Rendu évaluation 1 Docker

Groupe :

Rayan KHALFOUN / Nabil HATRI

Task1 :

1. Nous avons commencé par créer un réseau, que nous avons appelé : « mon_reseau_nabil_rayan », en utilisant la commande :

```
C:\Users\nabil>docker network create mon_reseau_nabil_rayan
f666a0620b4e8b3975682dc3f3e258ab48815d0faa1b7aa0fc6db2be7977769d
```

2. Une fois notre réseau créé, nous allons commencer par créer deux volumes pour la persistance de nos données qui seront associés à nos containers, voici les noms que nous avons attribué à nos volumes : **mariadb_data_nabil_rayan** pour le volume associé à notre backEnd qui est la base de donnée MariaDB, et **prestashop_data_nabil_rayan** pour le volume associé à Prestashop.

```
C:\Users\nabil>docker volume create mariadb_data_nabil_rayan
mariadb_data_nabil_rayan

C:\Users\nabil>docker volume create prestashop_data_nabil_rayan
prestashop_data_nabil_rayan
```

3. Une fois notre réseau et nos volumes créés, nous allons maintenant créer 2 containers qui vont représenter le backEnd et le frontEnd de notre application, voici les noms que nous avons donné aux deux containers :

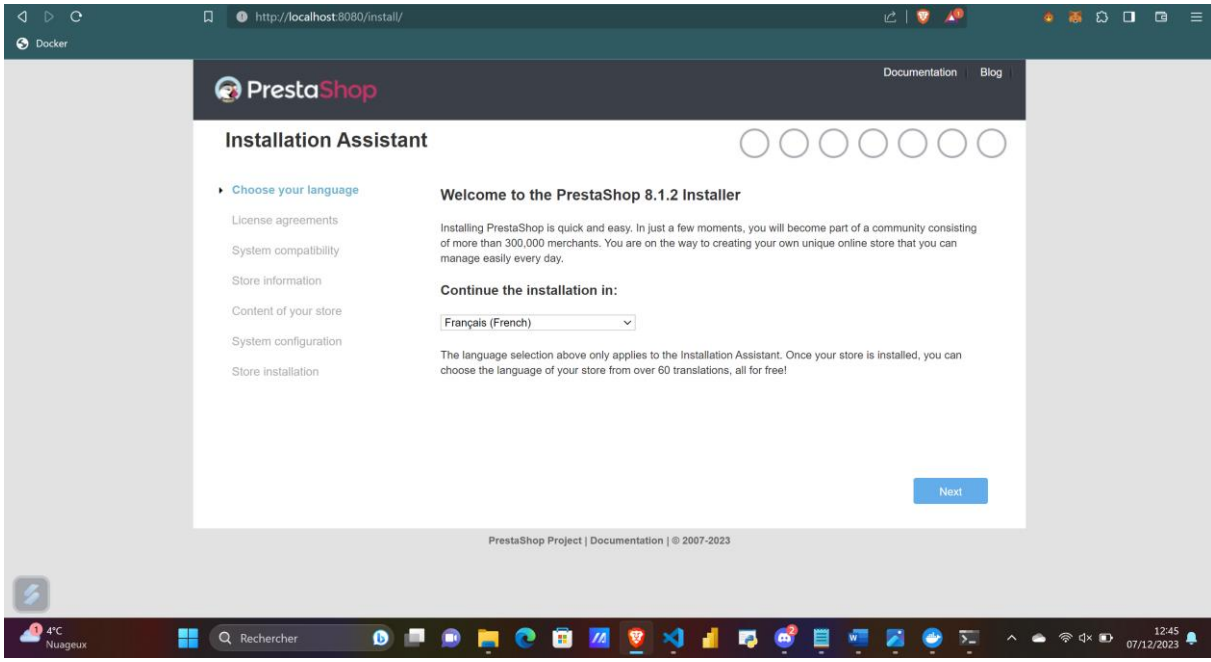
```
C:\Users\nabil>docker run -d --name mariadb_nabil_rayan --network mon_reseau_nabil_rayan -e MYSQL_ROOT_PASSWORD=0000 -e MYSQL_DATABASE=prest
ashop_db -e MYSQL_USER=prestashop_user -e MYSQL_PASSWORD=0000 -v mariadb_data_nabil_rayan:/var/lib/mysql mariadb
Unable to find image 'mariadb:latest' locally
latest: Pulling from library/mariadb
cbe3537751ce: Pull complete
5bfcd11f8751: Pull complete
ed018e89b8db: Pull complete
3e4cf40a46f9: Pull complete
938b1b815dca: Pull complete
07e09e75520d: Pull complete
82012f0ef36f: Pull complete
6430910462f4: Pull complete
Digest: sha256:c006c05608604cf21c9f5b13af3ba7d6ccb3df5bc042c3fe294e0b6d34689b55
Status: Downloaded newer image for mariadb:latest
268087caec71bca23d543d24286f5278645bc7e5241079bf780a5d856395d17f
```

```
C:\Users\nabil>docker run -d --name prestashop_nabil_rayan --network mon_reseau_nabil_rayan -e DB_SERVER=mariadb_nabil_rayan -e DB_NAME=pres
tashop_db -e DB_USER=prestashop_user -e DB_PASSWD=0000 -p 8080:80 -v prestashop_data_nabil_rayan:/var/www/html prestashop/prestashop
Unable to find image 'prestashop/prestashop:latest' locally
latest: Pulling from prestashop/prestashop
a803e7c4b030: Pull complete
84313b8f4350: Pull complete
94f42c54df3f: Pull complete
fac86b32c028: Pull complete
e326747c51cb: Pull complete
2241eb3f28be: Pull complete
82dc7266c2a7: Pull complete
03466a4f1518: Pull complete
cd65d11b092a: Pull complete
cea53147c430: Pull complete
d70222ea3c2a: Pull complete
573cf5bb7fa1: Pull complete
05a21f22b3ef: Pull complete
69b6682cef76: Pull complete
df6afbf38cd6: Pull complete
648b56fab1e7: Pull complete
6a477b6e6c67: Pull complete
85b3e6fb5238: Pull complete
56d09d835ca3: Pull complete
ef2451c0f597: Pull complete
2b3cb1cd78db: Pull complete
f7e55e482859: Pull complete
9095eba13757: Pull complete
b11a645ec861: Pull complete
b9a0496e02be: Pull complete
9f8132ecfa88: Pull complete
Digest: sha256:e7321ad71ad001c10942db594947d2b5d5f287a4fff6287a62dc53509f42c4de
Status: Downloaded newer image for prestashop/prestashop:latest
8c746ab83c23c95215da911488cfb5cf7b2d225b90eb5282432bba6cb3aa03b1
```

