# Fiche de procédure Docker

## 1. Présentation

Points forts

Docker est une plateforme de conteneurisation qui permet de créer, déployer et exécuter des applications de manière portable et efficace.

Voici quelques-unes des principales caractéristiques et concepts de Docker :

**Conteneurs**: Les conteneurs Docker sont des instances d'images Docker, qui sont des fichiers statiques qui contiennent tout ce dont une application a besoin pour s'exécuter, y compris son code, ses dépendances et les configurations système requises.

Les conteneurs sont légers et portables, et ils sont conçus pour fonctionner sur n'importe quel système d'exploitation et sur n'importe quelle infrastructure.

**Images**: Les images Docker sont des fichiers statiques qui contiennent toutes les informations nécessaires pour créer un conteneur. Une image peut inclure le code source d'une application, les fichiers de configuration, les dépendances et tout autre composant requis pour exécuter l'application. Les images Docker sont créées à partir d'un fichier Dockerfile, qui spécifie comment l'image doit être construite.

Points faibles

Points forts	Points faibles
Portabilité : Les conteneurs Docker sont	Complexité : Docker peut être complexe à
portables et peuvent être exécutés sur	configurer et à gérer, en particulier pour les
n'importe quel système d'exploitation et sur	utilisateurs novices.
n'importe quelle infrastructure, ce qui facilite le	
déploiement et la gestion d'applications dans	
des environnements différents.	
Légèreté : Les conteneurs Docker sont plus	Sécurité : Bien que Docker offre une isolation
légers que les machines virtuelles, ce qui les	des applications, les conteneurs peuvent encore
rend plus rapides à démarrer et moins	présenter des vulnérabilités de sécurité s'ils ne
gourmands en ressources système.	sont pas configurés et gérés correctement.
Isolation : Les conteneurs Docker offrent une	Dépendances : Les conteneurs Docker
isolation des applications qui permet d'éviter les	dépendent souvent de nombreux composants
conflits de dépendances et de garantir la	tiers, ce qui peut entraîner des problèmes de
sécurité des applications.	compatibilité et de dépendances complexes.

Réutilisabilité : Les images Docker peuvent être Performance : Bien que les conteneurs Docker réutilisées pour créer de nouveaux conteneurs, soient plus légers que les machines virtuelles, ils

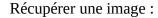
ce qui facilite le développement et le déploiement d'applications en fournissant des environnements de développement et de test cohérents.

peuvent encore avoir des performances inférieures à celles des applications exécutées directement sur un système d'exploitation hôte.

# 2. Installation

apt update apt install ca-certificates curl gnupg lsb-release mkdir -m 0755 -p /etc/apt/keyrings curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg echo "deb [arch=\$(dpkg --prinStyle de paragraphe par défautt-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \$ (lsb\_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null apt update apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-composeplugin docker run hello-world

# 3. Commande de base



docker pull{nom image}

### Liste des images :

docker images

Lancer un conteneur en arrière plan avec un partage du port interne 80 du conteneur sur le port externe 8080 :

docker run -d -p 8080:80 {nom image}

Il est possible d'utilise l'option --name {nom} pour assigner un nom au conteneur, sinon il en aura un par défaut.

### Liste des conteneurs créés :

docker ps -a

Liste des conteneurs en cours :

docker ps

Entrer une commande dans un conteneur up :

docker exec -it {nom conteneur} bash

Stopper un conteneur :

docker stop {nom conteneur}

Relancer un conteneur:

docker start {nom conteneur}

Supprimer complétement un conteneur :

docker rm {nom conteneur}

Supprimer complétement une image :

docker rmi {nom image}

# 4. Images docker personnalisées

On créer d'abord un répertoire qui contiendra notre Dockerfile :

mkdir {nom répertoire} && cd {nom répertoire}

On créer et on modifie le fichier Dockerfile :

nano Dockerfile

Exemple de Dockerfile pour afficher « Hello World! »:

# Utiliser Debian en tant que base d'image

FROM debian:latest

# Exécuter la commande 'echo' pour afficher le message 'Hello World!'

CMD echo "Hello World!"

Exemple de Dockerfile pour modifier la page web par défaut d'un conteneur NGINX :

# Utiliser l'image NGINX en tant que base d'image

### FROM nginx

# Copier le fichier index.html dans le répertoire /usr/share/nginx/html du conteneur

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

L'instruction COPY dans le Dockerfile prend en premier paramètre le chemin de la source depuis l'hôte et en deuxième paramètre le chemin de la destination du conteneur qui sera créé.

On construit ensuite notre image:

 $docker\ build\ -t\ \{nom\ image\}\{chemin\ absolu/relatif\ vers\ le\ fichier\ Dockerfile\}$ 

On démarre le conteneur :

docker run {nom image}

# 5. Docker-compose

Installer docker-compose:

```
apt install docker-compose
```

contenu du fichier docker-compose.yml pour nginx et accéssible à partir du port 8080 :

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

Récupérer les images contenues dans le fichier de configuration docker-compose.yml :

```
docker-compose pull
```

Lancer au premier plan le conteneur :

```
docker-compose up
```

Pour stopper un conteneur au premier plan :

```
ctrl + c
```

Lancer en arrière plan le conteneur :

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

Pour stopper un conteneur en arrière plan :

```
docker-compose down
```

Récupérer les logs d'un conteneur en arrière plan en temps réel :

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

Créer un conteneur (docker-compose.yml) avec un wordpress et une base de donnée mysql avec Les données de la base MySQL ainsi que celles de WordPress persistées dans un volume Docker, afin d'être conservées d'un redémarrage à l'autre :

```
version: "3"
services:
 db:
    image: mysql:5.7
   volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
     MYSQL_ROOT_PASSWORD: admin1234
     MYSQL_DATABASE: wordpress-tp
     MYSQL_USER: wordpress
     MYSQL_PASSWORD: admin
 wordpress:
   depends_on:
      - db
    image: wordpress:latest
   volumes:
      - wp_data:/var/www/html
    restart: always
    ports:
      - "8080:80"
    environment:
     WORDPRESS_DB_HOST: db:3306
     WORDPRESS_DB_USER: wordpress
     WORDPRESS_DB_PASSWORD: admin
     WORDPRESS_DB_NAME: wordpress-tp
volumes:
 db_data:
 wp_data:
```