Institut **S**upérieur d'**I**nformatique et de **T**echniques de **C**ommunication

H-Sousse

Administration et Sécurité des Réseaux

TP 1

Docker et Docker Compose

Réalisé Par

ELagas Amel

Classe : 2DNI G1

A.U :2020-2021

Plan

Introduction

A. Partie théorique

1. Qu'est-ce que Docker ?

2. Quelle différence avec la virtualisation ?

3. Quels sont les avantages de Docker ?

4. Quels sont les principaux services et technos Docker ?

B. Partie pratique

1. Comprendre Docker

2. Interagir avec les containers

3. Créer un projet sous Docker

4. Utiliser Docker-Compose

Conclusion

Introduction

Le terme « Docker » désigne plusieurs choses : le projet d'une communauté Open Source, les outils issus de ce projet Open Source, l'entreprise Docker Inc. qui constitue le principal soutien de ce projet, ainsi que les outils que l'entreprise prend officiellement en charge. Voici donc une rapide explication de cet état de fait.

- Le logiciel « Docker » est une technologie de conteneurisation qui permet la création et l'utilisation de conteneurs Linux.
- La communauté Open Source Docker travaille à l'amélioration de cette technologie disponible gratuitement pour tout le monde.
- L'entreprise Docker Inc. s'appuie sur le travail de la communauté Docker, sécurise sa technologie et partage ses avancées avec tous les utilisateurs. Elle prend ensuite en charge les technologies améliorées et sécurisées pour ses clients professionnels.

Avec la technologie Docker, nous pouvons traiter les conteneurs comme des machines virtuelles très légères et modulaires. En outre, ces conteneurs nous offrons une grande flexibilité : nous pouvons les créer, déployer, copier et déplacer d'un environnement à un autre, ce qui vous permet d'optimiser vos applications pour le cloud.

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

A.Partie théorique

1. Qu'est-ce que Docker ?

Docker permet d'embarquer une application dans un ou plusieurs containers logiciels qui pourra s'exécuter sur n'importe quel serveur machine, qu'il soit physique ou virtuel. Docker fonctionne sous Linux comme Windows Server. C'est une technologie qui a pour but de faciliter les déploiements d'application, et la gestion du dimensionnement de l'infrastructure sous-jacente. Elle est proposée par la société Docker, en partie en open source (sous licence Apache 2.0).

2. Quelle différence avec la virtualisation ?

La virtualisation permet, via un hyperviseur, de simuler une ou plusieurs machines physiques, et les exécuter sous forme de machines virtuelles (VM) sur un serveur ou un terminal.

Ces VM intègrent elles-mêmes un OS sur lequel des applications sont exécutées. Ce n'est pas le cas du container logiciel.

Le container fait en effet directement appel à l'OS de sa machine hôte pour réaliser ses appels système et exécuter ses applications, d'où son extrême légèreté. Dans le cas de Linux, les containers Docker exploitent un composant du noyau Linux baptisé LXC (ou Linux Container).

Au format Windows Server, ils s'adossent à une brique équivalente, appelée Windows Server Container.

Le moteur Docker normalise ces briques par le biais d'API dans l'optique d'exécuter les applications dans des containers standards, qui sont ensuite portables d'un serveur à l'autre.

3. Quels sont les avantages de Docker ?

Comme le container n'embarque pas d'OS à la différence de la machine virtuelle, il est plus léger que cette dernière. Il n'a pas besoin d'activer un second système d'exploitation pour exécuter ses applications.

Cela se traduit par un lancement beaucoup plus rapide. Pour parvenir au même résultat, les environnements de virtualisation ont besoin d'un pool de VM inactives provisionnées à l'avance. Avec Docker, nul besoin de pool, puisqu'un container est bootable en quelques secondes. IBM a publié en 2014 un comparatif de performance entre Docker et KVM.

Sa conclusion est sans appel : Docker égale ou excède les performances de l'hyperviseur et ce dans tous les cas testés. Pour Big Blue, la vélocité des containers Docker se rapproche même de celle des serveurs machines nus. En éliminant la couche de virtualisation, consommatrice en ressources, Docker permettrait de réduire la consommation de RAM de 4 à 30 fois.

