Mémento Docker

Docker est une plateforme de conteneurs lancée en 2013. Elle permet de créer facilement des conteneurs et des applications basées sur les conteneurs. Il en existe d'autres, mais celle-ci est la plus utilisée. Initialement conçue pour Linux, Docker permet aussi la prise en charge des containers sur Windows ou Mac grâce à une " layer " de virtualisation Linux entre le système d'exploitation Windows / macOS et l'environnement runtime Docker. Il est donc possible d'exécuter des conteneurs Windows natifs sur des environnements de conteneurs Windows ou Linux.

1-Installation (Linux)

Avant de pouvoir installer Docker, il faut d'abord configurer le dépôt APT en tant que root.

```
su -
apt-get update
apt-get install ca-certificates curl
```

Crée le répertoire pour stocker les clés GPG et ajouter la clé GPG officielle de Docker

```
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg -o
/etc/apt/keyrings/docker.asc sudo
chmod a+r /etc/apt/keyrings/docker.asc
```

Le dépôt Docker est ajouté aux sources APT. Cette commande permet de récupérer la version codename de votre distribution Debian (comme "bookworm" ou "bullseye") et de l'utiliser pour sélectionner le dépôt approprié.

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

Installer les paquets nécessaire a Docker

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

Pour terminer versifier l'installation exécuter la commande suivante :

```
docker run hello-world
```

Si votre 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.
```

2-Commandes de Base

2.1 Gestion des images

Télécharger une image : télécharge l'image d'une application depuis Docker Hub (le registre public d'images)

```
docker pull <nom_image>
```

Lister les images disponibles (deux méthodes sont disponibles) :

```
docker images
```

ou

```
docker image ls
```

Supprimer une image :

```
docker rmi <image_id>
```

2.2 Gestion des conteneurs

Lancer un conteneur : la commande ci-dessous permet de lancer un conteneur, modifier son contenu et l'effacer automatiquement lorsque l'utilisateur le quitte (supprimer le –rm pour que le conteneur s'efface pas automatiquement)

```
docker run -it --rm <nom de l'image>
```

Mapper des ports : vous pouvez aussi spécifier le mappage des ports entre la machine hôte et le conteneur avec le *-p*

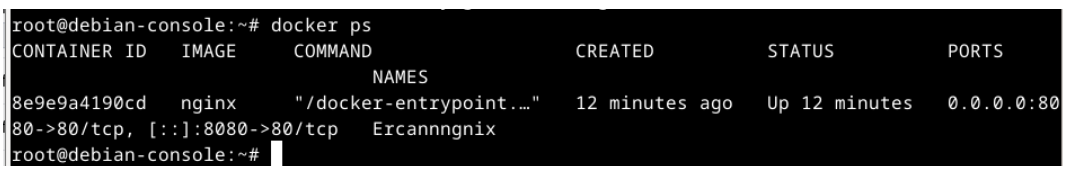
```
docker run -p <port_hôte>:<port_conteneur> <nom_image>
```

Lister les conteneurs : affiche la liste des conteneurs actifs et inactifs (ajouter `-a` pour tous les conteneurs)

```
docker ps
```

Vérifiez le statut Up

Exemple :



Accéder à un conteneur en cours d’exécution : permet d'exécuter des commandes à l'intérieur d'un conteneur actif.

```
docker exec -it <id_conteneur> /bin/bash
```

Démarrer un conteneur : relance un conteneur arrêté

```
docker start <id_conteneur>
```

Arrêter un conteneur : stoppe un conteneur actif

```
docker stop <id_conteneur>
```

Supprimer un conteneur : Supprime complètement un conteneur, qu'il soit arrêté ou non

```
docker rm -f <id_conteneur>
```

2.3 Logs et surveillance

Afficher les logs d’un conteneur : affiche les journaux d’un conteneur, utile pour déboguer ou surveiller son activité

```
docker logs <id_conteneur>
```

Suivre les logs en temps réel : ajoutez -f pour suivre les logs en continu

```
docker logs -f <id_conteneur>
```

2.4 Volumes Docker

Les volumes sont utilisés pour stocker des données persistantes entre différents conteneurs.

Lister les volumes Docker :

```
docker volume ls
```

Supprimer un volume :

```
docker volume rm <nom_volume>
```

3-Création d'images Docker

Création d’un Dockerfile :
Le fichier Dockerfile contient les instructions nécessaires à la création d’une image Docker

Exemple basique pour une application Nginx :

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

Construire une image Docker :

Une fois le Dockerfile prêt, utilisez la commande suivante pour construire l'image :

```
docker build -t <nom_image> .
```

Il faut se placer dans le dossier qui contient le Dockerfile ou à défaut, remplacer par le chemin du fichier

Vérifiez que l'image a été créée correctement avec la commande :

```
docker images
```

4-Automatisation avec Docker Compose

Docker Compose est un outil permettant de définir et de gérer des applications multi-conteneurs. Grâce à un fichier de configuration en format YAML (docker-compose.yml), vous pouvez automatiser la gestion de plusieurs services (conteneurs) au lieu de les gérer individuellement avec des commandes Docker

Créer un fichier docker-compose.yml (*touch docker-composer.yml*)

Le fichier docker-composer.yml définit les services à déployer dans un environnement Docker. Voici un exemple minimaliste pour déployer un serveur NGINX :

```yaml
version: '3'
services:
  web:
    image: nginx  # L'image NGINX sera téléchargée et utilisée pour créer le service
    ports:
      - "8080:80"  # Le port 8080 de la machine hôte est mappé sur le port 80 du conteneur
```

version 3 : Indique la version du format de fichier Docker Compose. La version 3 est courante et adaptée pour des déploiements en production

services : Définit les services (conteneurs) qui seront déployés

web : C'est le nom du service, ici il correspond au serveur NGINX

image: Utilise l'image officielle (nginx) du Docker Hub

ports : Mappe le port 8080 de la machine hôte sur le port 80 du conteneur pour permettre l'accès externe via http://localhost:8080

Télécharger les images sans lancer les conteneurs

Cette commande télécharge les images mentionnées dans le fichier docker-compose.yml sans démarrer les conteneurs

```
docker-compose pull
```

Lancer le conteneur au premier plan

```
docker-compose up
```

Cette commande démarre les services définis dans le fichier. Si vous ne spécifiez pas -d, les services s'exécutent au premier plan et affichent les logs directement dans votre terminal.
L'accès au service se fait en tapant http://localhost:8080 dans votre navigateur

Stopper le conteneur

```
docker-compose stop
```

Exemple pour WordPress et PostgreSQL :
Cet exemple illustre comment utiliser Docker Compose pour créer une installation multi-services avec WordPress et PostgreSQL

```yaml
version: '3'
services:
  db:
    image: postgres  # Utilise l'image officielle PostgreSQL
    volumes:
      - db_data:/var/lib/postgresql/data  # Les données de PostgreSQL sont stockées dans un volume Docker
    environment:
      POSTGRES_DB: wordpress
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password  # Définit la base de données et les accès

  wordpress:
    image: wordpress  # Utilise l'image officielle WordPress
    ports:
      - "8080:80"  # Le port 8080 de la machine hôte est mappé sur le port 80 du conteneur WordPress
    environment:
      WORDPRESS_DB_HOST: db  # Indique que WordPress utilisera le service 'db' pour la base de données
      WORDPRESS_DB_NAME: wordpress
      WORDPRESS_DB_USER: user
      WORDPRESS_DB_PASSWORD: password  # Paramètres de connexion à la base de données
volumes:
  db_data: {}  # Crée un volume Docker pour persister les données de la base PostgreSQL
```