

Compte-rendu

TD2 – Exploitation de Docker

YOUSSEF BOUARICHE

Sommaire

Installation et première utilisation de docker	1
1. Utiliser la machine Linux debian 13 qui est dans le commun 1l	1
2. Démarrez le conteneur en interactif avec un shell en lui donnant le nom « servWebStatique » avec une image d'une machine linux debian. Puis, mettez à jour les paquets du conteneur et installez le serveur Web apache2.....	1
3. Créez une nouvelle image (vos_initiales/debian:trixie-apache2) à partir du conteneur modifié.	2
4. Lancez 2 conteneurs (servweb1 et servweb2) à partir de cette nouvelle image permettant d'accéder à la page Web par défaut d'Apache (le premier mappé sur le port 8001 et le deuxième mappé sur le port 8002).....	3
5. Testez l'accès à chacun des services Web.....	3
6. Vérifier l'adresse ip de votre conteneur, de la machine ou est installé docker (docker network) Comment trouver l'adresse IP d'un conteneur Docker ? - Autre (softoban.com) mettre en place un réseau contenant les 2 sites web. (nom réseau : LAN) Quel est le fichier qui contient les adresses ip de vos conteneurs ?	4
7. Création du site web	4
8. Création du Dockerfile.....	5
9. Construction de l'image Docker	5
10. Exécution du conteneur Docker.....	5
11. Accès au site web	5
12. Installer Docker Compose	6
13. Lire et comprendre un fichier docker-compose.yml	6
14. Tester un docker-compose.yml.....	7

Installation et première utilisation de docker

1. Utiliser la machine Linux debian 13 qui est dans le commun 1l.

La machine linux debian 13 a bien été installé et configuré sur VirtualBox

2. Démarrez le conteneur en interactif avec un shell en lui donnant le nom « servWebStatique » avec une image d'une machine linux debian. Puis, mettez à jour les paquets du conteneur et installez le serveur Web apache2.

Afin de créer et démarrer un conteneur à partir d'une image, nous allons utiliser la commande **docker run**. Pour qu'elle ait le nom demandé et qu'elle soit en interactif, nous allons écrire la commande comme ceci :

docker run –name servWebStatique -it debian

```
root@srvdeb13:~# docker run --name servWebStatique -it debian
```

--name : permet de rajouter un nom à notre conteneur

-it : rends le conteneur interactif avec un shell (I pour interaction, T pour terminal)

debian : l'image dont va se servir le conteneur

Une fois la commande réalisée, on passera directement à l'intérieur du conteneur. Un moyen de vérifier est par le nom de l'invite de commande qui devrait avoir changé si l'on est bien dans le conteneur, avec l'ID du conteneur en question :

```
root@67bf248c2dad:/# .
```

Pour mettre à jour les paquets, nous devons utiliser la commande **apt update** DANS le conteneur, afin de mettre à jour la liste des paquets disponible. Ensuite, on mettra à jour avec la commande **apt upgrade**. une demande de confirmation sera demandée il faudra donc taper « Y » pour confirmer dans l'invite de commande.

```
root@67bf248c2dad:/# apt update
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
root@67bf248c2dad:/#
```

apt : gestionnaire de paquets Debian

Pour installer le serveur Web apache2, nous utiliserons la commande **apt-get install apache2**. Si aucun message d'erreur s'affiche, nous pouvons continuer :

```
root@67bf248c2dad:/# apt-get install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.65-2).
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

Il faut maintenant démarrer le service **apache2** que nous venons d'installer, avec la commande :
service apache2 start

```
root@67bf248c2dad:/# service apache2 start
Starting Apache httpd web server: apache2
[Thu Jul 20 11:00:00 2023] [warn] [client ::1:50000] AH00524: 'ServerName' directive globally to suppress this
.
root@67bf248c2dad:/# service apache2 status
apache2 is running.
```

service apache2 status permet de vérifier l'état du service **apache2**, si il est bien allumé (running) ou non (down).

3. Créez une nouvelle image (**vos_initiales/debian:trixie-apache2**) à partir du conteneur modifié.

Afin de créer une nouvelle image à partir de notre conteneur déjà existant (et modifié), nous allons utiliser la commande **docker commit**. Dans ce cas précis :

docker commit servWebStatique yb/debian:trixie-apache2

