= Introduction à Docker
Stéphane Deraco <stephane.deraco@dsi.cnrs.fr>
:toc:

Introduction à https://www.docker.com[Docker], pour la journée de veille technologique ARGOS : "Devops : de l'hyperviseur aux conteneurs" (http://indico.lal.in2p3.fr/event/2635/).

== Installation

```
➜  ~  sudo apt-get install docker.io # Sur Debian et dérivées
```

=== Démon docker

```
➜  ~  sudo service docker status
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; disabled)
   Active: active (running) since lun. 2014-12-08 17:25:22 CET; 17min ago
     Docs: http://docs.docker.com
 Main PID: 19291 (docker)
   CGroup: /system.slice/docker.service
           └─19291 /usr/bin/docker -d -H fd://
```

=== Premier appel de docker

```
➜  ~  docker version
Client version: 1.3.2
Client API version: 1.15
Go version (client): go1.3.2
Git commit (client): 39fa2fa
OS/Arch (client): linux/amd64
2014/12/08 17:48:41 Get http:///var/run/docker.sock/v1.15/version: dial unix
/var/run/docker.sock: permission denied #<1>
```

<1> Erreur de socket

=== Ajout du groupe docker

```
➜  ~  sudo gpasswd -a steph docker
➜  ~  docker version
Client version: 1.3.2
Client API version: 1.15
Go version (client): go1.3.2
Git commit (client): 39fa2fa
OS/Arch (client): linux/amd64
Server version: 1.3.2 # <1>
Server API version: 1.15
Go version (server): go1.3.2
Git commit (server): 39fa2fa
```

<1> On a les informations sur le _serveur_ docker

== Premier lancement d'un conteneur
=== Récupération d'une image
[source]

➜ ~ docker search debian | head
NAME
DESCRIPTION STARS OFFICIAL AUTOMATED
debian (Semi) Official Debian base
image. 253 [OK]
google/debian
25 [OK]
tianon/debian use "debian" instead -
https://index.docke... 13
yesnault/docker-phabricator Debian Jessie / Apache 2 / Mysql /
Phabric... 9 [OK]
eboraas/apache-php PHP5 on Apache (with SSL support), built
o... 8 [OK]
minimum2scp/es-kibana ElasticSearch + Kibana on Debian sid
amd64... 5 [OK]
hanswesterbeek/google-debian-oracle-jdk Oracle's JDK installed on top of Google's
... 5 [OK]
toke/owncloud7 Basic image with Owncloud 7 from debian
pa... 5 [OK]
eboraas/apache Apache (with SSL support), built on
Debian 5 [OK]

Bien faire attention à l'image que l'on récupère. Privilégier les images certifiées.

[source]

➜ ~ docker pull debian:latest
debian:latest: The image you are pulling has been verified

511136ea3c5a: Pull complete
f10807909bc5: Pull complete
f6fab3b798be: Pull complete
Status: Downloaded newer image for debian:latest

[source]

➜ ~ docker images
REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE
debian latest f6fab3b798be 4 weeks ago 85.1 MB

L'image a été récupérée et est disponible en local.

=== Lancement d'un conteneur
[source]

```
➜  ~  docker run debian echo Hello from docker
Hello from docker
--------------
```

Versions de Bash différentes entre l'hôte et le conteneur :

[source]
```
--------------
➜  ~  bash -version
GNU bash, version 4.3.30(1)-release (x86_64-pc-linux-gnu)

➜  ~  docker run debian bash -version
GNU bash, version 4.2.37(1)-release (x86_64-pc-linux-gnu)
--------------
```

Par contre, le conteneur utilise le noyau Linux de l'hôte :
[source]
```
--------------
➜  ~  uname -a
Linux debian 3.16.0-4-amd64 #1 SMP Debian 3.16.7-2 (2014-11-06) x86_64 GNU/Linux

➜  ~  docker run debian uname -a
Linux 87c1ea9f6a02 3.16.0-4-amd64 #1 SMP Debian 3.16.7-2 (2014-11-06) x86_64 GNU/Linux
--------------
```

