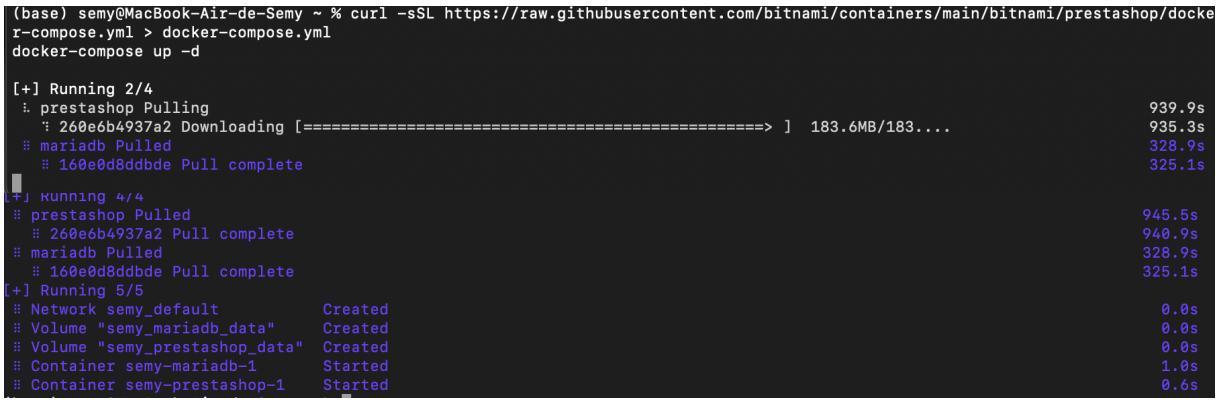
Configuration de réseaux Docker pour l'application PrestaShop
Membres de l'équipe :

- [Semy BOUACID]
- [Yoni MICHARD]

Ce document fournit un guide étape par étape pour configurer les réseaux Docker pour une application PrestaShop, en créant des sous-réseaux dédiés pour différents composants et en assurant la communication entre les conteneurs à l'aide de noms et d'adresses IP.

Prérequis : Installer l'image de Prestashop

- 

Tâche 1 : Déployer l'application dans un réseau

1. **Créer un réseau Docker - Réseau PrestaShop :**

Commande:

- docker network create prestashop-network

```
(base) semy@MacBook-Air-de-Semy ~ % docker network create prestashop-network

aca05a54b7bbe7431261494488b1255042e8256aeb846a6c8ca9ef9b8c86db9c
```

2. **Exécuter les conteneurs Frontend et Database dans le réseau :**

Commandes:

- docker run -d --name conteneur-bd --network prestashop-network -e POSTGRES_PASSWORD=admin postgres

- docker run --name conteneur-prestashop --network prestashop-network -e PRESTASHOP_DATABASE_PASSWORD=admin -d bitnami/prestashop

```
(base) semy@MacBook-Air-de-Semy ~ % docker run -d --name conteneur-bd --network prestashop-network -e POSTGRES_PASSWORD=admin postgres
96ec72bce20320efb336532c9b48b3813847651672cc3406b2901bfae83c8f47
(base) semy@MacBook-Air-de-Semy ~ % docker run --name conteneur-prestashop --network prestashop-network -e PRESTASHOP_DATABASE_PASSWORD=admin -d bitnami/presta
shop
f3608a4525dce405d8557a719ee806183c097880f2f66f83e1cfa36f69ae9ac2
(base) semy@MacBook-Air-de-Semy ~ % docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS          PORTS                                                NAMES
f3608a4525dc   bitnami/prestashop    "/opt/bitnami/script…"   4 seconds ago    Up 4 seconds    8080/tcp, 8443/tcp                                   conteneur-prestas
hop
96ec72bce203   postgres              "docker-entrypoint.s…"   16 seconds ago   Up 15 seconds   5432/tcp                                             conteneur-bd
9f9e929118b3   bitnami/prestashop:8  "/opt/bitnami/script…"   33 minutes ago   Up 33 minutes   0.0.0.0:80->8080/tcp, 0.0.0.0:443->8443/tcp          semy-prestashop-1
e620c8947443   bitnami/mariadb:11.1  "/opt/bitnami/script…"   33 minutes ago   Up 33 minutes   3306/tcp                                             semy-mariadb-1
(base) semy@MacBook-Air-de-Semy ~ %
```

