BARILLOT Esteban

Prise en main de docker

Etape 1 : Installation de docker

Se connecter sur la machine en tant qu'administrateur

```
apt-get update
apt-get install ca-certificates curl
install -m 0755 -d /etc/apt/keyrings
apt install curl
curl -fsSL https://download.docker.com/linux/debian/gpg -o
/etc/apt/keyrings/docker.asc
```

```
root@debian-console:~# curl -fsSL https://download.docker.com/linux/debian/gpg -
o /etc/apt/keyrings/docker.asc
```

```
chmod a+r /etc/apt/keyrings/docker.asc
```

```
root@debian-console:~# chmod a+r /etc/apt/keyrings/docker.asc
```

```
# Add the repository to Apt sources:
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
apt-get update
```

```
root@debian-console:~# echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://downlo
ad.docker.com/linux/debian \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
   tee /etc/apt/sources.list.d/docker.list > /dev/null
 apt-get update
Atteint :1 http://depot.stsio.lan/debian bookworm InRelease
Atteint :2 http://depot.stsio.lan/debian bookworm-updates InRelease
Atteint :3 http://depot.stsio.lan/security bookworm-security/updates InRelease
Réception de :4 https://download.docker.com/linux/debian bookworm InRelease [43,3 kB]
Réception de :5 https://download.docker.com/linux/debian bookworm/stable amd64 Packages [30,0 kB
]
73,4 ko réceptionnés en 0s (206 ko/s)
Lecture des listes de paquets... Fait
N: Le dépôt « Debian bookworm » a modifié sa valeur « firmware component » de « non-free » à « n
on-free-firmware »
N: Plus d'information disponible dans la note de mise à jour ici : https://www.debian.org/releas
es/bookworm/amd64/release-notes/ch-information.html#non-free-split
```

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

Pour vérifier que l'installation que Docker a bien fonctionné exécuter cette commande :

```
docker run hello-world
```

Si l'installation a bien fonctionné voici ce que cela vous affichera :

```
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/
```

Etape 2 : Découverte des commandes de base de Docker

1. Pour récupérer l'image du service NGINX
   `docker pull nginx`

```
root@debian-console:~# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
a2318d6c47ec: Pull complete
095d327c79ae: Pull complete
bbfaa25db775: Pull complete
7bb6fb0cfb2b: Pull complete
0723edc10c17: Pull complete
24b3fdc4d1e3: Pull complete
3122471704d5: Pull complete
Digest: sha256:04ba374043ccd2fc5c593885c0eacddebabd5ca375f9323666f28dfd5a9710e3
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
root@debian-console:~#
```

2. Vérifiez que vous avez bien récupéré l'image et qu'elle apparaît désormais dans votre liste d'images téléchargées.

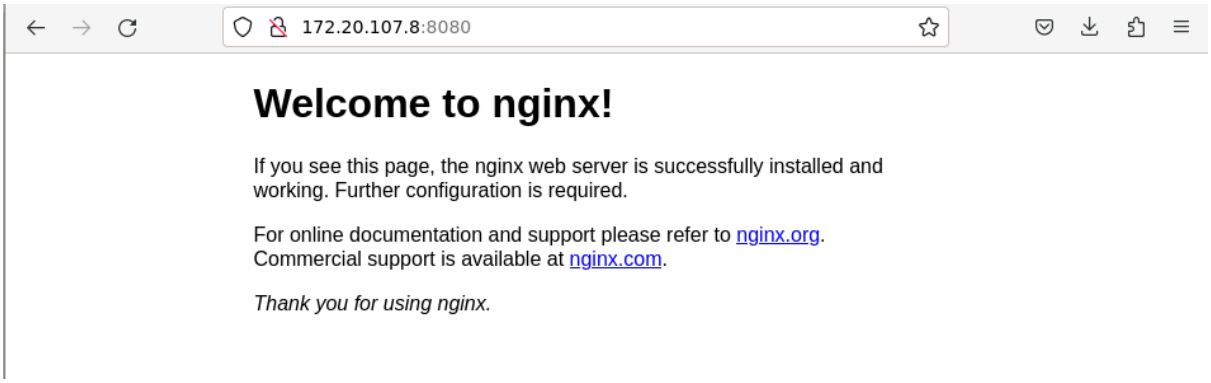
   `docker images`

```
root@debian-console:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
debian        latest    4fd3f4b75df3   11 days ago     117MB
nginx         latest    39286ab8a5e1   4 weeks ago     188MB
hello-world   latest    d2c94e258dcb   16 months ago   13.3kB
```

