

Université Cheikh Anta Diop



École Supérieure Polytechnique Département Génie Informatique Année Universitaire 2022-2023 Virtualisation et Cloud computing Atelier 1 : Plateforme de Conteneurisation Docker *Dr Mandicou BA*

L'objectif de cet atelier est de vous permettre de familiariser avec la plateforme de conteneurisation Docker. À la fin de cet atelier chaque étudiant doit être en mesure de pouvoir :

- ✓ Installer et de configurer correctement Docker
- ✓ Gérer efficacement des conteneurs en mode cli et graphique (**Docker cli, Portainer**).
- ✓ Télécharger ou téléverser des images Docker dans le Hub de Docker (**Docker Hub**)
- ✓ Gérer l'interconnexion entre conteneurs et leurs déploiements (**Docker Compose**)

Pré-requis :

1. Une machine virtuelle ou physique tournant sous l'os Debian 11.
2. Une connexion internet sur la machine
3. un compte sur Docker Hub

Ressources :

1. <https://www.mandicouba.net/vm/virtualbox/>
2. <https://www.mandicouba.net/os/debian-11.iso>

Tâche 1 : Mise en place d'une plateforme Docker sous Debian 11

1. Mise à jour de la liste de paquets existante

Avant de procéder à l'installation de docker la première chose à faire est de mettre à jour la liste des paquets de notre système Debian 11 Bullseye.

```
mandicou@Mandicou:~$ sudo apt update && sudo apt upgrade
```

2. Permettre à la commande apt d'utiliser les paquets via HTTPS

Nous allons installer **Docker** depuis le référentiel officiel afin d'être sûre de disposer la dernière version car les dépôts de **Debian** peuvent ne pas posséder la dernière version. Ainsi la première chose à faire est d'installer des prérequis qui vont nous permettre d'utiliser la commande **apt** pour installer des paquets à travers le protocole **https**.

```
mandicou@Mandicou:~$ sudo apt install apt-transport-https  
ca-certificates curl gnupg2 software-properties-common
```

3. Ajout de la clé GPG du référentiel officiel de Docker

La commande **apt** peut utiliser **https** comme mode transport des paquets nous allons ajouter la clé **GPG** du répertoire officiel de Docker afin de permettre le chiffrement et la signature de données.

```
mandicou@Mandicou:~$ curl -fsSL https://download.docker.com  
/linux/debian/gpg | sudo apt-key add -
```

Maintenant nous allons ajouter le dépôt officiel de **Docker** dans la liste des **dépôts** du système.

```
mandicou@Mandicou:~$ sudo add-apt-repository  
"deb [arch=amd64] https://download.docker.com  
/linux/debian $(lsb_release -cs) stable"
```

4. Mise à jour de la liste de paquets

Vu que nous avons ajouté une nouvelle entrée nous allons encore mettre à jour la liste des **paquets**

```
mandicou@Mandicou:~$ sudo apt update
```

5. Installation de Docker

Maintenant tout est prêt nous allons passer à l'installation de **Docker** proprement dit en utilisant la commande **apt**

```
mandicou@Mandicou:~$ sudo apt install docker-ce
```

6. Vérification de l'état de Docker

Après avoir installer **Docker**, nous allons vérifier si tout fonctionne correctement par exemple vérifier que le démon Docker a bien démarré de mêmes si le processus de démarrage automatique est activé afin de permettre à Docker de se lancer directement après le démarrage du Système.

```
mandicou@Mandicou:~$ sudo systemctl status docker  
docker.service - Docker Application Container Engine  
Loaded: loaded (/lib/systemd/system/docker.service;  
enabled; vendor preset: e  
Active: active (running) since Tue  
Docs: https://docs.docker.com  
Main PID: 580 (dockerd)  
Tasks: 11  
Memory: 130.3M  
CGroup: /system.slice/docker.service
```

7. Utiliser Docker sans Sudo

Par défaut, une commande **Docker** est toujours précédée du mot clé **sudo** autrement dit une commande **Docker** ne peut être exécutée que par le super admin **root** ou par un utilisateur du groupe docker. Nous allons lever cette contrainte afin d'utiliser **Docker** sans disposer des privilèges du super admin. Par défaut une fois que Docker est installé le groupe docker est automatiquement créé dans le doute taper la commande suivante pour créer le groupe

```
mandicou@Mandicou:~$ sudo groupadd -f docker
```

