabilité et de flexibilité dans le déploiement et l'exécution de l'application.

Docker repose sur une architecture client-serveur (image en dessous). Un client Docker communique avec le Docker daemon pour gérer la construction et le fonctionnement des conteneurs Docker ainsi que la distribution des images Docker issues d'un Docker Registry, public ou privé.



## I. Introduction à Docker

### Avantages de Docker

- Isolation : Docker permet d'isoler une application dans un conteneur, ce qui évite les conflits de dépendances et les interférences avec d'autres applications sur le même système.
- Portabilité : Les conteneurs Docker sont portables et peuvent être exécutés sur n'importe quelle plateforme compatible avec Docker, ce qui facilite le déploiement et la distribution d'applications.
- Légèreté : Les conteneurs Docker sont plus légers que les machines virtuelles, car ils partagent le même noyau avec le système hôte, ce qui les rend plus efficaces en termes de ressources système.
- Rapidité : Docker permet de créer, de démarrer et d'arrêter rapidement des conteneurs, ce qui facilite les tests et le développement d'applications.
- Gestion des versions : Docker facilite la gestion des différentes versions d'une application en permettant de créer et de conserver des images Docker pour chaque version.
- Évolutivité : Docker facilite l'évolutivité d'une application en permettant de créer plusieurs instances de conteneurs qui peuvent être gérées à l'aide d'outils tels que Docker **Swarm ou Kubernetes**.
- Flexibilité : Docker permet de créer des environnements de développement, de test et de production cohérents, ce qui facilite la gestion de l'infrastructure et la collaboration entre les équipes de développement et d'exploitation.

En résumé, Docker est une technologie très populaire et largement utilisée qui permet d'isoler, de distribuer et de gérer des applications de manière efficace et flexible, en facilitant la portabilité, l'évolutivité, la gestion des versions et la collaboration entre les équipes de développement et d'exploitation.

**Swarm ou Kubernetes** sont deux outils d'orchestration de conteneurs qui permettent de gérer de manière efficace et évolutive des environnements de conteneurs.

## II. Installation de Docker

### Prérequis

- Une machine (virtuelle) sous Linux avec (accès par pont)
- RAM : Docker nécessite au moins 2 Go de RAM pour fonctionner
- Accès root : Vous devez être connecté en tant qu'utilisateur root ou avoir des privilèges administratifs pour installer Docker.

### Installation de Docker (LINUX)

Pour ce faire, suivez les indications de la documentation officielle, disponible à cette adresse :

<https://docs.docker.com/engine/install/debian/>

Lisez bien l'intégralité des informations fournies avec les commandes. Toutes les commandes ne sont pas à exécuter, selon les situations !

Une fois docker installé, exécutez la commande suivante :

```
root@debian-xfce:~# docker run hello-world
```

Si l'installation s'est correctement déroulée, vous devriez lire un message tel que :

```
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/
```

### III. Utilisation de Docker

#### Gestion des conteneurs Docker

##### Démarrer un conteneur :

Pour lancer un conteneur, il faut utiliser la commande suivante :

```
root@debian-xfce:~# docker run -d -p 8080:80 nginx  
7d39a5b7903bb2ef3acb00de2bcd0e1ec4e95b32e35b8ceb214d4982a84fabdf
```

Cette commande va créer un nouveau conteneur NGINX en arrière-plan, qui écoute sur le port interne 80 et est accessible depuis le port externe 8080 de la VM/Machine.

##### Stopper un conteneur NGINX.

Pour arrêter le conteneur NGINX, il faut utiliser la commande suivante dans le terminal :

```
docker stop <nom_ou_ID_conteneur>
```

Remplacer <nom\_ou\_ID\_conteneur> par le nom ou l'ID du conteneur NGINX que noté précédemment.

Cette commande va arrêter le conteneur NGINX. Il est possible de vérifier que le conteneur est bien arrêté en utilisant la commande `docker ps -a` pour lister tous les conteneurs de votre machine, y compris les conteneurs arrêtés. Si le conteneur NGINX ne figure pas dans la liste des conteneurs en cours d'exécution, cela signifie qu'il a été arrêté avec succès.

