# TD 2

## Partie I : Interconnexion de conteneurs

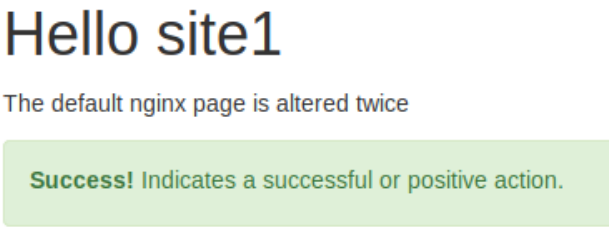
Nous allons ici créer 3 conteneurs qui seront interconnectés en réseaux.

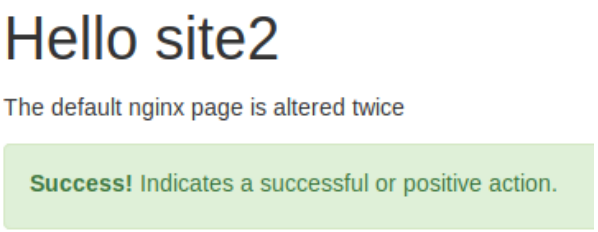
1)

Nous créons un nouveau projet avec la structure suivante :

```
td_docker
├── proxy
│   ├── Dockerfile
│   ├── site1.conf
│   └── site2.conf
├── site1
│   └── index.html
└── site2
    └── index.html
```

On modifie les fichiers index.html de façon à différencier les 2 sites :




2)

On lance un conteneur pour le site1 avec le port 8081 ainsi qu'un conteneur pour le site2 avec le port 8080.

Voici ci-dessous les commandes réalisées pour la création du conteneur du site1 :

```
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker/site1$ sudo docker run -d -p 8081:80 -v"$(pwd)":/usr/share
/nginx/html:ro --name site1 nginx
6c9a024f6c6866eeebeadcf74063adbe33ef94653a6446967699a7f4ce76e539
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker/site2$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                   NAMES
fd9384c1f284        nginx               "nginx -g 'daemon of…"   2 minutes ago       Up 2 minutes        0.0.0.0:8080->80/tcp    site2
6c9a024f6c68        nginx               "nginx -g 'daemon of…"   2 minutes ago       Up 2 minutes        0.0.0.0:8081->80/tcp    site1
```

```
"Mounts": [
    {
        "Type": "bind",
        "Source": "/home/orain/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker/site1",
        "Destination": "/usr/share/nginx/html",
        "Mode": "ro",
        "RW": false,
        "Propagation": "rprivate"
    }
],
```

3)

Pour configurer les fichiers de configuration du proxy et créer le conteneur proxy, il nous faut récupérer les adresses IP de chaque conteneur :

```
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35$ sudo docker inspect --format '{{ .NetworkSettings.IPAddress }}' site1
172.17.0.2
```

```
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35$ sudo docker inspect --format '{{ .NetworkSettings.IPAddress }}' site2
172.17.0.3
```

On peut maintenant configurer les fichiers du proxy

On configure d'abord site1.conf et site2.conf :

```
server {
    listen 8081;
    server_name site1;

    location / {
        proxy_pass http://172.17.0.2;
    }
}
```

```
server {
    listen 8080;
     server_name site2;

    location / {
        proxy_pass http://172.17.0.3;
    }
}
```

Dans dockerfile on va copier les fichiers de conf site1.conf et site2.conf :

```
FROM nginx
COPY site1.conf etc/nginx/conf.d/
COPY site2.conf etc/nginx/conf.d/
```

4)

On crée l'image « imgproxy » correspondant au proxy :

```
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker/proxy$ sudo docker build --tag imgproxy .
```

```
Sending build context to Docker daemon  4.096kB
Step 1/3 : FROM nginx
 ---> dbfc48660aeb
Step 2/3 : COPY site1.conf etc/nginx/conf.d/
 ---> cb39569f2ca3
Step 3/3 : COPY site2.conf etc/nginx/conf.d/
 ---> ef51727f35de
Successfully built ef51727f35de
Successfully tagged imgproxy:latest
```

```
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker/proxy$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
imgproxy            latest              ac8732d19fa9        9 minutes ago       109MB
```

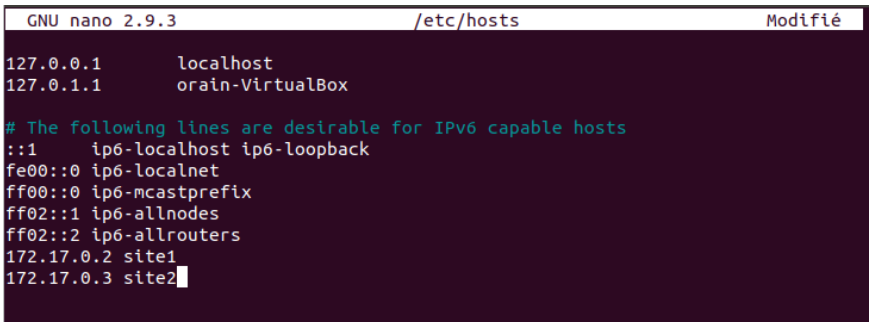
On peut maintenant lancer le conteneur sur le port 80 :

```
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker/proxy$ sudo docker run -d -p 80:80 --name proxy imgproxy
fb0877cb50f860eba3786c4c5d855a92dadf7be9237bcb1b67a8a5715cac1167
```

