

## TP5 – Docker Ubuntu

Dans un premier temps j'ai installé Docker avec les commandes suivantes, disponibles sur ce [site](#) (site avec de très bonnes explications) :

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-
common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"
apt-cache policy docker-ce
sudo apt install docker-ce
sudo systemctl status docker
```

Optionnel mais permet de ne pas à avoir à taper sudo à chaque fois :

```
sudo usermod -aG docker jade
su - jade
groups
```

Pour vérifier installation:

```
docker run hello-world
```

## Conteneur Alpine

Télécharger une image Alpine

Dans un premier temps j'ai regardé les images Alpine disponibles :

```
docker search alpine
```

```
jade@docker:~$ docker search alpine
NAME                DESCRIPTION                               STARS
alpine              A minimal Docker image based on Alpine Linux... 9578
alpinelinux/docker-cli Simple and lightweight Alpine Linux image wi... 7
alpinelinux/gitlab-runner Alpine Linux gitlab-runner (supports more ar... 4
alpinelinux/alpine-gitlab-ci Build Alpine Linux packages with Gitlab CI 3
alpinelinux/gitlab-runner-helper Helper image container gitlab-runner-helper ... 2
grafana/alpine      Alpine Linux with ca-certificates package in... 2
alpinelinux/gitlab  Alpine Linux based Gitlab image 2
alpinelinux/golang  Build container for golang based on Alpine L... 1
alpinelinux/package-builder Container to build packages for a repository 1
alpinelinux/unbound  1
rancher/alpine-git  1
alpinelinux/darkhttpd 1
```

J'ai téléchargé l'Alpine officielle :

```
docker pull alpine
```

Pour voir toutes les images téléchargées sur sa machine utiliser la commande « `docker images` » :

```
jade@docker:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
alpine        latest    042a816809aa   46 hours ago   7.05MB
hello-world   latest    feb5d9fea6a5   15 months ago  13.3kB
```

Lancer un conteneur en mode interactif

Il faut rajouter l'option `i` et `t` pour lancer le conteneur en mode interactif :

```
docker run -it --name alpine1 alpine /bin/sh
```

Pour sortir du conteneur il suffit de taper la commande « `exit` » dedans.

Commandes utiles

- La commande « `docker ps -a` » permet de vérifier les conteneurs qui ne sont pas lancés.

- La commande « `docker container ls` » est la même que « `docker ps` »
- La commande « `docker container ls -a` » qui affiche tous les conteneurs
- La commande « `docker container ls -aq` » qui affiche que les id de tous les conteneurs
- « `docker start $(docker container ls -aq)` » permet d'allumer tous les conteneurs en seulement une ligne
- Pour sortir d'un conteneur sans le stopper faire `ctrl+p` puis `ctrl+q`

Créez un fichier `toto.txt` dans le répertoire `/tmp` du conteneur

```
docker start alpine1
docker attach alpine1
cd tmp
touch toto.txt

exit
```

Arrêtez le conteneur

J'ai vérifié avec « `docker ps -a` » pour vérifier que mon conteneur était bien arrêté.

```
jade@docker:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS              PORTS          NAMES
1f11075656ae   alpine        "/bin/sh"               2 minutes ago   Exited (0) 26 seconds ago          alpine1
ab6763d95247   hello-world   "/hello"               2 hours ago    Exited (0) 12 minutes ago          vigilant_tu
```

Ce qui est le cas.

## Conteneur en mode démon

Lancer un nouveau conteneur Alpine en mode démon.

pour ouvrir directement le mode terminal sur le conteneur grâce à l'exécution du shell (`option`) la commande est :

```
docker run -it --name alpine2 alpine /bin/sh
```

Si on ne souhaite pas ouvrir directement le shell la commande est :

```
docker run -dit --name alpine2 alpine /bin/sh
```

Sans vous connecter dans le conteneur, récupérez l'ensemble des paramètres système du nouveau conteneur alpine

Pour récupérer les informations détaillées du conteneur j'ai utilisé la commande suivante :

```
docker inspect alpine2
```

Voici une partie du résultat obtenu :

```
    "WorkingDir": "",
    "Entrypoint": null,
    "OnBuild": null,
    "Labels": null
  },
  "Architecture": "amd64",
  "Os": "linux",
  "Size": 7049701,
  "VirtualSize": 7049701,
  "GraphDriver": {
    "Data": {
      "MergedDir": "/var/lib/docker/overlay2/97/merged",
      "UpperDir": "/var/lib/docker/overlay2/97/diff",
```

Testez l'existence du fichier précédemment créé, que remarquez-vous ?

Je me reconnecte au deuxième conteneur avec la commande :

```
docker attach alpine2
cd tmp
ls
```

J'ai bien remarqué que le fichier créé sur le premier conteneur n'était pas présent.

Sans vous connecter dans le conteneur, créez un fichier titi.txt dans le répertoire /tmp du conteneur

```
docker start alpine2
docker exec alpine2 touch /tmp/titi.txt
```

Arrêtez ce second conteneur

```
docker stop alpine2
```

Retour sur les deux conteneurs

Lister les conteneurs disponibles sur votre système

```
docker ps -a
```

```
jade@docker:~$ docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS              PORTS          NAMES
d7358c994e1e   alpine    "/bin/sh"               4 minutes ago Exited (137) 54 seconds ago           alpine2
1f11075656ae   alpine    "/bin/sh"               11 minutes ago Exited (0) 9 minutes ago           alpine1
ab6763d95247   hello-world "/hello"                2 hours ago   Exited (0) 21 minutes ago           vigilant_tu
```

Relancez les deux conteneurs créés en mode démon, vous en poursuivrez l'exécution

```
docker start $(docker container ls -aq)
docker ps //pour vérifier
```

Sans vous connecter dans les conteneurs, récupérez l'ensemble des paramètres systèmes des deux conteneurs

- Pour le premier :  
`docker stats alpine1`

```
CONTAINER ID   NAME      CPU %       MEM USAGE / LIMIT   MEM %       NET I/O       BLOCK I/O     PIDS
1f11075656ae   alpine1   0.00%       432KiB / 1.437GiB   0.03%       1.02kB / 0B    0B / 0B       1
```

- Pour le deuxième :  
`docker stats alpine2`

```
CONTAINER ID   NAME      CPU %       MEM USAGE / LIMIT   MEM %       NET I/O       BLOCK I/O     PIDS
d7358c994e1e   alpine2   0.00%       424KiB / 1.437GiB   0.03%       796B / 0B     0B / 0B       1
```

Testez l'existence des fichiers précédemment créés

Conteneur 1	Conteneur 2
<pre>docker exec alpine1 ls /tmp/ jade@docker:~\$ docker exec alpine1 ls /tmp/ toto.txt</pre>	<pre>docker exec alpine2 ls /tmp/ jade@docker:~\$ docker exec alpine2 ls /tmp/ titi.txt</pre>

Arrêtez les deux conteneurs

```
docker stop alpine1
```

```
docker stop alpine2
```

Détruisez les deux conteneurs

```
docker rm alpine1
```

```
docker rm alpine2
```