TP 1 Docker

Sommaire 1

Docker install – set-up

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/do
sudo 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] ht
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

```
ubuntu@ubuntu:~$ # Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo 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/ubun
tu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
Ign:1 cdrom://Ubuntu 24.04 LTS _Noble Numbat_ - Release amd64 (20240424) noble InRelease
Hit:2 cdrom://Ubuntu 24.04 LTS _Noble Numbat_ - Release amd64 (20240424) noble Release
Hit:3 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:5 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:7 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
The following additional packages will be installed:
  libcurl3t64-gnutls libcurl4t64
The following NEW packages will be installed:
  curl
The following packages will be upgraded:
```

```
ubuntu@ubuntu:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli
  docker-ce-rootless-extras docker-compose-plugin libslirp0 pigz slirp4netns
0 upgraded, 9 newly installed, 0 to remove and 146 not upgraded.
Need to get 121 MB of archives.
After this operation, 434 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 https://download.docker.com/linux/ubuntu noble/stable amd64 containerd.io amd64 1.6.32-1 [30.0 MB]
Get:2 http://archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble/main amd64 libslirp0 amd64 4.7.0-1ubuntu3 [63.8 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble/universe amd64 slirp4netns amd64 1.2.1-1build2 [34.9 kB]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-buildx-plugin amd64 0.14.0-1~ubuntu.24.04~noble
 [29.7 MB]
Get:6 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-cli amd64 5:26.1.3-1~ubuntu.24.04~noble [14.
6 MB]
Get:7 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce amd64 5:26.1.3-1~ubuntu.24.04~noble [25.3 MB
]
Get:8 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-ce-rootless-extras amd64 5:26.1.3-1~ubuntu.24.0
4~noble [9319 kB]
Get:9 https://download.docker.com/linux/ubuntu noble/stable amd64 docker-compose-plugin amd64 2.27.0-1~ubuntu.24.04~nobl
e [12.5 MB]
Fetched 121 MB in 4s (29.8 MB/s)
Selecting previously unselected package pigz.
(Reading database ... 217513 files and directories currently installed.)
```

```
ubuntu@ubuntu:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:266b191e926f65542fa8daaec01a192c4d292bff79426f47300a046e1bc576fd
Status: Downloaded newer image for hello-world:latest

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/
```

```
ubuntu@ubuntu:~/docker_yml$ sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(un
ame -s)-$(uname -m)" -o /usr/local/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 60.1M  100 60.1M    0     0  7283k      0  0:00:08  0:00:08 --:--:-- 9857k
ubuntu@ubuntu:~/docker_yml$ docker-compose --version
Docker Compose version v2.27.1
```

Git-hub install – set-up

```
ubuntu@ubuntu:~$ sudo apt-get install git-all
```

```
ubuntu@ubuntu:~$ git clone https://github.com/ACAPO21/devops.git
Cloning into 'devops'...
warning: You appear to have cloned an empty repository.
ubuntu@ubuntu:~$ touch test.txt
```

```
ubuntu@ubuntu:~$ git add test.txt
ubuntu@ubuntu:~$ git commit -m "first commit"
[main (root-commit) 2c59d37] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.txt
ubuntu@ubuntu:~$ git branch -M main
```

```
ubuntu@ubuntu:~$ git push -u origin main
Username for 'https://github.com': ACAPO21
Password for 'https://ACAPO21@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 211 bytes | 211.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ACAPO21/devops.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

5. Exécuter un serveur web (apache, nginx, …) dans un conteneur docker

a. Récupérer l'image sur le Docker Hub

```
ubuntu@ubuntu:~$ sudo docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
09f376ebb190: Pull complete
5529e0792248: Pull complete
9b3addd3eb3d: Pull complete
57910a8c4316: Pull complete
7b5f78f21449: Pull complete
b7923aa4e8a6: Pull complete
785625911f12: Pull complete
Digest: sha256:0f04e4f646a3f14bf31d8bc8d885b6c951fdcf42589d06845f64d18aec6a3c4d
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

b. Vérifier que cette image est présente en local

```
ubuntu@ubuntu:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
nginx         latest    4f67c83422ec   38 hours ago    188MB
hello-world   latest    d2c94e258dcb   13 months ago   13.3kB
```

c. Créer un fichier index.html simple

```
ubuntu@ubuntu:~$ touch index.html
ubuntu@ubuntu:~$ nano index.html
ubuntu@ubuntu:~$ cat index.html
<html>
<head>
        <title>PAGE DEVOPS</title>
</head>
</html>
```

