

*RF-232*

# Micronator

## boot2docker

Présentation

Premiers pas

Conteneurs

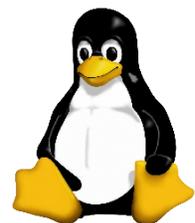
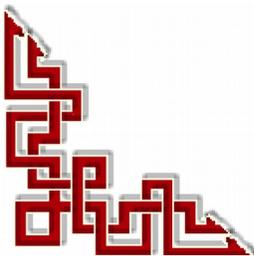
Images

Liaisons

Volumes

Docker Hub

Compose



© RF-232, Montréal 2015  
6447, avenue Jalobert, Montréal. Québec H1M 1L1

Tous droits réservés RF-232

### Licence publique générale GNU

Permission vous est donnée de copier, distribuer et/ou modifier ce document selon les termes de la **Licence publique générale GNU**, version 3, 29 juin 2007 publiée par la Free Software Foundation Inc; sans section inaltérable, sans texte de première page de couverture et sans texte de dernière page de couverture. Une copie de cette licence est incluse dans la section appelée **Licence publique générale GNU** de ce document, page: [134](#).

### AVIS DE NON-RESPONSABILITÉ

Ce document est uniquement destiné à informer. Les informations, ainsi que les contenus et fonctionnalités de ce document sont fournis sans engagement et peuvent être modifiés à tout moment. *RF-232* n'offre aucune garantie quant à l'actualité, la conformité, l'exhaustivité, la qualité et la durabilité des informations, contenus et fonctionnalités de ce document. L'accès et l'utilisation de ce document se font sous la seule responsabilité du lecteur ou de l'utilisateur.

*RF-232* ne peut être tenu pour responsable de dommages de quelque nature que ce soit, y compris des dommages directs ou indirects, ainsi que des dommages consécutifs résultant de l'accès ou de l'utilisation de ce document ou de son contenu.

Chaque internaute doit prendre toutes les mesures appropriées (*mettre à jour régulièrement son logiciel antivirus, ne pas ouvrir des documents suspects de source douteuse ou non connue*) de façon à protéger le contenu de son ordinateur de la contamination d'éventuels virus circulant sur la Toile.

---

#### Avertissement

*Bien que nous utilisions ici un vocabulaire issu des techniques informatiques, nous ne prétendons nullement à la précision technique de tous nos propos dans ce domaine.*

---

---

## *En un clin-d'oeil*

I-	Introduction.....	8
II-	Installation.....	12
III-	Rendre boot2docker permanent.....	19
IV-	Deuxième disque.....	25
V-	Login à distance.....	29
VI-	Docker Hub.....	31
VII-	Dockeriser des applications.....	33
VIII-	"Bonjour tout le monde" daemonisé.....	38
IX-	Client Docker.....	41
X-	Une application Web.....	43
XI-	Commandes diverses.....	46
XII-	Les images.....	50
XIII-	Nouvelle image.....	52
XIV-	Création d'une image.....	56
XV-	Dockerfile.....	59
XVI-	Liaisons inter-conteneurs.....	71
XVII-	Connexion par liens.....	75
XVIII-	Communication par liens.....	77
XIX-	Gestion des données dans les conteneurs.....	83
XX-	Conteneur-volume-de-données.....	87
XXI-	Sauvegarder, restaurer et migrer des volumes.....	89
XXII-	Nettoyage du serveur.....	96
XXIII-	Docker Hub.....	98
XXIV-	Recherche d'images.....	102
XXV-	Fonctionnalités de Docker Hub.....	103
XXVI-	Docker Compose.....	105
XXVII-	Démarrage rapide.....	109
XXVIII-	Création d'une image Docker.....	111
XXIX-	Compose et Wordpress.....	119

