Bachelor	
AIS	

Date :

TP Docker

Luczak.P

Durée : 3H

Introduction à Docker

Pré-requis : Avoir vu le cours sur docker et docker-compose

1 / Installation :

Installer la version stable de docker correspond à votre environnement en suivant les instructions données sur le site :

https://docs.docker.com/get-docker/

Assurez-vous que l'installation s'est bien déroulée et que docker-compose est également installé.

`docker -v` : vous obtiendrez la version de docker
`docker-compose -v` : vous obtiendrez la version de docker-compose

Lancer votre premier conteneur : `docker run hello-world`

Si tout s'est bien déroulé ….passons à la suite

2 / Images et containers:

Le docker hub contient des milliers d'images vous permettant de lancer votre container adapté à vos besoins

Saisir la commande : `docker run -p 8080:80 -d nginx`

`run` : lance le service
`-p 8080:80` : mappe le port 8080 de la machine host vers le port 80 du container
`-d` : mode détaché
`nginx` : image

Dans votre navigateur entrer : `localhost:8080`

Donner la fonction des commandes :

`docker ps`

`docker ps -a`

`docker images`

Comment arrêter un container ?
Comment supprimer un container ?
Comment supprimer une image ?

Télécharger l'image de Apache :

Lancer le container (en mode détaché) avec l'image apache en mappant le port 8081 du host vers le port 80 du container:

3 / Création de votre image :

Il vous est possible de créer votre propre image correspondant à votre besoin,pour cela nous allons créer un fichier nommé "Dockerfile"contenant l'ensemble de la recette décrivant l'image Docker dont vous avez besoin pour votre projet.

Exemple : Création de votre image btssio contenant un debian minimaliste et les outils vim,git,mc,tree et htop

Créer un répertoire lab0 puis saisir dans celui-ci le fichier dockerfile ci-dessous :

```
# -------------- DÉBUT COUCHE OS -------------------
FROM debian:stable-slim
# -------------- FIN COUCHE OS ---------------------

# MÉTADONNÉES DE L'IMAGE
MAINTAINER BTS SIO2 SISR

# -------------- DÉBUT COUCHE Installation  ---------------
RUN apt-get update \
&& apt-get install -y vim git htop mc \
&& apt-get clean \
&& rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
# -------------- FIN COUCHE Installation -----------------
```

Rq:La dernière ligne permet d'effectuer un nettoyage afin d'alléger l'image.

Quelques commandes :

FROM : qui vous permet de définir l'image source

RUN : qui vous permet d'exécuter des commandes dans votre conteneur

ADD : qui vous permet d'ajouter des fichiers dans votre conteneur

WORKDIR : qui vous permet de définir votre répertoire de travail

EXPOSE : qui permet de définir les ports d'écoute par défaut

VOLUME : qui permet de définir les volumes utilisables

CMD : qui permet de définir la commande par défaut lors de l'exécution de vos conteneurs Docker

Création de votre image : `docker build -t btssio:v1.0 .`

Rq : Ne pas oublier le point à la fin !!

Vérifier la présence de votre image et relever sa taille :

Lancer votre container à partir de votre image : `docker run -tid --name sisr btssio:v1.0`

Entrer dans le container puis vérifier le bon fonctionnement de vos applis : `docker exec -ti sisr bash`

Proposer une nouvelle image beaucoup plus légère permettant d'avoir les mêmes fonctionnalités.
(Indice : France en Formule1)

Rappel: Effacer vos containers et images après chaque fin de TP

3 / Orchestrer des containers :

Docker Compose va vous permettre d'orchestrer vos conteneurs, et ainsi de simplifier vos déploiements sur de multiples environnements. Docker Compose est un outil écrit en Python qui permet de décrire, dans un fichier YAML, plusieurs conteneurs comme un ensemble de services.

Cas d'usage : Votre équipe de dev vous demande de déployer et tester l'appli nécessitant les fichiers :

app.py et requirements.txt .

Ce dernier contient les applications qui seront installée via la commande pip. (ici redis et Flask)

L'application écrite en python utilise le framework Flask ainsi que la bdd redis ,elle affiche une page avec votre hostname ,un nom provenant d'une variable d'environnement et un compteur du nombre de visites .

Nous allons tout d'abord créer une image personnalisée (monimage) comprenant l'applicatif python des développeurs.

Créer un répertoire projet puis saisir le dockerfile ci-dessous :

```
FROM python:2.7-slim
WORKDIR /app
COPY . /app
RUN pip install -r requirements.txt
EXPOSE 80
ENV NOM BTS SIO2 SISR
CMD ["python", "app.py"]
```

A ce stade si vous lancez le container après avoir créer l'image ,vous aurez bien la page qui s'affichera mais sans le compteur de visites car redis n'est pas actif.
Pour palier à ce problème ,nous allons lancer un second container « relié » au premier afin qu'ils puissent fonctionner de concert .

Pour ce faire nous faisons appel à Docker-compose en créant un fichier docker-compose.yml

Attention de bien respecter l'indentation !!!

```
version: "3"
services:
  monapp:
    image: monimage
    depends_on:
      - redis
    ports:
      - "80:80"
    networks:
      - monreseau
    environment:
      - NOM=les SISR
  redis:
    image: redis
    networks:
      - monreseau

networks:
  monreseau:
```

Il faut ensuite lancer docker-compose : `docker-compose up`

Vous pouvez à présent afficher la page sur votre navigateur ,le compteur s'incrémentera à chaque actualisation de la page .

Le hostname affiché est celui du container ,effectuer les modifications afin qu'il soit : SaintLuc