### III. Utilisation de Docker

#### Gestion des conteneurs Docker

#### Supprimez complètement le conteneur NGINX.

Pour supprimer complètement le conteneur NGINX, utiliser la commande suivante dans le terminal :

```
docker rm <nom_ou_ID_conteneur>
```

```
root@debian-xfce:~# docker rm upbeat_hermann
upbeat_hermann
root@debian-xfce:~#
```

Remplacer <nom\_ou\_ID\_conteneur> par le nom ou l'ID du conteneur NGINX que vous avez noté précédemment.

Cette commande va supprimer le conteneur NGINX de votre machine. On peut vérifier que le conteneur a bien été supprimé en utilisant la commande `docker ps -a` pour lister tous les conteneurs sur votre machine, y compris les conteneurs arrêtés.

```
root@debian-xfce:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
7d39a5b7903b   nginx     "/docker-entrypoint..." 23 minutes ago Up 23 minutes 0.0.0.0:8080->80/tcp,
:::8080->80/tcp trusting_ptolemy
root@debian-xfce:~#
```

Si le conteneur NGINX ne figure plus dans la liste des conteneurs, cela signifie qu'il a été supprimé avec succès.

### III. Utilisation de Docker

#### Commandes de base de Docker

Récupérer tout d'abord l'image du service NGINX : `docker pull nginx`

Cette commande va télécharger l'image de NGINX depuis le Docker Hub et la stocker localement sur la machine.

Pour vérifier que l'image a bien été récupérée et qu'elle apparaît dans la liste des images téléchargées, vous pouvez exécuter la commande suivante :

```
root@debian-xfce:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx                latest             3f8a00f137a0       13 days ago        142MB
jetbrains/youtrack   2022.1.45133       d13474291c94       10 months ago      721MB
hello-world          latest             feb5d9fea6a5       17 months ago      13.3kB
```

Cette commande va afficher la liste des images Docker stockées localement sur votre machine. Si l'image NGINX apparaît dans la liste, cela signifie que vous l'avez bien récupérée depuis le Docker Hub.



### III. Utilisation de Docker

#### Commandes de base de Docker

Lancer un conteneur basé sur l'image NGINX, en arrière-plan, avec un partage du port interne 80 du conteneur sur le port externe 8080 de la VM/Machine :

Pour lancer un conteneur basé sur l'image NGINX en arrière-plan avec un partage du port interne 80 du conteneur sur le port externe 8080 de la VM, il faut utiliser la commande suivante :

```
root@debian-xfce:~# docker run -d -p 8080:80 nginx  
7d39a5b7903bb2ef3acb00de2bcd0e1ec4e95b32e35b8ceb214d4982a84fabdf
```

Cette commande va créer un nouveau conteneur NGINX en arrière-plan, qui écoute sur le port interne 80 et est accessible depuis le port externe 8080 de la VM/Machine.

Pour vérifier que l'on peut accéder à la page d'accueil de NGINX depuis le navigateur de l'ordinateur hôte, il faut récupérer l'adresse IP de la machine virtuelle Debian/machine physique.

Ensuite, ouvrez le navigateur web sur l'ordinateur hôte et entrer l'URL suivante dans la barre d'adresse :

**http://<adresse\_IP>:8080**



## 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](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

### III. Utilisation de Docker

#### Commandes de base de Docker

#### Récupérer la liste des conteneurs créés sur l'installation Docker

Utiliser la commande suivante dans le terminal : `docker ps -a`

Cette commande va afficher la liste de tous les conteneurs sur votre machine, y compris les conteneurs qui ne sont plus en cours d'exécution.

#### Vérifier que le conteneur NGINX est en cours d'exécution

Utiliser la commande suivante dans le terminal :

```
root@debian-xfce:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS
PORTS         NAMES
7d39a5b7903b   nginx     "/docker-entrypoint...." About a minute ago Up About
a minute      0.0.0.0:8080->80/tcp, :::8080->80/tcp trusting_ptolemy
f86d3f81f80e   nginx     "/docker-entrypoint...." 16 minutes ago Up 16 min
utes          0.0.0.0:80->80/tcp, :::80->80/tcp upbeat hermann
```

Cette commande va afficher la liste des conteneurs en cours d'exécution sur votre machine.