Publiée en août 2017 par le département de recherche informatique de l'université de Lund en Suède, une autre étude compare cette fois la performance des containers à celle des machines virtuelles VMware. Elle aboutit également à une conclusion en faveur de Docker.

4. Quels sont les principaux services et technos Docker ?

Docker Compose

Docker Compose est un outil développé par Docker pour créer les architectures logicielles containérisées. Dans cette logique, chaque brique de l'application (code, base de données, serveur web...) sera hébergée par un container.

Cet outil repose sur le langage YAML (pour Yet Another Markup Language) pour décrire l'architecture. Une fois celle-ci codée dans un fichier YAML, l'ensemble des services applicatifs seront générés via une commande unique.

Partie 1 : Comprendre Docker

2/ Commencer par un simple Hello-World

```
ubuntu@ubuntuserver:~$ sudo docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

ubuntu@ubuntuserver:~$
```

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

Ubuntu

4/ Créez un réseau virtuel dédié pour les conteneurs à installer.

```
ubuntu@ubuntuserver:~$ sudo docker network create -d bridge --subnet 10.0.0.0/24 test__network
[sudo] password for ubuntu:
```

Partie 2 : Interagir avec les containers

1/ listez les images importées avec la commande :

```
ubuntu@ubuntuserver:~$ sudo docker network create -d bridge --subnet 10.0.0.0/24 test__network
[sudo] password for ubuntu:
Error response from daemon: Pool overlaps with other one on this address space
ubuntu@ubuntuserver:~$
ubuntu@ubuntuserver:~$ sudo docker images
REPOSITORY                                TAG        IMAGE ID       CREATED         SIZE
project_web                               latest     ad79b4a15971   36 hours ago    937MB
postgres                                  latest     1f0815c1cb6e   4 days ago      314MB
python                                    3          2a93c239d591   7 days ago      885MB
nginx                                     latest     298ec0e28760   7 days ago      133MB
mongo                                     latest     ca8e14b1fda6   3 weeks ago     493MB
ubuntu                                    latest     f63181f19b2f   3 weeks ago     72.9MB
k8s.gcr.io/kube-proxy                     v1.20.2    43154ddb57a8   4 weeks ago     118MB
k8s.gcr.io/kube-controller-manager        v1.20.2    a27166429d98   4 weeks ago     116MB
k8s.gcr.io/kube-apiserver                 v1.20.2    a8c2fdb8bf76   4 weeks ago     122MB
k8s.gcr.io/kube-scheduler                 v1.20.2    ed2c44fbdd78   4 weeks ago     46.4MB
centos                                    latest     300e315adb2f   2 months ago    209MB
gcr.io/k8s-minikube/storage-provisioner   v4         85069258b98a   2 months ago    29.7MB
k8s.gcr.io/etcd                           3.4.13-0   0369cf4303ff   5 months ago    253MB
k8s.gcr.io/coredns                        1.7.0      bfe3a36ebd25   8 months ago    45.2MB
k8s.gcr.io/pause                          3.2        80d28bedfe5d   12 months ago   683kB
hello-world                               latest     bf756fb1ae65   13 months ago   13.3kB
ubuntu@ubuntuserver:~$
```

2/ Démarrez CentOS et Ubuntu :

```
ubuntu@ubuntuserver:~$ sudo docker run  -it ubuntu bash
root@a46b86df9db4:/# _
```

```
ubuntu@ubuntuserver:~$ sudo docker run -it centos bash

[root@25344f20919e /]#
```

3/ Confirmez les noms des systémes en marche avec **« os-release » :**

```
ubuntu@ubuntuserver:~$ sudo docker run  -it ubuntu bash
root@a46b86df9db4:/# cat /etc/os-release
NAME="Ubuntu"
VERSION="20.04.1 LTS (Focal Fossa)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.04.1 LTS"
VERSION_ID="20.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=focal
UBUNTU_CODENAME=focal
root@a46b86df9db4:/# _
```

