Rendu de TP Docker

Membres du binôme :

MAHAMAT Masse Alamine

MAKON Manyim Ma

Partie 0: Définitions

0.1- Rappeler le concept d'un Dockerfile => permet la création d'une image docker

0.2- Rappeler le concept d'un docker-compose => permet la mise en place d'une infrastructure au niveau d'un même host

0.3- C'est quoi le docker registry ? => Un annuaire public ou privé servant à la publication des images docker

Partie 1: Dockerfile

Télécharger le fichier zip suivant: http://demo.amamou.com/app.zip C'est une simple application permettant l'enregistrement par nom et email dans une base de donnée postgres

Après avoir installé postgresql, se connecter sur la base de données

```
alamine@alamine-VirtualBox:~/Documents/tp docker$ sudo apt install postgresql
[sudo] Mot de passe de alamine :
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
```

```
alamine@alamine-VirtualBox:~/Documents/tp docker$ su
Mot de passe :
root@alamine-VirtualBox:/home/alamine/Documents/tp docker# su postgress
su: l'identifiant postgress n'existe pas.
root@alamine-VirtualBox:/home/alamine/Documents/tp docker# su postgres
```

```
postgres@alamine-VirtualBox:/home/alamine/Documents/tp docker$ psql
psql (12.8 (Ubuntu 12.8-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# create database groupeX;
CREATE DATABASE
```

```
1  #!/bin/bash
2  set -e
3
4  flask db upgrade
5  flask run -h 0.0.0.0 -p 5000
6
7  export DBUSER=groupeX
8  export DBPASS=groupeX
9  export DBHOST=groupeX
10 export DBNAME=groupeX
```

Figure 1 Contenu du fichier entrypoint.sh

```
alamine@alamine-VirtualBox:~/Documents/tp docker/app$ sudo apt install python3-
pip
```

```
alamine@alamine-VirtualBox:~/Documents/tp docker/app$ pip3 install -r requireme
nts.txt
```

```
alamine@alamine-VirtualBox:~/Documents/tp docker/app$ sh entrypoint.sh
```

Figure 2 Installation de python

1.2- Créez une image docker a partir du Dockerfile fourni

```
alamine@alamine-VirtualBox:~/Documents/tp docker/app$ sudo docker build . -t tp
1_2
```

```
alamine@alamine-VirtualBox:~/Documents/tp docker/app$ sudo docker run tp1_2
```

1.3- Modifier votre dockerfile pour avoir votre serveur postgres au niveau de l'image générée

```
  GNU nano 4.8                          Dockerfile
FROM tiangolo/uwsgi-nginx-flask:python3.6
WORKDIR /app
RUN apt update && apt install postgresql -y

COPY . .
RUN pip install -r requirements.txt
RUN cp entrypoint.sh /docker-entrypoint.sh

USER postgres
RUN /etc/init.d/postgresql start &&\
psql --command "CREATE USER groupex with SUPERUSER PASSWORD 'groupeX';" &&\
createdb -O groupex groupex

USER root
expose 5000
ENV DBUSER=groupex
ENV DBPASS=groupeX
ENV DBHOST=localhost
ENV DBNAME=groupex

ENV FLASK_APP=app.py
RUN chmod 750  /docker-entrypoint.sh
ENTRYPOINT ["/docker-entrypoint.sh"]


                    [ Lecture de 23 lignes ]
^G Aide       ^O Écrire     ^W Chercher   ^K Couper     ^J Justifier
^X Quitter    ^R Lire fich. ^\ Remplacer  ^U Coller     ^T Orthograp.
```

Figure 3 Présention du fichier Dockerfile

```
alamine@alamine-VirtualBox:~/Documents/tp docker/app$ sudo docker run tp1_3
Starting PostgreSQL 11 database server: main.
INFO  [alembic.runtime.migration] Context impl PostgresqlImpl.
```

```
   WARNING: This is a development server. Do not use it in a production deploym
ent.
 * Running on http://172.17.0.3:5000/ (Press CTRL+C to quit)
```

