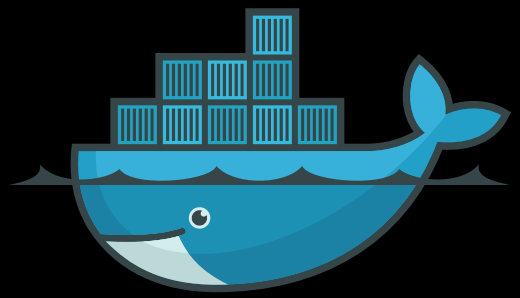
TP

—



docker

KALETA Maxime

BTS SIO

ensemble SAINT LUC Cambrai

Les chemins de la réussite

Sommaire

- Installation de docker
- Création d'une image
- Lancer une image
- Docker compose

KALETA Maxime

Installation de docker

Docker c'est quoi ?

Docker vous permet de concevoir, de tester et de déployer rapidement des applications sur vos machines avec simplicité. Cette solution permet d'optimiser votre temps de déploiement d'application

Ajouter la clé GPG de docker:

#sudo apt-get update

#sudo apt-get install ca-certificates curl

#sudo install -m 0755 -d /etc/apt/keyrings

#sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc

#sudo chmod a+r /etc/apt/keyrings/docker.asc

Ajouter le dépôt aux sources d'Apt :

#echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/debian \ $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \

#sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

#sudo apt-get update

KALETA Maxime

Installation de docker

Installation de la dernière version :

#sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Pour tester :

#sudo docker run hello-world

Pour un script d'installation je vous invite à consulter mon Github à l'adresse suivante :

https://github.com/d4rtox/tpbilanb2/blob/main/install_docker.sh

KALETA Maxime

Création d'une image

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.

Commande de construction de l'image :

#docker build -t <nom de 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

KALETA Maxime

Lancer une image

Lancer le container :

```
#docker run --name (nom du container) -p 8081:80 -d
  (détaché) (image)
```

Commandes :

- Permet de montrer uniquement les container actif par defaut :

#docker ps

- Permet de voir les container actif :

#docker ps –a

- Permet de voir toute les images :

#docker image

- Arreter un container :

```
#docker stop <id du conteneur>
```

- Supprimer un container ?

```
#docker rm container_id_or_name1 container_id_or_name2
```

- Supprimer une image ?

```
#docker rmi Image Image
```

- Téléchargement de l'image apache :

```
#docker pull <nom du pacquet vu sur le site DockerHub>
```

KALETA Maxime

Docker compose

Docker Compose est un **outil destiné à définir et exécuter des applications Docker à plusieurs conteneurs**. Dans Compose, vous utilisez un fichier YAML pour configurer les services de votre application. Ensuite, vous créez et vous démarrez tous les services à partir de votre configuration en utilisant une seule commande.

Orchestrer des containers :

Création d'un répertoire nommé « projet » à la racine

Création d'un fichier texte « dockerfile » avec inscrit dedans :

```
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"]
```

KALETA Maxime

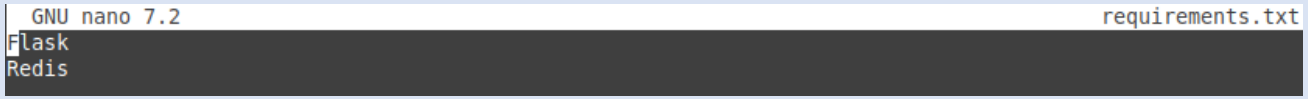
Docker compose

Importation des fichiers app.py et requirements.txt

App.py :

```
 GNU nano 7.2                                                    app.py
from flask import Flask
from redis import Redis, RedisError
import os
import socket

# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)

app = Flask(__name__)

@app.route("/")
def hello():
    try:
        visites = redis.incr("compteur")
    except RedisError:
        visites = "<i>Erreur de connection Redis, compteur desactive</i>"

    html = "<h3>Bonjour {nom}!</h3>" \
           "<b>Hostname:</b> {hostname}<br/>" \
           "<b>Visites:</b> {visites} <br/>" \
           "<p>La SIO Team !!</p>"
    return html.format(nom=os.getenv("NOM", "El Greco"), hostname=socket.gethostname(), visites=visites)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80)
```

requirement.txt :



```
 GNU nano 7.2                                                requirements.txt
Flask
Redis
```

Création de l'image :

```
docker build -t monimage .
```

Lancer l'image :

```
docker run --name dockerlie -p 8082:80 -d mon image
```

KALETA Maxime