=== Liste des conteneurs
[source]
```
--------------
➜  ~  docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES

➜  ~  docker ps --all
CONTAINER ID        IMAGE               COMMAND                CREATED
STATUS                      PORTS               NAMES
87c1ea9f6a02        debian:latest       "uname -a"             9 minutes ago       Exited
(0) 9 minutes ago                        clever_brattain
8568b22771e8        debian:latest       "bash -version"        10 minutes ago      Exited
(0) 10 minutes ago                       mad_carson
3f13f731f4a1        debian:latest       "echo Hello from doc   24 minutes ago      Exited
(0) 24 minutes ago                       boring_bohr
--------------
```

Par défaut `docker ps` ne liste que les conteneurs _actifs_.

Suppression de tous les conteneurs :
[source]
```
--------------
➜  ~  docker ps --all --quiet
87c1ea9f6a02
8568b22771e8
3f13f731f4a1
```

```
➜  ~  docker rm $(docker ps --all --quiet)
87c1ea9f6a02
8568b22771e8
3f13f731f4a1
--------------
```

Lancer un conteneur, sans conserver le résultat :

```
[source]
--------------
➜  ~  docker run --rm debian echo test
test

➜  ~  docker ps --all
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
--------------
```

=== Obtenir un shell dans le conteneur

```
[source]
--------------
➜  ~  docker run --rm debian bash
# Il ne se passe rien

➜  ~  docker run --rm --interactive --tty debian bash
root@60237777724e:/#
# On est dans le conteneur
--------------
```

== Configurer une image

Maintenant que l'on a un shell dans le conteneur, on peut y faire ce que l'on veut.
Par contre, il ne faut pas utiliser l'option `--rm` sinon on perd ce que l'on a fait.

```
[source]
--------------
➜  ~  docker run -it debian bash
root@60237777724e:/# # Par défaut, la commande ps n'est pas installée. On va l'installer
dans le conteneur.
root@60237777724e:/# apt-get update
Get:1 http://http.debian.net wheezy Release.gpg [1655 B]
# SNIP...
Reading package lists... Done

root@60237777724e:/# apt-get install procps
Reading package lists... Done
# SNIP...
Do you want to continue [Y/n]? Y
Get:1 http://http.debian.net/debian/ wheezy/main libncursesw5 amd64 5.9-10 [141 kB]
# SNIP...

root@60237777724e:/# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
```

```
root         1  0.0  0.1  17856  3112 ?        Ss   13:52   0:00 bash
root       135  0.0  0.0  15324  1904 ?        R+   13:55   0:00 ps aux

root@60237777724e:/# exit
--------------
```

On a maintenant un conteneur avec ce que l'on a fait :

```
[source]
--------------
➜  ~  docker ps --all
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS                      PORTS               NAMES
d29756010e67        debian:latest       "bash"              47 seconds ago      Exited (0)
17 seconds ago                          naughty_mestorf
--------------
```

Ce conteneur est arrêté. On peut le redémarrer :

```
[source]
--------------
➜  ~  docker start d297
d297

➜  ~  docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
d29756010e67        debian:latest       "bash"              56 minutes ago      Up 5
seconds                                 naughty_mestorf
--------------
```

Maintenant, pour s'y connecter :

```
[source]
--------------
➜  ~  docker attach d29
root@d29756010e67:/# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  17860  2840 ?        Ss+  14:55   0:00 bash
root        20  0.0  0.0  15324  1932 ?        R+   14:57   0:00 ps aux

root@d29756010e67:/# exit
--------------
```

Si ce conteneur nous parait bien, on peut le promouvoir en image :

```
[source]
--------------
➜  ~  docker commit d29 argos/demo
dadf613a22d31ff2f87db9590bebffa67b4ae6780befb027e83eb5a0ff1e1f24

➜  ~  docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
argos/demo          latest              dadf613a22d3        6 seconds ago       95.35 MB
debian              latest              f6fab3b798be        4 weeks ago         85.1 MB
--------------
```

On peut alors maintenant le lancer directement, on a `procps` installé :
[source]
```
➜  ~  docker run -it --rm argos/demo
root@60f22b0adb00:/# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  17860  2936 ?        Ss   15:04   0:00 bash
root         9  0.0  0.0  15324  1928 ?        R+   15:04   0:00 ps aux
root@60f22b0adb00:/# exit
```
A noter que l'on n'a pas précisé de commande. Dans ce cas, c'est la commande par défaut (colonne `COMMAND`) qui est utilisée.

