*Manipulations # 0 - INTRODUCTION - COMMENT SUPPRIMER DES CONTENEURS ET DES IMAGES*

*Manipulations # 1 - COMMANDES DE BASE 1 - COMMENT SUPPRIMER DES CONTENEURS ET DES IMAGES*

*Manipulations # 2 - CRÉER UN SERVEUR AVEC PYTHON + FLASK (DÉMO 1 -****DOCKERFILE****)*

*Manipulations # 3 - CRÉER UN SERVEUR AVEC ALMA + APACHE (DÉMO 2 -****DOCKERFILE****)*

*Manipulations # 4 - COMMANDES DE BASE 2 - SUPPRIMER DES IMAGES ET DES CONTAINERS && CRÉER UNE IMAGE NGINX*

*Manipulations # 5 - CRÉER UN SERVEUR AVEC NGINX && CHANGER LES PORTS INTERNES DE 80 à 90 (****DÉMO 3-DOCKERFILE****)*

*Manipulations # 6 - COMMANDES DE BASE 2 - SUPPRIMER DES CONTENEURS ET DES IMAGES*

*Manipulations # 7 - DEVOIR À LA MAISON (DÉMO 4 - SANS* ***DOCKERFILE****)*

*Manipulations # 8 - CORRECTION DEVOIR À LA MAISON*

*Manipulations # 9 - CORRECTION PARTIE 1 && -it && Créer un container avec plusieurs ports exposés && Créer une image à partir du container*

**Manipulations # 0 - *COMMENT SUPPRIMER DES CONTENEURS ET DES IMAGES***

*Supprimer les conteneurs*

docker ps

docker ps -a

docker ps -a -q

docker stop $(docker ps -a -q)

docker rm $(docker ps -a -q)

*Supprimer les images*

docker images

docker images -q

docker rmi $(docker images -q)

*Manipulations # 1 - COMMANDES DE BASE 1*

*Démo 1*

docker pull centos

docker images

docker run -d --name c1 nginx

docker images

docker ps

docker ps -a

docker stop c1

docker rm c1

docker ps

docker ps -a

docker images

*Manipulations # 2 - CRÉER UN SERVEUR AVEC PYTHON + FLASK (DÉMO 1 -DOCKERFILE)*

*Démo Dockerfile numéro #1 (PYTHON)*

<https://github.com/hrhouma/kubernetes1>

<https://github.com/hrhouma/kubernetes1/tree/kuberntes-v1/docker>

Dockerfile

FROM python:3.7

COPY myapp.py /tmp

RUN pip install flask

EXPOSE 8080

CMD ["python", "/tmp/myapp.py"]

myapp.py

from flask import Flask

import os

app = Flask(\_\_name\_\_)

@app.route('/')

def hello():

return('Hello from container..This is my first docker container\n')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=8080, debug=True)

Étapes:

mkdir exemple1

cd exemple1

nano Dockerfile

copier/coller le contenu de Dockerfile

ls

nano myapp.py

copier/coller le contenu de myapp.py

ls

docker build -t **monkiki .**

docker ps

docker run -d --name **c1monkiki** -p 9000:8080 monkiki

⇒ *monkiki* : Le nom de l’image que je vais créer

⇒ *c1monkiki* : Le nom de mon conteneur

*Manipulations # 3 - CRÉER UN SERVEUR AVEC ALMA + APACHE*

*(DÉMO 2 -DOCKERFILE)*

*Démo Dockerfile numéro #2*

*Démo Dockerfile numéro #2 (ALMA LINUX)- Voir annexe - Exercice 2 - Dockerfile*

Consultez ce lien sur github :

ou aller au dossier UTILE et télécharger les instructions depuis Exercice1- Dockerfile.txt

<https://drive.google.com/drive/folders/1EVIHqsCGu5lw-rpNCdcVPdlANrDqz75K?usp=sharing>

1-cd /home/hrehouma

2-mkdir exemple2

3-cd exemple2

4-nano Dockerfile

5-tu copies et tu colles le contenu de Dockerfile

6-nano script.sh

7-tu copies et tu colles le contenu de script.sh

8-nano index.html

9-tu copies et tu colles le contenu de index.html

10- tu t’assures que tous les fichiers sont dans le dossier exemple1 en exécutant ls (Dockerfile, script.sh et index.html)

11-exécute la commande suivante :

docker build -t monsiteapache . (le point est important)

12- exécute la commande suivante :

docker run -d -p 8080:80 --name siteapache monsiteapache

13- Allez dans le navigateur et tu vérifies si nous avons la page web

<http://localhost:8080>

*Supprimer les conteneurs*

docker ps

docker ps -a

docker ps -a -q

docker stop $(docker ps -a -q)

docker rm $(docker ps -a -q)

*Supprimer les images*

docker images

docker images -q

docker rmi $(docker images -q)

