LOG3000 - Processus du génie logiciel

Hiver 2024

TP No. 4

2137443 - Stefan Cotargasanu
2131367 - Faneva Rakotoarivony

Soumis à : Patrick Loic Foalem

21 mars 2024

4.1 - Questions d'analyse sur la section 3.1

1. Expliquez avec vos propres mots ce que fait la commande pull alpine.
La commande docker images récupère l'image dans le registre Docker Hub et la télécharge sur la machine locale.

2. Expliquez avec vos propres mots ce qui se passe derrière l'écran lorsque vous exécutez la commande docker run.
Lorsque l'on exécute la commande docker run, un conteneur lié à l'image spécifié est créé et exécuté.

3. Expliquez avec vos propres mots ce qui se passe derrière l'écran lorsque vous exécutez la commande docker run alpine echo "hello from alpine". S'agit-il d'une sortie de Docker ou de Linux Alpine ?
Lorsque cette commande est exécutée, un conteneur basé sur l'image alpine est créé et lancé, dans ce conteneur la commande echo est exécutée. La sortie est celle de Alpine Linux, car la commande echo est exécutée dans le conteneur. Cette sortie est redirigée par Docker au stdout.

4. Quelle est la différence entre une image et un conteneur ?
La différence est qu'une image est la source d'un conteneur, un conteneur n'existe que lorsque l'image est exécutée. Une image est exécutable, lorsqu'elle est exécutée, elle créé un conteneur basé sur cette image.

5. Quels sont les avantages d'un conteneur par rapport à une machine virtuelle ?
L'avantage d'un conteneur Docker comparé à une machine virtuelle se base sur le fait que les conteneurs partagent le même noyau que la machine hôte, ce qui permet de réduire leur taille ainsi que le temps de démarrage. En comparaison, une machine virtuelle doit avoir un kernel complet.

4.2 Questions d'analyse de la section 3.2

6. Expliquez avec vos propres mots chaque paramètre utilisé à l'étape 4 de la section 3.2 La commande lancée à l'étape 4 de la section 3.2 est docker run --name static-site -e AUTHOR="faneva_stefan" -d -P dockersamples/static-site. Nous commençons donc par **docker run** qui permet de lancer un conteneur basé sur l'image **dockersamples/static-site**. En suite, nous avons les différentes options utilisées :

— –**name static-site** : permet de donner un nom au conteneur, dans ce cas **static-site**.

— **-e AUTHOR="faneva_stefan"** : permet de définir une variable d'environnement **AUTHOR** avec la valeur **faneva_stefan**.
— **-d** : permet de lancer le conteneur en mode détaché, c'est-à-dire que la console n'est pas bloquée par le process.
— **-P** : permet de publier tous les ports exposés par le conteneur sur des ports aléatoires de la machine hôte.

4.3 Questions d'analyse sur la section 3.3

7. Expliquez la sortie de la commande docker images. Comment obtenir une version spécifique d'une image ?
La sortie de la commande docker images est une liste des images disponibles localement. Pour obtenir une version spécifique d'une image, il suffit de spécifier le nom de l'image suivi de la version désirée. Par exemple, pour obtenir la version 1.0 de l'image alpine, il suffit de lancer la commande docker pull alpine:1.0. Ou sinon si la question demandait comment avoir les informations sur une image spécifique, il est possible de lancer la commande docker images <image>:<version>.

8. Expliquez avec vos propres mots la différence entre base images et child images.
Les images Docker sont composées de plusieurs layers. L'image de base est la première image utilisée, celle mentionnée dans le **FROM**. les child images sont des images qui sont basées sur cette image mais avec des layers supplémentaires.

9. Expliquez avec vos propres mots la différence entre official images et user images.
Les official images sont des images prédéfinies, mises en places et maintenues par certains organismes (Docker, Canonical, RedHat, etc...) et qui sont disponibles à tous dans le registry Docker Hub. Les user images sont des images créées par des utilisateurs.

10. Expliquez avec vos propres mots ce qu'est un Dockerfile.
Un Dockerfile est un fichier contenant une série de commande permettant au Docker Builder de créér une image.