# Sommaire

<b>I-</b>	<b>Introduction.....</b>	<b>8</b>
	1. Docker.....	9
	2. Logiciels recommandés.....	10
	3. Particularités de ce document.....	10
	4. Commentaires et suggestions.....	11
<b>II-</b>	<b>Installation.....</b>	<b>12</b>
	1. Fichier ISO.....	12
	2. Machine virtuelle.....	12
	3. Connexion à distance avec PuTTY.....	12
	4. login.....	13
	5. root.....	14
	6. Commandes de base de docker.....	14
	7. Commandes Linux.....	16
	8. Infos de boot2docker.....	18
<b>III-</b>	<b>Rendre boot2docker permanent.....</b>	<b>19</b>
	1. Introduction.....	19
	2. Permanence de boot2docker.....	19
	3. Certificat.....	21
	4. Vérification.....	22
	5. Répertoire de stockage des conteneurs.....	23
	6. DNS.....	23
<b>IV-</b>	<b>Deuxième disque.....</b>	<b>25</b>
	1. Introduction.....	25
	2. Création du disque.....	25
	3. Permanence avec bootlocal.sh.....	27
	4. Conclusion de ce chapitre.....	28
<b>V-</b>	<b>Login à distance.....</b>	<b>29</b>
	1. Adresse IP du serveur boot2docker.....	29
	2. Login à distance.....	29
	3. root.....	30
<b>VI-</b>	<b>Docker Hub.....</b>	<b>31</b>
	1. Utilisation de Docker Hub?.....	31
	2. Création d'un compte Docker Hub.....	31
<b>VII-</b>	<b>Dockeriser des applications.....</b>	<b>33</b>
	1. Introduction.....	33
	2. Images et conteneurs.....	33
	3. "Bonjour tout le monde".....	33
	4. Historique d'une image.....	35
	5. Conteneur interactif.....	36

---

<b>VIII-</b>	<b>"Bonjour tout le monde" daemonisé.....</b>	<b>38</b>
	1. Commande.....	38
	2. Conséquences.....	39
<b>IX-</b>	<b>Client Docker.....</b>	<b>41</b>
	1. Introduction.....	41
	2. Client Docker.....	41
<b>X-</b>	<b>Une application Web.....</b>	<b>43</b>
	1. Exécution d'une application Web dans Docker.....	43
	2. Le conteneur de l'application Web.....	44
<b>XI-</b>	<b>Commandes diverses.....</b>	<b>46</b>
	1. Mappage du port interne du conteneur.....	46
	2. Journaux du conteneur.....	46
	3. Processus du conteneur.....	47
	4. Inspection du conteneur.....	47
	5. Arrêt du conteneur.....	47
	6. Redémarrage du conteneur.....	48
	7. Suppression du conteneur.....	49
<b>XII-</b>	<b>Les images.....</b>	<b>50</b>
	1. Introduction.....	50
	2. Liste des images sur l'hôte.....	50
<b>XIII-</b>	<b>Nouvelle image.....</b>	<b>52</b>
	1. Obtenir une nouvelle image.....	52
	2. Trouver des images.....	53
	3. Téléchargement (pull).....	54
<b>XIV-</b>	<b>Création d'une image.....</b>	<b>56</b>
	1. Introduction.....	56
	2. Mise à jour et commit d'une image.....	56
<b>XV-</b>	<b>Dockerfile.....</b>	<b>59</b>
	1. Introduction.....	59
	2. Répertoire et fichier Dockerfile.....	59
	3. Étiquette d'une image.....	67
	4. Téléversement d'une image.....	67
	5. Suppression d'une image.....	69
<b>XVI-</b>	<b>Liaisons inter-conteneurs.....</b>	<b>71</b>
	1. Introduction.....	71
	2. Connexion utilisant le mappage de port réseau.....	71
<b>XVII-</b>	<b>Connexion par liens.....</b>	<b>75</b>
	1. Introduction.....	75
	2. L'importance de donner des noms.....	75
<b>XVIII-</b>	<b>Communication par liens.....</b>	<b>77</b>
	1. Introduction.....	77
	2. Création d'un lien.....	77
	3. Avantages.....	78
	4. Variables d'environnement.....	79