```
exit
root@srvdeb13:~# docker commit servWebStatique yb/debian:trixie-apache2
sha256:debd93e8dd15bb13c8ef1947bf41afc7a2e1bdcda0c5fcf4cc1bf4d3620a1866
```

la ligne **sha256** nous permet de confirmer que l'image a bien été confirmé, car elle nous affiche justement son ID.

4. Lancez 2 conteneurs (servweb1 et servweb2) à partir de cette nouvelle image permettant d'accéder à la page Web par défaut d'Apache (le premier mappé sur le port 8001 et le deuxième mappé sur le port 8002)

Afin de lancer ces deux nouveaux conteneurs à partir de cette nouvelle image, nous allons utiliser **docker run**. Dans ce cas précis :

docker run -d -p 8001:80 –name servweb1 yb/debian:trixie-apache2

et

docker run -d -p 8002:80 –name servweb2 yb/debian:trixie-apache2

-d : permets de lancer le conteneur en arrière plan

-p : permets de mapper sur un port

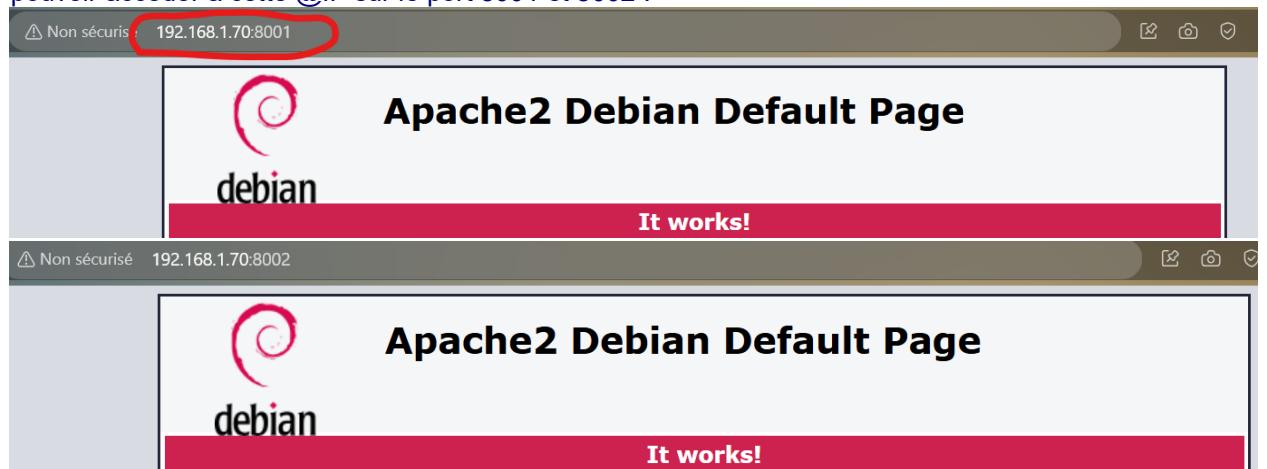
Si tout est bon, on devrait pouvoir accéder depuis la machine hôte au serveur web d'apache sur notre @IP dans le port 8001 et 8002.

5. Testez l'accès à chacun des services Web

Il faut d'abord trouver son @IP sur la machine virtuelle debian 13, en utilisant la commande **ip a** :

```
2: enp0s3: <BROADCAST,MULTICAST,NOARP>
      link/ether 08:00:27: altname enx080027879
      inet 192.168.1.70/24
```

Maintenant qu'on a pris connaissance de notre adresse IP, comme dit précédemment, on devrait pouvoir accéder à cette @IP sur le port 8001 et 8002 :



6. Vérifier l'adresse ip de votre conteneur, de la machine ou est installé docker (docker network) Comment trouver l'adresse IP d'un conteneur Docker ? - Autre (softoban.com) mettre en place un réseau contenant les 2 sites web. (nom réseau : LAN) Quel est le fichier qui contient les adresses ip de vos conteneurs ?

Pour trouver l'adresse IP d'un conteneur Docker, on peut utiliser la commande **docker inspect** qui permet d'afficher toutes les informations détaillées du conteneur.
(Par soucis de lisibilité je ne vais prendre en capture d'écran que l'adresse IP)