11. Expliquez chaque ligne du Dockerfile créé à l'étape 5 de la section 3.3.

— **FROM alpine :3.5** : spécifie l'image de base, dans ce cas l'image alpine version 3.5.

— **RUN apk add –update py2-pip** : installe le paquet py2-pip, qui est un gestionnaire de paquet pour Python.

— **COPY requirements.txt /usr/src/app/** : Copie le fichier requirements.txt dans le répertoire /usr/src/app/ du conteneur.

— **RUN pip install –no-cache-dir -r /usr/src/app/requirements.txt** : Installe les dépendances spécifiées dans le fichier requirements.txt.

— **COPY app.py /usr/src/app/** : Copie le script app.py dans le répertoire /usr/src/app/ du conteneur.

— **COPY templates/index.html /usr/src/app/templates/** : Copie le fichier index.html dans le répertoire /usr/src/app/templates/ du conteneur.'

— **EXPOSE 5000** : Expose le port 5000 du conteneur.

— **CMD ["python", "/usr/src/app/app.py"]** : Définit le point d'entrée du conteneur comme étant le script app.py défini plus tôt.

12. Expliquez ce qui se passe lorsque vous exécutez la commande
docker build -t <YOUR_USERNAME>/flask-app. Lorsque cette commande est lancée, le builder va exécuter chacune des commandes du Dockerfile, créant une nouvelle image.
Cette image est ensuite taggée avec le nom spécifié, dans ce cas **<YOUR_USERNAME>/flask-app**.

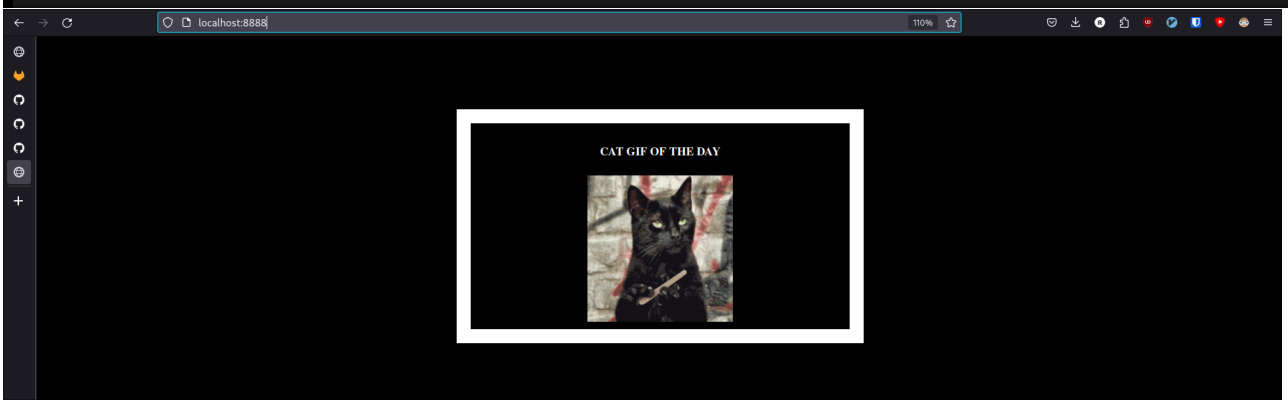
13. Décrivez chaque ligne du Dockerfile que vous avez créé. Ajoutez des captures d'écran et la sortie des commandes pour appuyer votre explication. Ajoutez également une image de votre navigateur montrant l'URL et le site Web en cours d'exécution. N'oubliez pas de soumettre le code source que vous avez développé
Le Dockerfile que nous avons utilisé se trouve en annexe. Ligne par ligne, voici ce qu'il fait :
FROM ubuntu :22.04 : spécifie l'image de base, dans ce cas l'image ubuntu version 22.04.
RUN apt-get update && apt install -y python3 python3-pip : Mets à jour les repos apt et install python ainsi que pip.
COPY requirements.txt /usr/src/app : Copie le fichier requirements.txt dans le dossier /usr/src/app/
RUN pip install –no-cache-dir -r /usr/src/app/requirements.txt : Installe les dépendances spécifiées dans le fichier requirements.txt.
COPY app.py /usr/src/app/ : Copie le script app.py de l'application dans le conteneur.
COPY templates/index.html /usr/src/app/templates/ : Copie le template html dans le dossier /usr/src/app/templates/
EXPOSE 5000 : Expose le port 5000 du conteneur.
CMD ["python3", "-m", "flask", "-A", "/usr/src/app/app.py", "run", "–host=0.0.0.0"] : Définit le point d'entrée du conteneur comme étant le script app.py défini plus tôt en utilisant Flask.