=== Notion de port
Cet exemple va installer `netcat` et l'objectif et de le faire écouter sur port, puis, y accéder en dehors du conteneur.

[source]
```
➜  ~  docker run -it argos/demo apt-get -y install netcat-traditional
Reading package lists... Done
# SNIP ...
update-alternatives: using /bin/nc.traditional to provide /bin/nc (nc) in auto mode

➜  ~  docker ps --all
CONTAINER ID        IMAGE               COMMAND                CREATED
STATUS                      PORTS               NAMES
50f182bf37ee        argos/demo:latest   apt-get -y install    20 seconds ago      Exited (0)
16 seconds ago                      elegant_galileo

➜  ~  docker commit 50f argos/netcat
f472bd490861d06368d17ead664945d97831f189a69694a21ee1d679e9eec594

➜  ~  docker run -p 1234:1234 -d argos/netcat nc -l -p 1234 # <1>
c2734d34b25e3c5fbb6de21588940223225b37c57761b7723e430f82b259f50c
```
<1> Le flag `-d` permet de lancer docker en mode détaché. Le flag `-p` permet de mapper un port exposé par le conteneur vers l'hôte.

[source]
```
➜  ~  docker ps
CONTAINER ID        IMAGE                  COMMAND             CREATED
STATUS              PORTS                     NAMES
c2734d34b25e        argos/netcat:latest   "nc -l -p 1234"     20 seconds ago      Up 19
seconds       0.0.0.0:1234->1234/tcp   furious_bell

➜  ~  docker top c27
UID                 PID                 PPID                C
STIME               TTY                 TIME                CMD
root                8488                1857                0
16:55               ?                   00:00:00            nc -l -p 1234
```

On a bien notre conteneur qui tourne, et qui écoute sur le port 1234.

```
[source]
--------------
➜  ~  telnet localhost 1234
Trying ::1...
Connected to localhost.
Escape character is '^]'.
coucou

➜  ~  docker logs c2734d34b25e
coucou
--------------
```

== Diff
La commande `docker diff` permet de visualiser les modificatons apportées à un conteneur.

```
[source]
--------------
➜  ~  docker run argos/netcat sh -c "echo hello > world.txt"
➜  ~  docker ps --all
CONTAINER ID        IMAGE                  COMMAND                CREATED
STATUS                      PORTS               NAMES
6bcb9a610377        argos/netcat:latest    "sh -c 'echo hello >   7 seconds ago       Exited
(0) 6 seconds ago                       lonely_archimedes
➜  ~  docker diff 6b
A /world.txt
--------------
```

== Exportation et importation d'images
La commande `save` permet d'exporter une image présente en local :

```
[source]
--------------
➜  ~  docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
argos/netcat        latest              f472bd490861        14 minutes ago      95.96 MB
argos/demo          latest              dadf613a22d3        58 minutes ago      95.35 MB
mongo               latest              0e68275c469e        2 weeks ago         391.6 MB
debian              latest              f6fab3b798be        4 weeks ago         85.1 MB

➜  ~  docker save argos/netcat > image_avec_netcat.tar

➜  ~  ll
-rw-r--r-- 1 steph steph  97M déc.   9 17:02 image_avec_netcat.tar
--------------
```

