

## TP Docker

- 1) Création du premier conteneur nommé containerbdd pour stocké la base de données avec l'image de postgres, création du deuxième conteneur nommé prestashop-container avec l'image de prestashop/prestashop puis lister avec « docker ps » la liste des conteneurs qui sont démarrés.

```
wassf@LAPTOP-4E65GICU MINGW64 ~
$ docker run -d --name containerbdd --network prestashop-network -e POSTGRES_USER=admin -e POSTGRES_PASSWORD=pass -e POSTGRES_DB=dbtp -e ALLOW_EMPTY_PASSWORD=yes -v tp_volume:/var/lib/postgresql/data -p 5433:5432 -d postgres
25a7b1bd772232144cee3078b94989f108532a7c3ac6deac19e9be0fe7e9bf3

wassf@LAPTOP-4E65GICU MINGW64 ~
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED        STATUS        PORTS                    NAMES
25a7b1bd7722        postgres           "docker-entrypoint.s..." 3 seconds ago  Up 3 seconds  0.0.0.0:5433->5432/tcp   containerbdd
bfba24d693a6        prestashop/prestashop "docker-php-entrypoi..." 5 minutes ago  Up 5 minutes   80/tcp                  prestashop-container

wassf@LAPTOP-4E65GICU MINGW64 ~
$ docker run --name prestashop-container --network prestashop-network -e PRESTASHOP_DATABASE_PASSWORD= -e ALLOW_EMPTY_PASSWORD=yes -d prestashop/prestashop
bfba24d693a6ce7c37786601d55d1ca6c411932699970f769b32e18af6e65457

wassf@LAPTOP-4E65GICU MINGW64 ~
$ docker run --name prestashop-container --network prestashop-network -e PRESTASHOP_DATABASE_PASSWORD= -e ALLOW_EMPTY_PASSWORD=yes -d prestashop/prestashop
Unable to find image 'prestashop/prestashop:latest' locally
latest: Pulling from prestashop/prestashop
a803e7c4b030: Pulling fs layer
84313b8f4350: Pulling fs layer
94f42c54df3f: Pulling fs layer
fac86b32c028: Pulling fs layer
e326747c51cb: Pulling fs layer
2241eb3f28be: Pulling fs layer
82dc7266c2a7: Pulling fs layer
0346eaf1518: Pulling fs layer
cd65d1b092a: Pulling fs layer
cea53147c430: Pulling fs layer
d70222ea3c2a: Pulling fs layer
573cf5bb7fa1: Pulling fs layer
05a21f2b3ef: Pulling fs layer
fac86b32c028: Waiting
69b682cef76: Pulling fs layer
e326747c51cb: Waiting
2241eb3f28be: Waiting
82dc7266c2a7: Waiting
df6afb38cd6: Pulling fs layer
648b56fable7: Pulling fs layer
6a477b6e6c67: Pulling fs layer
```

- 2) On rentre dans le conteneur crée :

```
wassf@LAPTOP-4E65GICU MINGW64 ~
$ winpty docker exec -it prestashop-container bash
I have no name!@bd047acb1ad9:/$ exit
exit
```

- 3) On crée désormais les sous-réseaux backend et frontend :

```
wassf@LAPTOP-4E65GICU MINGW64 ~
$ docker network create --subnet 10.0.0.0/24 ynov-backend-network
95237d3cf755be83ecf6a81378a1eb38e9bc9d4c1588697eb1b5db7fd5e67623

wassf@LAPTOP-4E65GICU MINGW64 ~
$ docker network create --subnet 10.0.1.0/24 ynov-frontend-network
60aea6ec48f9395b61e4d476e4f5735dde2551654c44d118f9bcdd98f29ea715
```

- 4) On crée le conteneur gateway qui va servir de routeur entre les 2 autres conteneurs puis on va connecter le conteneur gateway-container dans les 2 réseaux backend et frontend. Puis on liste pour voir les conteneurs démarré.

```
wassf@LAPTOP-4E65GICU MINGW64 ~  
$ docker run -d --name gateway-container nginx:latest  
65cf95d84fd5b6ba760ad880b585c9e251abb53d8a6bc1f059c259dda005df43  
wassf@LAPTOP-4E65GICU MINGW64 ~  
$ docker network connect ynov-frontend-network gateway-container  
wassf@LAPTOP-4E65GICU MINGW64 ~  
$ docker network connect ynov-backend-network gateway-container  
wassf@LAPTOP-4E65GICU MINGW64 ~  
$ docker ps  
CONTAINER ID        IMAGE               COMMAND                  CREATED            STATUS              PORTS                               NAMES  
65cf95d84fd5        nginx:latest        "/docker-entrypoint..." 22 seconds ago    Up 21 seconds      80/tcp                             gateway-container  
153a71dac81c        bitnami/prestashop "/opt/bitnami/script..." 12 minutes ago    Up 43 seconds      8080/tcp, 8443/tcp                prestashop-container  
db3201ac08f3        postgres            "docker-entrypoint.s..." 12 minutes ago    Up 12 minutes      5432/tcp                          containerbdd
```