Explication des différents paramètres présents dans notre ligne de code :

a. **-d**: Cette option signifie "detached", ce qui permet au conteneur de s'exécuter en arrière-plan.
b. **–name :** permet de donner un nom au container
c. **–network :** permet de spécifier le réseau dans lequel les containers seront connectés
d. **-e :** permet d'instancier des variables d'environnement, qui sont les suivantes :
    i. **MYSQL_ROOT_PASSWORD :** spécifie le mot de passe root de MariaDB
    ii. **MYSQL_DATABASE :** cela crée une base de données nommée "prestashop_db" dans MariaDB
    iii. **MYSQL_USER :** Cela crée un utilisateur dans MariaDB avec le nom "prestashop_user"
e. **Mariadb :** C'est le nom de l'image Docker à utiliser pour créer le conteneur. Dans ce cas, il utilise l'image officielle MariaDB provenant du Docker Hub.

4. Maintenant que toute notre architecture est créé nous allons procéder aux tests pour confirmer que ça fonctionne et que les conteneurs communiquent entre eux dans notre réseau, pour cela on procède en 2 étapes :
    a. **Etape 1** : afficher notre frontEnd sur le port que nous avons spécifié plus haut : 8080, ça fonctionne bien et voici une capture du résultat :



    b. **Etape 2** : tester avec le ping sur l'invite de commandes, à noter que nous avons du installer ping dans nos containers pour pouvoir l'utiliser, nous avons utilisé ces 2 lignes de commandes pour installer ping et l'utiliser par la suite :

```
C:\Users\nabil>docker exec -it prestashop_nabil_rayan apt-get update
Hit:1 http://deb.debian.org/debian bookworm InRelease
Hit:2 http://deb.debian.org/debian bookworm-updates InRelease
Hit:3 http://deb.debian.org/debian-security bookworm-security InRelease
Reading package lists... Done

C:\Users\nabil>docker exec -it prestashop_nabil_rayan apt-get install -y iputils-ping
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Nous voyons que le container appelé répond, ce qui confirme que nos 2 containers communiquent bien entre eux :

```
C:\Users\nabil>docker exec -it prestashop_nabil_rayan bash
root@8c746ab83c23:/var/www/html# ping mariadb_nabil_rayan
PING mariadb_nabil_rayan (172.18.0.2) 56(84) bytes of data.
64 bytes from mariadb_nabil_rayan.mon_reseau_nabil_rayan (172.18.0.2): icmp_seq=1 ttl=64 time=3.37 ms
64 bytes from mariadb_nabil_rayan.mon_reseau_nabil_rayan (172.18.0.2): icmp_seq=2 ttl=64 time=0.118 ms
64 bytes from mariadb_nabil_rayan.mon_reseau_nabil_rayan (172.18.0.2): icmp_seq=3 ttl=64 time=0.140 ms
64 bytes from mariadb_nabil_rayan.mon_reseau_nabil_rayan (172.18.0.2): icmp_seq=4 ttl=64 time=0.138 ms
64 bytes from mariadb_nabil_rayan.mon_reseau_nabil_rayan (172.18.0.2): icmp_seq=5 ttl=64 time=0.141 ms
64 bytes from mariadb_nabil_rayan.mon_reseau_nabil_rayan (172.18.0.2): icmp_seq=6 ttl=64 time=0.140 ms
^C
```

Task2 :

1. Sur la capture d'écran ci-dessous, nous avons combiné 3 étapes :
    a. Création du premier network en lui donnant un subnet (entre Prestashop (frontEnd) et Nginx)
    b. Création du deuxième network en lui donnant un subnet (entre mariadb (backEnd) et Nginx)
    c. Enfin nous avons créé le container Nginx qui prendra le rôle de routeur, pour permettre à nos 2 réseaux de communiquer entre eux