Voici des captures d'écran :

```
❯ matricules; docker build -t tp4cats .
2131367-2137443
2024-03-21 14:18:13
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  23.73MB
Step 1/8 : FROM ubuntu:22.04
 ---> fd1d8f58e8ae
Step 2/8 : RUN apt update && apt install -y python3 python3-pip
 ---> Using cache
 ---> 9261a617aa88
Step 3/8 : COPY requirements.txt /usr/src/app/
 ---> Using cache
 ---> b0606133f9c5
Step 4/8 : RUN python3 -m pip install --no-cache-dir -r /usr/src/app/requirements.txt
 ---> Using cache
 ---> a31a001fbc4b
Step 5/8 : COPY app.py /usr/src/app/
 ---> de48baad95e6
Step 6/8 : COPY templates/index.html /usr/src/app/templates/
 ---> 9ece455a0695
Step 7/8 : EXPOSE 5000
 ---> Running in c97fd4b66e14
 ---> Removed intermediate container c97fd4b66e14
 ---> 933230eae0a1
Step 8/8 : CMD ["python3", "-m", "flask", "-A", "/usr/src/app/app.py", "run", "--host=0.0.0.0"]
 ---> Running in 8d50a42526f2
 ---> Removed intermediate container 8d50a42526f2
 ---> 70b066e185b2
Successfully built 70b066e185b2
Successfully tagged tp4cats:latest
```

```
~/ecole/poly/Session6/LOG3000/TP4/flask_app on git main ?13                    took 4s at 14
❯ docker run -p 8888:5000 tp4-log3000 sh
❯ matricules; docker build -t tp4cats .
❯ docker run -p 8888:5000 tp4cats
 * Serving Flask app '/usr/src/app/app.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
172.17.0.1 - - [21/Mar/2024 18:43:36] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [21/Mar/2024 18:43:36] "GET /favicon.ico HTTP/1.1" 404 -
```



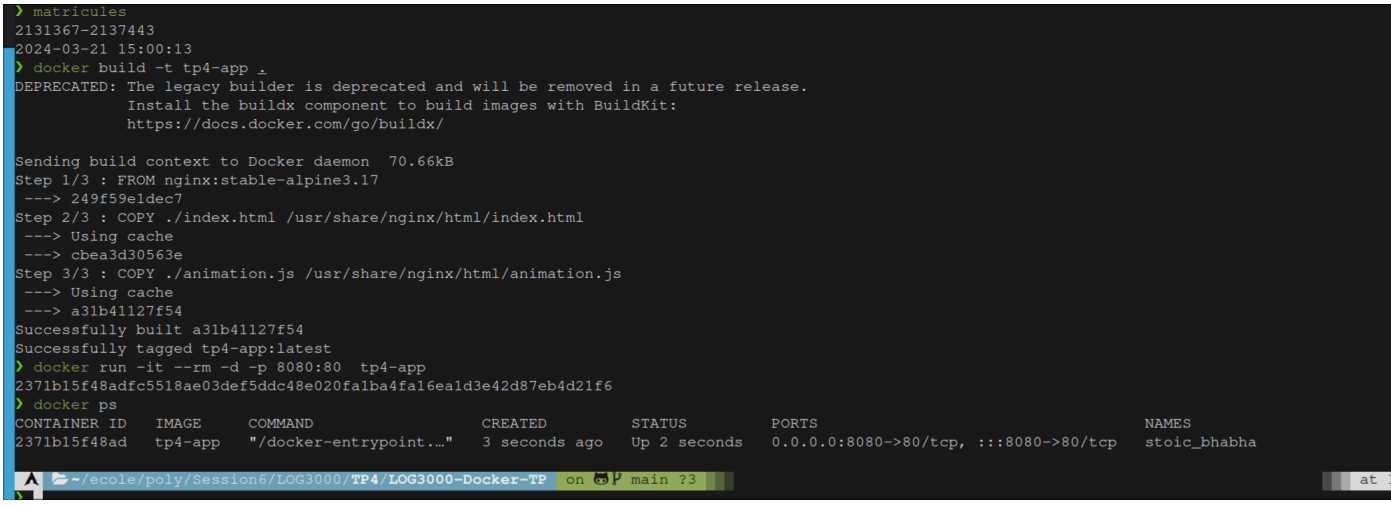
4.4 Questions d'analyse sur la section 3.4