*Manipulations # 4 - COMMANDES DE BASE 2 -*

*SUPPRIMER DES IMAGES ET DES CONTAINERS*

*&& CRÉER UNE IMAGE NGINX*

docker run -it -d -p 8080:80 --name c1 nginx (testez localhost:8080)

docker run -it -d -p 8081:80 --name c2 nginx (testez localhost:8081)

docker run -it -d -p 8082:80 --name c3 nginx (testez localhost:8082)

docker ps

docker ps -q (donne les ids)

docker stop a69983c1a0d3 ou docker stop a69

docker stop 262c9da97eb4

docker stop 66793fb59d9

docker ps

docker ps -a

docker start a69983c1a0d3

docker start 262c9da97eb4

docker start 66793fb59d9

docker ps

docker stop c1

docker stop c2

docker stop c3

docker ps

docker start c1

docker start c2

docker start c3

docker rm a69983c1a0d3 ou docker rm c1

Si on veut supprimer tous les containers :

docker **stop** $(docker ps -a -q) ***ensuite*** docker **rm** $(docker ps -a -q)

Astuce : Exécutez cette commande en premier pour comprendre la signification de -q

docker ps -a -q **(ça donne les ids)**

⇒ ça ne fonctionne pas !!!! En effet, les containers doivent être stoppés avant de les supprimer

docker stop $(docker ps -a -q)

docker rm $(docker ps -a -q)

Est-ce que nous avons tout supprimé ? ⇒ la réponse est non ! Il reste les images

docker images

docker images -q

docker rmi $(docker images -q)

docker images

docker run -it -d -p 8080:80 --name c1 nginx

*Manipulations # 5 - CRÉER UN SERVEUR AVEC NGINX*

*&& CHANGER LES PORTS INTERNES DE 80 à 90*

*(DÉMO 3 -DOCKERFILE)*

*Démo Dockerfile numéro #3*

Consultez ce lien sur github :

mkdir exemple3

cd exemple3

nano Dockerfile

*Contenu du Dockerfile*

# Utiliser une image Nginx

FROM nginx

# Copier le fichier HTML personnalisé dans le conteneur

COPY index.html /usr/share/nginx/html/index.html

nano index.html

*Contenu du index.html*

<html> Hello from nginx </html>

docker build -t img\_site\_nginx .

docker images

docker run -d -p 9001:90 --name c1\_site\_nginx img\_site\_nginx

docker ps

Essayez <http://localhost:9001> (ça ne fonctionne pas, le port interne est à 80 !!!)

docker exec -it c1\_site\_nginx **bash**

ls

cd /etc/

cd nginx

**nano** /etc/nginx/conf.d/default.conf (https://unix.stackexchange.com/questions/355911/trouble-installing-nano)

apt-get update

apt-get install **nano**

nano /etc/nginx/conf.d/default.conf

Changez le port 80 à 90

server {

listen 90;

listen [::]:90;

server\_name localhost;

nginx -s reload (https://docs.nginx.com/nginx/admin-guide/basic-functionality/runtime-control/)

exit

Essayez <http://localhost:9001>

*Manipulations # 6 - SUPPRIMER DES CONTENEURS ET DES IMAGES*

*Supprimer les conteneurs*

docker ps

docker ps -a

docker ps -a -q

docker stop $(docker ps -a -q)

docker rm $(docker ps -a -q)

docker ps -a

*Supprimer les images*

docker images

docker images -q

docker rmi $(docker images -q)

docker images

*Manipulations # 7 - COMMANDES DE BASE 2*

docker --help

docker ps --help

docker build --help

*Changer l’image :*

Reprenons l’exemple de **l'exercice 1** ci-haut (*Démo Dockerfile numéro #1 (PYTHON))* et construisons un container exercice 1

nano Dockerfile

(**Exemple 1** - voir page 4 *Manipulations # 2 - CRÉER UN SERVEUR AVEC PYTHON + FLASK (DÉMO 1 -DOCKERFILE)*)

docker build -t myimage-13oct:1.0

docker images

docker run -d --name c1 -p 8081:8080 **myimage-13oct:1.0**

docker ps

*D’autres conteneurs :*

docker run -d --name c2 -p 8082:8080 **myimage-13oct:1.0**

docker run -d --name c3 -p 8083:8080 **myimage-13oct:1.0**

docker run -d --name c4 -p 8084:8080 **myimage-13oct:1.0**

docker run -d --name c5 -p 8085:8080 **myimage-13oct:1.0**

docker ps

docker images

docker exec -it c1 bash

cd /tmp

ls (on va trouver myapp.py)

exit

*Manipulations # 8 - DEVOIR À LA MAISON*

*DEVOIR*

*Faites le devoir avec deux méthodes (1-sans Dockerfile et 2- avec Dockerfile)*

* Créez un conteneur avec une image Ubuntu (avec le nom de conteneur c1)
* Connectez-vous au conteneur — docker exec -it c1 bash
* Mettez à jour le dépôt (apt update) dans le conteneur
* Installez apache2 dans le conteneur (apt install apache2).

