ré-requis : une VM, les proxy configurés, docker et docker-compose installés

Création d'une image alpine+python3

1/ Créer un répertoire alpine-python3 et s'y rendre

mkdir alpine-python3

cd alpine-python3

2/ créer un fichier appelé « Dockerfile », qui contiendra les instructions suivantes :

```
FROM alpine:latest

ENV http_proxy 'http://openwatt-proxy-np.itn.ftgroup:3128/'
ENV https_proxy 'http://openwatt-proxy-np.itn.ftgroup:3128/'
RUN apk add --no-cache python3 && \
  python3 -m ensurepip && \
  rm -r /usr/lib/python*/ensurepip && \
  pip3 install --upgrade pip setuptools && \
  rm -r /root/.cache
```

3/ la commande de build d'image

sudo docker build –no-cache –t alpine-python3

on obtient sur sa machine une image docker « alpine-python3 » que l'on pourra utiliser dans les projets python3

Création d’une image alpine+mongodb

1/créer un répertoire alpine-mongodb et s’y rendre

```
mkdir alpine-mongodb
cd alpine-mongodb
```

2/ créer un fichier nommé Dockerfile et le remplir avec les instructions suivantes :

```
FROM alpine:latest

ENV http_proxy 'http://openwatt-proxy-np.itn.ftgroup:3128/'
ENV https_proxy 'http://openwatt-proxy-np.itn.ftgroup:3128/'

RUN \
echo http://dl-4.alpinelinux.org/alpine/edge/testing >> /etc/apk/repositories && \
apk add --no-cache mongodb && \
rm /usr/bin/mongosniff /usr/bin/mongoperf

VOLUME /data/db
EXPOSE 27017 28017

COPY run.sh /root
ENTRYPOINT [ "/root/run.sh" ]
CMD [ "mongod" ]
```

2/ il faut aussi créer un fichier « run.sh » dont le contenu sera le suivant :

```
#!/bin/sh

# Docker entrypoint (pid 1), run as root

[ "$1" = "mongod" ] || exec "$@" || exit $?


# Make sure that database is owned by user mongodb

[ "$(stat -c %U /data/db)" = mongodb ] || chown -R mongodb /data/db


# Drop root privilege (no way back), exec provided command as user mongodb

cmd=exec; for i; do cmd="$cmd '$i'"; done

exec su -s /bin/sh -c "$cmd" mongodb
```

3/ la commande de build de l'image

```
sudo docker build –no-cache –t alpine-mongodb
```

on obtient une image appellee « alpine-mongodb » que l'on pourra utiliser pour tout projet utilisant une base mongodb.

Creation d'une image pour notre projet API library

1/ créer un répertoire « library » et s'y rendre

```
mkdir library

cd library
```

2/ copier le code source de notre projet

Fichier app.py

Répertoire controlers

Répertoire swagger

3/ créer un fichier nommé Dockerfile qui contiendra les instructions suivantes :

```
FROM alpine-python3
ARG http_proxy
ARG https_proxy
COPY . /app
WORKDIR /app
RUN pip3 install --user -r requirements.txt
```

4/ créer un fichier nommé requirements.txt qui contiendra les instructions suivantes :

```
connexion>=1.0
pymongo
```

5/ créer un fichier nommé docker-compose.yml qui contiendra les instructions suivantes :

```
version: '2'
services:
  library:
    build:
      context: .
```

```
    args:
     - http_proxy=http://openwatt-proxy-np.itn.ftgroup:3128/
     - https_proxy=http://openwatt-proxy-np.itn.ftgroup:3128/
   command: python3 -u app.py
   ports:
    - "8080:8080"
   volumes:
    - .:/app
   links:
    - db
  db:
   image: alpine-mongodb
```

Remarques : le link permet de « relier » nos deux containers micro-services, les « args » permettent de fournir les proxy pour le dockerfile et la command

6/ modifier le nom de la machine qui héberge mongodb dans le fichier default_controller.py

On avait :

```
client = MongoClient('localhost', 27017)
```

on doit avoir :

```
client = MongoClient(db, 27017)
```

remarque : « db » est le nom du service que l'on a indiqué dans le fichier docker-compose.yml

7/ démarrer notre projet

sudo docker-compose up

on pourra ajouter l'option –d si on souhaite que notre projet fonctionne en arriere-plan (mode « detached »)

8/ avec son navigateur web, se connecter sur http://@ip-flottante-de-ma-vm:8080/api/v1/ui

9/ on observera la taille de l'image de notre micro-service : environ 80 Mb seulement ! facile à déployer sur n'importe quelle machine disposant de docker/docker-compose.

IMPORTANT :

Si on supprime le container, les données disparaissent aussi car nous n'avons pas utilisé de volume « externe aux containers » pour stocker les données mongodb

Les commandes pour stopper/supprimer les containers et les images

sudo docker stop $(sudo docker ps -a -q)

sudo docker rm $(sudo docker ps -a -q)

sudo docker rmi $(sudo docker images -q)