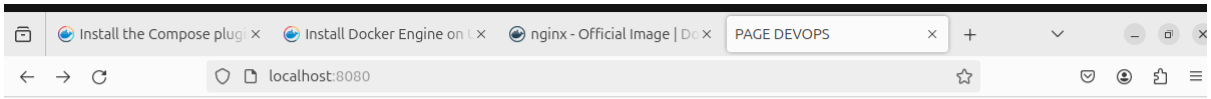
```
ubuntu@ubuntu:~$ mkdir my_website
ubuntu@ubuntu:~$ mv index.html /m
media/ mnt/
ubuntu@ubuntu:~$ mv index.html my_website/
ubuntu@ubuntu:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos  devops  my_website  snap  test.txt
```

d. Démarrer un conteneur et servir la page html créée précédemment à l'aide d'un volume (option -v de docker run)

```
ubuntu@ubuntu:~$ sudo docker run --name my_nginx -v ~/my_website:/usr/share/nginx/html:ro -p 8080:80 -d nginx
df3d13379f3492a5a38c44cb36d37b17f827148a7e5728b79d79c8946a6b7f15
```

```
ubuntu@ubuntu:~$ sudo docker exec -it my_nginx /bin/bash
root@df3d13379f34:/# ls
bin   dev                  docker-entrypoint.sh  home  lib64  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint.d  etc                   lib   media  opt  root  sbin  sys  usr
root@df3d13379f34:/# cd usr/
root@df3d13379f34:/usr# ls
bin  games  include  lib  lib64  libexec  local  sbin  share  src
root@df3d13379f34:/usr# cd share
root@df3d13379f34:/usr/share# ls
X11             ca-certificates  doc         gcc       libc-bin     maven-repo  pam-configs  terminfo
base-files      common-licenses  doc-base    gdb       libgcrypt20  menu        perl5        util-linux
base-passwd     debconf          dpkg        info      lintian      misc        pixmaps      xml
bash-completion debianutils      fontconfig  java      locale       nginx       polkit-1     zoneinfo
bug             dict             fonts       keyrings  man          pam         tabset       zsh
root@df3d13379f34:/usr/share# cd nginx/
root@df3d13379f34:/usr/share/nginx# ls
html
root@df3d13379f34:/usr/share/nginx# cd html/
root@df3d13379f34:/usr/share/nginx/html# ls
index.html
root@df3d13379f34:/usr/share/nginx/html# cat index.html
<html>
<head>
        <title>PAGE DEVOPS</title>
</head>
</html>
```

On voit bien PAGE DEVOPS héberger en localhost :



e. Supprimer le conteneur précédent et arriver au même résultat que précédemment à l'aide de la commande docker cp

```
ubuntu@ubuntu:~$ sudo docker stop my_nginx
my_nginx
ubuntu@ubuntu:~$ sudo docker rm my_nginx
my_nginx
ubuntu@ubuntu:~$ sudo docker run --name my_nginx -p 8080:80 -d nginx
936de70ee94dd8074c1b499d05515d90cfae23aa25ffc5cccf6bc58c5620fdae
ubuntu@ubuntu:~$ sudo docker cp \~my_website/index.html my_nginx:usr/share/nginx/html/index.html
Successfully copied 2.05kB to my_nginx:usr/share/nginx/html/index.html
```

