# Fiche de procédure : Utilisation de Docker

Cette fiche de procédure contient les explications de docker ainsi que les commandes de bases pour l'utilisation depuis un Debian en mode console.

## Explication de docker :

Docker est une technologie de virtualisation légère qui permet d'emballer une application et ses dépendances dans un conteneur, de sorte qu'elle puisse être exécutée de manière isolée sur n'importe quel système d'exploitation compatible Docker, sans avoir besoin de réinstaller ou de reconfigurer l'application. En d'autres termes, Docker facilite la création, la distribution et l'exécution d'applications en fournissant un environnement cohérent et portable.

| **Avantages** | **Inconvénients** |
|---|---|
| Isolation des applications : Docker permet de créer des conteneurs isolés qui encapsulent une application et toutes ses dépendances, ce qui facilite le déploiement et la gestion des applications. | Utilisation de ressources supplémentaires : Docker nécessite l'utilisation de ressources supplémentaires pour exécuter les conteneurs, ce qui peut avoir un impact sur les performances de l'hôte. |
| Portabilité : Les conteneurs Docker sont portables et peuvent être exécutés sur n'importe quel système compatible Docker, ce qui facilite le déploiement des applications dans des environnements différents. | Configuration complexe : La configuration de Docker peut être complexe, en particulier lorsqu'il s'agit de gérer des conteneurs à grande échelle. |
| Rapidité de déploiement : Docker permet de déployer des applications rapidement, car il est possible de créer des conteneurs en quelques minutes et de les exécuter immédiatement. | Sécurité : Bien que Docker soit relativement sécurisé, il existe toujours des risques de failles de sécurité, en particulier si les conteneurs ne sont pas correctement configurés ou si des images malveillantes sont utilisées. |
| Gestion des ressources : Docker permet de gérer efficacement les ressources système en allouant des ressources spécifiques à chaque conteneur. | Difficulté à gérer les données : La gestion des données dans Docker peut être difficile, en particulier lorsqu'il s'agit de stocker des données sur des conteneurs à long terme. |
| Facilité de mise à l'échelle : Docker permet de mettre à l'échelle facilement les applications en ajoutant ou en supprimant des conteneurs en fonction de la demande. | Coût de maintenance : La maintenance de Docker peut être coûteuse, en particulier si elle est utilisée à grande échelle, car elle nécessite une surveillance constante pour garantir la disponibilité et la sécurité des conteneurs. |

## Installation de Docker :

En mode « root » !

```
apt install \  ca-certificates \   curl \   gnupg \   lsb-release
```

Pour ajouter la clé GPG officiel de Docker :

```
mkdir -m 0755 -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Commande pour configurer les dépôt : /etc/apt/sources.list.d

```
echo \  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian \ $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
```

Mettre de nouveau à jour les paquets :

```
apt update
```

Pour installer la dernière version :

```
apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

La commande pour vérifier que docker est bien installé est :

```
docker run hello-world
```

## Découverte des commandes de base de Docker :

Pour télécharger l'image de NGINX depuis docker :

```
docker pull nginx
```

Pour voir que le téléchargement à bien eu lieux :

```
docker images
```

Pour lancer un conteneur (image) NGINX sous docker sous le port 8080 :

```
docker run -d -p 8080:80 nginx
```

Pour vérifier que nginx est accessible depuis un navigateur:

```
http://adresse-ipv4:8080/
```

La liste des conteneurs créés est :

```
docker ps -a
```

La vérification que le conteneur NGINX est en cours d'exécution :

```
docker top <nom ou ID du conteneur>
```

L'accès au conteneur NGINX sous docker en bash :

```
sudo docker exec -it <nom ou ID du conteneur> bash
```

Pour afficher les informations du système :

```
uname -a
```

Les commandes pour arrêter et pour relancer un conteneur :

```
docker stop nom ou ID du conteneur | docker start nom ou ID du conteneur
```

La suppression d'un conteneur se fait avec la commande :

```
docker rm <nom ou ID du conteneur>
```

Afin pour vérifier que le conteneur avec l'image en questions à bien été supprimé :

```
docker rmi nginx | docker images
```

## Création de vos propres images Docker :

Créer un fichier Dockerfile avec comme contenue :

```
FROM debian:latest

CMD ["echo", "Hello World!"]
```

Ensuite, nous construisons l'image en utilisant la commande suivante:

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

Pour créer un fichier index.html dans le dossier de travail courant, utilisez la commande suivante

```
touch index.html
```

Pour vérifier que le fichier HTML créé est présent dans le dossier courant, utilisez la commande suivante:

```
cat index.html
```

Pour construire une nouvelle image nommée "my-nginx" incluant le fichier HTML précédemment, nous avons besoin de créer un Dockerfile avec le contenu suivant :

```
FROM nginx

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

Ensuite, l'image va se construire en utilisant la commande suivante:

```
docker build -t my-nginx .
```

Pour lancer un conteneur basé sur l'image "my-nginx" créer, utilisez la commande suivante:

```
docker run -d -p 80:80 my-nginx
```

## Automatisation du déploiement de vos conteneurs :

Pour créer le fichier de configuration, faut créer un fichier docker-compose.yml avec comme informations :

```
version: '3'
services:
 nginx:
  image: nginx
  ports:
   - "8080:80"
  volumes:
   - ./index.html:/usr/share/nginx/html/index.html:ro
```

Pour récupérer les images sans lancer les conteneurs, il faut exécuter la commande suivante :

```
docker-compose pull
```

Pour lancer le conteneur en premier plan, il faut exécuter la commande suivante :

```
docker-compose up
```

Ensuite, il faut ouvrir un navigateur sur l'ordinateur hôte et taper l'adresse IP de la machine virtuelle Docker suivie du port 8080 (par exemple http://adress-ipv4:8080) pour accéder à la page d'accueil de NGINX. Pour stopper le conteneur, il faut interrompre la commande docker-compose up avec **Ctrl-C**.

Pour voir les logs, pour lancer les conteneurs en arrière-plan et stopper, il faut exécuter la commande suivante :

```
docker-compose logs -f | docker-compose down | docker-compose up -d
```

Pour créer un fichier de configuration permettant d'obtenir une installation WordPress + MySQL fonctionnelle avec les données persistées dans un volume Docker, il faut créer un fichier docker-compose.yml dans le dossier de travail courant avec le contenu suivant :

```
version: '3'
services:
  db:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: emrys-root
      MYSQL_DATABASE: td-docker
      MYSQL_USER: emrys
      MYSQL_PASSWORD: emrys-password
    volumes:
      - db_data:/var/lib/mysql
  wordpress:
    depends_on:
      - db
    image: wordpress
    ports:
      - "8080:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: emrys
      WORDPRESS_DB_PASSWORD: emrys-password
      WORDPRESS_DB_NAME: td-docker
    volumes:
      - wordpress_data:/var/www/html
volumes:
  db_data:
  wordpress_data:
```

Ensuite, il faut exécuter la commande suivante pour lancer les conteneurs :

```
docker-compose  up -d
```