3. Lancez 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. Vérifiez ensuite depuis le navigateur de votre ordinateur hôte que vous accédez bien à la page d'accueil de NGINX, en tapant comme URL l'IP de votre VM Debian, suivie du port 8080. Vous devriez accéder à une page indiquant « Welcome to nginx! ».
   `docker run -d --name my-nginx -p 8080:80 nginx`

```
root@debian-console:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
debian        latest    4fd3f4b75df3   11 days ago     117MB
nginx         latest    39286ab8a5e1   4 weeks ago     188MB
hello-world   latest    d2c94e258dcb   16 months ago   13.3kB
root@debian-console:~# docker run -d --name EstebanNginx -p 8080:80 nginx
6a79ab2832da0f065562860cda694b78e178cc3554cbccec46101822238a7e6e
```

Puis par la suite ouvrez votre navigateur et tapez l'adresse IP de votre machine suivi du port 8080.



4. Récupérez la liste des conteneurs créés sur votre installation Docker. Notez le nom et l'ID du conteneur de NGINX, ils vous serviront dans les questions suivantes.

   `docker ps -a`

```
root@debian-console:~# docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS                      PORTS                                       NAMES
6a79ab2832da   nginx         "/docker-entrypoint.…"   11 minutes ago   Up 11 minutes               0.0.0.0:8080->80/tcp, [::]:8080->80/tcp     EstebanNginx
ee38b0662f21   hello-world   "/hello"                 38 minutes ago   Exited (0) 38 minutes ago                                               blissful_yalow
root@debian-console:~#
```

5. Vérifiez que le conteneur est bien en cours de fonctionnement en listant les processus de conteneurs en cours. Notez le statut actuel du conteneur.

   `docker ps`

```
root@debian-console:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                       NAMES
6a79ab2832da   nginx     "/docker-entrypoint.…"   13 minutes ago   Up 13 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp     EstebanNginx
root@debian-console:~#
```

6. A l'intérieur du conteneur NGINX, exécutez, par le biais d'un invité de commande bash, la commande *uname -a*. Notez la sortie de cette commande.

   `uname -a`

```
root@debian-console:~# uname -a
Linux debian-console 6.1.0-23-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.99-1 (2024-07-15) x86_64 GNU/Linux
```

7. Stoppez le conteneur NGINX.

   `docker ps`

```
root@debian-console:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED      STATUS      PORTS                                       NAMES
6a79ab2832da   nginx     "/docker-entrypoint.…"   6 days ago   Up 6 days   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp     EstebanNginx
```

   Conserver l'id du conteneur pour le mettre dans la commande suivante
   `docker stop <CONTAINER_ID>`

```
root@debian-console:~# docker stop 6a79ab2832
6a79ab2832
```

8. Vérifiez en réexécutant la commande permettant d'identifier les processus en cours que le conteneur est bien arrêté. Notez le statut actuel du conteneur.

   ```
   docker ps -a
   ```

   ```
   root@debian-console:~# docker ps -a
   CONTAINER ID   IMAGE         COMMAND                  CREATED      STATUS                          PORTS     NAMES
   6a79ab2832da   nginx         "/docker-entrypoint.…"   6 days ago   Exited (0) About a minute ago             EstebanNginx
   ee38b0662f21   hello-world   "/hello"                 6 days ago   Exited (0) 6 days ago                     blissful_yalow
   ```

9. Relancez à présent le conteneur NGINX précédemment créé.

   ```
   docker start
   ```

   ```
   root@debian-console:~# docker start 6a79ab2832da
   6a79ab2832da
   ```

10. Vérifiez en réexécutant la commande permettant d'identifier les processus en cours que le conteneur est bien relancé. Notez à nouveau le statut actuel du conteneur et vérifiez qu'il correspond au même statut que celui obtenu à la question 5.