6. Builder une image

a. A l'aide d'un Dockerfile, créer une image (commande docker build)

```
ubuntu@ubuntu:~$ mkdir ~docker_image
ubuntu@ubuntu:~$ cd \~docker_image/
ubuntu@ubuntu:~/~docker_image$ ls
ubuntu@ubuntu:~/~docker_image$ echo "<h1> HELLO DEVOPS </h1>" > index.html
ubuntu@ubuntu:~/~docker_image$ nano Dockerfile
ubuntu@ubuntu:~/~docker_image$ ls
Dockerfile  index.html
ubuntu@ubuntu:~/~docker_image$ cat Dockerfile
FROM nginx:lastest
COPY index.html /usr/share/nginx/html/index.html
```

```
ubuntu@ubuntu:~/~docker_image$ sudo docker build -t nginx_image .
[+] Building 0.9s (7/7) FINISHED                                                docker:default
 => [internal] load build definition from Dockerfile                                      0.0s
 => => transferring dockerfile: 105B                                                      0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                           0.0s
 => [internal] load .dockerignore                                                         0.0s
 => => transferring context: 2B                                                           0.0s
 => [internal] load build context                                                         0.0s
 => => transferring context: 61B                                                          0.0s
 => [1/2] FROM docker.io/library/nginx:latest                                             0.0s
 => [2/2] COPY index.html /usr/share/nginx/html/index.html                                0.3s
 => exporting to image                                                                    0.5s
 => => exporting layers                                                                   0.5s
 => => writing image sha256:3bf69b258a765e5d8f308678a9ece629803b5c3befd16fc2fb6ee1c22806f95b  0.0s
 => => naming to docker.io/library/nginx_image                                            0.0s
```

b. Exécuter cette nouvelle image de manière à servir la page html (commande docker run)

```
ubuntu@ubuntu:~/~docker_image$ sudo docker run --name nginx_image_container -p 8080:80 -d nginx_image
64568c138910930a50b7c076d426443bb15386eb574c61ab08e6f634d491d56f
```

c. Quelles différences observez-vous entre les procédures 5. et 6. ? Avantages et inconvénients de l'une et de l'autre méthode ?

La principale différence entre l'utilisation d'un Dockerfile et une procédure manuelle (comme dans la procédure 5) réside dans la facilité de déploiement et d'automatisation. Avec un Dockerfile, tout est facilement déployable de manière automatisée, tandis qu'avec la procédure manuelle, chaque étape doit être exécutée manuellement, ce qui peut être fastidieux et sujet à des erreurs.

- **Avantages** :
  Crée une image réutilisable et portable.
  Simplifie le déploiement car tout est défini dans le Dockerfile.
  Meilleure pratique pour les environnements de production.
- **Inconvénients** :
  Nécessite la construction de l'image après chaque modification.

Peut-être plus lent à itérer pendant le développement.

