1 Docker : Aide mémoire

1.1 Installation (Debian/Ubuntu)

Ajouter la clé de dépôt

```
# apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80
--recv-keys 58118E89F3A912897C070ADBF76221572C52609D
```

Créer le fichier /etc/apt/sources.list.d/docker.list et écrire ça dedans :

```
deb https://apt.dockerproject.org/repo debian-jessie main
```

Puis quitter le fichier et lancer

```
# apt-get update
# apt-get install docker-engine
# service docker start
```

Pour tester, on peut lancer

```
# docker run hello-world
```

1.2 Manipulation d'images

Prendre une Debian sur le dépôt officiel de Docker et se connecter dessus :

```
# docker pull debian
# docker run -i -t debian /bin/bash
```

1.2.1 Faire tout ce qu'on veut sur la nouvelle image

```
root@xxxxxx# ...
```

1.2.2 Et sauvegarder les changements

```
root@xxxxxx# exit
# docker commit xxxxxx le_nom_de_l_image
```

1.2.3 Supprimer les images

```
# docker rmi <id | nom_de_l_image>
```

1.2.4 Création d'images par Dockerfile

Les principales directives des fichiers Dockerfiles :

- **MAINTAINER** : nom et courriel de mainteneur du conteneur ;
- **FROM :** image de base (Ubuntu, Debian) ;
- **VOLUME** : point de montage ;
- **RUN** : commande à exécuter pour installer le conteneur ;
- **ENTRYPOINT** : commande qui s'exécute au démarrage du conteneur (une seule sera exécutée) ;
- **CMD** : commande qui s'exécute au démarrage du conteneur ;
- **ADD** : copier un fichier du répertoire courant dans le système de fichiers du conteneur ;
- **USER** : utilisateur qui exécute les commandes dans le conteneur ;
- **EXPOSE** : port(s) à exposer à l'extérieur.

NB : *Il est possible de créer un fichier .dockerignore*
Ce fichier contiendra les fichiers à ignorer dans la contruction

1.3 Manipuler un conteneur

```
# JOB1=$(docker run -d conteneur)
# docker logs $JOB1
# docker stop $JOB1
```

1.3.1 Voir les conteneurs qui tournent

```
# docker ps
```

ou

```
# docker ps -a
```

Tous les conteneurs (même inactifs)

1.3.2 Supprimer un ou tous les conteneurs

```
# docker rm $JOB1
# docker rm id_du_conteneur
# docker rm `docker ps -a -q`
```

1.3.3 Construire un conteneur

Créer un fichier nommé Dockerfile avec les directives Docker, puis :

```
# docker build -t nom_du_conteneur .
```

1.3.4 Lancement

```
# docker run -d ubuntu /bin/sh -c "while true; do echo Hello
world; sleep 1; done"
```

docker run [OPTIONS] IMAGE [COMMAND] [ARG…]

Les options :

- -d lance le conteneur en tâche de fond
- --name nomme le conteneur
- -v <vol src>:<vol cont> partage un répertoire entre la machine et le conteneur
- -p <port src>:<port cont> relie le port de la machine au port du conteneur
- --dns=[] défini des dns propres au conteneur
- -e <ENV> défini une variable d'environnement
- -it lance le conteneur en interactif
- --link= lie le conteneur à un autre
-

1.3.5 Arret

```
$ docker stop <JOB>
```

1.3.6 Démarrage

```
$ docker start <JOB>
```

1.3.7 Relance

```
$ docker restart <JOB>
```

1.3.8 Tuer un conteneur

```
$ docker kill <JOB>
```

1.3.9 Supprimer un conteneur

Un conteneur doit être stoppé pour être supprimé

```
$ docker rm <JOB>
```

1.3.10 Entrer dans un conteneur actif

```
$ docker exec -it <JOB> <programme>
```

1.4 Le réseau

Docker autorise le fonctionnement des conteneurs sur le réseau au moyen de dirvers. Par défaut docker fournit 2 drivers de réseau : bridge et overlay. Il est possible d'écrire son propre driver.
L'installation de Docker installe trois type de driver :

```
$ docker network ls
NETWORK ID          NAME                DRIVER
18a2866682b8        none                null
c288470c46f6        host                host
7b369448dccb        bridge              bridge
```

Par défaut, c'est le bridge qui est utilisé. Pour retrouver l'adresse IP :

```
$ docker inspect <JOB>
```

1.4.1 Créer un bridge dédié

Un réseau de type bridge est limité à une seule machine, tandis qu'un Overlay peut inclure plusieurs hôtes. Pour créer un bridge dédié :

```
$ docker network create -d bridge my-bridge-network
```