Tâche 2 : Créer des réseaux avec des sous-réseaux et configurer le routage

1. **Créer des réseaux avec des CIDR - Frontend et Backend :**

- docker network create --subnet=192.168.1.0/24 eval-frontend-network

- docker network create --subnet=192.168.2.0/24 eval-backend-network

```
(base) semy@MacBook-Air-de-Semy ~ % docker network create --subnet=192.168.1.0/24 eval-frontend-network

c112a04f8031f58a93956c2df97b6be690e58a6cff36b4aa0244826de8864440
(base) semy@MacBook-Air-de-Semy ~ % docker network create --subnet=192.168.2.0/24 eval-backend-network

1b833720b8d55fd1bd979fef3f23d9285a0605fb87335c6d68692c52e5f0e342
(base) semy@MacBook-Air-de-Semy ~ % docker network ls
NETWORK ID     NAME                    DRIVER    SCOPE
7a632b813a11   bridge                  bridge    local
1b833720b8d5   eval-backend-network    bridge    local
c112a04f8031   eval-frontend-network   bridge    local
77993d481519   host                    host      local
30f902cb1ab1   none                    null      local
aca05a54b7bb   prestashop-network      bridge    local
56ee2f9564f6   semy_default            bridge    local
```

2. **Créer un conteneur passerelle/router :**

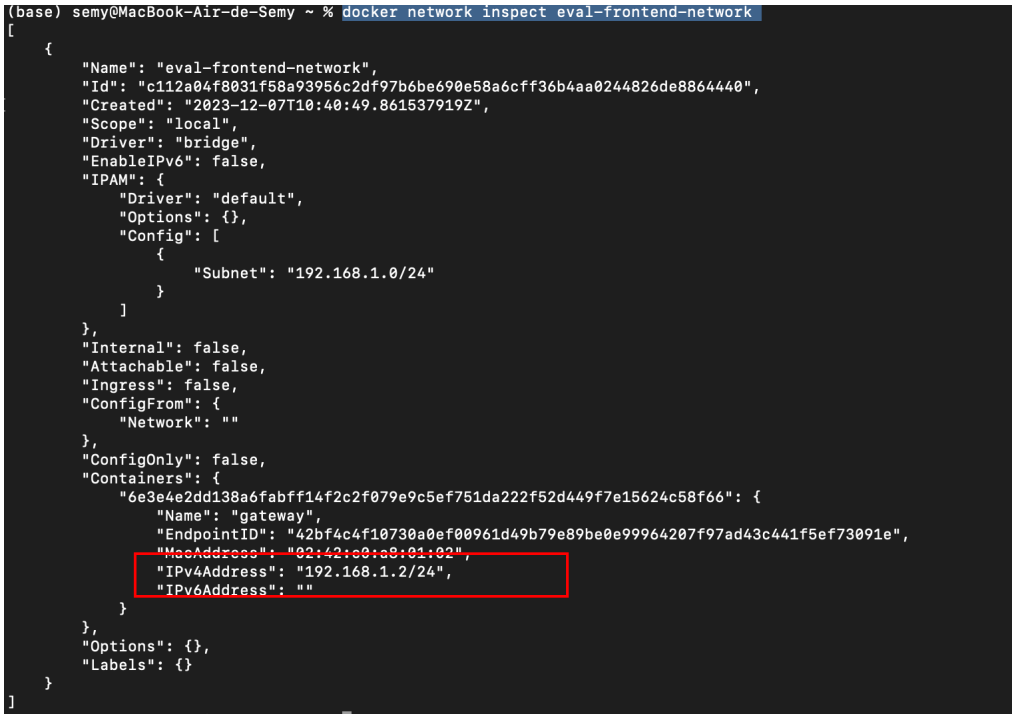
- docker run -d --name gateway --network eval-frontend-network nginx
- docker network connect eval-backend-network gateway

```
(base) semy@MacBook-Air-de-Semy ~ % docker run -d --name gateway --network eval-frontend-network nginx
6e3e4e2dd138a6fabff14f2c2f079e9c5ef751da222f52d449f7e15624c58f66
(base) semy@MacBook-Air-de-Semy ~ % docker network connect eval-backend-network gateway
```