Voici le Dockerfile que nous avons mis en place :

```
1  FROM nginx:stable-alpine3.17
2
3  COPY ./index.html /usr/share/nginx/html/index.html
4  COPY ./animation.js /usr/share/nginx/html/animation.js
```

Nous nous basons sur une image de nginx qui se base sur la distribution linux d'Alpine. Ensuite, nous copions les fichiers index.html et animation.js dans le dossier /usr/share/nginx/html/ du conteneur. Cette page index est directement exposée par nginx.

Voici quelques captures d'écran de la création et du déploiement de l'application :




localhost:8080
110%
Bravo!!!! Vous avez créé votre première application avec docker!

4.5 Question de rétroaction

Nous travaillons à l'amélioration continue des travaux pratiques de LOG3000. Cette question peut être répondue très brièvement.

1. Combien de temps avez-vous passé au travail pratique, en heures-personnes, en sachant que deux personnes travaillant pendant trois heures correspondent à six heures-personnes. Est-ce que l'effort demandé pour ce laboratoire est adéquat ?

Nous avons passé environ 3 heures-personnes pour ce laboratoire. La partie faire le laboratoire était simple, la partie la plus longue était apprendre à utiliser LaTeXpour la rédaction du rapport. L'effort est adéquat, le sujet est très cool. Il pourrait être pertinent d'aborder docker-compose qui est plus complet pour des applications avec plusieurs conteneurs ! On est un peu parti hors-sujet pour la section 4.3 mais ça reste pertinent.

2. Quelles difficultés avez-vous rencontré lors de ce laboratoire ?

Surtout LaTeX.

Vous pourrez trouver tout le dossier source de ce TP sur le repo gitub suivant : `https://github.com/Fanevark/faneva_stfn_docker_tp`.

.1 Dockerfile 4.3

```
1  FROM ubuntu:22.04
2  RUN apt update && apt install -y python3 python3-pip
3  COPY requirements.txt /usr/src/app/
4  RUN python3 -m pip install --no-cache-dir -r /usr/src/app/requirements.txt
5  COPY app.py /usr/src/app/
6  COPY templates/index.html /usr/src/app/templates/
7  EXPOSE 5000
8  CMD ["python3", "-m", "flask", "-A", "/usr/src/app/app.py", "run",
   ↪  "--host=0.0.0.0"]
```

.2 Flask App 4.3

```
from flask import Flask, render_template
import random
app = Flask(__name__)

# list of cat images
images = [
        "https://media.giphy.com/media/BzyTuYCmvSORqs1ABM/giphy.gif",
        "https://media.giphy.com/media/xUPGcyi4YxcZp8dWZq/giphy.gif",
        "https://media.giphy.com/media/1iu8uG2cjYFZS6wTxv/giphy.gif",
        "https://media.giphy.com/media/mlvseq9yvZhba/giphy.gif",
        "https://media.giphy.com/media/lJNoBCvQYp7nq/giphy.gif"
]

@app.route('/')
def index():
        url = random.choice(images)
        return render_template('index.html', url=url)

if __name__ == " main ":
        app.run(host="0.0.0.0")
```