- 5) On inspecte désormais le réseau backend et on vérifie bien que le conteneur gateway-container et le containerbdd sont bien dans le réseau backend :

```
wassf@LAPTOP-4E65GICU MINGW64 ~  
$ docker inspect ynov-backend-network  
[  
  {  
    "Name": "ynov-backend-network",  
    "Id": "95237d3cf755be83ecf6a81378a1eb38e9bc9d4c1588697eb1b5db7fd5e67623",  
    "Created": "2023-12-07T10:27:52.062861208Z",  
    "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": {  
      "65cf95d84fd5b6ba760ad880b585c9e251abb53d8a6bc1f059c259dda005df43": {  
        "Name": "gateway-container",  
        "EndpointID": "57204a357749df84aec39fbd37b119ad08578444764ec27e91a6c3607deb62fb",  
        "MacAddress": "02:42:0a:00:00:02",  
        "IPv4Address": "10.0.0.2/24",  
        "IPv6Address": ""  
      },  
      "db3201ac08f3302a1e1e32cc95a569df5ad0b1797c24fb5b41e31c610010cf1f": {  
        "Name": "containerbdd",  
        "EndpointID": "065a7fb5d079ca6c03e07c99ccad2e7e6164911e3b7a960677811b3363d5c475",  
        "MacAddress": "02:42:0a:00:00:03",  
        "IPv4Address": "10.0.0.3/24",  
        "IPv6Address": ""  
      }  
    },  
    "Options": {},  
    "Labels": {}  
  }  
]
```

- 6) On inspecte désormais le réseau frontend et on vérifie que le conteneur gateway-container et le prestashop-container sont bien dans le réseau frontend :

```
wassf@LAPTOP-4E65GICU MINGW64 ~  
$ docker inspect ynov-frontend-network  
[  
  {  
    "Name": "ynov-frontend-network",  
    "Id": "60aea6ec48f9395b61e4d476e4f5735dde2551654c44d118f9bcd98f29ea715",  
    "Created": "2023-12-07T10:28:59.653198285Z",  
    "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": {  
      "153a71dac81c287f7998961ff557570536910c5ee3bc1fd4e1533d12a05ed58a": {  
        "Name": "prestashop-container",  
        "EndpointID": "a7d6ffc2021809c77e3415b024f18f32b3f2164d564b7562fad836350b2ab9a2",  
        "MacAddress": "02:42:0a:00:01:03",  
        "IPv4Address": "10.0.1.3/24",  
        "IPv6Address": ""  
      },  
      "cd9f886078c0a9926365cc9286cd6c109f1a9e22fc4e951af76981d1cb045613": {  
        "Name": "gateway-container",  
        "EndpointID": "47efeb778a817496fa32776bb4742d69995a983a398798b111c61e46e558ea30",  
        "MacAddress": "02:42:0a:00:01:02",  
        "IPv4Address": "10.0.1.2/24",  
        "IPv6Address": ""  
      }  
    },  
    "Options": {},  
    "Labels": {}  
  }  
]
```

- 7) Pour voir tous les réseaux créés :

```
wassf@LAPTOP-4E65GICU MINGW64 ~  
$ docker network ls  
NETWORK ID          NAME                DRIVER              SCOPE  
ef7b2ce45db3        bridge             bridge              local  
a884f11e803a        frontend           bridge              local  
c250ef5a32c4        host               host                local  
dc67ca20c221        none               null                local  
088b0593fe74        prestanet          bridge              local  
ab84480e5988        prestashop-network bridge              local  
bdc42636f45e        reseautp           bridge              local  
95237d3cf755        ynov-backend-network bridge              local  
60aea6ec48f9        ynov-frontend-network bridge              local
```

- 8) On rentre dans le conteneur gateway qui sert de routeur :

```
wassf@LAPTOP-4E65GICU MINGW64 ~  
$ winpty docker exec -it gateway-container bash  
root@cd9f886078c0:/# apt-get update && apt-get upgrade
```

On installe dans les 2 conteneurs iputils-ping, ici dans containerbdd :

```
wassf@LAPTOP-4E65GICU MINGW64 ~  
$ winpty docker exec -it containerbdd bash  
root@25a7b1bd7722:/# apt-get install iputils-ping  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done
```

