# Activité : créer une image Docker

Pour cette découverte de Docker, vous pouvez réaliser au préalable :

- avoir **installé** Docker.
- utiliser l'environnement de découverte appelé **Play with Docker (PWD)** : https://labs.play-with-docker.com/

### Créer une image personnalisée

L'image qui a servi à la création d'un conteneur n'est jamais modifiée par ce qui est fait à l'intérieur d'un conteneur. Pour que les modifications puissent faire partie d'une image, il faut créer une nouvelle, dans laquelle une nouvelle couche logicielle sera ajoutée et cette couche supplémentaire contiendra les modifications par rapport à l'image de base.

Pour créer une nouvelle image basée sur l'image ubuntu de base mais intégrant les modifications souhaitées comme le fichier docker.txt ou l'installation de paquets logiciels voici comment procéder.

Tout d'abord créez un conteneur et personnalisez-le :

```
btssio@ubuntudocker:~$ docker run -i -t ubuntu
root@407248dafa24:/# cd /root
root@407248dafa24:~# touch docker.txt
root@407248dafa24:~# ls
docker.txt
root@407248dafa24:~# apt-get update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
…
root@407248dafa24:~# apt-get install net-tools
Reading package lists... Done
…
root@407248dafa24:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 2109  bytes 16445383 (16.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1570  bytes 110630 (110.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
root@407248dafa24:~# exit
btssio@ubuntudocker:~$
```

En visualisant les conteneurs existants, vous retrouvez celui que vous venez de créer (ID 407248dafa24) avec le nom généré par Docker (festive*sammet) <code shell> btssio@ubuntudocker:~$ docker ps -a CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 407248dafa24 ubuntu "/bin/bash" 2 m.. Up 2 minutes festive*sammet </code>

Puis créez votre nouvelle image en utilisant son ID ou son nom et en lui associant un nouveau nom d'image préfixé par une information qui vous identifie :

```
btssio@ubuntudocker:~$ docker commit festive_sammet techer/ubuntu_nettools
sha256:2cd084ad8053b2c7d5747a7b0975006681981ea9f8e8dfea6f20c77bb361ff25
btssio@ubuntudocker:~ $ docker images
REPOSITORY             TAG     IMAGE ID      CREATED           SIZE
techer/ubuntu_nettools latest  2cd084ad8053  7 seconds ago     128MB
ubuntu                 latest  d131e0fa2585  11 days ago       102MB
…
```

Une **nouvelle image** existe avec comme nom techer/ubuntu_nettools et d'une **taille supérieure** à celle de ubuntu compte tenu du paquet logiciel net Tools installé.

La création d'un nouveau conteneur à partir de cette nouvelle image montre qu'elle a été personnalisée :

```
btssio@ubuntudocker:~ $ docker run -it techer/ubuntu_nettools
root@88085b7f2c50:/# ls /root
docker.txt
root@88085b7f2c50:/#
```

### Sauvegarder une image en local

Il peut être utile de sauvegarder une image localement à des fins d'exploitation sur un autre PC (en attendant de publier notre image sur le Hub officiel ou un autre Hub) :

```
docker save <image> > <nom_fichier.tar>
```

Par exemple :

```
btssio@ubuntudocker:~ $ docker save ar/ubuntu:ssh > serv_ubuntu-ssh.tar
```

### Restaurer depuis un conteneur en local

```
docker load -i <nom_fichier.tar>
```

Par exemple :

```
btssio@ubuntudocker:~ $docker load -i serv_ubuntu-ssh.tar
```

# Retour Accueil Docker

- Docker