4/ Arrêtez les conteneurs et relancez-les avec le réseau virtuel dédie avec la commande :

```
ubuntu@ubuntuserver:~$ sudo docker run --network=test_network --ip=10.0.0.8 -it ubuntu bash
root@3fe85629cb61:/#
```

```
ubuntu@ubuntuserver:~$ sudo docker run --network=test_network --ip=10.0.0.58 -it centos
[root@887a15a410a5 /]#
```

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

```
ubuntu@ubuntuserver:~$ ping 10.0.0.58 10.0.0.18
PING 10.0.0.18 (10.0.0.18) 56(124) bytes of data.
```

Partie 3 : Créer un projet sous Docker

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

```
ubuntu@ubuntuserver:/usr$ sudo su
[sudo] password for ubuntu:
root@ubuntuserver:/usr# mkdir project
root@ubuntuserver:/usr# cd project
```

2. Copiez le contenu du fichier « Dockerfile » :

➔ On va utiliser :

FileZilla est un client FTP (File Transfert Protocol).

Ce logiciel vous permet de charger ou télécharger les fichiers sur un serveur distant comme par exemple les éléments de votre site web chez vous ou depuis votre hébergeur.

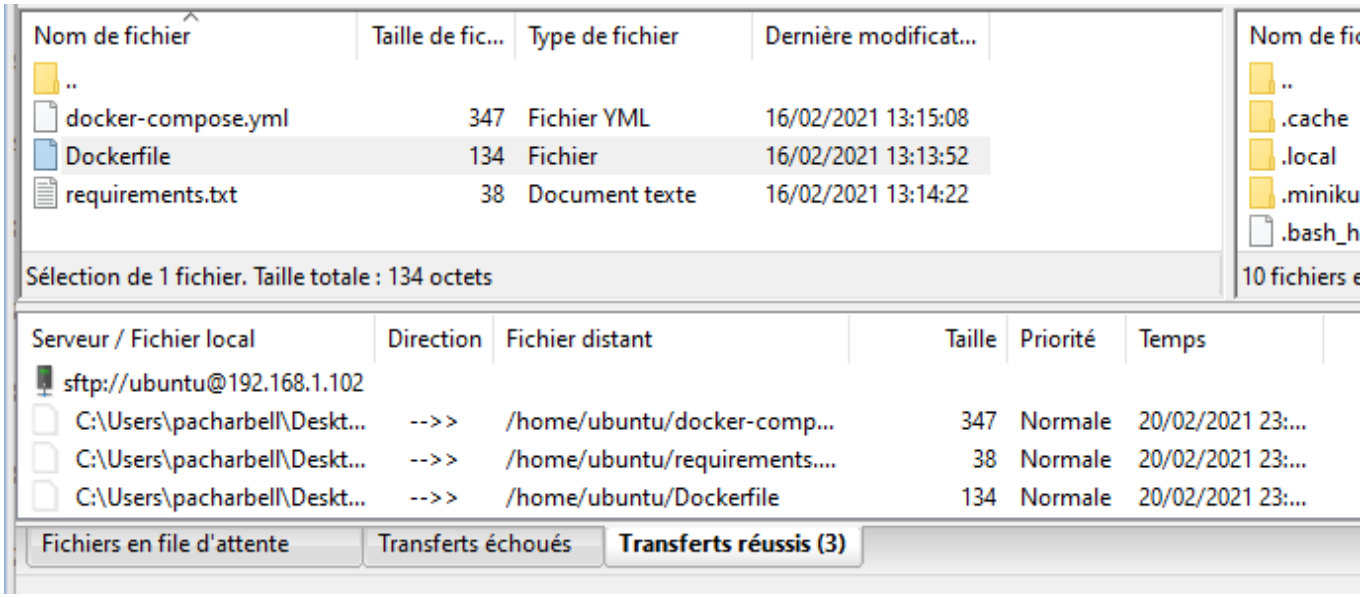
Il possède une interface utilisateur graphique intuitive.