9) On fait communiquer 2 conteneurs dans un même réseau avec un PING (task 1)

```
wassif@LAPTOP-4E65GICU MINGW64 ~  
$ winpty docker exec -it containerbdd ping prestashop-container  
PING prestashop-container (172.19.0.2) 56(84) bytes of data:  
64 bytes from prestashop-container.prestashop-network (172.19.0.2): icmp_seq=1 ttl=64 time=0.729 ms  
64 bytes from prestashop-container.prestashop-network (172.19.0.2): icmp_seq=2 ttl=64 time=0.209 ms  
64 bytes from prestashop-container.prestashop-network (172.19.0.2): icmp_seq=3 ttl=64 time=0.137 ms  
64 bytes from prestashop-container.prestashop-network (172.19.0.2): icmp_seq=4 ttl=64 time=0.063 ms  
64 bytes from prestashop-container.prestashop-network (172.19.0.2): icmp_seq=5 ttl=64 time=0.106 ms  
64 bytes from prestashop-container.prestashop-network (172.19.0.2): icmp_seq=6 ttl=64 time=0.099 ms  
^C  
-- prestashop-container ping statistics --  
6 packets transmitted, 6 received, 0% packet loss, time 5070ms  
rtt min/avg/max/mdev = 0.063/0.223/0.729/0.230 ms
```

On vérifie que les 2 conteneurs sont bien dans le même réseau :

```
wassif@LAPTOP-4E65GICU MINGW64 ~  
$ docker inspect prestashop-network  
[  
  {  
    "Name": "prestashop-network",  
    "Id": "ab84480e59883ae928611f7795aebfbb5f4b86c0b83aa464194425131259764b",  
    "Created": "2023-12-07T09:38:35.572728436Z",  
    "Scope": "local",  
    "Driver": "bridge",  
    "EnableIPv6": false,  
    "IPAM": {  
      "Driver": "default",  
      "Options": {},  
      "Config": [  
        {  
          "Subnet": "172.19.0.0/16",  
          "Gateway": "172.19.0.1"  
        }  
      ]  
    },  
    "Internal": false,  
    "Attachable": false,  
    "Ingress": false,  
    "ConfigFrom": {  
      "Network": ""  
    },  
    "ConfigOnly": false,  
    "Containers": {  
      "25a7b1bd7722332144cee3078b94989f108532a7c5ac6deac19e9be0fe7e9bf3": {  
        "Name": "containerbdd",  
        "EndpointID": "b7335f612bcd3385309ed71a0d959bb5089edc57e44d117057bc43116d854f1",  
        "MacAddress": "02:42:ac:13:00:03",  
        "IPv4Address": "172.19.0.3/16",  
        "IPv6Address": ""  
      },  
      "bfba24d693a6ce7c37786601d55d1ca6c411932699970f769b32e18af6e65457": {  
        "Name": "prestashop-container",  
        "EndpointID": "627dcdedd38f12da9977cd6f19bc1195f9295b9464d72938fba3d47c3c7badb",  
        "MacAddress": "02:42:ac:13:00:02",  
        "IPv4Address": "172.19.0.2/16",  
        "IPv6Address": ""  
      }  
    },  
    "Options": {},  
    "Labels": {}  
  }  
]
```

- 10) On installe dans un premier temps postgres sql client dans le container prestashop-container puis on lance la commande pour nous connecter au conteneur de la base de données ici containerbdd dans le conteneur prestashop-container. (task 1)