7. Utiliser une base de données dans un conteneur docker

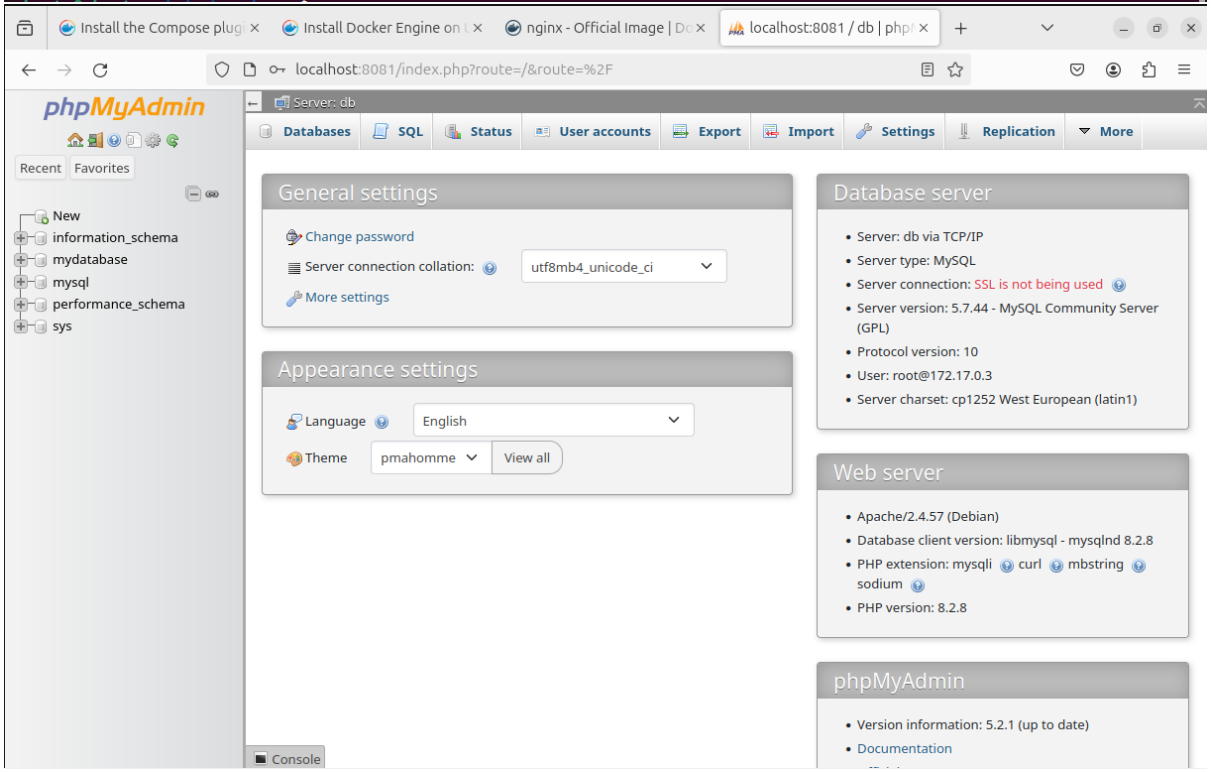
a. Récupérer les images mysql:5.7 et phpmyadmin depuis le Docker Hub

```
ubuntu@ubuntu:~/~docker_image$ sudo docker pull mysql:5.7
5.7: Pulling from library/mysql
20e4dcae4c69: Pull complete
1c56c3d4ce74: Pull complete
e9f03a1c24ce: Pull complete
68c3898c2015: Pull complete
6b95a940e7b6: Pull complete
90986bb8de6e: Pull complete
ae71319cb779: Pull complete
ffc89e9dfd88: Pull complete
43d05e938198: Pull complete
064b2d298fba: Pull complete
df9a4d85569b: Pull complete
Digest: sha256:4bc6bc963e6d8443453676cae56536f4b8156d78bae03c0145cbe47c2aad73bb
Status: Downloaded newer image for mysql:5.7
docker.io/library/mysql:5.7
```

```
ubuntu@ubuntu:~/~docker_image$ sudo docker pull phpmyadmin/phpmyadmin
Using default tag: latest
latest: Pulling from phpmyadmin/phpmyadmin
faef57eae888: Pull complete
989a1d6c052e: Pull complete
0705c9c2f22d: Pull complete
621478e043ce: Pull complete
98246dcca987: Pull complete
bfed8c155cb6: Pull complete
7a7c2e908867: Pull complete
d176994b625c: Pull complete
2d8ace6a2716: Pull complete
c70df516383c: Pull complete
15e1b44fe4c7: Pull complete
65e50d44e95a: Pull complete
77f68910bc0a: Pull complete
605dd3a6e332: Pull complete
99ce27188f07: Pull complete
74d64e32c5d5: Pull complete
ef5fc9928b9f: Pull complete
163f3256e112: Pull complete
Digest: sha256:67ba2550fd004399ab0b95b64021a88ea544011e566a9a1995180a3decb6410d
Status: Downloaded newer image for phpmyadmin/phpmyadmin:latest
docker.io/phpmyadmin/phpmyadmin:latest
```

b. Exécuter deux conteneurs à partir des images et ajouter une table ainsi que quelques enregistrements dans la base de données à l'aide de phpmyadmin

```
ubuntu@ubuntu:~/~docker_image$ sudo docker run --name mysql_container -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=myda
tabase -d mysql:5.7
d2525518e96c562e60c1fc0f7b09e3df6284ddc0626f082599c2d598acebc224
ubuntu@ubuntu:~/~docker_image$ sudo docker run --name php_container --link mysql_container:db -p 8081:80 -d phpmyadmin/p
hpmyadmin
24c2201fd3de3ed421c83fd2c8a0d7500c5bb8a036683a7e78c4a219166e4ff3
```