Partie 2: Docker-compose

2.1- - créer un fichier docker-compose permettant de mettre en place deux containers avec un pour l'application et un deuxième pour la base de données

```
1 version: "3.7"
2 services:
3  db:
4   image: postgres
5   restart: always
6   environment:
7    POSTGRES_USER: adex
8    POSTGRES_PASSWORD: adex
9    POSTGRES_DB: adex
10  app:
11   build: .
12   restart: always
13   ports:
14    -"5000:5000"
15   environment:
16    DBUSER: adex
17    DBPASS: adex
18    DBHOST: db
19    DBNAME: adex
20    restart: always
21    FLASK_APP: app.py
22   depends_on:
23    - db
```

Figure 1 docker-compose v1 par ManyimMa

```
alamine@alamine-VirtualBox:~/Documents/tp docker/app$ sudo docker-compose up
Creating network "app_default" with the default driver
```

2.2- créer un deuxième fichier docker compose permettant cette fois ci de générer deux containers pour l'application avec deux accès différents au niveau de la db. L'application A1 aura le droit d'écriture sur la DB l'application A2 copie de l'application A1 n'aura que les droits de lecture sur la DB

```yaml
version: "3.7"
services:
 db:
  image: postgres
  volumes:
   - ./init.sql:/docker-entrypoint-initdb.d/init.sql
  environment:
   POSTGRES_DB: adex
   POSTGRES_HOST: localhost
   POSTGRES_USER: adex
   POSTGRES_PASSWORD: adex
 app1:
  build: .
  restart: always
  ports:
   - "5000:5000"
  environment:
   FLASK_APP: app.py
   FLASK_ENV: development
   FLASK_RUN_HOST: 0.0.0.0
   DBUSER: adex1
   DBPASS: adex
   DBHOST: db
   DBNAME: adex

  depends_on:
   - db
 app2:
  build: .
  restart: always
  ports:
   - "5050:5000"
  environment:
   FLASK_APP: app.py
   FLASK_ENV: development
   FLASK_RUN_HOST: 0.0.0.0
   DBUSER: adex2
   DBPASS: adex2
   DBHOST: db
   DBNAME: adex
  depends_on:
   - db
```

```
db_1   | 2021-10-13 17:45:45.018 UTC [1] LOG:  database system is ready to acce
pt connections
```

```
n deployment.
app_1  |  * Running on http://172.18.0.3:5000/ (Press CTRL+C to quit)
```

Figure 2 docker-compose v2 par ManyimMa

2.3- modifier le code de A2 puisse démarrer car elle ne peut plus être utilisée vu que l'accès à la DB se fait uniquement en lecture

Partie 3: Docker-registry

3.1- Mettrez en place localement un docker registry

```
registry:
  container_name: registry
  restart: always
  image: registry:2
  ports:
    - 5000:5000
  volumes:
    - "/storage/registry/data:/var/lib/registry"
    - "/storage/registry/certs:/certs"
    - "/storage/registry/auth:/auth"
  environment:
    - REGISTRY_HTTP_TLS_CERTIFICATE=/certs/example.com.crt
    - REGISTRY_HTTP_TLS_KEY=/certs/example.com.key
    - REGISTRY_AUTH=htpasswd
    - REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm
    - REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd
```

Figure 4Docker-registry

3.2- tagger les images générées précédemment et envoyer les au niveau de votre registry local

```
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app# docker run -d -p 5000:5000 --restart always --name registry registry:2
6708fa5e2c810982c6782424075686709 93a3fb601b923870121367 19ab16436
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                          PORTS                                   NAMES
6708fa5e2c81        registry:2          "/entrypoint.sh /etc…"   5 seconds ago       Up 3 seconds                    0.0.0.0:5000->5000/tcp                  registry
42275fcdb35c        app_app             "/docker-entrypoint.…"   7 minutes ago       Up 7 minutes                    80/tcp, 443/tcp, 0.0.0.0:80->5000/tcp   app_app_1
3c0e81128b1c        app_app2            "/docker-entrypoint.…"   7 minutes ago       Restarting (1) 51 seconds ago                                           app_app2_1
61fe31ec4d04        postgres            "docker-entrypoint.s…"   7 minutes ago       Up 7 minutes                    5432/tcp                                app_db_1
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app# docker tag app_app localhost:5000/app_image
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app# docker tag postgres localhost:5000/postgres_image
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app#
```