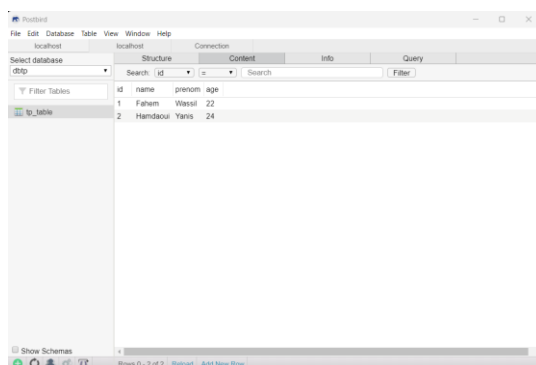
```
mass@BLAPTOP-4E65GICU MINGW64 ~
$ winpty docker exec -it prestashop-container bash
root@bfa24d693a6:/var/www/html# apt-get install postgresql-client
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libpq5 netbase postgresql-client-15 postgresql-client-common sensible-utils
Suggested packages:
  postgresql-15 postgresql-doc-15
The following NEW packages will be installed:
  libpq5 netbase postgresql-client-15 postgresql-client-common sensible-utils
0 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
Need to get 1961 kB of archives.
After this operation, 9208 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://deb.debian.org/debian bookworm/main amd64 netbase all 6.4 [12.8 kB]
Get:2 http://deb.debian.org/debian bookworm/main amd64 sensible-utils all 0.0.17+nmu1 [19.0 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security/main amd64 libpq5 amd64 15.5-0+deb12u1 [187 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 postgresql-client-common all 248 [35.1 kB]
Get:5 http://deb.debian.org/debian-security bookworm-security/main amd64 postgresql-client-15 amd64 15.5-0+deb12u1 [1697 kB]
Get:6 http://deb.debian.org/debian bookworm/main amd64 postgresql-client all 15+248 [10.1 kB]
Fetched 1961 kB in 2s (972 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package netbase.
(Reading database ... 15627 files and directories currently installed.)
Preparing to unpack .../0-netbase_6.4_all.deb ...
base: psql: command not found
root@bfa24d693a6:/var/www/html# psql -h containerbdd -U admin -d dbtp -w
psql (15.5 (Debian 15.5-0+deb12u1), server 16.1 (Debian 16.1-1.pgdg120+1))
WARNING: psql major version 15, server major version 16.
Some psql features might not work.
Type "help" for help.

dbtp=# INSERT INTO exotable3 (nom, prenom, age) VALUES ('Fahem','Wassil', 22);
ERROR: relation "exotable3" does not exist
LINE 1: INSERT INTO exotable3 (nom, prenom, age) VALUES ('Fahem','Wa...
                    ^
dbtp=# CREATE TABLE example_table (
dbtp=#         id SERIAL PRIMARY KEY,
dbtp=#         name VARCHAR(50),
dbtp=#         age INT
dbtp=# AC
dbtp=# CREATE TABLE tp_table (id SERIAL PRIMARY KEY, name VARCHAR (50), Prenom VARCHAR (50), age INT);
CREATE TABLE
dbtp=# INSERT INTO tp_table (nom, prenom, age) VALUES ('Fahem','Wassil', 22);
ERROR: column "nom" of relation "tp_table" does not exist
LINE 1: INSERT INTO tp_table (nom, prenom, age) VALUES ('Fahem','Was...
                    ^
dbtp=# INSERT INTO tp_table (name, prenom, age) VALUES ('Fahem','Wassil', 22);
INSERT 0 1
dbtp=# select * from tp_table;
 id | name | prenom | age
----+-----+-----+----
  1 | Fahem | Wassil | 22
(1 row)

dbtp=# INSERT INTO tp_table (name, prenom, age) VALUES ('Hamdaoui','Yanis', 24);
INSERT 0 1
dbtp=# select * from tp_table;
 id | name | prenom | age
----+-----+-----+----
  1 | Fahem | Wassil | 22
  2 | Hamdaoui | Yanis | 24
(2 rows)

dbtp=#
```

- 11) On peut vérifier sur Postbird en y accédant via les données que l'on a renseigné lors de la création du conteneur, qu'on a créée la table ci-dessus et les données ci-dessus : (Task 1)



The screenshot shows the Postbird application interface. On the left, a sidebar lists the database 'dbtp' and its tables 'tp\_table' and 'example\_table'. The main window displays the 'Structure' tab for 'tp\_table', showing columns: id (SERIAL), name (VARCHAR(50)), prenom (VARCHAR(50)), and age (INT). Below this, the 'Content' tab shows the data rows:

id	name	prenom	age
1	Fahem	Wassil	22
2	Hamdaoui	Yanis	24

At the bottom, it indicates 'Rows 0 - 2 of 2' and provides options to 'Refresh' or 'Add New Row'.

12) On ping depuis le gateway vers prestashop-contenair.

```
root@fdb1873659dd:/# ping 10.0.1.3
PING 10.0.1.3 (10.0.1.3) 56(84) bytes of data.
64 bytes from 10.0.1.3: icmp_seq=1 ttl=64 time=1.98 ms
64 bytes from 10.0.1.3: icmp_seq=2 ttl=64 time=0.058 ms
64 bytes from 10.0.1.3: icmp_seq=3 ttl=64 time=0.098 ms
64 bytes from 10.0.1.3: icmp_seq=4 ttl=64 time=0.187 ms
64 bytes from 10.0.1.3: icmp_seq=5 ttl=64 time=0.105 ms
64 bytes from 10.0.1.3: icmp_seq=6 ttl=64 time=0.112 ms
64 bytes from 10.0.1.3: icmp_seq=7 ttl=64 time=0.091 ms
64 bytes from 10.0.1.3: icmp_seq=8 ttl=64 time=0.079 ms
64 bytes from 10.0.1.3: icmp_seq=9 ttl=64 time=0.080 ms
^C
--- 10.0.1.3 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8248ms
rtt min/avg/max/mdev = 0.058/0.309/1.976/0.590 ms
root@fdb1873659dd:/# exit
exit
```