De même le socket utilisé par **Docker** doit appartenir au groupe docker. La commande suivante permet de définir l'appartenance.

```
mandicou@Mandicou:~$ sudo chown root:docker /var/run/docker.sock
```

Maintenant nous devons ajouter l'utilisateur courant au groupe **docker** pour lui permettre d'utiliser les commandes docker sans au préalable taper **sudo**

```
mandicou@Mandicou:~$ sudo usermod -a -G docker "$(whoami)"
```

Pour terminer, on va appliquer les changements et redémarrer **Docker**

```
mandicou@Mandicou:~$ newgrp docker
```

```
mandicou@Mandicou:~$ sudo systemctl restart docker
```

8. Lancement de l'image hello-world

Nous allons lancer notre première image prédéfinie dans Docker, il s'agit de l'image **hello-world** utiliser d'habitude pour vérifier que **Docker** est bien installé et que tout fonctionne à merveille

```
mandicou@Mandicou:~$ 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.
```

9. L'aide sous Docker

La commande ci-dessous permet d'afficher la liste des commandes Docker

```
mandicou@Mandicou:~$ docker --help
```

Pour obtenir de l'aide sur une commande docker il faut utiliser la commande `docker help` suivis de la commande dont on veut afficher le manuel :

```
mandicou@Mandicou:~$ docker help nom_de_la_commande
```

Tâche 2 : Gestion des conteneurs Docker

1. Création d'une image

Pour créer une image docker nous allons commencer par créer un dossier qui va contenir les données de notre image ainsi que le fichier Dockerfile.

```
mandicou@Mandicou:~$ mkdir $HOME/premiere-image
mandicou@Mandicou:~$ cd $HOME/premiere-image
mandicou@Mandicou:~$ mkdir mes-donnees
mandicou@Mandicou:~$ touch $HOME/premiere-image/mes-dossier/file {1..4}
mandicou@Mandicou:~$ nano Dockerfile
```

Contenu du fichier Dockerfile :

```
# Définir l'image source
FROM debian
# Exécuter des commandes dans le conteneur
RUN apt-get update -yq
# Ajouter des fichiers dans le conteneur
ADD . /mes-dossier/
# Définir le répertoire de travail
WORKDIR /mes-donnees
# Définir les ports d'écoute par défaut
EXPOSE 80 443
# Définir la commande par défaut lors de l'exécution du conteneur
CMD echo "Bienvenu sur votre premier conteneur" && bash
```

Une fois le fichier Dockerfile créé, nous allons utiliser la commande build pour créer notre image.

```
mandicou@Mandicou:~$ docker build -t premiere-image .
```

Maintenant que l'image est créée nous allons lancer notre conteneur à l'aide de la commande

```
mandicou@Mandicou:~$ docker run -ti premiere-image
```

Nous allons installer manuellement quelques programmes usuels dans notre conteneur

```
mandicou@Mandicou:~$ docker run -ti premiere-image
apt install net-tools curl apache2
echo "ServerName localhost" >> /etc/apache2/apache2.conf
/etc/init.d/apache2 restart
```

2. Lister les conteneurs actifs

```
mandicou@Mandicou:~$ docker ps -a
```

3. Status du dernier conteneur créé

```
mandicou@Mandicou:~$ docker ps -l
```

4. Démarrer un conteneur

```
mandicou@Mandicou:~$ docker start ID
```

5. Arrêter un conteneur

```
mandicou@Mandicou:~$ docker stop ID
```

6. Supprimer un conteneur

```
mandicou@Mandicou:~$ docker rm ID
```

7. Installation automatique des programmes précédemment installés

```
mandicou@Mandicou:~$ nano Dockerfile
```

Contenu du fichier Dockerfile

```
FROM ubuntu
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update
RUN apt-get install net-tools -y
RUN apt-get install curl -y
RUN apt-get install apache2 -y
RUN apt-get install apache2-utils -y
RUN apt-get clean
EXPOSE 80 443
CMD echo "ServerName localhost" >> /etc/apache2/apache2.conf
&& /etc/init.d/apache2 restart && bash
```

```
mandicou@Mandicou:~$ docker build -t server-web .
mandicou@Mandicou:~$ docker run -ti server-web
```

Tâche 3 : Docker Hub

1. Recherche d'une image

Docker Hub est le dépôt distant géré par l'organisation Docker, nous pouvons y déposer nos images ou bien même télécharger des images depuis les serveurs Docker Hub. Pour rechercher une image dans le Docker Hub, nous allons utiliser la commande **docker search** suivis du nom de l'image à rechercher.

```
mandicou@Mandicou:~$ docker search mysql
```