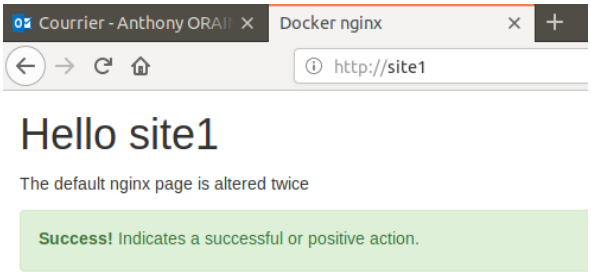
```
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker/proxy$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                   NAMES
627a3267e582        imgproxy            "nginx -g 'daemon of…"   24 seconds ago      Up 22 seconds       0.0.0.0:80->80/tcp      proxy
2ea7e125d22c        nginx               "nginx -g 'daemon of…"   20 minutes ago      Up 20 minutes       0.0.0.0:8080->80/tcp    site2
e67a8e62200b        nginx               "nginx -g 'daemon of…"   21 minutes ago      Up 21 minutes       0.0.0.0:8081->80/tcp    site1
```

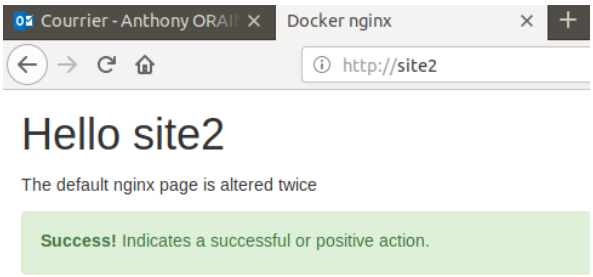
5)

Le proxy fonctionne selon le nom de domaine saisi dans le navigateur. On peut modifier le fichier /etc/hosts dans l'optique d'ajouter la résolution des sites site1 et site2 :




On peut à présent tester dans un navigateur http://site1 et http://site2 :



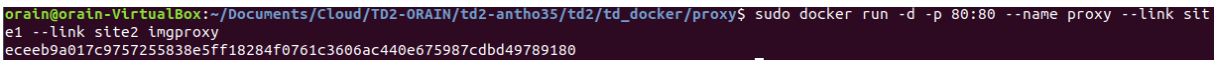

6)

L'inconvénient à cette solution est le suivant : au redémarrage d'un conteneur, il peut changer d'adresse IP. Nous devons alors modifier le fichier de configuration à chaque fois pour que le processus fonctionne.

## Docker link

1)

Une autre possibilité consiste à créer un lien entre les conteneurs. On peut réaliser cela par l'intermédiaire de la commande *–link*. On commence par stopper le conteneur proxy puis on le supprime. On peut maintenant établir le lien entre le site1 et le site2 :




Le lien créé va se manifester par la création de variables d'environnement du conteneur bénéficiaire. Cela se réalise via la commande : *sudo docker exec proxy printenv*

Liste des variables d'environnement du conteneur proxy :

```
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker/proxy$ sudo docker exec proxy printenv
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=eceeb9a017c9
SITE1_PORT=tcp://172.17.0.2:80
SITE1_PORT_80_TCP=tcp://172.17.0.2:80
SITE1_PORT_80_TCP_ADDR=172.17.0.2
SITE1_PORT_80_TCP_PORT=80
SITE1_PORT_80_TCP_PROTO=tcp
SITE1_NAME=/proxy/site1
SITE1_ENV_NGINX_VERSION=1.15.5-1~stretch
SITE1_ENV_NJS_VERSION=1.15.5.0.2.4-1~stretch
SITE2_PORT=tcp://172.17.0.3:80
SITE2_PORT_80_TCP=tcp://172.17.0.3:80
SITE2_PORT_80_TCP_ADDR=172.17.0.3
SITE2_PORT_80_TCP_PORT=80
SITE2_PORT_80_TCP_PROTO=tcp
SITE2_NAME=/proxy/site2
SITE2_ENV_NGINX_VERSION=1.15.5-1~stretch
SITE2_ENV_NJS_VERSION=1.15.5.0.2.4-1~stretch
NGINX_VERSION=1.15.5-1~stretch
NJS_VERSION=1.15.5.0.2.4-1~stretch
HOME=/root
```

2)

Nous devons vérifier que le fichier /etc/hosts du proxy a bien été mis à jour. Malheureusement, aucune modification n'est apparente dans ce fichier. Il n'y a ni mise à jour, ni association nom/@ip.