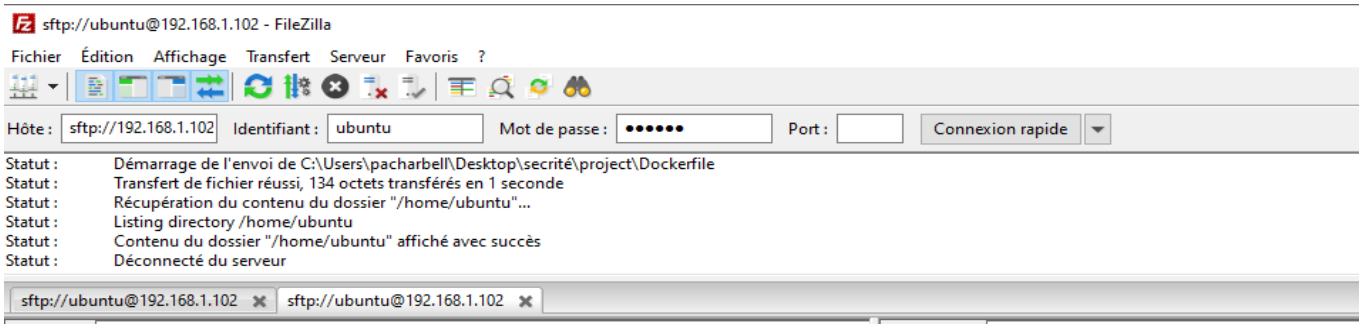
Rapide et fiable, Filezilla est gratuit et multi-plateforme : il fonctionne sur tout système d'exploitation et supporte plusieurs types de connexion : client FTP, FTPS et SFTP.

On utilise l'**@ IP** de notre machine virtuelle comme hôte, l'identifiant **ubuntu,** le mot de passe **ubuntu** et **22** comme numéro de port.

- l'**@ IP** de notre machine virtuelle : **192.168.1.102**

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.102  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::a00:27ff:feb8:909c  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:b8:90:9c  txqueuelen 1000  (Ethernet)
        RX packets 914  bytes 126941 (126.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 189  bytes 30167 (30.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```





➔ Nous avons donc copier le contenu du 3 fichiers :

- **« requirements.txt »**
- **« Dockerfile »**
- **« docker-compose.yml »**

```
root@ubuntuserver:/usr/bin/project# ls
docker-compose.yml  Dockerfile  requirements.txt
root@ubuntuserver:/usr/bin/project#
```

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 .

```
root@ubuntuserver:/usr/bin/project# sudo  docker-compose run web django-admin.py startproject compos
eexample
Starting project_db_1 ... done
Creating project_web_run ... done
root@ubuntuserver:/usr/bin/project# _
```

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

```
root@ubuntuserver:/usr/bin/project# sudo chown -R $root : $root.
root@ubuntuserver:/usr/bin/project#
```

```
root@ubuntuserver:/usr/project# ls -l
total 20
drwxr-xr-x 2 root   root   4096 Feb 25 11:51 composeexample
-rw-rw-r-- 1 ubuntu ubuntu  347 Feb 25 10:59 docker-compose.yml
-rw-rw-r-- 1 ubuntu ubuntu  134 Feb 25 10:59 Dockerfile
-rwxr-xr-x 1 root   root    670 Feb 25 11:44 manage.py
-rw-rw-r-- 1 ubuntu ubuntu   38 Feb 25 10:59 requirements.txt
root@ubuntuserver:/usr/project#
```

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

```
root@ubuntuserver:/usr/project# ls
composeexample  docker-compose.yml  Dockerfile  manage.py  requirements.txt
root@ubuntuserver:/usr/project# cd composeexample
root@ubuntuserver:/usr/project/composeexample# ls
asgi.py  __init__.py  settings.py  urls.py  wsgi.py
root@ubuntuserver:/usr/project/composeexample#  vi settings.py_
```

```
ALLOWED_HOSTS = [*]
```

```
~
"/manage-py" [New] 2L, 21C written
root@ubuntuserver:/usr/bin/project/composeexample# _
```