Pour importer une image depuis un fichier, c'est la commande `load` :

```
[source]
--------------
➜  ~  docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
argos/demo          latest              dadf613a22d3        About an hour ago   95.35 MB
mongo               latest              0e68275c469e        2 weeks ago         391.6 MB
debian              latest              f6fab3b798be        4 weeks ago         85.1 MB

➜  ~  docker load < image_avec_netcat.tar
```

```
➜  ~  docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
argos/netcat        latest              f472bd490861        17 minutes ago      95.96 MB
argos/demo          latest              dadf613a22d3        About an hour ago   95.35 MB
mongo               latest              0e68275c469e        2 weeks ago         391.6 MB
debian              latest              f6fab3b798be        4 weeks ago         85.1 MB
--------------
```

```
== Historique
[source]
--------------
➜  ~  docker history argos/netcat
IMAGE               CREATED             CREATED BY                                      SIZE
f472bd490861        27 minutes ago      apt-get -y install netcat-traditional
615.5 kB
dadf613a22d3        About an hour ago   bash
10.24 MB
f6fab3b798be        4 weeks ago         /bin/sh -c #(nop) CMD [/bin/bash]               0 B
f10807909bc5        4 weeks ago         /bin/sh -c #(nop) ADD file:01b419e635eb6bec68   85.1
MB
511136ea3c5a        18 months ago                                                       0 B
--------------
```

```
== Dockerfile
Pour créer une image qui correspond aux besoins, c'est fastidieux et propice aux
erreurs.
Pour cela, il y a la possibilité d'automatiser la création d'images.

WARNING: Utiliser un répertoire dédié par Dockerfile

NOTE: Docker recommande l'utilisation de l'image de base Debian
(https://docs.docker.com/articles/dockerfile_best-practices/)
```

```
=== Exemple simple :
Cet exemple reprend la construction d'une image avec netcat qui écoute sur le port 1234.
Pour cela, créer un fichier `Dockerfile` avec le contenu suivant :

[source]
-------
FROM debian:wheezy # <1>

MAINTAINER Stéphane Deraco <stephane.deraco@dsi.cnrs.fr> # <2>

RUN apt-get update && apt-get install -y \ # <3>
    netcat-traditional \
    procps

CMD ["nc", "-l", "-p", "1234"] # <4>

EXPOSE 1234 # <5>
-------
<1> `FROM` indique quelle est l'image de base à utiliser
```

<2> `MAINTAINER` est juste indicatif
<3> `RUN` permet de lancer des commandes. On peut avoir plusieurs `RUN` dans un fichier.
<4> `CMD` indique la commande par défaut à lancer
<5> `EXPOSE` indique que cette image expose les ports suivants

Une fois le fichier `Dockerfile` créé, on peut lancer la construction avec la commande `build` :

[source]

➜ simple docker build -t argos/simple .
Sending build context to Docker daemon 2.56 kB
Sending build context to Docker daemon
Step 0 : FROM debian:wheezy
debian:wheezy: The image you are pulling has been verified
511136ea3c5a: Already exists
f10807909bc5: Already exists
f6fab3b798be: Already exists
Status: Image is up to date for debian:wheezy
 ---> f6fab3b798be
Step 1 : MAINTAINER Stéphane Deraco <stephane.deraco@dsi.cnrs.fr>
 ---> Running in e219524c0ba3
 ---> 6c4f5fe676fb
Removing intermediate container e219524c0ba3
Step 2 : RUN apt-get update && apt-get install -y   netcat-traditional   procps
 ---> Running in 0149937f3c47
Get:1 http://security.debian.org wheezy/updates Release.gpg [836 B]
Get:2 http://security.debian.org wheezy/updates Release [102 kB]
Get:3 http://http.debian.net wheezy Release.gpg [1655 B]
SNIP...
invoke-rc.d: policy-rc.d denied execution of start.
Setting up psmisc (22.19-1+deb7u1) ...
 ---> 81c7260a472c
Removing intermediate container 0149937f3c47
Step 3 : CMD nc -l -p 1234
 ---> Running in b02a36b71701
 ---> 41ed8f4b74e7
Removing intermediate container b02a36b71701
Step 4 : EXPOSE 1234
 ---> Running in b59936eb9d47
 ---> 338ec341a27d
Removing intermediate container b59936eb9d47
Successfully built 338ec341a27d

On peut voir dans la construction de l'image les différentes couches qui s'ajoutent.
Notre image est maintenant présente, on peut la lancer.

