B1 – Prise en main de Docker

C’est quoi Docker ?

Docker est une plateforme ouverte de conteneurisation pour le développement, l'expédition et l'exécution d'applications. Grâce à la virtualisation basée sur LXC, Docker vous permet d’isoler les applications que vous hébergez afin de fournir rapidement des logiciels et services. Avec Docker, vous pouvez gérer votre infrastructure de la même manière que vos applications.

Les conteneurs Dockers sont portables. Ils utilisent des images enveloppant des solutions logicielles.

Docker est largement utilisé en DevOps pour l’orchestration (l'expédition, le test et le déploiement du code), pour réduire considérablement le délai entre l'écriture du code et son exécution en production.

Docker offre de hautes performances, proches de celles de l’hôte.

Note : il faut exécuter les commandes « `docker` » et « `docker-compose` » en superutilisateur (« root »).

Vous trouverez le contenu des fichiers utilisés dans l’Annexe.

Installation de Docker (Debian)

Ajout du dépot:

Ajout de la clé GPG:

```
sudo apt-get update

sudo apt-get install ca-certificates curl gnupg

sudo install -m 0755 -d /etc/apt/keyrings

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

sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

Ajout du dépot Docker au sources.list:

```
echo \

  "deb [arch="$(dpkg --print-architecture)" signed-
by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian \

  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \

  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
```

Installation de Docker:

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

Utilisation de Docker

Pour récupérer une image:

```
docker pull <image>[:version]
```

Par exemple celle de NGINX:

```
docker pull nginx:latest
```

Pour lister les images téléchargées:

```
docker image ls / docker images
```

Pour lancer un conteneur en arrière-plan:

```
docker run -d <image>
```

Pour lancer un conteneur, en exposant le port interne 80 vers le port externe 8080:

```
docker run -p 8080:80 <image>
```

Pour lister les processus Docker en cours:

```
docker ps
```

Pour lister tous les conteneurs Docker créés:

```
docker ps --all
```

Exécuter une commande dans un conteneur:

```
docker exec <id/nom_conteneur> bash -c "<commande>"
```

Par exemple, afficher la version de Linux utilisée par le conteneur:

```
docker exec <id/nom_conteneur> bash -c "uname -a"
```

Note : on peut également lancer des commandes en lançant un conteneur avec un terminal interactif (option "-it")

Stopper un conteneur:

```
docker stop <id/nom_conteneur>
```

Relancer un conteneur précédemment créé:

```
docker start <id/nom_conteneur>
```

Supprimer un conteneur:

```
docker rm <id/nom_conteneur>
```

Supprimer une image:

```
docker image rm <image> / docker rmi <image>
```

Créer une image Docker à partir d'un Dockerfile:

`nano Dockerfile`

`docker build -t <nom_image_creee> - < Dockerfile` (ou répertoire courant)

`docker run <nom_image_creee>`

Par exemple, créer une image Docker NGINX à partir d'un Dockerfile:

`mkdir my-nginx`

`nano index.html`

Contenu du fichier index.html utilisé pour remplacer la page par défaut de NGINX

```
  GNU nano 7.2                          testserv/index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <p>
        Test
    </p>
</body>
</html>
```

`nano Dockerfile`

Contenu du Dockerfile

```
  GNU nano 7.2                          testserv/Dockerfile
FROM nginx
COPY ./index.html /usr/share/nginx/html/index.html
```

`docker build -t my-nginx .`

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

Créer un projet docker-compose (NGINX):

```
mkdir <nom_projet>
```

```
nano docker-compose.yml
```

Contenu du docker-compose.yml (exemple pour NGINX)

```
  GNU nano 7.2              testcompose/docker-compose.yml
services:
  nginx:
    image: nginx:latest
    ports:
      - "8080:80"
    volumes:
      - ./:/usr/share/nginx/html
```

```
docker-compose up
```

Créer un projet docker-compose (Wordpress + PostgreSQL):

```
mkdir testwordpress
```

```
nano docker-compose.yml
```

Contenu du docker-compose.yml (Wordpress + PostgreSQL)

```
  GNU nano 7.2                                  docker-compose.yml
services:
  wordpress:
    image: wordpress:latest
    container_name: my-wordpress
    ports:
      - "8080:80"
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    volumes:
      - wordpress_data:/var/www/html

  db:
    image: postgres:latest
    container_name: my-postgresql
    environment:
      POSTGRES_DB: wordpress
      POSTGRES_USER: wordpress
      POSTGRES_PASSWORD: wordpress
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  wordpress_data:
  postgres_data:
```

```
docker-compose up
```

Annexe

Fichier index.html (à titre d'exemple...):

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>Document</title>

</head>

<body>

    <p>

        Test

    </p>

</body>

</html>
```

Fichier Dockerfile (Hello World):

```
FROM debian:latest

CMD echo "Hello World"
```

Fichier Dockerfile (NGINX):

```
FROM nginx

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

Fichier docker-compose.yml (NGINX):

```
services:
  nginx:
    image: nginx:latest
    ports:
      - "8080:80"
    volumes:
      - ./:/usr/share/nginx/html
```

Fichier docker-compose.yml (Wordpress + PostgreSQL):

Note : privilégier une version fixe plutôt que « latest ». Egalement, les noms de conteneurs sont à titre d'exemple.

```
services:
  wordpress:
    image: wordpress:latest
    container_name: my-wordpress
    ports:
      - "8080:80"
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: <utilisateur>
      WORDPRESS_DB_PASSWORD: <mdp>
      WORDPRESS_DB_NAME: <nom_db>
    volumes:
      - wordpress_data:/var/www/html

  db:
    image: postgres:latest
    container_name: my-postgresql
    environment:
      POSTGRES_DB: <nom_db>
      POSTGRES_USER: <utilisateur>
      POSTGRES_PASSWORD: <mdp>
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  wordpress_data:
  postgres_data:
```