3.3- reconstruire les images après un légère modification du code et taguer les avec un nouveau tag permettant de les différencier de la premiere version

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                          PORTS                                   NAMES
6708fa5e2c81        registry:2          "/entrypoint.sh /etc…"   5 minutes ago       Up 5 minutes                    0.0.0.0:5000->5000/tcp                  registry
42275fcdb35c        app_app             "/docker-entrypoint.…"   12 minutes ago      Up 12 minutes                   80/tcp, 443/tcp, 0.0.0.0:80->5000/tcp   app_app_1
3c0e81128b1c        app_app2            "/docker-entrypoint.…"   12 minutes ago      Restarting (1) 24 seconds ago                                           app_app2_1
61fe31ec4d04        postgres            "docker-entrypoint.s…"   12 minutes ago      Up 12 minutes                   5432/tcp                                app_db_1
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app# docker tag app_app localhost:5000/app_image2
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app# docker tag postgres localhost:5000/postgres_image2
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app#
```

3.4- push les nouvelles images sur votre registry local

```
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app# docker push localhost:5000/app_image2
The push refers to repository [localhost:5000/app_image2]
6b376a181593: Pushed
90cc633f8860: Pushed
23ea7026bfc1: Pushed
7fb60d5b5e82: Pushed
5ccca9f5f27f: Pushed
1971b933c82a: Pushed
596464cb926b: Pushed
07289a4a4d58: Pushed
4249ef06ae69: Pushed
e775b278161a: Pushed
27d9a6169c9f: Pushed
b9557a3d31d9: Pushed
c8f52965417f: Pushed
9cde96ca61c2: Pushed
f3fd11fd6d6c: Pushed
e553743c668b: Pushed
88af765ea71f: Pushed
b47c358499df: Pushed
cbd550635f44: Pushed
2efef025159b: Pushed
3acbf6947195: Pushed
f90fe4978ca2: Pushed
88383b8e3cb5: Pushed
e93627dfc607: Pushed
8f23b00cc77f: Pushed
cf691a2ea3f9: Pushed
3d3e92e98337: Pushed
8967306e673e: Pushed
9794a3b3ed45: Pushed
5f77a51ade6a: Pushed
e40d297cf5f8: Pushed
latest: digest: sha256:fda5c75ce996f6b41c952254943cd0a0d47f8fd02017512598ccd4d477d9ad51 size: 6807
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app# docker push localhost:5000/postgres_image2
The push refers to repository [localhost:5000/postgres_image2]
119c31fbdbd6: Pushed
15378556b393: Pushed
5be11eea2fcb: Pushed
53b19c82edbf: Pushed
efe23cbeb2c2: Pushed
e6d39f005225: Pushed
57b15b12c86d: Pushed
20db730be8b4: Pushed
```

3.5- supprimer de votre registry local les premières image tagués

```
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app# docker build -t img-groupe4 .
Sending build context to Docker daemon  32.77kB
Step 1/15 : FROM tiangolo/uwsgi-nginx-flask:python3.6
 ---> a16ce562e863
Step 2/15 : RUN apt-get install postgresql -y
 ---> Running in 888603f6aa57
Reading package lists...
Building dependency tree...
Reading state information...
E: Unable to locate package postgresql
The command '/bin/sh -c apt-get install postgresql -y' returned a non-zero code: 1
00
root@tp-docker:/home/tp-docker/Bureau/tp-docker/app#
```

3.6- déclencher le garbage collector au niveau de votre registry pour nettoyer les blob non référencés (suite à la suppression du tag)

Partie 4: Gestion du cycle de vie des images:

Construire un script qui détecte la dernière version de l'image hébergée sur le registry et tag automatiquement la nouvelle image avec la version suivante. Exemple : dernier image tag=1.2 image suivante tag > 1.2 vous pouvez choisir librement les règles de tag

4.2- modifier le script de 4-1 pour automatiquement tester le bon fonctionnement de votre application, tag et push l'image.