3. **Configurer la table de routage à l'intérieur du conteneur passerelle/router :**

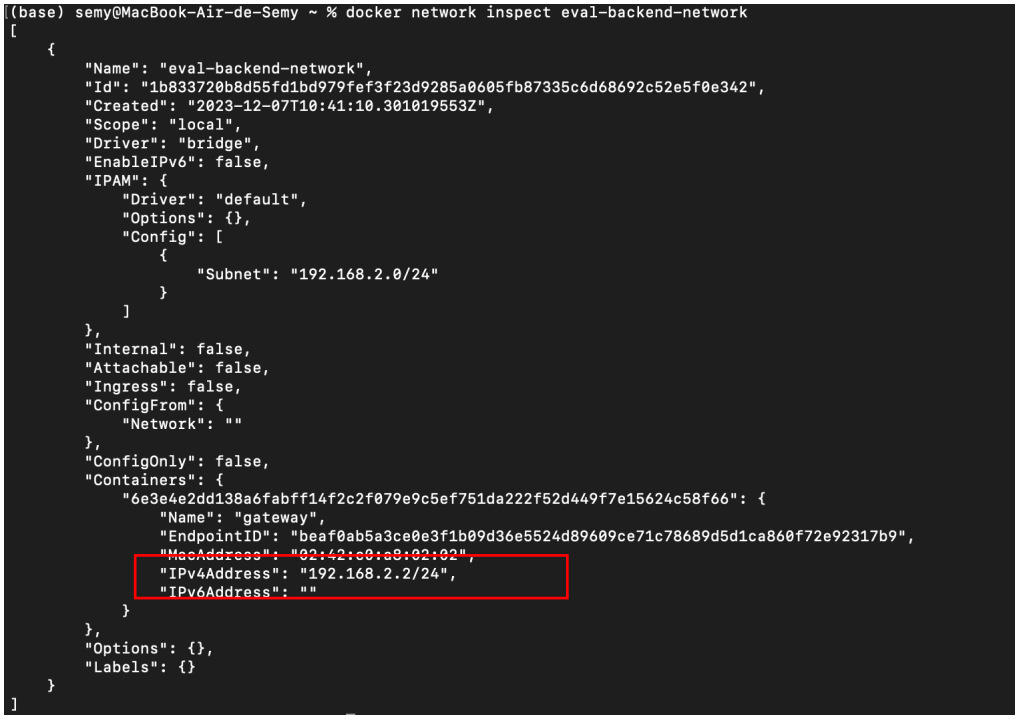
<u>On doit d'abord récupérer la passerelle des deux sous-réseaux crées :</u>

- docker network inspect eval-frontend-network



```
(base) semy@MacBook-Air-de-Semy ~ % docker network inspect eval-frontend-network
[
    {
        "Name": "eval-frontend-network",
        "Id": "c112a04f8031f58a93956c2df97b6be690e58a6cff36b4aa0244826de8864440",
        "Created": "2023-12-07T10:40:49.861537919Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "192.168.1.0/24"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "6e3e4e2dd138a6fabff14f2c2f079e9c5ef751da222f52d449f7e15624c58f66": {
                "Name": "gateway",
                "EndpointID": "42bf4c4f10730a0ef00961d49b79e89be0e99964207f97ad43c441f5ef73091e",
                "MacAddress": "02:42:c0:a8:01:02",
                "IPv4Address": "192.168.1.2/24",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

- docker network inspect eval-backend-network