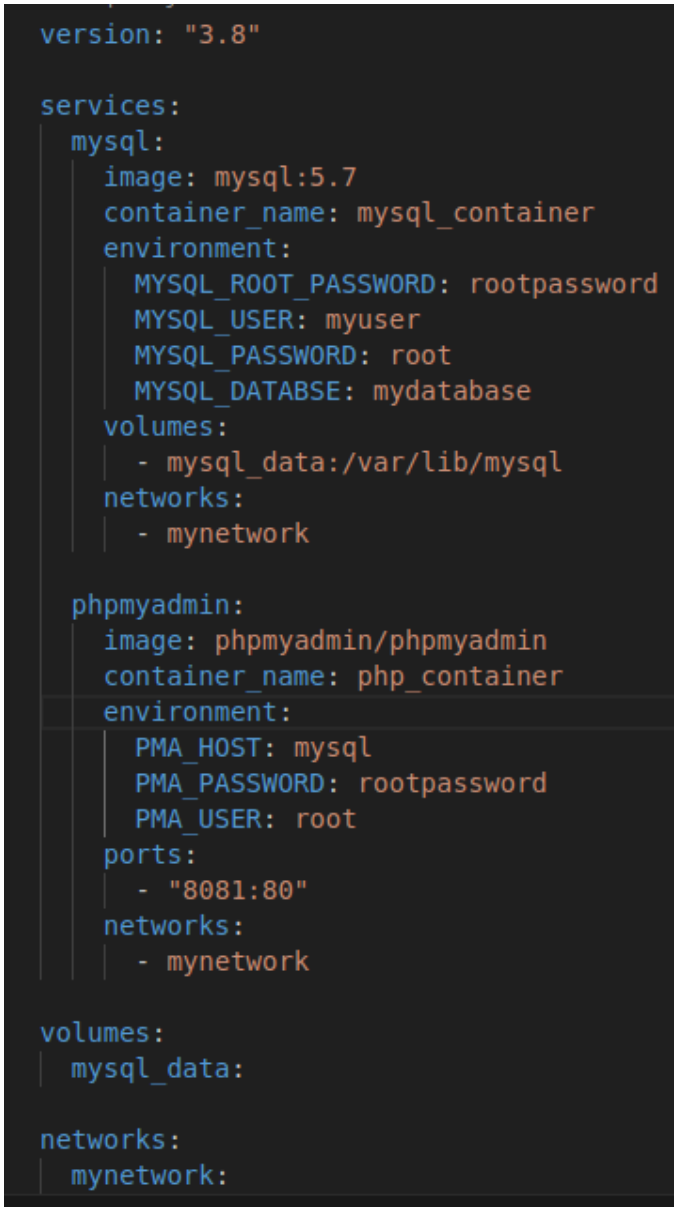


8. Faire la même chose que précédemment en utilisant un fichier

a. Qu'apporte le fichier docker-compose par rapport aux commandes docker run ? Pourquoi est-il intéressant ? (cf. ce qui a été présenté pendant le cours)

- **Simplicité** : Vous pouvez définir et gérer plusieurs conteneurs avec un seul fichier YAML, ce qui simplifie la configuration et la gestion des conteneurs.
- **Portabilité** : Le fichier `docker-compose.yml` peut être partagé et versionné, facilitant la collaboration et le déploiement sur différents environnements.
- **Répétabilité** : Les configurations sont reproductibles. Vous pouvez facilement recréer l'environnement de conteneur en exécutant `docker-compose up`.
- **Gestion des dépendances** : Vous pouvez définir des dépendances entre les services, garantissant ainsi que les conteneurs démarrent dans le bon ordre.
- **Variables d'environnement** : Vous pouvez utiliser des variables d'environnement pour paramétrer vos services, rendant la configuration flexible et sécurisée.

b. Quel moyen permet de configurer (premier utilisateur, première base de données, mot de passe root, …) facilement le conteneur mysql au lancement ?