L'option -d spécifie à docker le type de driver à utiliser. Cette option peut être laissée vide vu que c'est le fonctionnement par défaut.

```
$ docker network ls
NETWORK ID          NAME                DRIVER
7b369448dccb        bridge              bridge
615d565d498c        my-bridge-network   bridge
18a2866682b8        none                null
c288470c46f6        host                host
```

Une inspection du bridge le déclare vide

```
$ docker network inspect my-bridge-network
[
    {
        "Name": "my-bridge-network",
        "Id":
"5a8afc6364bccb199540e133e63adb76a557906dd9ff82b94183fc48c40857ac
",
        "Scope": "local",
        "Driver": "bridge",
        "IPAM": {
            "Driver": "default",
            "Config": [
                {}
            ]
        },
        "Containers": {},
        "Options": {}
    }
]
```

1.4.2 Ajouter un conteneur à un réseau

```
$ docker run -d --net=my-bridge-network --name db
training/postgres
```

1.4.3 Déplacer un conteneur sur un autre réseau

```
$ docker run -d --name web training/webapp python app.py
$ docker network connect my-bridge-network web
```

2 Exemple

Image LAMP – MariaDB, vous devez avoir un répertoire avec les fichiers suivants :

- ✗ Dockerfile
- ✗ foreground.sh
- ✗ index.phpstart.sh
- ✗ supervisord.conf

2.1 Dockerfile

```
# lamp (d'un M qui veut dire Maria)
# Pour Debian Wheezy
#
# VERSION 0.0.1
#

FROM debian:wheezy
MAINTAINER Nico Dewaele "nico@adminrezo.fr"

ENV DEBIAN_FRONTEND noninteractive

# Depots, mises a jour et installs de Apache/PHP5

RUN echo "deb http://ftp.fr.debian.org/debian/ wheezy main non-
free contrib" > /etc/apt/sources.list
RUN (apt-get update && apt-get upgrade -y -q && apt-get dist-
upgrade -y -q && apt-get -y -q autoclean && apt-get -y -q
autoremove)
RUN apt-get install -y -q vim ssh supervisor python-software-
properties apache2 libapache2-mod-php5 php5-cli php5-mysql

# Installation et configuration de MariaDB

RUN apt-key adv --recv-keys --keyserver keyserver.ubuntu.com
0xcbcb082a1bb943db
RUN add-apt-repository 'deb
http://mirrors.linsrv.net/mariadb/repo/5.5/debian wheezy main'
RUN apt-get update && apt-get install -y mariadb-server
RUN service mysql start
RUN mysql -v -uroot -e'UPDATE user SET host = "%" WHERE user =
"root" LIMIT 1; DELETE FROM user WHERE user = "root" AND host !=
"%"' mysql
RUN service mysql stop

# Config de Apache
```

```
ADD foreground.sh /etc/apache2/foreground.sh
ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2

# Verification de PHP

ADD index.php /var/www/index.php
RUN rm /var/www/index.html

# Demarrage des services

RUN mkdir -p /var/log/supervisor
ADD supervisord.conf /etc/supervisor/conf.d/supervisord.conf
ADD start.sh /start.sh

EXPOSE 80 22
# Si MariaDB doit être exposée à l'extérieur
# EXPOSE 3306

CMD ["/bin/bash", "-e", "/start.sh"]
```

2.2 foreground.sh

Lance le serveur apache en arrière plan

```
#!/bin/bash
```

2.3 start.sh

```
read pid cmd state ppid pgrp session tty_nr tpgid rest <
/proc/self/stat
trap "kill -TERM -$pgrp; exit" EXIT TERM KILL SIGKILL SIGTERM
SIGQUIT

source /etc/apache2/envvars
apache2 -D FOREGROUND
```

2.4 index.php

```
<?php
Phpinfo() ;
?>
```

2.5 start.sh

Lance le supervisor (qui lancera à son tour Apache et MariaDB)

```
#!/bin/bash
supervisord
```

2.6 supervisord.conf

Fichier de configuration de supervisor

```
[supervisord]
nodaemon=true

[program:sshd]
command=/usr/sbin/sshd -D
stdout_logfile=/var/log/supervisor/%(program_name)s.log
stderr_logfile=/var/log/supervisor/%(program_name)s.log
autorestart=true

[program:httpd]
command=/usr/sbin/apache2ctl start
stopsignal=6

[program:mariadb]
command=/usr/sbin/mysqld
stdout_logfile=/tmp/%(program_name)s.stdout
stderr_logfile=/tmp/%(program_name)s.stderr
stopsignal=6
```