5.	Notes importantes sur les variables d'environnement Docker.....	80
6.	Mise à jour du fichier /etc/hosts.....	81
<b>XIX-</b>	<b>Gestion des données dans les conteneurs.....</b>	<b>83</b>
1.	Introduction.....	83
2.	Volumes de données.....	83
3.	Création d'un volume de données.....	83
4.	Monter un répertoire hôte en tant que volume de données.....	84
5.	Monter un fichier de l'hôte en tant que volume de données.....	86
<b>XX-</b>	<b>Conteneur-volume-de-données.....</b>	<b>87</b>
1.	Introduction.....	87
2.	Mise en garde importante.....	87
3.	Création d'un conteneur-volume-de-données.....	87
4.	Suppression de conteneurs avec volume.....	88
<b>XXI-</b>	<b>Sauvegarder, restaurer et migrer des volumes.....</b>	<b>89</b>
1.	Rappel de la mise en garde.....	89
2.	Contenu du volume.....	89
3.	Sauvegarde d'un volume.....	91
4.	Ajout au volume.....	92
5.	Restauration d'un volume à son emplacement original.....	93
6.	Restauration d'un volume à un nouvel emplacement.....	95
<b>XXII-</b>	<b>Nettoyage du serveur.....</b>	<b>96</b>
1.	Rappel de la mise en garde.....	96
2.	Nettoyage.....	96
3.	Prochaine étape.....	97
<b>XXIII-</b>	<b>Docker Hub.....</b>	<b>98</b>
1.	Introduction.....	98
2.	Utilisation de Docker Hub.....	98
3.	Création d'un compte Docker Hub.....	99
4.	Login, fichier .dockercfg et logout.....	100
<b>XXIV-</b>	<b>Recherche d'images.....</b>	<b>102</b>
1.	Introduction.....	102
2.	Contribuer à Docker Hub.....	102
3.	Pousser une image vers Docker Hub.....	102
<b>XXV-</b>	<b>Fonctionnalités de Docker Hub.....</b>	<b>103</b>
1.	Introduction.....	103
2.	Registres privés.....	103
3.	Organisations et équipes.....	103
4.	Construction automatisée.....	103
5.	Déclencheur de construction.....	104
6.	Point Web d'accueil logiciel (webhook).....	104
7.	Prochaine étape.....	104
<b>XXVI-</b>	<b>Docker Compose.....</b>	<b>105</b>
1.	Remarque importante.....	105
2.	Introduction.....	105
3.	Utilisation.....	106
4.	Documentation.....	106

---

5.	Installation de Compose.....	106
6.	Commandes docker-compose disponibles.....	108
<b>XXVII-</b>	<b>Démarrage rapide.....</b>	<b>109</b>
1.	Répertoire de travail.....	109
2.	Fichier app.py.....	109
3.	Dépendances Python.....	110
<b>XXVIII-</b>	<b>Création d'une image Docker.....</b>	<b>111</b>
1.	Définition de l'environnement avec le fichier Dockerfile.....	111
2.	Définition des services avec le fichier docker-compose.yml.....	111
3.	Construction et exécution de l'application à l'aide de Compose.....	112
4.	Mode daemon.....	115
5.	Variables d'environnement.....	116
6.	Arrêt de Compose.....	117
<b>XXIX-</b>	<b>Compose et Wordpress.....</b>	<b>119</b>
1.	Introduction.....	119
2.	Téléchargement de Wordpress.....	119
3.	Fichier Dockerfile.....	120
4.	Fichier docker-compose.yml.....	120
5.	Fichier de configuration de Wordpress.....	121
6.	Génération du projet.....	123
7.	Config de Wordpress.....	124
8.	Arrêt des conteneurs.....	126
9.	Documentation de Compose.....	126
	Crédits.....	127

# I- Introduction

Ce document se veut une introduction au conteneurs **Docker**. Une connaissance de base en informatique et une certaine expérience de travail à la console sont requises. Quelques notions de Linux sont préférables pour la compréhension de certains termes et commandes utilisées. Il n'est pas nécessaire d'être un expert, le simple fait de vouloir devrait être amplement suffisant.

Les systèmes utilisés sont tous des machines virtuelles et roulent sous **VirtualBox**. Des liens sont fournis pour l'installation des logiciels nécessaires pour la création d'un environnement de travail avec lequel vous pourrez expérimenter tous les exemples de ce document.

Une première lecture en diagonale de ce document est suggéré pour avoir une vue d'ensemble de **boot2docker** et ce qu'il peut vous apporter pour faciliter votre travail.

- On débute par l'installation de boot2docker et la façon d'en faire un système d'exploitation permanent pour la gestion d'images et de conteneurs Docker.
- Les premiers pas permettent de se loguer à distance, comprendre un dépôt d'images, réaliser un "pull" et lancer un conteneur.
- L'apprentissage des images Docker est suivi d'une exploration plus détaillée. Ces chapitres vous familiarisent avec les commandes de base.
- Viennent ensuite les conteneurs et les liens qu'ils peuvent avoir avec l'extérieur et surtout entre-eux.
- Le système est étendu aux volumes qui sont en réalité de simples répertoires de l'hôte accessibles par les conteneurs.
- On améliore notre flux de travail avec **Docker Hub**.
- Pour terminer, une introduction à **Compose** permet la création d'images et conteneurs.