Si le conteneur NGINX est en cours d'exécution, une ligne avec son nom et son ID dans la liste des conteneurs. Il est possible de vérifier le statut actuel du conteneur dans la colonne "STATUS". Si le conteneur est en cours d'exécution, le statut devrait être "Up" avec une durée de fonctionnement indiquée.

### III. Utilisation de Docker

#### Commandes de base de Docker

Exécuter un shell interactif dans le conteneur NGINX et afficher la sortie de la commande : `docker exec -it <nom_ou_ID_conteneur> bash -c "uname -a"`

```
root@debian-xfce:~# docker exec -it upbeat_hermann bash -c "uname -a"
Linux f86d3f81f80e 5.10.0-18-amd64 #1 SMP Debian 5.10.140-1 (2022-09-02) x86_64 GNU/Linux
root@debian-xfce:~#
```

La sortie de cette commande devrait afficher des informations sur le noyau Linux du conteneur NGINX, comme son nom, sa version, son ID de version et d'autres informations système.

## IV. Gestion des images Docker

### Gestion d'une image Docker

#### Démarrage d'une image Docker

Pour démarrer une image Docker, vous pouvez utiliser la commande docker run. Voici les étapes à suivre :

- Avoir le terminal ouvert
- Utiliser la commande `docker images` pour lister toutes les images Docker présentes sur votre système. Notez le nom ou l'ID de l'image que vous souhaitez démarrer.
- Utiliser la commande docker run suivie du nom ou de l'ID de l'image pour démarrer l'image.  
Par exemple, si on veut démarrer l'image test:image, utiliser la commande suivante :

```
docker run test:image
```

Cette commande va démarrer un conteneur basé sur l'image test:image et ouvrir un terminal à l'intérieur du conteneur.

#### Arrêt d'une image Docker

Voici les étapes à suivre pour arrêter une image Docker :

- Vérifier les images en cours d'exécution à l'aide de la commande « `docker ps` »
- Identifier l'ID ou le nom de l'image que vous souhaitez arrêter
- Exécutez la commande docker stop suivie de l'ID ou du nom de l'image :

```
docker stop <ID ou nom de l'image>
```

Vérifiez que l'image a bien été arrêtée en exécutant la commande docker ps.

Il est possible d'arrêter aussi l'image immédiatement, en utilisant la commande docker kill suivi du nom ou de l'ID à la place de docker stop.

## IV. Gestion des images Docker

### Gestion d'une image Docker

### Supprimer une image Docker

Pour supprimer une image Docker, il faut utiliser la commande :

```
docker rmi <suivi du nom ou de l'ID de l'image>
```

Il est également possible de supprimer plusieurs images en même temps en spécifiant leurs noms ou leurs IDs séparés par des espaces :

```
docker rmi mon-image1 mon-image2
```

Si l'image est actuellement utilisée par un conteneur, il faudra d'abord arrêter le conteneur avant de pouvoir supprimer l'image.

Il est important de noter que la suppression d'une image est définitive et qu'elle ne peut pas être récupérée.

## IV. Gestion des images Docker

### Création d'une image Docker

Voici les étapes nécessaires à la création d'une image Docker nommée my-hello-world, qui affiche le message "Hello World !" lorsqu'elle est démarrée :

Créer un nouveau répertoire pour votre projet :

```
mkdir my-hello-world
```

```
cd my-hello-world
```

Créer un nouveau fichier appelé Dockerfile dans ce répertoire :

```
touch Dockerfile
```

Ouvrir le fichier Dockerfile avec votre éditeur de texte préféré et ajoutez les instructions suivantes :

```
FROM debian:latest  
CMD echo "Hello World !"
```

Ces instructions indiquent à Docker de créer une image basée sur la dernière version de Debian et de définir la commande à exécuter lorsque le conteneur est démarré pour afficher le message "Hello World !". Enregistrer et fermer le fichier Dockerfile.

Dans le terminal, il faut se placer dans le répertoire du projet my-hello-world. Construire l'image Docker à partir du fichier Dockerfile avec la commande suivante :

```
docker build -t my-hello-world .
```

Cela va créer une nouvelle image Docker appelée my-hello-world à partir du fichier Dockerfile dans le répertoire actuel.