```
docker inspect servweb1 : {"IPAddress": "172.17.0.4",
```

```
docker inspect servweb2 : {"IPAddress": "172.17.0.5",
```

Maintenant qu'on connaît l'adresse ip de nos conteneurs, on peut mettre en place notre réseau de nom LAN, grâce à la commande **docker network create LAN**. Si la commande a bien été effectuée, un ID devrait s'afficher après la commande :

```
root@srvdeb13:~# docker network create LAN  
7135569c3b1d9a696bb26573dfb2b48e9658fa0267343bc087358455ae0ab352
```

On mets ensuite en lien au réseau LAN nos deux conteneurs avec la commande **docker network connect** :

```
root@srvdeb13:~# docker network connect LAN servweb1 && docker network connect LAN servweb3  
root@srvdeb13:~#
```

Une manière de s'assurer que nos deux conteneurs sont bien connectés à notre réseau LAN et en utilisant de nouveau la commande **docker inspect** mais cette fois si pour le réseau LAN :

```
"Containers": {  
    "293f5591ea4411d4ccfd0e59e38edf7f572607dd49330f606bb7e4c3c54a72f": {  
        "Name": "servweb1",  
        "EndpointID": "6fc7e722577ba5a04da3a53043988b182d3283b25cae704e00a177b54605974b",  
        "MacAddress": "fa:4c:e7:85:f1:db",  
        "IPv4Address": "172.18.0.2/16",  
        "IPv6Address": ""  
    },  
    "fccc0050a2fc206c30002acbb699850f24bcd716de5fe0bd06f7ef3516b5abd8": {  
        "Name": "servweb3",  
        "EndpointID": "810553c5cba281c650bd8e0392928481708171882f35d7032de0320086d93c57",  
        "MacAddress": "ea:80:e8:f1:42:d3",  
        "IPv4Address": "172.18.0.3/16",  
        "IPv6Address": ""  
    }  
},
```

On peut retrouver dans la catégorie « Containers » (conteneurs en français) nos deux conteneurs **servweb**, on peut voir qu'une nouvelle adresse IP leur a été assigné dans le réseau LAN.

Le fichier qui contient les adresses ip de nos conteneurs est **/etc/hosts**. Attention, ce fichier est à chercher **DANS** les conteneurs, il ne sera pas présent sur le fichier **/etc/hosts** de notre machine virtuelle Debian 13.

7. Crédation du site web

- a) Créez un répertoire vide dans /root pour votre projet. Vous pouvez le nommer, par exemple, "siteweb" : on se place dans /root avec la commande « cd /root », puis on utilise la commande « mkdir siteweb »
- b) À l'intérieur de ce répertoire, créez un fichier HTML nommé index.html en utilisant l'éditeur nano. : on commence par se placer dans ce répertoire avec la commande « cd siteweb », puis on utilise « nano index.html » afin de créer ce fichier
- c) Dans index.html, ajoutez un contenu HTML de base

Voici le résultat attendu une fois avoir réalisé les commandes donné sur le TD :

```
GNU nano 8.4
<html>
<head>
    <title>Mon Site Web Docker</title>
</head>
<body> <h1>Bienvenue sur mon site web Docker !</h1>
</body> </html>
```

8. Crédation du Dockerfile

- d) À l'intérieur du répertoire "siteweb", créez un fichier nommé Dockerfile (sans extension) : **nano Dockerfile**
- e) Dans ce fichier, ajoutez les instructions suivantes pour créer une image Docker minimale avec un serveur web Apache :

```
# Installation de l'image httpd
FROM httpd:latest
# Copie du contenu du répertoire local dans le répertoire du site web Apache
COPY ./index.html /usr/local/apache2/htdocs/
```

9. Construction de l'image Docker

- f) Accédez au répertoire "siteweb" : **chemin absolu pour accéder au répertoire = cd /root/siteweb**
- g) Utilisez la commande suivante pour construire l'image Docker à partir du Dockerfile : **dockerbuild -t siteweb** .

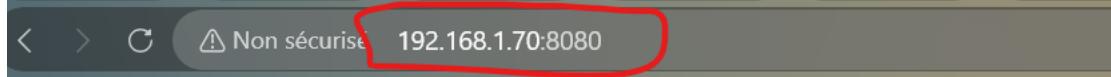
Résultat attendu : `root@srvdeb13:~/siteweb# docker build -t siteweb .
[+] Building 2.9s (8/8) FINISHED`

10. Exécution du conteneur Docker

- h) Une fois l'image construite, exécutez un conteneur Docker à partir de cette image avec la commande suivante : `root@srvdeb13:~/siteweb# docker run -d -p 8080:80 siteweb
91e16c10cf36cb3002d87ef11218806fc835017d79a383d0f1d109fb58ebcacc`

11. Accès au site web

- i) Ouvrez votre navigateur web sur la machine Windows 10 et accédez à l'adresse <http://@IP de la machine docker:8080> pour voir votre site web Docker en action :



Bienvenue sur mon site web Docker !

12. Installer Docker Compose

- a) Vérifier la version de Docker : **docker version**