Ce document se veut une introduction à Docker et ne prétend nullement être une référence et faire de vous un expert mais simplement vous mettre en contact avec une technologie qui en est à ses premiers balbutiements organisés et dont la croissance fulgurante lui promet un avenir des plus spectaculaires.

La plupart des chapitres sont des traductions du [guide de l'utilisateur](#) de Docker avec une mise en page un peu différente et quelques détails supplémentaires.

Bonne lecture,

Michel-André

### 1. Docker

Référence: [http://fr.wikipedia.org/wiki/Docker\\_%28logiciel%29](http://fr.wikipedia.org/wiki/Docker_%28logiciel%29).



Présentement boot2docker, compilé en 32 bits, ne fonctionne qu'avec des systèmes 64 bits.

Docker est un [logiciel open source](#) qui automatise le déploiement d'applications dans des conteneurs logiciels. Selon la firme de recherche sur l'industrie, 451 Research, "Docker est un outil qui peut empaqueter une application et ses dépendances dans un conteneur virtuel qui pourra être exécuté sur n'importe quel serveur Linux". Ceci permet d'étendre la flexibilité et la portabilité d'exécution d'une application, que ce soit sur la machine locale, un cloud privé ou public, une machine nue, etc.

Docker étend le format de Conteneur Linux standard, [LXC](#), avec une [API](#) de haut niveau fournissant une solution de virtualisation qui exécute les processus de façon isolée. Docker utilise LXC, [cgroups](#), et le [noyau Linux](#) lui-même. Contrairement aux machines virtuelles traditionnelles, un conteneur Docker n'inclut pas de système d'exploitation, à la place il s'appuie sur les fonctionnalités du système d'exploitation fournies par l'infrastructure sous-jacente.

La technologie de conteneur de Docker peut être utilisée pour étendre des systèmes distribués de façon à ce qu'ils s'exécutent de manière autonome depuis une seule machine physique ou une seule instance par nœud; ce qui permet aux nœuds d'être déployés au fur et à mesure que les ressources sont disponibles, offrant un déploiement transparent et similaire aux [Paas](#) pour des systèmes comme [Apache Cassandra](#), [Riak](#) ou d'autres systèmes distribués.

#### 1.1. Historique

Docker a été développé comme un projet interne de dotCloud par Solomon Hykes, une société proposant une [Plate-forme en tant que service](#), avec les contributions d'Andrea Luzzardi et Francois-Xavier Bourlet, également employés de dotCloud. Docker est une évolution basée sur les technologies propriétaires de dotCloud, elles-mêmes construites sur des projets open-sources tels que Cloudlets.

Docker a été distribué en tant que projet open source à partir de mars 2013.

Au 18 novembre 2013, le projet a été mis en favoris plus de 7 300 fois sur [GitHub](#) (14e projet le plus populaire), avec plus de 900 forks et 200 contributeurs.

Au 9 mai 2014, le projet a été mis en favoris plus de 11 769 fois sur [GitHub](#), avec plus de 1 912 forks et 423 contributeurs.

## 2. Logiciels recommandés

### 2.1. VirtualBox

Logiciel de virtualisation: <https://www.virtualbox.org/>.

### 2.2. SME-9/64

Système d'exploitation Linux: [http://wiki.contribs.org/SME\\_Server:Download](http://wiki.contribs.org/SME_Server:Download).

### 2.3. digestIT 2004

Logiciel de calcul de somme de contrôle: <http://www.colonywest.us/digestit/>.

### 2.4. Notepad++

Éditeur de texte ASCII: <http://notepad-plus-plus.org/fr/>.

### 2.5. WinSCP

Logiciel de téléversement: <http://winscp.net/eng/docs/lang:fr>.

### 2.6. PuTTY

Logiciel d'accès SSH: <http://www.putty.org/>

## 3. Particularités de ce document

### 3.1. Notes au lecteur

\* Les captures d'écrans ne sont que des références.

\*\* Les informations écrites ont préséance sur celles retrouvées dans les captures d'écrans. Veiller à se référer aux différents tableaux lorsque ceux-ci sont présents.