Vérifier que l'image a bien été créée en utilisant la commande docker images pour lister toutes les images sur votre machine.

Il est maintenant possible de lancer un nouveau conteneur basé sur cette image en utilisant la commande docker run my-hello-world. Le message "Hello World !" s'affichera dans la console.

## IV. Gestion des images Docker

### Création d'une image Docker

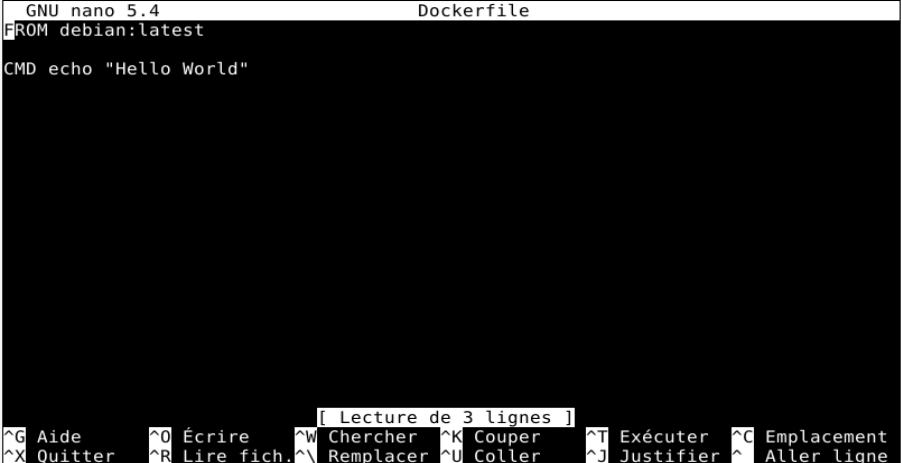
#### Vue sur le terminal des commandes :

```
root@debian-xfce:~# mkdir my-hello-world
root@debian-xfce:~# cd my-hello-world/
root@debian-xfce:~/my-hello-world# touch Dockerfile
root@debian-xfce:~/my-hello-world# nano Dockerfile
root@debian-xfce:~/my-hello-world# docker build -t my-hello-world
"docker build" requires exactly 1 argument.
See 'docker build --help'.

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile
root@debian-xfce:~/my-hello-world# docker build -t my-hello-world .
Sending build context to Docker daemon  2.048kB
Step 1/2 : FROM debian:latest
latest: Pulling from library/debian
32fb02163b6b: Pull complete
Digest: sha256:f81bf5a8b57d6aa1824e4edb9aea6bd5ef6240bcc7d86f303f197a2eb77c430f
Status: Downloaded newer image for debian:latest
--> 72b624312240
Step 2/2 : CMD echo "Hello World"
--> Running in 60149862fb48
Removing intermediate container 60149862fb48
--> 7a4367a89880
Successfully built 7a4367a89880
```

#### touch DockerFile :



The screenshot shows a terminal window with the nano text editor open. The title bar indicates 'GNU nano 5.4' and 'Dockerfile'. The editor contains three lines of text: 'FROM debian:latest', 'CMD echo "Hello World"', and a blank line. At the bottom, a status bar shows 'Lecture de 3 lignes' and a list of keyboard shortcuts: ^G Aide, ^O Écrire, ^W Chercher, ^K Couper, ^T Exécuter, ^C Emplacement, ^X Quitter, ^R Lire fich., ^\ Remplacer, ^U Coller, ^J Justifier, ^\_ Aller ligne.

#### Commande de lancement :

```
root@debian-xfce:~/my-hello-world# docker run my-hello-world
```

## IV. Gestion des images Docker

### Création d'une image Docker

Construire une image Docker contenant un fichier HTML et exécuter un conteneur à partir de cette image sur un port donné.