```
(base) semy@MacBook-Air-de-Semy ~ % docker network inspect eval-backend-network
[
    {
        "Name": "eval-backend-network",
        "Id": "1b833720b8d55fd1bd979fef3f23d9285a0605fb87335c6d68692c52e5f0e342",
        "Created": "2023-12-07T10:41:10.301019553Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "192.168.2.0/24"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "6e3e4e2dd138a6fabff14f2c2f079e9c5ef751da222f52d449f7e15624c58f66": {
                "Name": "gateway",
                "EndpointID": "beaf0ab5a3ce0e3f1b09d36e5524d89609ce71c78689d5d1ca860f72e92317b9",
                "MacAddress": "02:42:c0:a8:02:02",
                "IPv4Address": "192.168.2.2/24",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

On ajout les deux addresses de passerelle aux commandes qui nous permettent de configure les tables de routage :

- docker exec -it gateway /bin/bash ip route 192.168.1.0 via 192.168.1.2 ip route add 192.168.2.0 via 192.168.2.2
- `docker exec ––privileged gateway ip route replace 192.168.1.0 via 192.168.1.2`
- `docker exec ––privileged gateway ip route replace 192.168.2.0 via 192.168.2.2`

Vérifier la connectivité en utilisant les adresses IP :
À l'intérieur du conteneur passerelle/router :

ping <container_IP_eval-frontend-network> ping <container_IP_eval-backend-network>

Objectif :

Le but de cette tâche était de configurer les réseaux Docker pour une application PrestaShop, en créant des réseaux distincts avec des sous-réseaux dédiés pour les composants frontend et backend, et en établissant une communication entre ces réseaux à l'aide d'une passerelle/router.
Méthodologie :

1. Déploiement de l'Application dans un Réseau Unique :
    - J'ai créé un réseau Docker nommé prestashop-network.
    - J'ai lancé les conteneurs frontend et database (PrestaShop) en les attachant au réseau prestashop-network.
2. Création de Réseaux avec Subnets Dédiés :
    - J'ai créé deux réseaux distincts, ynov-frontend-network et ynov-backend-network, avec des sous-réseaux spécifiés (192.168.1.0/24 et 192.168.2.0/24 respectivement).
3. Configuration d'une Passerelle/Router et des Routes :
    - J'ai essayé d'utiliser un conteneur nommé gateway comme passerelle/router pour connecter les deux réseaux et configurer les routes pour permettre la communication entre les sous-réseaux.

Problèmes Rencontrés :

- Problème d'exécution des commandes de routage : Malgré plusieurs tentatives, j'ai rencontré des difficultés pour exécuter les commandes de configuration de la table de routage à l'intérieur du conteneur gateway. L'erreur "cannot execute binary file" a été rencontrée, suggérant un problème avec l'exécution des commandes ip.

Étapes de Résolution :

1. Vérification des Outils Disponibles :
    - J'ai tenté d'exécuter d'autres commandes réseau (ifconfig) pour vérifier la disponibilité des outils réseau à l'intérieur du conteneur gateway.
2. Recherche de Solutions Alternatives :
    - J'ai exploré la possibilité d'utiliser une image Docker différente ou de rechercher une image contenant les outils réseau nécessaires pour configurer les routes.

3. Vérification des Autorisations :
    - J'ai vérifié les autorisations pour m'assurer que j'avais les droits nécessaires pour effectuer des modifications réseau à l'intérieur du conteneur.

Conclusion :

Malgré les efforts déployés, la configuration des routes entre les réseaux ynov-frontend-network et ynov-backend-network à l'intérieur du conteneur gateway n'a pas abouti en raison des problèmes rencontrés avec l'exécution des commandes réseau à l'intérieur du conteneur.

Suggestions :

- Explorer des images Docker alternatives avec les outils réseau requis pour configurer les routes.
- Vérifier attentivement les autorisations et les privilèges pour exécuter les commandes réseau à l'intérieur des conteneurs Docker.