UNIVERSITE DE SOUSSE Institut Supérieur d'Informatique et des Techniques de Communication Hammam Sousse		جـامعة سوسة المعهد العالي للإعلامية وتقنيات الاتصال بحمام سوسة

TP1 : Docker et DockerCompose — A.U. : 2020/2021

Matière : Administration et Sécurité des Réseaux
Enseignant : Salah Gontara
Classe : 2DNI

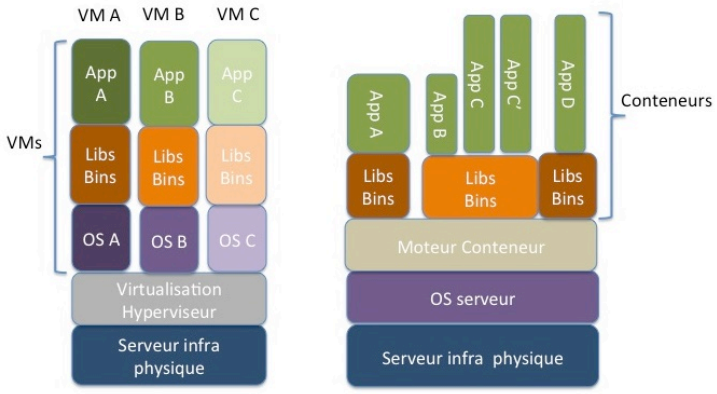
OBJECTIF(S):

- Comprendre le fonctionnement de base de Docker
- Interagir efficacement avec les containers
- Utiliser Docker-Compose pour la conteneurisation d'application
- Intégrer un projet sous Docker avec un framework tel que Django

OUTILS UTILISÉS :

Docker, les containers images : CentOS, Ubuntu, Django Project, Python, Postgres

Partie 1 : Comprendre Docker



Docker permet aux utilisateurs d'installer des applications dans des conteneurs de logiciels. Dans ce cas, les applications peuvent être isolées du système d'exploitation sur lequel elles s'exécutent. Dans les conteneurs Docker, chaque application est autonome. Par conséquent, la configuration et la suppression sont simples et n'ont que peu ou pas d'effet sur le système d'exploitation.

Comparaison VM et Docker

Machines Virtuelles	Docker Containers
Chaque VM a un système d'exploitation propre à elle.	Chaque conteneur peut partager des ressources du système d'exploitation hôte
Démarre en quelques minutes	Démarre en quelques secondes
Plus d'utilisation des ressources	Moins d'utilisation des ressources
Les machines virtuelles peuvent être facilement déplacées vers un nouveau système d'exploitation hôte.	Les conteneurs sont détruits et recréés plutôt que déplacés (le volume de données est sauvegardé).

1. Installez Docker Toolbox et assurez-vous de l'installation avec les commandes suivantes :
   sudo docker version
   sudo docker-compose version
   Notez les versions de Docker et Docker-compose

2. Commencez par un simple Hello World !
   sudo docker run hello-world

3. Donnez le nom du noyau sur lequel docker est installé.

4. Créez un réseau virtuel dédié pour les conteneurs à installer.
   Utilisez la commande : **sudo docker network create -d bridge --subnet 10.0.0.0/24 test_network**

Partie 2 : Interagir avec les containers

1. Listez les images importées avec la commande :
   sudo docker images

2. Démarrez CentOS et Ubuntu avec la commande :
   sudo docker run -it image_name bash

3. Confirmez les noms des systèmes en marche avec « os-release ».

4. Arrêtez les conteneurs et relancez-les avec le réseau virtuel dédié avec la commande :
   sudo docker run --network=[…] --ip=[…] -it image_name bash

5. Faites un ping entre les conteneurs pour confirmer l'appartenance au même réseau virtuel.

Partie 3 : Créer un projet sous Docker

1. Créez le dossier « **project** » dans votre répertoire utilisateur et positionnez-vous dessus avec la commande **cd**

2. Copiez le contenu du fichier « Dockerfile » :

```
FROM python:3
ENV PYTHONUNBUFFERED 1
RUN mkdir /code
WORKDIR /code
ADD requirements.txt /code/
RUN pip install -r requirements.txt
ADD . /code/
```

3. Copiez le contenu du fichier « requirements.txt »:

```
Django>=3.0,<4.0
psycopg2-binary>=2.8
```

4. Copiez le contenu du fichier « docker-compose.yml »:

```
version: "3.9"

services:
  db:
    image: postgres
    environment:
      - POSTGRES_DB=postgres
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    depends_on:
      - db
```

Partie 4 : Utiliser Docker-Compose

1. A la racine du projet, lancez la configuration avec docker-compose run
   sudo docker-compose run web django-admin.py startproject composeexample .

2. Changez les droits d'accès au dossier avec la commande suivate :
   sudo chown -R $USER:$USER .

3. Modifiez la permission de cette IP dans le fichier composeexample/settings.py:

```
ALLOWED_HOSTS = [*]
```

4. Editez la section « DATABASES » dans le fichier composeexample/settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'postgres',
        'USER': 'postgres',
        'PASSWORD': 'postgres',
        'HOST': 'db',
        'PORT': 5432,
    }
}
```

5. Démarrez le projet avec docker-compose :
   sudo docker-compose up

6. Accéder à Django:

   - Sous windows : **192.168.56.101:8000**
   - Sous linux : **localhost:8000**