```
root@srvdeb13:~/siteweb# docker version
Client: Docker Engine - Community
  Version:           29.0.0
```

- b) Vérifier si Docker Compose est déjà installé : **docker compose version**. On peut ainsi remarquer que Docker Compose n'est pas installé sur la machine virtuelle. **On passe donc à l'étape c)** pour l'installer.

- c) Installer Dockercompose

On commence par utiliser **apt update**, puis on installe Docker Compose avec **apt install docker-compose-plugin**. Si Docker Compose a bien été installé, on devrait obtenir ceci lorsqu'on tape la commande **docker compose version** :

```
root@srvdeb13:~/siteweb# docker compose version
Docker Compose version v2.40.3
```

13. Lire et comprendre un fichier docker-compose.yml

- d) Créez un répertoire de test **mkdir tp_docker_compose && cd tp_docker_compose**

On n'oublie pas de se replacer sur root pour ne pas créer nos répertoires dans le répertoire siteweb.
On tape ensuite la commande pour créer le répertoire de test et se placer directement dessus.

- e) Créez un fichier docker-compose.yml à la racine du dossier

On commence par créer le fichier avec la commande **nano docker-compose.yml**, on écrit ensuite dans ce fichier ce qui est demandé dans le TD :

```
GNU nano 8.4
version: '3.8'

services:
  web:
    image: nginx
    ports:
      - "8080:80"

  db:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: testdb
      MYSQL_USER: user
      MYSQL_PASSWORD: userpass
```

f) Annoter par les étudiants le rôle de chaque ligne

version : 3.8 = La version de Docker Compose qu'on va utiliser

services := listes des conteneurs à lancer

web := nom du conteneur, permet ensuite de définir en dessous sa configuration (avec indentation)

image : **nginx** = l'image utilisé pour le conteneur « **web** »

ports : = permet de mapper, de définir la redirection de ports

 ‘**8080:80**’ = numéro de port choisi pour le conteneur « **web** »

(retour en début de ligne pour sortir de la configuration du conteneur « **web** »)

db = nom du deuxième conteneur qu'on va configurer (avec indentation)

image : **mysql :8.0** = l'image + la version utilisé pour le conteneur « **db** »

environment = liste des variables envoyées à *MySQL* (avec une indentation en plus)

MYSQL_ROOT_PASSWORD : **root** = mot de passe du compte root MySQL

MYSQL_DATABASE : **testdb** = crée une base de données nommée « **testdb** »

MYSQL_USER : **user** = crée un utilisateur nommé « **user** »

MYSQL_PASSWORD : **userpass** = définit le mot de passe de l'utilisateur « **user** »

14. Tester un docker-compose.yml

g) Étapes à suivre :

a. Lancer l'environnement : **docker compose up -d**, les dernières lignes doivent nous confirmer que les 2 conteneurs « **web** » et « **db** » ont bien été lancé :

```
[+] Running 3/3
  └─ Network tp_docker_compose_default   Created
    └─ Container tp_docker_compose-web-1  Started
      └─ Container tp_docker_compose-db-1  Started
```

b. Lister les conteneurs actifs : **docker compose ps**, cela nous permet de voir les conteneurs actifs, voici ce que j'ai pu voir :

NAME	IMAGE	SERVICE	STATUS
tp_docker_compose-db-1	mysql:8.0	db	UP
tp_docker_compose-web-1	nginx	web	UP

La colonne STATUS permet de confirmer que les conteneurs sont bien actif (UP).

c. Tester dans le navigateur :



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

d. La base MySQL tourne en arrière-plan (non visible mais accessible). Tester.

On sait grâce à la commande **docker compose ps** que MySQL est actif (UP). Pour montrer qu'elle est accessible, on peut faire un simple test en exécutant une commande **echo** afin de voir si elle reçoit bien la commande et si elle nous répond.

(Commande complète dans ce cas : **docker exec tp_docker_compose-db-1 echo « test »**)