### 3.2. Conventions

Toutes les commandes à entrer à la console sont en **gras**. Les affichages à surveiller sont en **rouge**, **bleu**, **orange** ou **magenta**.

```
# ping 192.168.1.149
192.168.1.149 is alive
#
```

Les liens de référence internet sont en **bleu** et ceux intra document en **bleu**.



Manipulation, truc ou ruse pour se tirer d'embarras.



Une recommandation ou astuce.



Une note.



Une étape, note ou procédure à surveiller.



Paragraphe non complété ou non vérifié.



Cet icône indique que cette commande est sur une seule ligne. Le **PDF** la mettra sur deux lignes avec un [CR] [LF] entre les deux. Il faudra donc copier la commande entière dans un éditeur de texte ASCII et la mettre sur une seule ligne avant de la copier à la console.

#### 4. Commentaires et suggestions

**RF-232** apprécie énormément échanger avec ses internautes. Vos commentaires et suggestions sont indispensables à l'amélioration de la documentation et du site **micronator.org**.

N'hésitez pas à nous transmettre vos commentaires et à nous signaler tout problème d'ordre technique que vous avez rencontré ou n'arrivez pas à résoudre. Tous vos commentaires seront pris en considération et nous vous promettons une réponse dans les plus brefs délais.



**Brancher les aînés,  
encourager l'Informatique Libre et la diffusion du savoir**







On vérifie l'utilisateur.

```
docker@boot2docker:~$ whoami
docker
docker@boot2docker:~$
```

## 5. root

Avec une connexion à distance, pour devenir **root**, il suffit de lancer la commande suivante:

```
docker@boot2docker:~$ sudo -s
root@boot2docker:/home/docker#
```

On vérifie l'utilisateur.

```
root@boot2docker:/home/docker# whoami
root
root@boot2docker:/home/docker#
```

On se rend dans le répertoire personnel de root.

```
root@boot2docker:/home/docker# cd
root@boot2docker:~#
```

On vérifie.

```
root@boot2docker:~# pwd
/root
root@boot2docker:~#
```

## 6. Commandes de base de docker

On peut afficher les commandes de base de docker.

```
root@boot2docker:~# docker --help
Usage: docker [OPTIONS] COMMAND [arg...]

A self-sufficient runtime for linux containers.

Options:
  --api-enable-cors=false      Enable CORS headers in the remote API
  -b, --bridge=""             Attach containers to a pre-existing network bridge
                              use 'none' to disable container networking
  --bip=""                    Use this CIDR notation address for the network
                              bridge's IP, not compatible with -b
  -D, --debug=false           Enable debug mode
  -d, --daemon=false          Enable daemon mode
  --dns=[]                    Force Docker to use specific DNS servers
  --dns-search=[]             Force Docker to use specific DNS search domains
  -e, --exec-driver="native" Force the Docker runtime to use a specific exec
                              driver
  --fixed-cidr=""             IPv4 subnet for fixed IPs (e.g. 10.20.0.0/16)
                              this subnet must be nested in the bridge subnet
  --fixed-cidr-v6=""          IPv6 subnet for fixed IPs (e.g.: 2001:a02b/48)
  (which is defined by -b or --bip)
```