[source]

➜ simple docker images
REPOSITORY   TAG   IMAGE ID   CREATED   VIRTUAL SIZE
argos/simple   latest   338ec341a27d   2 minutes ago   95.46 MB
debian   latest   f6fab3b798be   5 weeks ago   85.1 MB
debian   wheezy   f6fab3b798be   5 weeks ago   85.1 MB

```
➜  simple  docker run -p 1234:1234 -d argos/simple
f5a378074e0b75ab4b7262481b3afa03be9b431606fa4077eef8b76f5c98d2f6

➜  simple  telnet localhost 1234
Trying ::1...
Connected to localhost.
Escape character is '^]'.
----
```

On peut voir l'historique de construction de cette image :

```
[source]
----
➜  simple  docker history argos/simple
IMAGE               CREATED             CREATED BY                                      SIZE
338ec341a27d        4 minutes ago       /bin/sh -c #(nop) EXPOSE map[1234/tcp:{}]       0 B
41ed8f4b74e7        4 minutes ago       /bin/sh -c #(nop) CMD [nc -l -p 1234]           0 B
81c7260a472c        4 minutes ago       /bin/sh -c apt-get update && apt-get install
10.36 MB
6c4f5fe676fb        4 minutes ago       /bin/sh -c #(nop) MAINTAINER Stéphane Deraco    0 B
f6fab3b798be        5 weeks ago         /bin/sh -c #(nop) CMD [/bin/bash]               0 B
f10807909bc5        5 weeks ago         /bin/sh -c #(nop) ADD file:01b419e635eb6bec68   85.1
MB
511136ea3c5a        18 months ago                                                       0 B
----
```

On retrouve les différentes commandes présentes dans le Dockerfile.

=== Exemple un peu moins simple

Un conteneur docker s'arrête dès que le processus lancé dans ce conteneur se termine. Cela est donc un inconvénient dans les cas où l'on souhaite lancer plusieurs processus, qui plus est en arrière-plan. Pour cela, il existe divers projet, dont http://supervisord.org/[Supervisord] qui répondent à ce besoin.

Cet exemple montre la mise en place d'un conteneur avec ssh et apache.

On écrit un nouveau fichier Dockerfile :

```
[source]
----
FROM debian:wheezy

# Exemple provenant de https://docs.docker.com/articles/using_supervisord/

MAINTAINER Exemple tiré du site de Docker

RUN apt-get update && apt-get install -y \
    openssh-server \
    apache2 \
    supervisor

RUN mkdir -p /var/lock/apache2 /var/run/apache2 /var/run/sshd /var/log/supervisor

COPY supervisord.conf /etc/supervisor/conf.d/supervisord.conf # <1>
```

```
EXPOSE 22 80 # <2>

CMD ["/usr/bin/supervisord"] # <3>
```

<1> `COPY` permet d'inclure dans l'image des fichiers ; ici, on ajoute un fichier de configuration pour supervisord
<2> 2 ports seront accessibles
<3> le processus principal est `supervisord`

Il faut également créer le fichier `supervisord.conf` :

[source,ini]

```
[supervisord]
nodaemon=true ; <1>

[program:sshd]
command=/usr/sbin/sshd -D ; <2>

[program:apache2]
command=/bin/bash -c "source /etc/apache2/envvars && exec /usr/sbin/apache2 -DFOREGROUND" ;
<3>
```

<1> indique à supervisord de tourner au premier plan
<2> commande pour lancer le démon SSH
<3> commande pour lancer Apache

Avec cette configuration, c'est supervisord qui va s'occuper de lancer les processus. Il permet
également de les surveiller, de les relancer si besoin, ...

Pour construire l'image, on lance un _build_, toujours en se plaçant dans le répertoire où se trouve le fichier Dockerfile et le fichier de configuration de supervisord :