```
rayanus@MacBook-Pro-de-rayan tpYnov % docker network create --subnet=10.0.0.0/24 ynov-frontend-network

200a26fbc54fceea2cd8ab28598ed889ca57fb7324bb24153edd9e8a11a82582
rayanus@MacBook-Pro-de-rayan tpYnov % docker network create --subnet=10.0.1.0/24 ynov-backend-network

f3f9c8f2db47a94dbb8c96581949413c1f40999a21b86d38421b4a6ad135c105
rayanus@MacBook-Pro-de-rayan tpYnov % docker run -d --name nginx-router --network ynov-frontend-network nginx

12ccbbdb9199220e666aba52bf6f88ec50bd95c1804d5a16b6ff08d1c39d620b
```

```
rayanus@MacBook-Pro-de-rayan tpYnov % docker network connect ynov-backend-network nginx-router
```

2. Ensuite on a utilisé « ip route add » pour définir comment le traffic doit être acheminé d'une destination à l'autre en créant une table de routage

```
rayanus@MacBook-Pro-de-rayan tpYnov % docker exec nginx-router ip route add 10.0.0.0/24 via 10.0.0.1

RTNETLINK answers: File exists
rayanus@MacBook-Pro-de-rayan tpYnov % docker exec nginx-router ip route add 10.0.1.0/24 via 10.0.1.1

RTNETLINK answers: File exists
```

3. Ci-dessous les information pour nos deux networks qu'on obtient :

```
rayanus@MacBook-Pro-de-rayan tpYnov % docker network inspect ynov-frontend-network

[
    {
        "Name": "ynov-frontend-network",
        "Id": "200a26fbc54fceea2cd8ab28598ed889ca57fb7324bb24153edd9e8a11a82582",
        "Created": "2023-12-07T15:05:22.853595605Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "10.0.0.0/24"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "6fb57ad48506895cd09067c3b84a67cc0eee2abfccfcd558b634f4ced0179450": {
                "Name": "nginx-router",
                "EndpointID": "3ace5d6e7aac5240948ed3daf87118815a691b059d05edd1acd097b35b4535bb",
                "MacAddress": "02:42:0a:00:00:02",
                "IPv4Address": "10.0.0.2/24",
                "IPv6Address": ""
            },
            "bbb8ef9a2ad76ddf6eeb35c436f94c8bb989a9021e328f6f1549d78fbc6939db": {
                "Name": "prestashop_nabil_rayan",
                "EndpointID": "08ee9796da076d47676ebb15e0eadd6a1a0d0282ae158d389e8b1cd4b649fecc",
                "MacAddress": "02:42:0a:00:00:03",
                "IPv4Address": "10.0.0.3/24",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
```

```
rayanus@MacBook-Pro-de-rayan tpYnov % docker network inspect ynov-backend-network

[
    {
        "Name": "ynov-backend-network",
        "Id": "f3f9c8f2db47a94dbb8c96581949413c1f40999a21b86d38421b4a6ad135c105",
        "Created": "2023-12-07T15:05:49.374895593Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "10.0.1.0/24"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "3f4034f7916454fe297b6c6a84eb58f2a8f33cad2fe1eff9ea0b16fbc6003276": {
                "Name": "mariadb_nabil_rayan",
                "EndpointID": "b252f043276a34898e5cbd4d681e16379e43db3eb466dcbfcf5ef6ecfd2358bc",
                "MacAddress": "02:42:0a:00:01:03",
                "IPv4Address": "10.0.1.3/24",
                "IPv6Address": ""
            },
            "6fb57ad48506895cd09067c3b84a67cc0eee2abfccfcd558b634f4ced0179450": {
                "Name": "nginx-router",
                "EndpointID": "0a272a1013dae6c297596c83fa122dd31f6707ce5998c18fc4825c55ab1f429e",
                "MacAddress": "02:42:0a:00:01:02",
                "IPv4Address": "10.0.1.2/24",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
```

4. La table de routage de Nginx est bien paramétrée, comme on peut le voir ci-dessous :

```
rayanus@MacBook-Pro-de-rayan tpYnov %  docker exec nginx-router ip route
default via 10.0.1.1 dev eth1
10.0.0.0/24 dev eth0 proto kernel scope link src 10.0.0.2
10.0.1.0/24 dev eth1 proto kernel scope link src 10.0.1.2
```

5. On peut tester dans le container DB que si l'on essaye de faire un trace route jusqu'à l'adresse ip du container PrestaShop il arrive à se connecter +

```
traceroute to 10.0.0.2 (10.0.0.2), 30 hops max, 60 byte packets
 1  10.0.1.1 (10.0.1.1)  0.681 ms  0.029 ms  0.013 ms
 2  * * *
 3  * * *
 4  * * *
```