    Pour vérifier que le conteneur NGINX est bien relancé, écrivez la commande suivante :

    ```
    docker ps -a
    ```

    ```
    root@debian-console:~# docker ps -a
    CONTAINER ID   IMAGE         COMMAND                  CREATED      STATUS                  PORTS                                   NAMES
    6a79ab2832da   nginx         "/docker-entrypoint.…"   6 days ago   Up 4 seconds            0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   EstebanNginx
    ee38b0662f21   hello-world   "/hello"                 6 days ago   Exited (0) 6 days ago                                           blissful_yalow
    ```

    Vous pouvez ainsi voir que le statut est donc passé en "Up 4 seconds".

11. Supprimez complètement le œ NGINX.

    Tout d'abord vous devez stopper le conteneur

    ```
    docker stop <CONTAINER_ID>
    ```

    ```
    root@debian-console:~# docker stop 6a79ab2832da
    6a79ab2832da
    ```

    ```
    docker rm <CONTAINER_ID>
    ```

    ```
    root@debian-console:~# docker rm 6a79ab2832da
    6a79ab2832da
    ```

12. Vérifiez en réexécutant la commande permettant de lister les conteneurs créés que le conteneur NGINX a bien disparu de la liste.

    ```
    docker container ls -a
    ```

```
root@debian-console:~# docker container ls -a
CONTAINER ID   IMAGE         COMMAND    CREATED      STATUS                  PORTS     NAMES
ee38b0662f21   hello-world   "/hello"   7 days ago   Exited (0) 7 days ago             blissful_yalow
```

13. Supprimer l'image de NGINX.

    Pour voir les images :

    ```
    docker images
    ```

    Puis pour supprimer l'image :
    ```
    docker rmi <IMAGE_ID>
    ```

```
root@debian-console:~# docker rmi 39286ab8a5e1
Untagged: nginx:latest
Untagged: nginx@sha256:04ba374043ccd2fc5c593885c0eacddebabd5ca375f9323666f28dfd5a9710e3
Deleted: sha256:39286ab8a5e14aeaf5fdd6e2fac76e0c8d31a0c07224f0ee5e6be502f12e93f3
Deleted: sha256:d71f9b66dd3f9ef3164d7023cc99ce344d209decd5d6cd56166c0f7a2f812c06
Deleted: sha256:634d30adf8a2232256b2871e268c8f0fdb2c348374cd8510920a76db56868e16
Deleted: sha256:f230be3f4e104c7414b7ce9c8d301f37061b4e06afe010878ea55f858d89f7f3
Deleted: sha256:c5210c8480131b7dbc5ad8adc425d68cd7a8848ee2e07de3c69cb88a4b8fd662
Deleted: sha256:d4f588811a337e0b01da46772d02f7f82ee5f9baff6886365ffb912d455f4f53
Deleted: sha256:d73e21a1e27b0184b36f6578c8d0722a44da253bc74cd72e9788763f4a4de08f
Deleted: sha256:8e2ab394fabf557b00041a8f080b10b4e91c7027b7c174f095332c7ebb6501cb
```

14. Vérifiez en réexécutant la commande permettant de lister les images téléchargées que l'image a bien disparu de la liste.

    ```
    docker images
    ```

```
root@debian-console:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
debian        latest    4fd3f4b75df3   2 weeks ago     117MB
hello-world   latest    d2c94e258dcb   16 months ago   13.3kB
```

Étape 3 – Création de vos propres images Docker

1. Pour vous familiariser avec l'environnement de construction d'images de Docker, créez une image nommée *my-hello-world*, basée sur Debian dans sa dernière version et qui aura pour seul objectif, une fois démarrée, d'afficher dans la console le message *« Hello World ! »*, le tout à l'aide d'un Dockerfile. Notez bien chaque étape nécessaire à la réalisation de cette phase, vous en aurez à nouveau besoin par la suite.

```
touch Dockerfile
```

```
nano Dockerfile
```

puis :

```
FROM debian:latest

COPY hello.sh /hello.sh

RUN chmod +x /hello.sh

CMD ["/hello.sh"]
```

Sauvegarder le fichier

```
touch hello.sh
nano hello.sh
```

puis :

```
#!/bin/sh
echo "Hello World !"
```

Sauvegarder le fichier, puis :

```
chmod +x hello.sh
```

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

pour vérifier :

```
docker images
```

```
root@debian-console:~# docker images
REPOSITORY       TAG       IMAGE ID       CREATED          SIZE
my-hello-world   latest    cd44e35f49e4   23 seconds ago   117MB
debian           latest    4fd3f4b75df3   2 weeks ago      117MB
hello-world      latest    d2c94e258dcb   16 months ago    13.3kB
```

pour exécuter le conteneur :

```
docker run my-hello-world
```

```
root@debian-console:~# docker run my-hello-world
Hello World !
```