3)

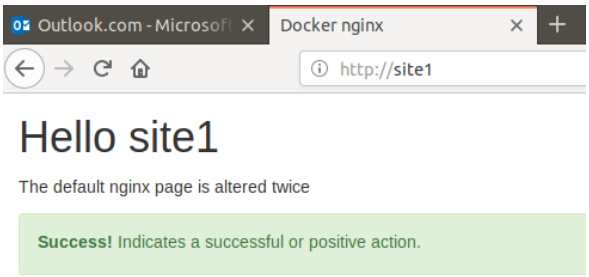
La liaison entre conteneurs va permettre de lier ces derniers par nom de conteneur plutôt que par adresses IP. On va modifier en conséquence les fichiers de configuration du serveur proxy en remplaçant les adresses IP par les noms des conteneurs.

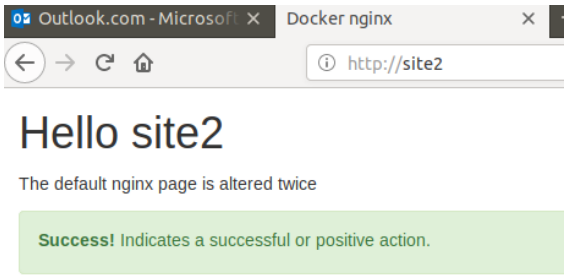
```
server {
    listen 8081;
    server_name site1;

    location / {
        proxy_pass http://site1;
    }
}
```

```
server {
    listen 8080;
     server_name site2;

    location / {
        proxy_pass http://site2;
    }
}
```




4)

```
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker/proxy$ sudo docker stop proxy
proxy
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker/proxy$ sudo docker stop site2
site2
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker/proxy$ sudo docker stop site1
site1
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker/proxy$ sudo docker start proxy
Error response from daemon: Cannot link to a non running container: /site1 AS /proxy/site1
Error: failed to start containers: proxy
```

Dans la figure ci-dessus, on voit que lorsque l'on éteint tous les conteneurs puis qu'on les rallume, le conteneur proxy ne veut pas démarrer en premier car il ne peut pas faire le lien vers les deux autres conteneurs. Par conséquent, il faut d'abord allumer les conteneurs site1 et site2 pour pouvoir ensuite lancer proxy.

## Partie II – Docker-compose

Les liens peuvent s'avérer utile pour la réalisation d'une composition simple. Néanmoins, la ligne de commande peut rapidement devenir surchargée. Pour éviter cela, nous allons utiliser *docker-compose*.

1)

On commence par créer un nouveau projet avec la structure ci-dessous, à la différence que la racine s'appellera td_docker-compose de façon à le différencier du répertoire utilisé dans la partie précédente.

```
td_docker
├── docker-compose.yml
├── proxy
│   ├── Dockerfile
│   ├── site1.conf
│   └── site2.conf
├── site1
│   ├── Dockerfile
│   └── index.html
└── site2
    ├── Dockerfile
    └── index.html
```

2)

Après avoir lu la documentation de référence liée au docker-compose-file, on crée un fichier docker-compose.yml à la racine de notre projet. Il est configuré de la manière suivante :

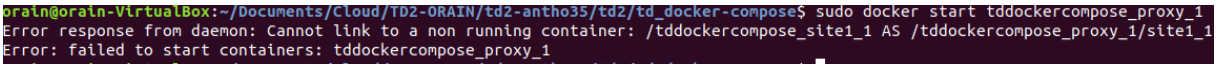
```yaml
proxy:
   image: imgproxy
   ports:
   - 80:80
   links:
   - site1
   - site2
site1:
   image: nginx
site2:
   image: nginx
```

3)

Dans la figure ci-dessous, on exécute un *docker-compose build* puis on lance la composition (*docker-compose up*) en tâche de fond, en utilisant le -d :

```
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker-compose$ sudo docker-compose build
site2 uses an image, skipping
site1 uses an image, skipping
proxy uses an image, skipping
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker-compose$ sudo docker-compose up -d
Creating tddockercompose_site2_1 ...
Creating tddockercompose_site1_1 ...
Creating tddockercompose_site2_1
Creating tddockercompose_site1_1 ... done
Creating tddockercompose_proxy_1 ...
Creating tddockercompose_proxy_1 ... done
orain@orain-VirtualBox:~/Documents/Cloud/TD2-ORAIN/td2-antho35/td2/td_docker-compose$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                  NAMES
82f563c3b4e6        imgproxy            "nginx -g 'daemon of…"   54 seconds ago      Up 52 seconds       0.0.0.0:80->80/tcp     tddockercompose_proxy_1
2acba3e1fdec        nginx               "nginx -g 'daemon of…"   56 seconds ago      Up 53 seconds       80/tcp                 tddockercompose_site1_1
a253d0688ef9        nginx               "nginx -g 'daemon of…"   56 seconds ago      Up 53 seconds       80/tcp                 tddockercompose_site2_1
1ea9265b37c3        nginx               "nginx -g 'daemon of…"   30 minutes ago      Up 30 minutes       0.0.0.0:8080->80/tcp   site2
583938ca1323        nginx               "nginx -g 'daemon of…"   30 minutes ago      Up 30 minutes       0.0.0.0:8081->80/tcp   site1
```

4)




Lorsque l'on stoppe les conteneurs et qu'on les redémarre, on obtient une erreur : le conteneur lié au proxy ne peut démarrer car il a besoin des deux autres conteneurs pour démarrer.