Créer un fichier nommé index.html dans le répertoire courant avec la commande

```
touch index.html
```

Modifier le fichier Dockerfile pour qu'il utilise l'image NGINX comme image de base, puis copie le fichier index.html dans le répertoire /usr/share/nginx/html du conteneur. Le nouveau contenu du fichier Dockerfile sera le suivant :

```
FROM nginx  
COPY index.html /usr/share/nginx/html
```

Démarrer un nouveau conteneur Docker à partir de l'image my-hello-world créé précédemment en utilisant la commande suivante :

```
docker run --name my-hello-world-container -p 8080:80 -d my-hello-world
```

--name my-hello-world-container donne un nom au conteneur.

-d exécute le conteneur en arrière-plan (en mode détaché).

my-hello-world est le nom de l'image à partir de laquelle le conteneur sera créé.

Vérifier que le conteneur fonctionne correctement en accédant à l'adresse IP de la machine Docker sur le port 8080 depuis un navigateur web. Le contenu de index.html devrait être affiché dans le navigateur.



## V. Automatisation des conteneurs

Utilisation de Docker Compose pour déployer un conteneur NGINX.

Créez un nouveau fichier nommé docker-compose.yml dans votre dossier de travail courant.

```
touch docker-compose.yml
```

Ajoutez les lignes suivantes pour spécifier les détails de votre conteneur :

```
version: '3'
services:
  my-nginx:
    image: my-nginx
    ports:
      - "8080:80"
```

Le fichier docker-compose.yml contient une seule définition de service nommée my-nginx. Cette définition utilise l'image my-nginx créée précédemment et configure le port 80 du conteneur pour être exposé sur le port 8080 de la machine hôte.

Ensuite, démarrer le conteneur : `docker-compose up`

Résultat :

```
root@debian-console:~# docker-compose up
Starting root_nginx_1 ... done
Attaching to root_nginx_1
nginx_1 | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform
nginx_1 | configuration
nginx_1 | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default
nginx_1 | sh
nginx_1 | 10-listen-on-ipv6-by-default.sh: info: IPv6 listen already enabled
nginx_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginx_1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx_1 | /docker-entrypoint.sh: Configuration complete; ready for start up
nginx_1 | 2023/03/08 13:51:09 [notice] 1#1: using the "epoll" event method
nginx_1 | 2023/03/08 13:51:09 [notice] 1#1: nginx/1.23.3
nginx_1 | 2023/03/08 13:51:09 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
nginx_1 | 2023/03/08 13:51:09 [notice] 1#1: OS: Linux 5.10.0-8-amd64
nginx_1 | 2023/03/08 13:51:09 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
nginx_1 | 2023/03/08 13:51:09 [notice] 1#1: start worker processes
nginx_1 | 2023/03/08 13:51:09 [notice] 1#1: start worker process 24
nginx_1 | 172.20.110.15 - - [08/Mar/2023:13:51:12 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0" "-"
nginx_1 | 172.20.110.15 - - [08/Mar/2023:13:51:12 +0000] "GET /favicon.ico HTTP/1.1" 404 153
nginx_1 | "http://172.20.107.27:8080/" "Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0" "-"
```

## V. Automatisation des conteneurs

### Configuration d'un environnement WordPress + MySQL persistant avec Docker Compose

Créer un fichier de configuration Docker Compose permettant d'installer WordPress avec MySQL, vous pouvez modifier le fichier docker-compose.yml en y ajoutant les lignes suivantes :

```
version: '3.9'
services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: your_mysql_root_password
      MYSQL_DATABASE: your_mysql_database
      MYSQL_USER: your_mysql_user
      MYSQL_PASSWORD: your_mysql_user_password

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: your_mysql_user
      WORDPRESS_DB_PASSWORD: your_mysql_user_password
      WORDPRESS_DB_NAME: your_mysql_database
    volumes:
      db_data:
```

Ces lignes de code définissent deux services : un service MySQL nommé db et un service WordPress nommé wordpress. Le service MySQL utilise l'image Docker mysql:5.7 et crée un volume nommé db\_data pour stocker les données de la base de données MySQL. Le service WordPress utilise l'image Docker wordpress:latest et dépend du service MySQL pour fonctionner. Le port 8000 est exposé pour accéder au site WordPress.