2. Créez un fichier index.html dans votre dossier de travail courant.

```
touch index.html
```

3. Ajoutez à l'intérieur quelques lignes de code HTML, elles serviront pour les tests à la fin de la procédure.

```
nano index.html
```

Puis écrivez ceci :

```
<!DOCTYPE html>

<html lang="fr">

<head>

    <meta charset="UTF-8">

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

    <title>Ma Page Web</title>

    <meta name="description" content="Description de ma page
web">

</head>

<body>

    <h1>Bonjour, monde !</h1>

    <p>Ceci est ma première page web.</p>

<!-- Ajoutez ici d'autres éléments HTML selon vos besoins -->
```

```
</body>

</html>
```

4. Vérifiez que le fichier HTML créé est bien présent dans le dossier courant et qu'il contient bien le contenu prévu.

   ```
   cat ./index.html
   ```

5. En vous basant sur une image NGINX, construisez maintenant une nouvelle image nommée *my-nginx* incluant le fichier HTML précédemment créé et qui aura pour objectif de servir ce fichier.

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

6. A l'aide des commandes vues à l'étape 2 de ce TP, lancez un conteneur basé sur l'image fraîchement créée.

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

7. Vérifiez à présent, en vous rendant depuis le navigateur du poste hôte de votre machine virtuelle Docker, que la page HTML précédemment créée s'affiche bien en lieu et place du message *« Welcome to nginx! »*.
   Dans votre navigateur

Étape 4 – Automatisation du déploiement de vos conteneurs

1. Pour vous familiariser avec Docker Compose, créez le fichier de configuration nécessaire qui permet de déployer le même conteneur que celui obtenu à l'étape 2 – question 3 de cette activité.

   Créer un fichier docker-compose.yml

   ```
   version: '3'

   services:

     nginx:

       image: nginx

       ports:

         - "8080:80"
   ```

2. Récupérez d'abord les images contenues dans le fichier de configuration, sans en lancer les conteneurs.

   ```
   docker-compose pull
   ```

3. Lancez maintenant au premier plan le conteneur compris dans le fichier de configuration. Vérifiez ensuite depuis le navigateur de votre ordinateur hôte que vous accédez bien à la page d'accueil de NGINX, en tapant comme URL l'IP de votre VM Debian, suivie du port 8080. Vous devriez accéder à une page indiquant « *Welcome to nginx!* ».

   ```
   docker-compose up
   ```

4. Stoppez le conteneur.

   ```
   docker-compose stop
   ```

5. Lancez cette fois en arrière-plan les conteneurs compris dans le fichier de configuration de la question 3.

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

6. Trouvez un moyen de récupérer en temps-réel les logs de console du conteneur lancé en arrière-plan à la question précédente.

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

7. Stoppez le conteneur. Vous remarquerez que le processus d'arrêt n'est pas le même qu'à la question 4.

```
docker-compose down
```

8. Créez à présent un fichier de configuration permettant d'obtenir une installation WordPress + PostgreSQL fonctionnelle. Les données de la base PostgreSQL ainsi que celles de WordPress devront être persistées dans un volume Docker, afin d'être conservées d'un redémarrage à l'autre. L'installation WordPress devra être accessible de l'extérieur depuis le port 8080 de la VM.

   Créer une fichier docker-compose.yml pour WordPress et PostgreSQL

```
version: '3'
services:
  db:
    image: postgres
    volumes:
      - db_data:/var/lib/postgresql/data
    environment:
      POSTGRES_DB: wordpress
      POSTGRES_USER: wordpress
      POSTGRES_PASSWORD: wordpress_password

  wordpress:
    image: wordpress
    ports:
      - "8080:80"
    volumes:
      - wp_data:/var/www/html
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_NAME: wordpress
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress_password

volumes:
  db_data:
  wp_data:
```

   Lance les conteneurs avec la commande :
   `docker-compose up -d`
   Vérifier l'accès à WordPress grâce à votre navigateur en utilisant l'IP de la VM et le port 8080.