## Installation

```
-G, --group="docker"      Group to assign the unix socket specified by -H
                          when running in daemon mode
                          use '' (the empty string) to disable setting of a
                          group
-g, --graph="/var/lib/docker"  Path to use as the root of the Docker runtime
-H, --host=[]             The socket(s) to bind to in daemon mode or connect
                          to in client mode, specified using one or more
                          tcp://host:port, unix:///path/to/socket,
                          fd://* or fd://socketfd.
-h, --help=false         Print usage
--icc=true               Allow unrestricted inter-container and Docker
                          daemon host communication
--insecure-registry=[]   Enable insecure communication with specified
                          registries (no certificate verification for HTTPS
                          and enable HTTP fallback) (e.g., localhost:5000
                          or 10.20.0.0/16)
--ip=0.0.0.0            Default IP address to use when binding container
                          ports
--ip-forward=true        Enable net.ipv4.ip_forward and IPv6 forwarding if
                          --fixed-cidr-v6 is defined. IPv6 forwarding may
                          interfere with your existing IPv6 configuration
                          when using Router Advertisement.
--ip-masq=true           Enable IP masquerading for bridge's IP range
--iptables=true         Enable Docker's addition of iptables rules
--ipv6=false            Enable IPv6 networking
-l, --log-level="info"   Set the logging level (debug, info, warn, error,
                          fatal)
--label=[]              Set key=value labels to the daemon (displayed in
                          `docker info`)
--mtu=0                 Set the containers network MTU if no value is
                          provided: default to the default route MTU or 1500
                          if no default route is available
-p, --pidfile="/var/run/docker.pid" Path to use for daemon PID file
--registry-mirror=[]    Specify a preferred Docker registry mirror
-s, --storage-driver="" Force the Docker runtime to use a specific storage
                          driver
--selinux-enabled=false Enable selinux support. SELinux does not presently
                          support the BTRFS storage driver
--storage-opt=[]        Set storage driver options
--tls=false             Use TLS; implied by --tlsverify flag
--tlscacert="/root/.docker/ca.pem" Trust only remotes providing a certificate signed
by the CA given here
--tlscert="/root/.docker/cert.pem" Path to TLS certificate file
--tlskey="/root/.docker/key.pem" Path to TLS key file
--tlsverify=false       Use TLS and verify the remote (daemon: verify
client, client: verify daemon)
-v, --version=false     Print version information and quit
```

### Commands:

```
attach    Attach to a running container
build     Build an image from a Dockerfile
commit    Create a new image from a container's changes
cp        Copy files/folders from a container's filesystem to the host path
create    Create a new container
diff      Inspect changes on a container's filesystem
events    Get real time events from the server
exec      Run a command in a running container
export    Stream the contents of a container as a tar archive
history   Show the history of an image
images    List images
import    Create a new filesystem image from the contents of a tarball
info      Display system-wide information
inspect   Return low-level information on a container or image
kill      Kill a running container
load      Load an image from a tar archive
login     Register or log in to a Docker registry server
logout    Log out from a Docker registry server
logs      Fetch the logs of a container
```

```
port      Lookup the public-facing port that is NAT-ed to PRIVATE_PORT
pause     Pause all processes within a container
ps        List containers
pull      Pull an image or a repository from a Docker registry server
push      Push an image or a repository to a Docker registry server
rename    Rename an existing container
restart   Restart a running container
rm        Remove one or more containers
rmi       Remove one or more images
run       Run a command in a new container
save      Save an image to a tar archive
search    Search for an image on the Docker Hub
start     Start a stopped container
stats     Display a live stream of one or more containers' resource usage statistics
stop      Stop a running container
tag       Tag an image into a repository
top       Lookup the running processes of a container
unpause   Unpause a paused container
version   Show the Docker version information
wait      Block until a container stops, then print its exit code
```

Run 'docker COMMAND --help' for more information on a command.  
root@boot2docker:~#



Les commandes ci-dessus sont seulement celles utilisées avec le préfixe **docker**.

```
root@boot2docker:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
root@boot2docker:~#
```

## 7. Commandes Linux

Les principales commandes Linux sont aussi disponibles.

```
root@boot2docker:~# echo $PATH
/
/home/docker/.local/bin:/usr/local/sbin:/usr/local/bin:/apps/bin:/usr/sbin:/usr/bin:/sbin:/bin
root@boot2docker:~#
```

Les commandes Linux disponibles à tous.

```
root@boot2docker:~# ls /bin/
addgroup      dd             getopt        lsmod         printenv      su
adduser       delgroup      grep          mkdir         ps            sync
ash           deluser      gunzip        mknod         pwd           tar
busybox       df            gzip          mktemp        rev           touch
busybox.suid  dmesg        hostname     more          rm            true
cat           dnsdomainname ipcalc       mount         rmdir        umount
chgrp         dumpkmap     kill         mv            rpm          uname
chmod         echo          linux32      netstat       sed          uncompress
chown         egrep        linux64      nice          setarch      usleep
cp           false        ln           pidof         sh           vi
cpio         fdflush     login        ping          sleep        watch
date         fgrep        ls           ping6         stty         zcat
root@boot2docker:~#
```

