Université de Picardie Jules Verne

Informatique – Master CCM

INSSET – Saint-Quentin

Conteneurs Applicatifs et Micro-Services M2

C. Drocourt

cyril.drocourt@u-picardie.fr

Cours 3 : Docker Compose

V2023.1

Table des matières

1 - Introduction et installation

Si on désire exécuter plusieurs conteneurs pour un projet, qui vont donc avoir des liens entre eux, il est préférable d'utiliser un outil spécifique. Il en existe plusieurs mais celui officiel de Docker s'appelle « docker-compose ».

Sous Ubuntu il suffit d'utiliser la gestion standard des paquets :

```
[root@ubuntu22 ~]# apt install docker-compose
```

2 - Format du fichier version 1.0

Il est d'abord nécessaire de créer un répertoire de travail pour ce projet, dans notre exemple nous allons instancier deux conteneurs :

```
[root@drocourt ~]# mkdir my_project
[root@drocourt ~]# cd my_project
```

Il est ensuite nécessaire de créer un fichier de configuration qui sera nommé « docker-compose.yml » dans le répertoire du projet :

```
[root@drocourt ~/my_project]# vi docker-compose.yml
```

Il va être nécessaire dans ce fichier de décrire les deux environnements, liens qui les unissent, les ports, l'initialisation des variables, …

```
  cont1:
    image: machin:latest
    volumes:
      - vol1:/var/lib/clopi
    restart: always
    environment:
      MYVAR1: valeur1
      MYVAR2: valeur2

  cont2:
    links:
      - cont1
    image: truc:latest
    ports:
      - "8034:92"
    restart: always
    environment:
      MYVARX: toto
      MYVARY: patrick
```

Puis on exécute l'environnement :

```
[root@drocourt ~/my_project]# docker-compose up -d
Pulling cont1 (machin:3.4)...
3.4: Pulling from machin
Creating myproketct_cont1_1...
Pulling cont2 (truc:latest)…
```

On vérifie :

```
[root@drocourt ~/my_project]# docker-compose ps
         Name                       Command           State       Ports
-------------------------------------------------------------------------------
myproject_cont1_1       docker-entrypoint.sh machind  Up   3412/tcp
myproject_cont2_1       docker-entrypoint.sh trucd    Up   0.0.0.0:8034->92/tcp
```

Pour stopper :

```
[root@drocourt ~/my_project]# docker-compose stop
```

Pour supprimer les containers ainsi que les réseaux associés (l'option -v permet de demander également la suppression des volumes) :

```
[root@drocourt ~/my_project]# docker-compose down -v
```

Dans le fichier, il est possible également d'utiliser les options suivantes :

- Pour exposer un port :

```
expose:
  - "3812"
```

- Pour un volume d'un autre conteneur :

```
volumes_from:
  - <machine>
```

De plus, la commande « docker-compose » possède les commandes suivantes :

- `rm` : pour supprimer tous les conteneurs,
- `scale <nom>=X` : Pour lancer X instances du conteneur <nom>,
- `logs` : pour afficher tous les logs de l'ensemble,
- `restart` : pour redémarrer l'ensemble des conteneurs,

3 - Version 2, sections services volumes et networks

La version 2 du fichier de l'outil « docker-compose » est utilisable à partir de la version 1.10 de Docker, et introduit les éléments suivants :

- Le ficher doit débuter avec la ligne « `version: '2'` »
- Les container sont regroupés dans une section « `services` »
- Les partages sont regroupés dans une section « `volumes` »
- Les directives réseaux sont regroupés dans une section « `networks` »

Un fichier possède donc la structure suivante :

```
version: '2'
services:

volumes:

networks:
```

3.1 - Les services

La section service contient la définition des containers comme dans la version 1 de « docker-compose » avec quelques différences :

- Ajout d'une directive « depends_on » pour indiquer des dépendances à d'autres containers et ainsi les démarrer avant :

```
depends_on:
  - dbpostgres
```

- Ajout d'une directive « networks » dans laquelle on peut définir les réseaux auxquels le container appartient, à condition de les avoir déclarer dans la section « networks » :

```
networks:
  - frontend
  - backend
```

- Suppression possible des directives « link » car docker-compose va créer automatiquement un réseau par défaut dans lequel il place les containers.

3.2 - Les volumes

La section volume est censée contenir la définition des volumes utilisés par l'ensemble des containers de la section service, par exemple pour déclarer un volume nommé partage :

```
  <nom_partage>:
        driver: local
```

ou simplement :

```
  <nom_partage>: {}
```

On peut également indiquer que le volume existe déjà par la directive « external: true » et même indiquer un nom alternatif si besoin :

```
<nom_partage>:
    external: true
    name: <autre_nom>
```

Dans la section service, il s'utilise simplement :

```
service :
   ...
   volumes:
     - <nom_partage>:/mnt/appl
```

3.3 - Les réseaux

En général, la section indique simplement le réseau par défaut :

```
networks:
    default:
```

Il est cependant possible de préciser la création d'autres réseaux en indiquant simplement leurs noms, qui seront ensuite utilisables dans la section « networks » des services :

```
networks:
    proxy:
    db:
```

Dans la section service, il s'utilise simplement :

```
service :
   ...
   network:
     - proxy
```

4 - Compose et build

Dans le fichier « docker-compose.yml », il est également possible de construire les conteneurs, en supprimant la directive « image : XXX », et en plaçant la directive :

```
build: ./directory
```

Le répertoire « directory » va contenir l'ensemble des fichiers nécessaires à la création du conteneur, dont le fichier « Dockerfile ».

Il est possible d'indiquer un autre nom pour le fichier « dockerfile » par la directive :

```
dockerfile : df-myname
```

Il est néanmoins possible d'utiliser la directive « image : xx:tag », qui sera alors le nom utilisé pour la construction.