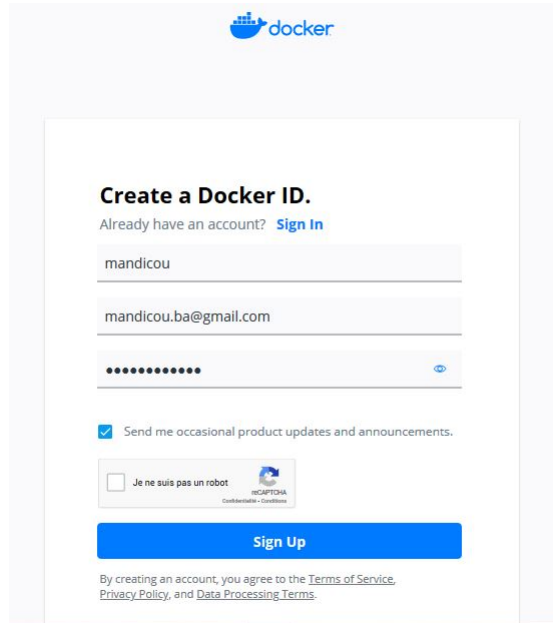
2. Téléchargement du conteneur

Pour télécharger un conteneur depuis Docker Hub nous allons utiliser la commande **docker pull** suivi de l'image qu'on veut télécharger

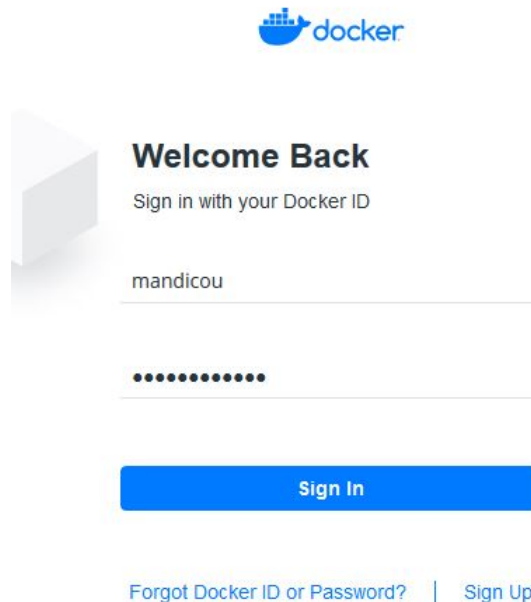
```
mandicou@Mandicou:~$ docker pull mysql/mysql-server
Using default tag: latest
latest: Pulling from mysql/mysql-server
c7127dfa6d78: Pull complete
530b30ab10d9: Pull complete
59c6388c2493: Pull complete
cca3f8362bb0: Pull complete
Status: Downloaded newer image for mysql/mysql-server:latest
docker.io/mysql/mysql-server:latest
mandicou@Mandicou:~$ docker images
docker run -ti mysql/mysql-server
```