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

```
root@ubuntuserver:/usr/project/composeexample# vi settings.py_
```

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

```
STATIC_URL = '/static/'
"settings.py" 124L, 3190C written
root@ubuntuserver:/usr/project/composeexample# _
```

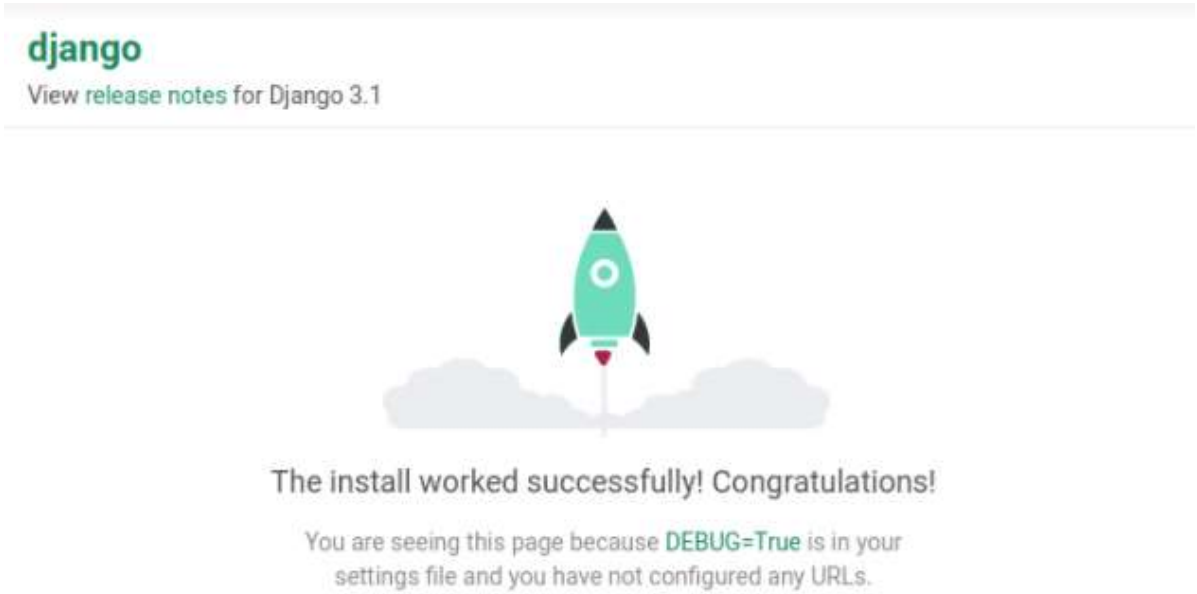
5. Démarrez le projet avec docker-compose : sudo docker-compose up

```
root@ubuntuserver:/usr/project# sudo docker-compose run web django-admin.py startproject composeexam
ple  .
Creating project_web_run ... done
```

```
db_1    | 2021-02-25 12:56:35.623 UTC [27] LOG:  database system was shut down at 2021-02-25 12:54:31
 UTC
db_1    | 2021-02-25 12:56:35.645 UTC [1] LOG:  database system is ready to accept connections
web_1   | Watching for file changes with StatReloader
web_1   | Performing system checks...
web_1   |
web_1   | System check identified no issues (0 silenced).
web_1   |
web_1   | You have 18 unapplied migration(s). Your project may not work properly until you apply the
migrations for app(s): admin, auth, contenttypes, sessions.
web_1   | Run 'python manage.py migrate' to apply them.
web_1   | February 25, 2021 - 12:56:42
web_1   | Django version 3.1.6, using settings 'composeexample.settings'
web_1   | Starting development server at http://0.0.0.0:8000/
web_1   | Quit the server with CONTROL-C.
```

6. Accéder à Django:

- Sous windows : 192.168.56.101:8000



Conclusion

Au cours de ce TP , nous avons commencer par comprendre la notion du Docker, ensuite nous avons bien interagir avec les containers, nous avons utiliser Docker-compose qui est un outil développé par Docker pour créer les architectures logicielles containérisées.