```
root@srvdeb13:~/tp_docker_compose# docker exec tp_docker_compose-db-1 echo "test"
test
```

Le conteneur nous a bien répondu en nous renvoyant le mot « test ».

e. Afficher les logs.

Pour afficher les logs, on utilise la commande « **docker compose logs -f** ». rajouter **-f** à la commande permet d'afficher les logs en temps réel, ils se mettent à jour constamment.

Voici mes logs :

```
initial-insecure option.
db-1 | 2025-11-28 08:09:14+00:00 [Note] [Entrypoint]: Database files initialized
db-1 | 2025-11-28 08:09:14+00:00 [Note] [Entrypoint]: Starting temporary server
db-1 | 2025-11-28T08:09:14.717287Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release use SET GLOBAL host_cache_size=0 instead.
db-1 | 2025-11-28T08:09:14.722180Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.44) starting as process 119
db-1 | 2025-11-28T08:09:14.802825Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
db-1 | 2025-11-28T08:09:16.088761Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
db-1 | 2025-11-28T08:09:17.601819Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
db-1 | 2025-11-28T08:09:17.601559Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported by this channel.
db-1 | 2025-11-28T08:09:17.613060Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is exposed to all OS users. Consider choosing a different directory.
db-1 | 2025-11-28T08:09:17.655327Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Socket: /var/run/mysqld/mysql.sock
db-1 | 2025-11-28T08:09:17.655936Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.44' socket: '/var/run/mysql.sock' port: 0 MySQL Community Server - GPL.
db-1 | 2025-11-28 08:09:17+00:00 [Note] [Entrypoint]: Temporary server started.
db-1 | 2025-11-28 08:09:17+00:00 [Note] [Entrypoint]: '/var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'
db-1 | Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.
db-1 | Warning: Unable to load '/usr/share/zoneinfo/leap-seconds.list' as time zone. Skipping it.
db-1 | Warning: Unable to load '/usr/share/zoneinfo/leapseconds' as time zone. Skipping it.
db-1 | Warning: Unable to load '/usr/share/zoneinfo/tzdata.zi' as time zone. Skipping it.
db-1 | Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.
db-1 | Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.
db-1 | 2025-11-28 08:09:39+00:00 [Note] [Entrypoint]: Creating database testdb
db-1 | 2025-11-28 08:09:39+00:00 [Note] [Entrypoint]: Creating user user
db-1 | 2025-11-28 08:09:39+00:00 [Note] [Entrypoint]: Giving user user access to schema testdb
db-1 |
db-1 | 2025-11-28 08:09:43+00:00 [Note] [Entrypoint]: Stopping temporary server
db-1 | 2025-11-28T08:09:39.729820Z 13 [System] [MY-013172] [Server] Received SHUTDOWN from user root. Shutting down mysqld (Version: 8.0.44).
db-1 | 2025-11-28T08:09:43.144218Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.44) MySQL Community Server - GPL
db-1 | 2025-11-28 08:09:43+00:00 [Note] [Entrypoint]: Temporary server stopped
db-1 |
db-1 | 2025-11-28 08:09:43+00:00 [Note] [Entrypoint]: MySQL init process done. Ready for start up.
db-1 |
db-1 | 2025-11-28T08:09:44.041925Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release use SET GLOBAL host_cache_size=0 instead.
db-1 | 2025-11-28T08:09:44.044132Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.44) starting as process 1
db-1 | 2025-11-28T08:09:44.054912Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
db-1 | 2025-11-28T08:09:45.243998Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
db-1 | 2025-11-28T08:09:46.116981Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
db-1 | 2025-11-28T08:09:46.117939Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported by this channel.
db-1 | 2025-11-28T08:09:46.138130Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is exposed to all OS users. Consider choosing a different directory.
db-1 | 2025-11-28T08:09:46.156206Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysql.sock
db-1 | 2025-11-28T08:09:46.156721Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.44' socket: '/var/run/mysql.sock' port: 3306 MySQL Community Server - GPL.
```