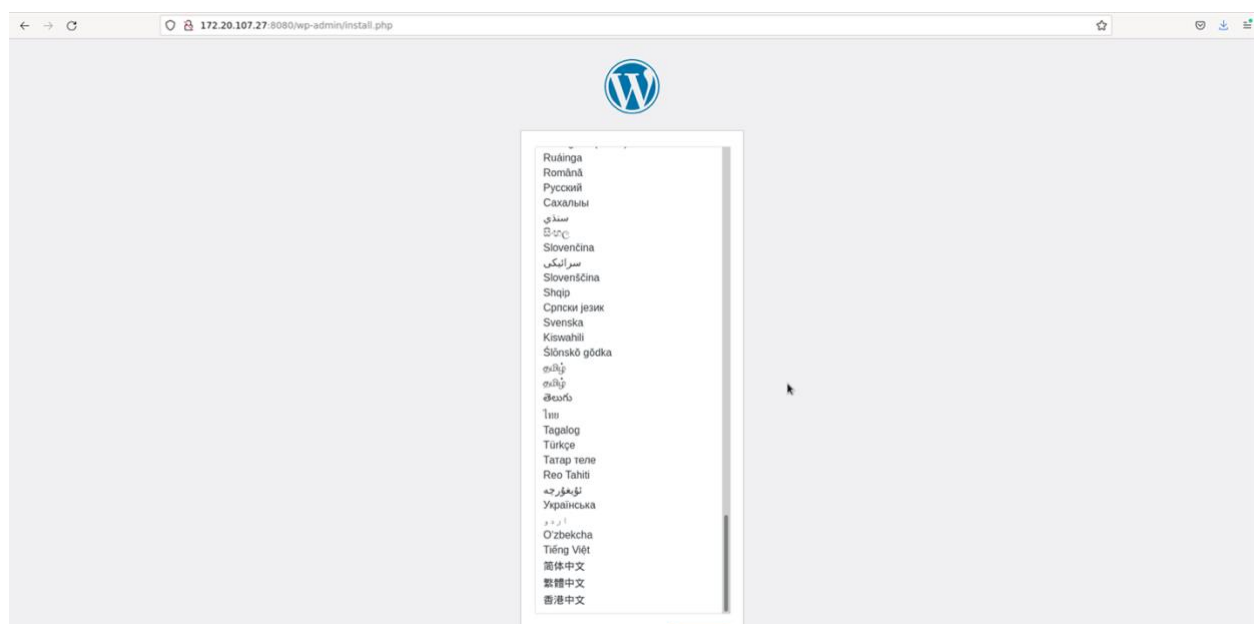
Vous devez remplacer les valeurs your\_mysql\_root\_password, your\_mysql\_database, your\_mysql\_user et your\_mysql\_user\_password par vos propres informations de configuration. Ces informations seront utilisées pour créer un utilisateur MySQL et une base de données pour WordPress.

## V. Automatisation des conteneurs

### Configuration d'un environnement WordPress + MySQL avec Docker Compose

Ouvrir son navigateur et insérer l'IP de la VM/ Machine avec le port 8080.

### Résultat :



## V. Automatisation des conteneurs

### Commande de conteneurs Docker via Docker Compose.

Pour démarrer le conteneur en utilisant Docker Compose, utilisez la commande suivante :

```
docker-compose up
```

Cela démarre le conteneur. Il est possible d'ajouter l'option -d à la fin (docker-compose up -d) qui signifie que le conteneur sera exécuté en mode détaché, ce qui permet de continuer à utiliser le terminal.

Pour arrêter et supprimer le conteneur, utilisez la commande suivante :

```
docker-compose down
```

Cela arrêtera et supprimera le conteneur créé avec la commande docker-compose up.

Pour recupérer en temps-réel les logs de console du conteneur lancé en arrière-plan, utilisez la commande suivante :

```
docker-compose logs -f
```

La commande `docker-compose kill` est utilisée pour forcer l'arrêt des conteneurs d'un projet Docker Compose. Cette commande envoie un signal SIGKILL aux processus en cours d'exécution dans les conteneurs pour les arrêter immédiatement.

Contrairement à la commande docker-compose down, qui arrête les conteneurs de manière ordonnée en leur permettant de terminer proprement leurs tâches en cours, docker-compose kill force les conteneurs à s'arrêter immédiatement, sans terminer proprement leur travail.