Docker – Python Data

Ce document est un rapport contenant toutes les informations liées au TP de Python Data. Il donnera toutes les finormations nécéssaires à la comprehension de Docker.

1. Introduction

Docker est un outil open source permettant de faciliter le lancement d'applications via la création de conteneurs. Ainsi, il permet au développeur de pouvoir exécuter des applications depuis n'importe quel serveur sans se soucier du système d'exploitation de la machine hôte permettant ainsi une plus grand flexibilité et portabilité vise à vis d'une application. Le processus de conteneurisation va créer des conteneurs qui seront construits sur des capacités du noyau linux.

L'objectif d'un conteneur est le même que pour un serveur dédié virtuel : héberger des services sur un même serveur physique tout en les isolant les uns des autres. Un conteneur est cependant moins figé qu'une machine virtuelle en matière de taille de disque et de ressources allouées.Un conteneur permet d'isoler chaque service : le serveur web, la base de données, une application peuvent être exécutés de façon indépendante dans leur conteneur dédié, contenant uniquement les dépendances nécessaires.

2. Premier Conteneur docker

Tout d'abord, pour installer Docker, il suffit de se rendre sur le lien suivant :

https://www.docker.com/get-started

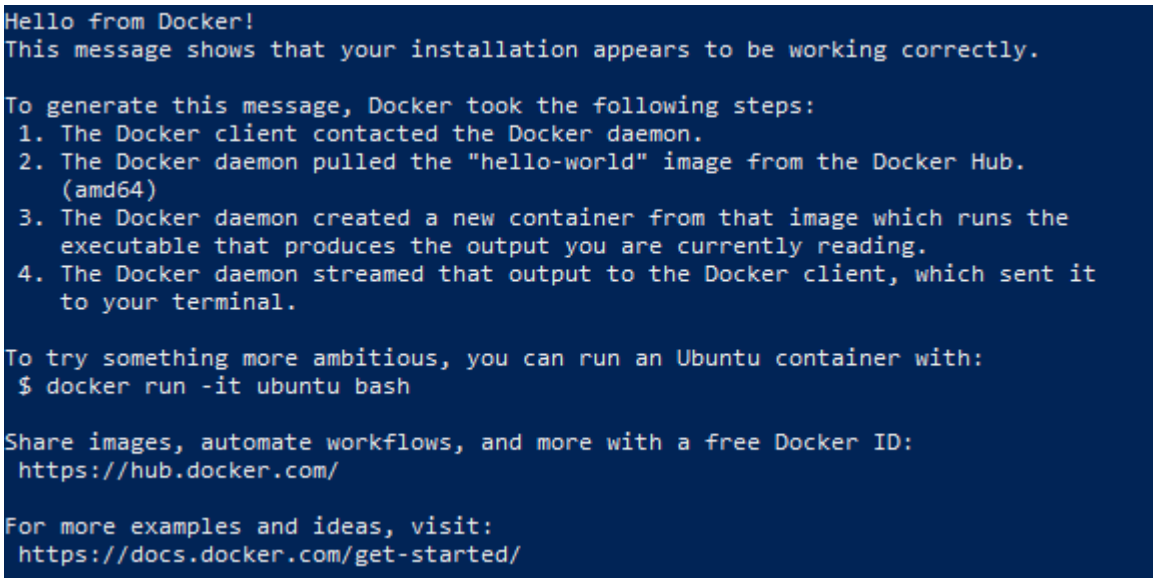
Une fois docker installé, nous allons exécuter notre premier container Windows. Pour cela, il suffit juste d'effectuer la commande suivante dans le répertoire de votre choix :

docker run -it ubuntu bash

Rajouter à la commande :

--detach (-d) pour pas que le container s'arrête.

Le résultat suivant est censé apparaitre :



Annexe 1 : Résultat affichage docker run

On peut ensuite afficher l'image via la commande suivante :

docker images

3. Data engineering sandbox

Dans cette partie, nous allons créer un DockerFile. Celui-ci va contenir toutes les informations nécessaires à la création de notre environnement docker. Pour cela, on créer un dossier workspace dans lequel on aura notre DockerFile. On va insérer ce même DockerFile le contenu suivant :

```
FROM ubuntu

WORKDIR /usr/src/workspace
COPY . .
RUN apt-get -y update

RUN apt-get -y install python3 \
&& apt-get -y install python3-pip \
&& apt-get install -y git \
&& apt-get install -y vim \
&& pip3 install -r requirements.txt \
&& ipykernel install --user \
&& git config --global user.name alphab2000x \
&& git config --global user.email alphab200x@hotmail.fr \
&& git clone https://github.com/alphab2000x/docker.git

EXPOSE 8000
CMD ["jupyter", "notebook", "--port=8000", "--no-browser", "--ip=0.0.0.0", "--allow-root"]
```

On va créer le fichier requirement.txt et y insérer le contenu suivant dedans :

```
numpy==1.16.0
pandas
seaborn
flask
jupyter
scikit-learn
sklearn
tensorflow
```

Cela correspond aux librairies que l'on veut importer pour notre jupyter notebook.

Enfin, on va créer notre docker compose. Pour cela, ont créé un fichier .yml que l'on nommera « **docker-compose** » à la racine, fichier qui va contenir les lignes suivantes :

```
version: "3.2"
services:
   python:
      build: python
      ports:
        - "8000:8000"
      networks:
        node_net:
          ipv4_address: 172.28.1.4

networks:
  node_net:
    ipam:
      driver: default
      config:
       - subnet: 172.28.0.0/16
```

Docker Compose est un outil qui permet de décrire (dans un fichier YAML) et gérer (en ligne de commande) plusieurs conteneurs comme un ensemble de services interconnectés. Si nous travaillons sur une application Rails, je vais par exemple décrire un ensemble composé de 3 conteneurs :

- un conteneur PostgreSQL
- un conteneur Redis
- un conteneur pour le code de mon application

La version de Docker Compose doit être spécifiée en haut du fichier YAML. Comme indiqué dans la commande suivante, la version "3.2"devrait fonctionner pour les versions stables les plus récentes de Docker.

Ne pas préciser la version pourrait empêcher certain composant de fonctionner et générer des erreurs.

Pour lancer notre docker compose, on va exécuter la commande suivante :

docker-compose up -d

Une suite d'instruction va se lancer, une image va se créer et sera alors visible via la commande suivante :

docker images

Puis pour exécuter jupyter notebook :

docker-compose up --build

Il nous suffira juste d'ouvrir notre jupyter via l'URL :

http://127.0.0.1:8000/?token=91ba6672758020b63ef10bc7fbab87aa30bdba0bdd0a046e

L'URL avec le token sont disponible suite à l'exécution de la commande précédente.

4. Commande

Commande	Description
docker ps	Informations concernant les processus docker
Docker images	Informations concernant l'image docker
docker stop $(docker ps -a -q)	Arrêter tout les conteneurs
docker rm $(docker ps -a -q)	Supprimer tous les conteneurs
docker stop NomDuConteneur	Pour arrêter le conteneur
docker rm NomDuConteneur	Supprimer le conteneur
docker rmi NomDuConteneur	Supprimer l'image
docker build -t NomImageQueJeVeux .	construire l'image
docker run -p 3306:3306(ou autre) NomImageQueJeVeux.	Pour exécuter l'image
docker volume ls	Affiche la liste des volumes
docker volume rm monVolume	Supprime le volume
docker network ls	Affiche la liste des network
docker network prune	
docker exec -it monContainer bash	Exécuter mon container et me met dans un terminal