Vérification de l'état de santé des applications à l'aide des tests unitaires et l'instruction HEALTHCHECK

Objectif : L'objectif de cette manipulation est de contrôler la bonne exécution de vos applications en utilisant les tests unitaires avec python et l'instruction HEALTHCHECK.

1. Les tests unitaitres

Exemple 1 : test le bon fonctionnement d'un conteneur basé sur la servlet tomcat

- Création de conteneur basé sur la servlet tomcat

```
docker run -d --name servletTomcat -p 8080:8080  tomcat:8.0.20-jre8
```

- Préparer le script python avec les tests unitaires, voici un exemple :

```
import docker
import os
import unittest

class TestStringMethods(unittest.TestCase):

    def test_is_running(self):
        client = docker.from_env()
        container_name = "servletTomcat"
        RUNNING = 'running'
        container = client.containers.get(container_name)
        container_state = container.attrs['State']
        bool_state = container_state['Status'] == RUNNING
        self.assertTrue (bool_state, "Running failure!")

if __name__ == '__main__':
    unittest.main()
```

- Préparer l'image à l'aide de Dockerfile pour empaqueter le script de test dans un conteneur python

```
FROM alpine:latest

RUN apk add python3 \
    && apk add py3-pip \
    && pip install docker \
    && pip install unittest2

COPY conftest.py .

CMD ["conftest.py"]
ENTRYPOINT ["python3"]
```

- Construire l'image à partir du fichier Dockerfile préparé

```
docker build -t testctomcat .
```

- Créer le conteneur de test

```
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock --name
testctomcat testctomcat
```

- Voici un aperçu de l'exécution de test

```
[root@localhost testunit]# docker run --rm -v /var/run/docker.sock:/var/run/docker.sock --name testtomcat testctomcat
.
----------------------------------------------------------------------
Ran 1 test in 0.032s

OK
[root@localhost testunit]#
```

Exercice d'application :

1. Importez le référentiel suivant : https://gitlab.com/meddeb/docker-testunit
2. Adaptez le script de test pour vérifier le bon lancement des deux conteneurs "web" et "php"

2. L'instruction HEALTHCHECK

Le moteur Docker, à partir de la version 1.12, fournit un moyen de définir des commandes personnalisées en tant que vérifications de l'état.

Une vérification de l'état personnalisée est spécifiée dans un Dockerfile à l'aide de la directive HEALTHCHECK.

Selon la documentation officielle de Docker, ces codes de sortie peuvent être:

- 0: succès - le conteneur est sain et prêt à être utilisé
- 1: défectueux - le conteneur ne fonctionne pas correctement
- 2: réservé - ne pas utiliser ce code de sortie"

Préparation de l'image - Dockerfile :

```
FROM nginx:1.17.7
RUN apt-get update && apt-get install -y wget
HEALTHCHECK CMD wget -q --method=HEAD localhost/system-status.txt
```

⇨ La routine HEALTHCHECK été défini.
⇨ La vérification de l'état comprend un appel, à l'aide wget, pour récupérer l'URL spécifique du fichier personnalisé system-status.txt.

⇨ Dans cet exemple on suppose que le bon fonctionnement de l'application exige la présence du fichier system-status.txt (healthy), sinon il a eu un problème (unhealthy)

- Tester le bilan de santé : En utilisant le Dockerfile créé au ci-dessus, créer une nouvelle image et exécuter le conteneur.

```
[root@localhost health]# docker build -t docker-health .
Sending build context to Docker daemon  2.048kB
Step 1/3 : FROM nginx:1.17.7
 ---> c7460dfcab50
Step 2/3 : RUN apt-get update && apt-get install -y wget
 ---> Using cache
 ---> 36cce44ee5fa
Step 3/3 : HEALTHCHECK CMD wget -q --method=HEAD localhost/system-status.txt
 ---> Using cache
 ---> 2b4dd34919b5
Successfully built 2b4dd34919b5
Successfully tagged docker-health:latest
[root@localhost health]# docker run --rm --name c-health -p 8080:80 docker-health
```

- Vérification de l'état à l'aide du docker ps

```
[root@localhost health]# docker run -d -p 8081:80 --name webNginx docker-health
a2a4ce59573247e668d64e86d6522dede62f84af3c577a1546a10ec78284e8f1
[root@localhost health]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS                            PORTS                                   NAMES
a2a4ce595732   docker-health   "nginx -g 'daemon of…"   7 seconds ago   Up 4 seconds (health: starting)   0.0.0.0:8081->80/tcp, :::8081->80/tcp   webNginx
[root@localhost health]#
```

⇨ On constate qu'avec l'état du conteneur (Up 4 seconds), il y a l'ajout d'une indication sur la santé du conteneur (**health : strating**).

