Fiche de procédure : Docker

Docker est un outil de mise en œuvre de conteneurisation, il permet de créer et de gérer des conteneurs. On verra ainsi comment l'installer sur une machine virtuelle, ses différentes commandes de base, la création d'images docker et une automatisation de son déploiement.

Sommaire :

Installation de Docker :

Afin d'installer Docker, il faut tout d'abord créer une machine virtuelle en Linux console, dans notre cas on utilisera VirtualBox pour créer la machine virtuelle.

Une fois sur la machine virtuelle il faut installer le docker **apt** archivage , on utilisera les commandes suivantes :

```
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
```

On ajoute ensuite le répertoire pour docker :

```
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
```

Lorsque c'est installé on peut enfin installer les paquets docker avec la commande suivante :

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


Carteron Mathieu
BTS SIO

On vérifiera ensuite si l'installation à bien eu lieu avec la commande :

sudo docker run hello-world

Si le message de confirmation apparaît, le docker est bien installé !

```
root@debian-console:/home/etu1# sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:54e66cc1dd1fcb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Status: Downloaded newer image for hello-world:latest

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/

root@debian-console:/home/etu1#
```

Différentes commandes de Docker :

On peut par exemple récupérer une image avec :
docker pull [image] (ex: docker pull nginx)
On peut créer avec cette commande un container :
docker run -d -p 8080:80 --name myapp nginx
Puis vérifier que le container est bien créé : *docker container ls*

Gestion de container :

→ **Lancer** : *docker start nomcontainer*
→ **Stopper** : *dockier stop nomcontainer*
→ **Vérifier les docker** lancés : *docker status*

```
CONTAINER ID   NAME       CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O     PIDS
7e5da5d90261   appNginx   0.00%     2.402MiB / 1.928GiB   0.12%     1.23kB / 126B   0B / 12.3kB   2
```

→ **Vérifier les processus** de conteneur : *docker container top nomcontainer*
→ **Supprimer** un conteneur : *docker container rm nomcontainer*
→ **Lister** les conteneurs : *docker container ls*
On pourra par exemple utiliser la commande après une suppression pour voir s'il n'est plus dans la liste.
(Pour supprimer les images on utilisera la même commande avec *image* à la place de container)

Carteron Mathieu
BTS SIO

Création d'images Docker :

Affichage de texte :

Une image docker est un fichier contenant l'application ainsi que les librairies (package).
Une fois lancé on obtient un conteneur.

Afin de créer une image à l'aide de **DockerFile**, on créé d'abord un dossier (*mkdir*) et on s'y rends (cd), on créé ensuite un fichier en .sh avec un texte à l'intérieur du fichier (ex : Hello world !).

On écrit ensuite le Dockerfile (dans un fichier) :

```
FROM debian:latest
COPY my-hello-world.sh /my-hello-world.sh
CMD ["/my-hello-world.sh"]
```

Une fois écrit on construit l'image Docker :

```
root@debian-console:/home/etu1/my-hello-world# docker build -t my-hello-world .
[+] Building 4.5s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 126B
 => [internal] load metadata for docker.io/library/debian:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build context
 => => transferring context: 65B
 => [1/2] FROM docker.io/library/debian:latest@sha256:833c135acfe9521d7a0035a296076f98c182c542a2b6b5a0fd7063d355d696be
 => => resolve docker.io/library/debian:latest@sha256:833c135acfe9521d7a0035a296076f98c182c542a2b6b5a0fd7063d355d696be
 => => sha256:15b1d8a5ff03aeb0f14c8d39a60a73ef22f656550bfa1bb90d1850f25a0ac0fa 49.28MB / 49.28MB
 => => sha256:833c135acfe9521d7a0035a296076f98c182c542a2b6b5a0fd7063d355d696be 8.93kB / 8.93kB
 => => sha256:56b68c54f22562e5931513fabfc88a23670faf16bbe82f2641d8a2c836ea30fc 1.02kB / 1.02kB
 => => sha256:999ffdddc1528999603ade1613e0d336874d84448a74db8f981c6fae4db91ad7 451B / 451B
 => => extracting sha256:15b1d8a5ff03aeb0f14c8d39a60a73ef22f656550bfa1bb90d1850f25a0ac0fa
 => [2/2] COPY my-hello-world.sh /my-hello-world.sh
 => exporting to image
 => => exporting layers
 => => writing image sha256:1c596e3d6c5e3877d3107ca7cc94cf88f63fbfdd5283fe041cc175a6a7a153b2
 => => naming to docker.io/library/my-hello-world
```

Affichage de code web :

Afin d'afficher du code au lieu du texte on crée une image de la même façon que précédemment, en y incluant un fichier html avec le code souhaité.
Exemple :

```
<!DOCTYPE html>
<html>
<body>

<h1> Titre de la page </h1>

<p> Contenu de la page </p>

</body>
<html>
```

On lance ensuite le container (comme vu dans les différentes commandes docker)
Une fois lancé, on vérifie avec docker run que le code apparaît correctement !

Carteron Mathieu
BTS SIO

Automatisation du déploiement de conteneurs Docker :

On utilisera ensuite le Docker Compose, cet outil permet de définir et de gérer plusieurs containers à la fois, ce qui est très pratique pour des applis.

On commence tout d'abord par créer un fichier de configuration comme vu précédemment pour déployer un conteneur (voir les différentes commandes docker).
On récupère ensuite les images dans le fichier de config et on lance le conteneur (comme vu précédemment) et on vérifie qu'il fonctionne correctement.

On lance ensuite le conteneur en arrière plan, pour cela on utilisera :
docker compose up -d nomConteneur (le -d permettant de lancer en arrière plan).

On cherchera maintenant à récupérer les logs du conteneur lancé en arrière plan, pour cela on utilisera la commande : *docker compose logs -f* (Le -f permet de les suivre en temps réel).

Carteron Mathieu
BTS SIO