3. Téléverser une image dans le docker Hube

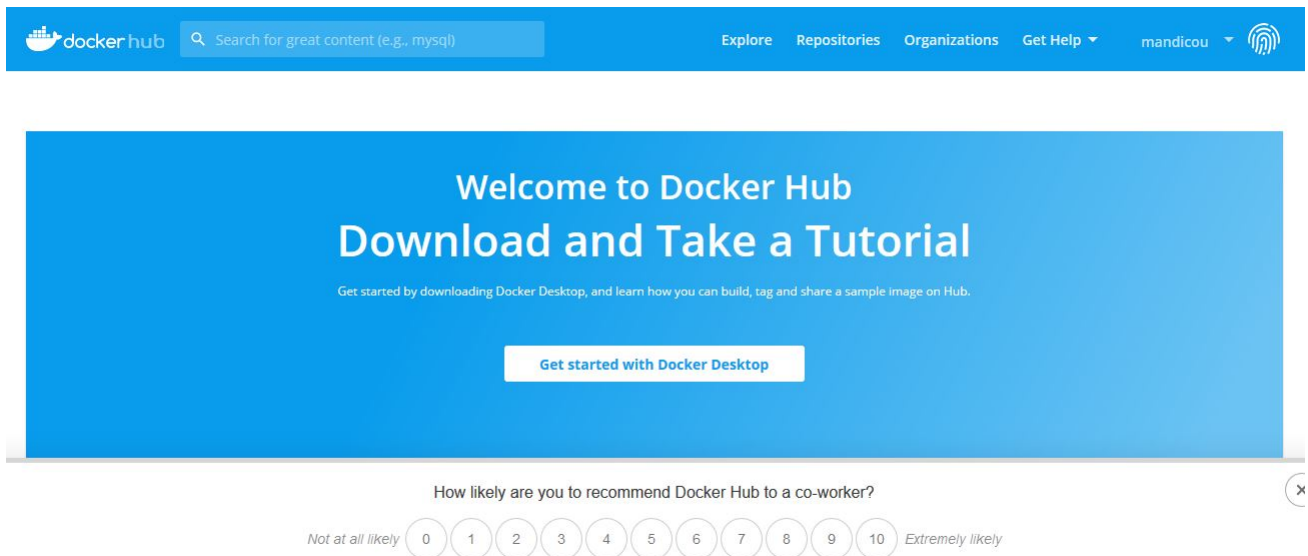
Docker Hub ne permet pas uniquement de télécharger des images mais permet aussi de sauvegarder en ligne nos propres images. Pour cela il faut disposer d'un compte docker hub. La création d'un compte Docker Hub se fait au niveau du site officiel de docker <https://hub.docker.com>



The screenshot shows the 'Create a Docker ID' page on Docker Hub. At the top is the Docker logo. Below it, the heading 'Create a Docker ID.' is followed by a link 'Already have an account? Sign In'. The form contains three input fields: a username field with 'mandicou', an email field with 'mandicou.ba@gmail.com', and a password field with masked characters. Below the password field is a checkbox labeled 'Send me occasional product updates and announcements.' which is checked. Underneath is a CAPTCHA section with a checkbox labeled 'Je ne suis pas un robot' and a CAPTCHA image. A blue 'Sign Up' button is at the bottom of the form. Below the button, a small text line states: 'By creating an account, you agree to the Terms of Service, Privacy Policy, and Data Processing Terms.'



The screenshot shows the 'Welcome Back' sign-in page on Docker Hub. At the top is the Docker logo. To the left of the text is a 3D cube graphic. The heading 'Welcome Back' is followed by the text 'Sign in with your Docker ID'. The form has two input fields: a username field with 'mandicou' and a password field with masked characters. A blue 'Sign In' button is at the bottom of the form. Below the button, there are two links: 'Forgot Docker ID or Password?' and 'Sign Up'.



une fois le compte crée nous allons utiliser la commande **docker login** pour se connecter a notre dépôt distant docker

```
mandicou@Mandicou:~/ docker login
Login with your Docker ID to push and
pull images from Docker Hub. If you don't
have a Docker ID, head over to https://hub.docker.com to create one.
Username: mandicou
Password:
Login Succeeded
mandicou@Mandicou:~/
```

Après avoir se connecter à notre dépôt distant nous pouvons maintenant déposer notre image pour cet exemple nous allons sauvegarder sur Docker Hub l'image précédemment crée.

```
mandicou@Mandicou:~$ docker tag 93566 mandicouba/premiere --image:v1.0
mandicou@Mandicou:~$ docker image push mandicouba/premiere --image:v1.0
```

Tâche 4 : Docker Compose

Docker compose est un outil qui permet de gérer nos conteneurs comme un ensemble de services interconnectés à l'aide d'un fichier YAML. Pour cet exemple nous allons gérer l'interconnexion des deux connecteurs précédemment créés à l'aide du fichier **Dockerfile** et notre image **MYSQL** téléchargée depuis Docker Hub.

1. Installation de docker compose

Docker compose ne faisant pas parti des outils de base l'or de l'installation de Docker, nous allons l'installer a l'aide de la commande suivante

```
sudo curl -o /usr/local/bin/docker-compose -L
"https://github.com/docker/compose/releases/download/1.8.1/
docker-compose-$(uname -s)-$(uname -m)"
```

```
sudo chmod +x /usr/local/bin/docker-compose
```

Pour vérifier que docker compose a bien été installé nous allons utiliser la commande suivante qui va afficher la version de **Docker Compose** installée.

```
mandicou@Mandicou:~$ sudo docker-compose -v
docker-compose version 1.8.1, build 878cff1
mandicou@Mandicou:~$
```