⇨ Docker engine essaye continuellement de vérifier l'état de santé du conteneur à l'aide de la commande wget pour récupérer le fichier "system-status.txt", après un laps de temps il mettra à jour l'état

```
[root@localhost health]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS                            PORTS                                   NAMES
4234dec782b3   docker-health   "nginx -g 'daemon of…"   About a minute ago   Up About a minute (unhealthy)     0.0.0.0:8081->80/tcp, :::8081->80/tcp   webNginx
[root@localhost health]#
```

- Injectons d'un processus dans le conteneur pour forcer la création du fichier "system-status.txt"

```
docker exec webNginx sh -c 'echo OK > /usr/share/nginx/html/system-status.txt'
```

- Observation du comportement du docker engine à l'aide du docker ps

```
[root@localhost health]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                    PORTS                                   NAMES
4234dec782b3   docker-health   "nginx -g 'daemon of…"   13 minutes ago   Up 13 minutes (healthy)   0.0.0.0:8081->80/tcp, :::8081->80/tcp   webNginx
[root@localhost health]#
```

- Modifications de l'instruction HEALTHCHECK pour spécifier l'intervalle de vérification et le nombre de tentatives de test
  - --interval=DURATION (default: 30s) La vérification de l'état s'exécutera d'abord à "interval" secondes après le démarrage du conteneur, puis à nouveau à "interval" secondes après la fin de chaque vérification précédente.

- --timeout=DURATION (default : 30s) Si une seule exécution de la vérification prend plus de secondes que le "Timeout", la vérification est considérée comme ayant échoué.
- --retries=N (default : 3) nombre de tentatives pour décider que le conteneur soit considéré comme défectueux.
- --start-period=DURATION (default : 0s)

Exemple , pour vérifier toutes les cinq minutes environ qu'un serveur Web est en mesure de diffuser la page principale du site pendant trois secondes:

```
HEALTHCHECK --interval=5m --timeout=3s CMD curl -f http://localhost || exit 1
```

- Dockerfile modifié

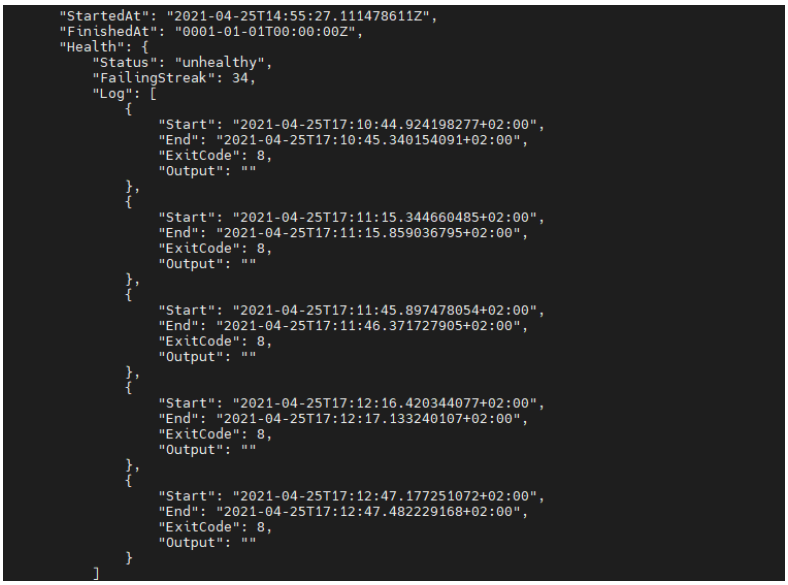
```
FROM nginx:1.17.7

RUN apt-get update && apt-get install -y wget

HEALTHCHECK --interval=5s --timeout=3s --retries=2  CMD wget -q --method=HEAD
localhost/system-status.txt || exit 1
```

⇨ La commande exit 1 pour personnaliser le code de retour dans le cas d'échec

- Observation du comportement du docker engine à l'aide du docker inspect



Exercice d'application :

- Créer un Docker file pour empaqueter l'application Flask
- Ajoutez le contrôle de santé de l'application à l'aide de l'utilitaire curl
- Créer un conteneur et vérifier l'état de santé du conteneur créé