[source]

```
➜  moinsSimple  docker build -t argos/demo2 .
Sending build context to Docker daemon 3.584 kB
Sending build context to Docker daemon
Step 0 : FROM debian:wheezy
 ---> f6fab3b798be
Step 1 : MAINTAINER Exemple tiré du site de Docker
 ---> Running in 5037387f5f98
 ---> 2bd2367ebbdf
Removing intermediate container 5037387f5f98
Step 2 : RUN apt-get update && apt-get install -y     openssh-server     apache2
supervisor
 ---> Running in 0bade040a671
Get:1 http://security.debian.org wheezy/updates Release.gpg [836 B]
# SNIP...
Setting up apache2 (2.2.22-13+deb7u3) ...
Setting up libswitch-perl (2.16-2) ...
Processing triggers for python-support ...
 ---> 2b21999e018b
Removing intermediate container 0bade040a671
Step 3 : RUN mkdir -p /var/lock/apache2 /var/run/apache2 /var/run/sshd /var/log/supervisor
```

```
 ---> Running in d026eb40c63d
 ---> d07d1c38a5b3
Removing intermediate container d026eb40c63d
Step 4 : COPY supervisord.conf /etc/supervisor/conf.d/supervisord.conf
 ---> c6b062c3cde4
Removing intermediate container 0cd21167b8e0
Step 5 : EXPOSE 22 80
 ---> Running in 9b413586ac97
 ---> c32ba4407b76
Removing intermediate container 9b413586ac97
Step 6 : CMD /usr/bin/supervisord
 ---> Running in 737c5802c66a
 ---> 19bec3ddf5d5
Removing intermediate container 737c5802c66a
Successfully built 19bec3ddf5d5
```

Notre image a été construite :

```
➜  moinsSimple  docker images argos/demo2
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
argos/demo2         latest              19bec3ddf5d5        4 minutes ago       190 MB
```

On peut maintenant lancer un conteneur, en mode détaché, et vérifier que SSH et Apache sont bien lancés :

```
➜  moinsSimple  docker run -d -P argos/demo2 # <1>
9e5eb88612b049333d830ab5b03fcc1578850125fc1ddcfa828fc3d10ab14b7e

➜  moinsSimple  docker logs 9e5
2014-12-12 13:13:08,395 CRIT Supervisor running as root (no user in config file)
2014-12-12 13:13:08,395 WARN Included extra file "/etc/supervisor/conf.d/supervisord.conf"
during parsing
2014-12-12 13:13:08,446 INFO RPC interface 'supervisor' initialized
2014-12-12 13:13:08,446 WARN cElementTree not installed, using slower XML parser for XML-RPC
2014-12-12 13:13:08,447 CRIT Server 'unix_http_server' running without any HTTP
authentication checking
2014-12-12 13:13:08,447 INFO supervisord started with pid 1
2014-12-12 13:13:09,450 INFO spawned: 'sshd' with pid 10
2014-12-12 13:13:09,451 INFO spawned: 'apache2' with pid 11
2014-12-12 13:13:10,512 INFO success: sshd entered RUNNING state, process has stayed up for
> than 1 seconds (startsecs)
2014-12-12 13:13:10,512 INFO success: apache2 entered RUNNING state, process has stayed up
for > than 1 seconds (startsecs)

➜  moinsSimple  docker ps
CONTAINER ID        IMAGE               COMMAND               CREATED
STATUS              PORTS                                        NAMES
9e5eb88612b0        argos/demo2:latest   "/usr/bin/supervisor   About a minute ago   Up
About a minute   0.0.0.0:49153->22/tcp, 0.0.0.0:49154->80/tcp   sick_poincare

➜  moinsSimple  telnet localhost 49153
Trying ::1...
```

```
Connected to localhost.
Escape character is '^]'.
SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2

➜  moinsSimple  curl localhost:49154
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
➜  moinsSimple
```

<1> l'option `-P` attribue tous les ports exposés par l'images à des ports aléatoires, que l'on peut identifier avec `docker ps` ou `docker inspect`. On aurait pu aussi utiliser par exemple `-p 20:20 -p 80:80`.

=== Reconstruction d'images

Les images sont en fait des empilements de couches (_layers_) successives. On le voit quand on demande l'historique d'une image :