2. création du fichier docker-compose.yml

Docker compose permet de gérer en même temps plusieurs images Docker à l'aide du fichier **docker-compose.yml**. Nous allons utiliser nos deux images précédemment créées à l'aide du fichier **Dockerfile** et l'image **MySQL** obtenue à l'aide de Docker Hub. Une fois le fichier créé nous pouvons démarrer et interconnecter nos deux conteneurs en une seule commande. nous allons créer le fichier **docker-compose.yml**

```
mandicou@Mandicou:~$ touch dockerfile docker-compose.yml
index.php host.conf
```

Contenu du fichier dockerfile

```
FROM ubuntu:16.04
RUN apt-get update && \
apt-get install -y apache2 php libapache2-mod-php
mysql-client php7.0-mysql

RUN sed -i 's/;extension=php_mysql.dll/
extension=php_mysql.dll/' /etc/php/7.0/apache2/php.ini
RUN sed -i 's/;extension=php_pdo_mysql.dll/extension=
php_pdo_mysql.dll/' /etc/php/7.0/apache2/php.ini

ADD ./host.conf /etc/apache2/sites-enabled/000-default.conf
ADD ./index.php /var/www/html/index.php

ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Contenu du fichier docker-compose.yml

```
db:
image: mysql/mysql-server:5.7
ports:
- "3306:3306"
environment:
- "MYSQL_ROOT_PASSWORD=passer"
- "MYSQL_USER=root"
- "MYSQL_PASSWORD=passer"
- "MYSQL_DATABASE=mysql"
serveur-web:
build: ./
ports:
- "80:80"
links:
- "db:db"
working_dir: "/home/mandicou/app"
```


Contenu du fichier Index.php

```
<?php

$dbh = new PDO( 'mysql:host=db;dbname=mysql', 'root', 'passer' );

foreach ( $dbh->query( 'SHOW DATABASES' ) as $row ) {
echo $row[0]. '<br/>';
}
?>
```

Contenu du fichier host.conf

```
<VirtualHost *:80>
ServerAdmin webmaster@localhost
DocumentRoot /var/www/html
</VirtualHost>
```

3. lancement du docker compose

Nous allons maintenant lancer **Docker Compose** une fois lancé nos deux conteneur seront automatiquement lancés.

```
docker-compose up -d
```

Tâche 5 : Portainer

Il est possible de gérer les conteneurs docker depuis un interface graphique, plusieurs outils permettent de le faire nous allons prendre un des outils qui s'appelle **Portainer** qui va permettre depuis son interface web de gerer les images de Docker.

1. installation de Portainer

Nous allons Commencer par créer un volume qui va accueillir les données de **Portainer**.

```
docker volume create portainer_data
```


Après avoir créer le volume nous allons maintenant démarrer Portainer depuis le port 9000

```
docker run -d -p 9000:9000 -p 8000:8000 --name portainer --restart always --
```

2. Utilisation de Portainer

une fois que Portainer est lancé nous pouvons ouvrir notre navigateur et saisir l'adresse suivant

```
http://localhost:9000
```



Please create the initial administrator user.


Username


Password

Confirm password


✓


✓ The password must be at least 8 characters long


 **Create user**




Connect Portainer to the Docker environment you want to manage.

 **Local**
Manage the local Docker environment

 **Remote**
Manage a remote Docker environment

 **Agent**
Connect to a Portainer agent

 **Azure**
Connect to Microsoft Azure ACI

Information


Manage the Docker environment where Portainer is running.


⚠ Ensure that you have started the Portainer container with the following Docker flag:

```
-v "/var/run/docker.sock:/var/run/docker.sock" (Linux).
```


or

```
-v \\.\pipe\docker_engine:\\.\pipe\docker_engine (Windows).
```

 **Connect**




- Home
- LOCAL
- Dashboard
- App Templates
- Stacks
- Containers
- Images
- Networks
- Volumes
- Events
- Host
- SETTINGS
- Extensions
- Users
- Endpoints
- Registries
- Settings

 portainer.io 1.23.1


Dashboard


Endpoint summary

[Portainer support](#)
[mandicou](#)
[my account](#)
[log out](#)

 **Endpoint info**

Endpoint	local 1 3.1 GB - Standalone 19.03.6
URL	/var/run/docker.sock
Tags	-

 **1**
Stack


 **18**
Containers

1 healthy


3 running


0 unhealthy

15 stopped

 **7**
Images

1.4 GB

 **2**
Volumes

 **3**
Networks