Note :

* Créez un fichier index.html sur la machine hôte avec le contenu ci-dessous :

<h1>Hello world.. running in apache2</h1>

* Copiez ce fichier de l'hôte au conteneur à l'emplacement /var/www/html/index.html
* Vérifiez si vous pouvez accéder à cette page depuis un navigateur

*Manipulations # 9 - CORRECTION PARTIE 1*

*&& -it*

*&& Créer un container avec plusieurs ports exposés*

*&& Créer une image à partir du container*

***Partie 9-1***

docker run -d --name myapache -p 81:80 ubuntu

Si vous suivez ces commandes, vous verrez que le conteneur Apache ne fonctionne pas (en utilisant 'docker ps'). Mais si vous utilisez ('docker ps -a'), vous verrez qu'il a été lancé mais s'est arrêté automatiquement (car il n'y a rien à l'intérieur du conteneur qui fonctionne).

docker rm myapache

docker run -d **-it** --name myapache -p 81:80 ubuntu

docker ps

docker exec -it myapache bash

cat /etc/os-release

exit

apt update

apt install apache2

apt install curl

curl localhost:80

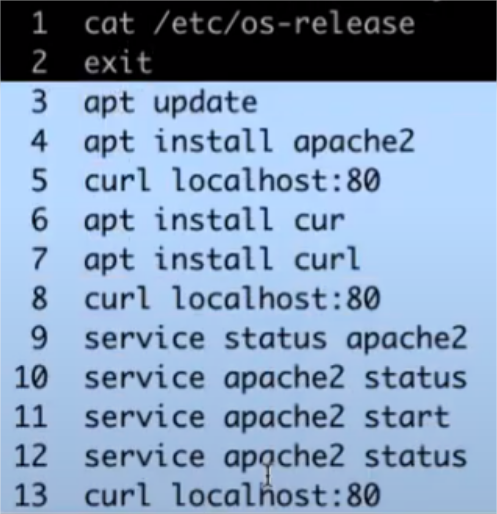
service apache2 status

service apache2 start

curl localhost:80

exit

**RÉSUMÉ :**



***Partie 9-2***

mkdir correctiondevoir

cd correctiondevoir

vi index.html

<h1> Hello world ..running in apache 2 </h1>

Quitter le fichier escape ensuite :wq!

Essayez la commande top

Nous allons copier le fichier index.html ci-haut

cp index.html myapache:/var/www/html/index.html

curl localhost:81

docker exec -it myapache bash

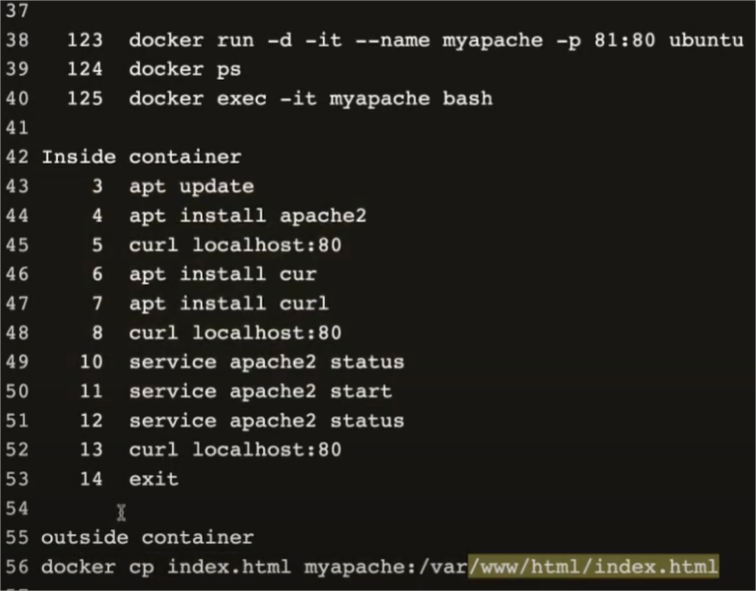
curl localhost:80

exit

curl localhost:81

Allez au navigateur et saisir : *localhost:81*

**RÉSUMÉ :**



***Partie 9-3***

docker stop myapache

docker rm myapache

Créer un container avec plusieurs ports exposés

docker run -d -it --name myapache -p 81:80 -p 82:80 ubuntu

docker ps

docker exec -it myapache bash

apt update

apt install curl

apt install apache2

service apache2 start

exit

curl localhost:81 (ou copier localhost:81 dans le navigateur)

docker exec -it myapache bash

echo “Hello test 123” > /var/www/html/index.html

exit (je quitte le container/intérieur = port 80)

curl localhost:81 (je suis à l’extérieur)

curl localhost:82

Créer une image à partir du container

Pour créer une image à partir du container en cours d'exécution ( ⇒ create an image from a running container)

docker commit myapache apache:1.0

docker images