[source]

```
➜  moinsSimple  docker history argos/demo2
IMAGE               CREATED             CREATED BY                                      SIZE
19bec3ddf5d5        14 minutes ago      /bin/sh -c #(nop) CMD [/usr/bin/supervisord]    0 B
c32ba4407b76        14 minutes ago      /bin/sh -c #(nop) EXPOSE map[22/tcp:{} 80/tcp   0 B
c6b062c3cde4        14 minutes ago      /bin/sh -c #(nop) COPY file:1f541edbe16baba58   179 B
d07d1c38a5b3        14 minutes ago      /bin/sh -c mkdir -p /var/lock/apache2 /var/ru   0 B
2b21999e018b        14 minutes ago      /bin/sh -c apt-get update && apt-get install
104.9 MB
2bd2367ebbdf        15 minutes ago      /bin/sh -c #(nop) MAINTAINER Exemple tiré du    0 B
f6fab3b798be        5 weeks ago         /bin/sh -c #(nop) CMD [/bin/bash]               0 B
f10807909bc5        5 weeks ago         /bin/sh -c #(nop) ADD file:01b419e635eb6bec68   85.1
MB
511136ea3c5a        18 months ago                                                       0 B
```

L'intérêt est que si on modifie le Dockerfile, alors toutes les couches précédent cette modification ne seront pas reconstruites. Par exemple, faisons la modification suivante, pour indiquer à supervisord un fichier de log (option `-l`) :

[source,diff]

```
--- Dockerfile  2014-12-11 10:03:43.874372513 +0100
+++ Dockerfile2 2014-12-12 14:22:30.298929916 +0100
@@ -15,4 +15,4 @@

 EXPOSE 22 80

-CMD ["/usr/bin/supervisord"]
+CMD ["/usr/bin/supervisord", "-l", "supervisord.log"]
```

On relance la construction :

[source]

```
➜  rebuild  docker build -t argos/demo3 .
Sending build context to Docker daemon 3.584 kB
Sending build context to Docker daemon
Step 0 : FROM debian:wheezy
 ---> f6fab3b798be
Step 1 : MAINTAINER Exemple tiré du site de Docker
 ---> Using cache
 ---> 2bd2367ebbdf
Step 2 : RUN apt-get update && apt-get install -y     openssh-server     apache2
supervisor
 ---> Using cache
 ---> 2b21999e018b
Step 3 : RUN mkdir -p /var/lock/apache2 /var/run/apache2 /var/run/sshd /var/log/supervisor
 ---> Using cache
 ---> d07d1c38a5b3
Step 4 : COPY supervisord.conf /etc/supervisor/conf.d/supervisord.conf
 ---> Using cache
 ---> eaca2b843728
Step 5 : EXPOSE 22 80
 ---> Using cache
 ---> 9d7b8de63ebc
Step 6 : CMD /usr/bin/supervisord -l supervisord.log
 ---> Running in 322e34179dca
 ---> 882e5e45ee4b
Removing intermediate container 322e34179dca
Successfully built 882e5e45ee4b
----
```

Les lignes intéressants sont celles indiquant _Using cache_. On peut voir que l'image de base, et le résultat de `apt-get` ont été réutilisés par exemple.

On peut vérifier que les changements ont bien été pris en compte :

[source]
```
➜  rebuild  docker run -d -P argos/demo3
4846580d32b1970a2fb7e6c97a17d0409583ec20c843822deba70b7ec4e18ba7

➜  rebuild  docker exec -it 48465 bash
root@4846580d32b1:/# tail supervisord.log
2014-12-12 13:59:08,258 CRIT Supervisor running as root (no user in config file)
2014-12-12 13:59:08,259 WARN Included extra file "/etc/supervisor/conf.d/supervisord.conf"
during parsing
2014-12-12 13:59:08,293 INFO RPC interface 'supervisor' initialized
2014-12-12 13:59:08,293 WARN cElementTree not installed, using slower XML parser for XML-RPC
2014-12-12 13:59:08,294 CRIT Server 'unix_http_server' running without any HTTP
authentication checking
2014-12-12 13:59:08,294 INFO supervisord started with pid 1
2014-12-12 13:59:09,297 INFO spawned: 'sshd' with pid 10
2014-12-12 13:59:09,304 INFO spawned: 'apache2' with pid 11
2014-12-12 13:59:10,334 INFO success: sshd entered RUNNING state, process has stayed up for
> than 1 seconds (startsecs)
2014-12-12 13:59:10,334 INFO success: apache2 entered RUNNING state, process has stayed up
for > than 1 seconds (startsecs)
root@4846580d32b1:/# exit
----
```