Devops

TD2 Docker

Tom MAFILLE

1. Intro

1. C'est quoi httpd ?

   -> httpdaemon, pour serveurs http. Ici c'est l'image docker associée au serveur apache

2. Qu'est-ce que alpine ?

   -> distrib linux légère, ici le serveur est basé sur alpine

2. Réseau et gestion des ports

2.1 Mise en pratique et découverte du réseau avec Docker

1. A quoi correspond l'option -d ?

   -> `-d` permet de lancer le docker en background. Utilité : utiliser le terminale en parallèle.

2. A l'aide des commanded docker inspect et ifconfig, expliquez la configuration réseau créée par Docker.

   -> `ip a : bash 5: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default link/ether 02:42:14:23:fb:1b brd ff:ff:ff:ff:ff:ff inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0   valid_lft forever preferred_lft forever inet6 fe80::42:14ff:fe23:fb1b/64 scope link proto kernel_ll   valid_lft forever preferred_lft forever`

   -> `docker inspect httpd1` Donne l'adresse `172.17.0.2`

   -> Le docker expose une page http accessible depuis l'hôte. Le local host est `.1` et la page accessible est `.2`.

3. Pouvez-vous accedez à la page web créée par le conteneur via votre navigateur ?

   -> Oui

4. Si oui comment ?

   -> En allant sur l'adresse `172.17.0.2`.

2.2 Mappage des ports entre l'hôte et le conteneur

1. Quelle option de `docker run` permet de mapper un port du conteneur sur un port de l'hôte ?

   -> `-p`

2. Appliquez cette option à la commande docker run précédente pour mapper le port 80 du conteneur sur le port 80 de votre hôte.

   -> `ddocker run -p 80:80 -d --name httpd1 --rm httpd:2-alpine`

3. Essayez de lancer un autre conteneur httpd (nommé httpd2) sur le même port d'hôte. Que se passe-t-il ?

   ->

`docker: Error response from daemon: driver failed programming external connectivity on endpoint httpd1`

4. Comment peut-on lire les logs de notre premier conteneur httpd ?

   -> `docker logs CONTAINER_ID` ou `docker logs -f CONTAINER_ID` (temps réel).

5. Maintenant que les ports 80 sont mappés, avec quelle adresse peut-on joindre notre serveur web depuis un navigateur ?

   -> sur localhost.

6. Dans ces logs, trouvez l'adresse IP appellante quand consulter la page d'accueil de httpd. A quoi correspond-t-elle ?

   -> `172.17.0.1 - - [10/Dec/2024:13:08:35 +0000] "GET / HTTP/1.1" 200 45` Cela donne l'adresse IP du docker, la date et la requete http

2.3. Docker et les réseaux

1. Dans quel "réseau docker" est le conteneur httpd ?

   -> En faisant `docker network ls`, on trouve les NETWORK ID correspondants aux différents réseaux. Ici, on a 051b02b054c9 qui correspond à bridge, et qui est le même ID que lorsqu'on effectue `docker inspect httpd1`. Donc on est sur le bridge.

2. Trouvez la commande pour créer un réseau Docker et créez un réseaux 'frontend'

   -> `docker network create frontend`

3. Peut-on connecter le conteneur dans ce nouveau réseau sans l'éteindre ?

   -> `docker network connect frontend httpd1` OUI

4. Trouvez la commande pour lister les réseaux

   -> `docker network ls`

5. Trouvez la commande pour connaitre les adresses IPs d'un réseau

   -> `docker network inspect NETWORK_NAME`

6. Peut-on connecter un conteneur sur plusieurs réseaux en même temps ?

   -> OUI

7. Quel peut bien en être l'intérêt ?

   -> Se co à plusieurs sous-réseaux (segmenter accès externes, + modulaire par ex).

3. Partager les fichiers dynamiquement entre l'hote et le conteneur

1. Dans la doc de httpd, trouvez où est situé le fichier qui s'affiche sur votre navigateur web quand vous vous connecter à l'interface web du conteneur.

   -> `/usr/local/apache2/htdocs/index.html`

2. C'est quoi un volume docker ?

   -> Un volume Docker -> permet de partager des fichiers entre un conteneur et son hôte ou entre conteneurs

3. Quelles sont les possibilités de 'montage' d'un volume ?

   -> Montage bind : associe un répertoire de l'hôte à un conteneur. Volume Docker standard : persistance gérée par Docker.

4. Trouvez l'option pour monter le volume entre le fichier seance2/index.html et le chemin de la question 3.1 dans le conteneur

-> `docker run -d --name httpd1 -p 8080:80 -v $(pwd)/seance2/index.html:/usr/local/apache2/htdocs/i`
`httpd`

5. Quelles peuvent être les autres utilités d'un volume ? Lors de changement de version par exemple ?

   -> Garder les données même après suppression du docker, partager les données entre conteneurs

4. Utilisation basique de Docker compose

4.1. Première partie: reproduire la configuration précédement créée

FAIT.

4.2 Manipulation de quelques commandes docker-compose

1. Expliquez la commande `docker-compose -f seance2/docker-compose.yml up`

   -> Lance l'ensemble de conteneurs défini dans un fichier `docker-compose.yml`

2. Que se passe-t-il si vous rajoutez -d en fin de commande ?

   -> En arrière plan (libère le terminal)

3. Que fait la commande docker-compose -f seance2/docker-compose.yml stop ?

   -> Arrête les conteneurs sans les supprimer

4. Quelle est la différence si vous remplacer stop par down ?

   -> Arrête les conteneurs + supprime les ressources

5. Les données sont-elles conservées après un stop + restart ?

   -> Oui

6. Les données sont-elles conservées après un down + up ?

   -> Non

4.3. Pour aller plus loin : une mini appli python avec sa base de donnée

1. C'est quoi Redis ?

   -> Redis est une base de données clé-valeur en mémoire, souvent utilisée pour le caching ou les compteurs simples. Dans l'exemple, il stocke le nombre de visites d'une page web.

2. -> fichier .py :

```python
import time
import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379) # TODO: Ici nous allons devoir définir le nom d'hôte qui

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
```

```
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times. /n'.format(count)
```

-> fichier yml :

```
version: "3"
services:
web:
    build:
    context: .
    ports:
    - "5000:5000"
    networks:
    - python_network

redis:
    image: "redis:5.0.10-alpine"
    networks:
    - python_network

# Déclaration du réseau
networks:
python_network:
    driver: bridge
```

4.4. Maintenant, nous souhaitons changer de version de Redis

1. Sur le docker hub, trouvez le tag qui correspond à l'image qu'on veut.

   -> image sur docker hub : redis:6-alpine

2. Quel champ du fichier docker-compose doit-on modifier ?

   -> modifs dans le fichier yml :

   ```
   redis:
   image: "redis:6-alpine"
   ```

3. Les données vont-elles être conservées lors du changement de version d'image ?

   -> Non

4. Pourquoi ?

   -> Car dockers, non persistants

5. Que pouvons nous utiliser pour conserver les données lors de la montée en version ?

   -> Des volumes