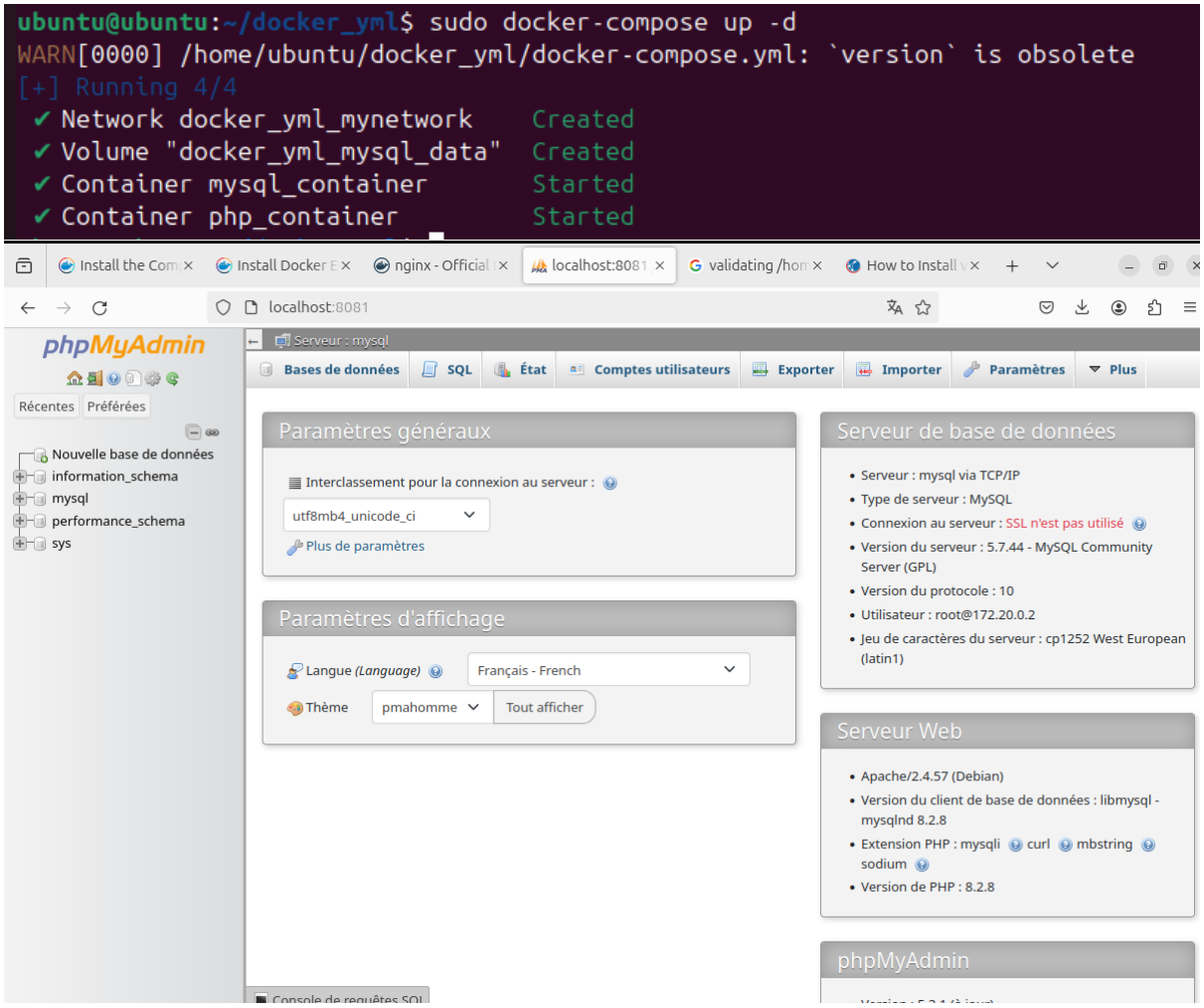
```yaml
version: "3.8"

services:
  mysql:
    image: mysql:5.7
    container_name: mysql_container
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_USER: myuser
      MYSQL_PASSWORD: root
      MYSQL_DATABSE: mydatabase
    volumes:
      - mysql_data:/var/lib/mysql
    networks:
      - mynetwork

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: php_container
    environment:
      PMA_HOST: mysql
      PMA_PASSWORD: rootpassword
      PMA_USER: root
    ports:
      - "8081:80"
    networks:
      - mynetwork

volumes:
  mysql_data:

networks:
  mynetwork:
```


ubuntu@ubuntu:~/docker_yml$ sudo docker-compose up -d
WARN[0000] /home/ubuntu/docker_yml/docker-compose.yml: `version` is obsolete
[+] Running 4/4
✔ Network docker_yml_mynetwork Created
✔ Volume "docker_yml_mysql_data" Created
✔ Container mysql_container Started
✔ Container php_container Started
localhost:8081
phpMyAdmin
Récentes
Préférées
Nouvelle base de données
information_schema
mysql
performance_schema
sys
Serveur : mysql
Bases de données
SQL
État
Comptes utilisateurs
Exporter
Importer
Paramètres
Plus
Paramètres généraux
Interclassement pour la connexion au serveur :
utf8mb4_unicode_ci
Plus de paramètres
Paramètres d'affichage
Langue (Language)
Français - French
Thème
pmahomme
Tout afficher
Serveur de base de données
Serveur : mysql via TCP/IP
Type de serveur : MySQL
Connexion au serveur : SSL n'est pas utilisé
Version du serveur : 5.7.44 - MySQL Community Server (GPL)
Version du protocole : 10
Utilisateur : root@172.20.0.2
Jeu de caractères du serveur : cp1252 West European (latin1)
Serveur Web
Apache/2.4.57 (Debian)
Version du client de base de données : libmysql - mysqlnd 8.2.8
Extension PHP : mysqli curl mbstring sodium
Version de PHP : 8.2.8
phpMyAdmin

9. Observation de l’isolation réseau entre 3 conteneurs

a. A l’aide de docker-compose et de l’image praqma/network-multitool disponible sur le Docker Hub créer 3 services (web, app et db) et 2 réseaux (frontend et backend). Les services web et db ne devront pas pouvoir effectuer de ping de l’un vers l’autre

```yaml
version: "3.8"

services:
  web:
    image: praqma/network-multitool
    container_name: web_container
    networks:
      - frontend
  app:
    image: praqma/network-multitool
    container_name: app_container
    networks:
      - frontend
      - backend
  db:
    image: praqma/network-multitool
    container_name: db_container
    networks:
      - frontend

networks:
  frontend:
  backend:
```

```
ubuntu@ubuntu:~/compose$ sudo docker-compose up -d
WARN[0000] /home/ubuntu/compose/docker-compose.yml: `version` is obsolete
[+] Running 10/10
 ✔ db Pulled
 ✔ web Pulled
 ✔ app Pulled
   ✔ 5758d4e389a3 Pull complete
   ✔ 89d2c42e021e Pull complete
   ✔ c56ef2f6b498 Pull complete
   ✔ fb4370a69dda Pull complete
   ✔ 003f3d74368c Pull complete
   ✔ cd3def2cca55 Pull complete
   ✔ ba5a2b2d204e Pull complete
[+] Running 5/5
 ✔ Network compose_frontend  Created
 ✔ Network compose_backend   Created
 ✔ Container db_container    Started
 ✔ Container web_container   Started
 ✔ Container app_container   Started
```

```
ubuntu@ubuntu:~/compose$ sudo docker exec -it web_container /bin/sh
/ # ping db_container
PING db_container (172.21.0.2) 56(84) bytes of data.
64 bytes from db_container.compose_frontend (172.21.0.2): icmp_seq=1 ttl=64 time=0.124 ms
64 bytes from db_container.compose_frontend (172.21.0.2): icmp_seq=2 ttl=64 time=0.168 ms
64 bytes from db_container.compose_frontend (172.21.0.2): icmp_seq=3 ttl=64 time=0.500 ms
64 bytes from db_container.compose_frontend (172.21.0.2): icmp_seq=4 ttl=64 time=0.070 ms
64 bytes from db_container.compose_frontend (172.21.0.2): icmp_seq=5 ttl=64 time=0.216 ms
64 bytes from db_container.compose_frontend (172.21.0.2): icmp_seq=6 ttl=64 time=0.084 ms
64 bytes from db_container.compose_frontend (172.21.0.2): icmp_seq=7 ttl=64 time=0.218 ms
^C
--- db_container ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6181ms
rtt min/avg/max/mdev = 0.070/0.197/0.500/0.135 ms
```

b. Quelles lignes du résultat de la commande docker inspect justifient ce comportement ?

c. Dans quelle situation réelles (avec quelles images) pourrait-on avoir cette configuration réseau ? Dans quel but ?

Cette configuration réseau peut être utile dans une architecture microservices où vous souhaitez isoler différents services pour des raisons de sécurité ou de gestion du trafic.