## Installation

```
root@boot2docker:~# ls /usr/bin/

[          exittc          nohup          tce-fetch.sh
[[        expr            nslookup      tce-load
ar        fdformat        od            tce-remove
arping    fgconsole       ondemand     tce-run
aubrsync  filetool.sh    openvt       tce-setdrive
aubusy    find           passwd       tce-setup
auchk     flock          patch        tce-size
aufhsm    fold          pgrep        tce-status
autoscan-devices free          pkill        tce-update
awk       fromISOfile    printf       tcemirror.sh
backup    ftpget         provides.sh  tee
basename  ftpput        readlink     telnet
bbcheck.sh fuser         realpath     test
bcrypt    getTime.sh    renice       text2lp0
bigHomeFiles.sh getdisklabel reset         tftp
bunzip2   getopt        resize       time
bzipcat   head          rev          timeout
bzip2     hexdump       rotdash     top
cal       history       rpm2cpio    tr
calc      hostid        script       traceroute
chkonboot.sh id            search.sh    tty
chrt      install       seekWinPartition udpsvd
chvt      ipcrm         select       uniq
cksum     ipc          seq          unix2dos
clear     killall       sethostname  unxz
cliorex  killall5     setkeycodes  unzip
cmp       last          setsid       uptime
comm      ldd           shasum       uuid
crontab   length        showbootcodes uuidgen
cut       less          sort         version
dc        loadfont      split        vi
deallocvt locale        start-stop-daemon vlock
diff      logger        strings      wc
dirname  logname       sudo         wget
nsdomainname logread       sudoedit     which
dos2unix  lsof          sum          who
du        md5sum        tail         whoami
dumpleases mesg          taskset      xargs
eject     microcom      tc-terminal-server xz
env       mkfifo        tce          xzcat
ether-wake mountables.sh tce-ab       yes
exitcheck.sh nc            tce-audit    zsync
root@boot2docker:~#
```

Les commandes Linux réservées à **root** et autres super-usagers.

```
root@boot2docker:~# ls /sbin/

adjtimex    fsck.ext2      loadcpufreq  modprobe     slogin
arp         fsck.ext3     loadkmap     mount.aufs   swapoff
aibusy     fsck.ext4     logread      mount.nfs    swapon
aumvdown   fsck.ext4dev  losetup      mount.vboxsf switch_root
auplink    getty         lsmod        nameif       sysctl
autologin  halt          lspcmcia     pccardctl   syslogd
blkid      hdparm        mke2fs       pivot_root  tune2fs
ctty-hack  hwclock      mkfs.ext2    poweroff     udevadm
depmod     ifconfig     mkfs.ext3    reboot       udevd
e2fsck    init         mkfs.ext4    rmmod       udhpcp
fdisk     insmod       mkfs.ext4dev route        udpsvd
freeramdisk klogd       mkswap       setconsole   umount.aufs
fsck      ldconfig     modinfo      start-stop-daemon vconfig
root@boot2docker:~#
```

## 8. Infos de boot2docker

On peut afficher les informations de **boot2docker**.

```
root@boot2docker:~# docker info

Containers: 0
Images: 0
Storage Driver: aufs
  Root Dir: /mnt/sda2/var/lib/docker/aufs
  Backing Filesystem: extfs
  Dirs: 0
Execution Driver: native-0.2
Kernel Version: 3.18.5-tinycore64
Operating System: Boot2Docker 1.5.0 (TCL 5.4); master : a66bce5 - Tue Feb 10 23:31:27 UTC 2015
CPUs: 2
Total Memory: 1.961 GiB
Name: boot2docker
ID: S3TT:EINM:MM3P:JLPV:57FZ:RLCA:KCAJ:BVKI:T2FH:R5EF:T7BK:MLEA
Debug mode (server): true
Debug mode (client): false
Fds: 11
Goroutines: 15
EventsListeners: 0
Init Path: /usr/local/bin/docker
Docker Root Dir: /mnt/sda2/var/lib/docker
root@boot2docker:~#
```

## III- Rendre boot2docker permanent

### 1. Introduction

Vu que nous avons amorcé depuis le **CD** de **boot2docker**, rien de ce qu'on fait ne sera permanent. Au prochain réamorçage, la machine redevient à son état initial.

### 2. Permanence de boot2docker

On peut rendre boot2docker permanent sur le disque de notre machine virtuelle en effectuant les étapes ci-dessous.

#### 2.1. Transfert de l'ISO vers le disque

Après un amorçage de la machine virtuelle avec l'**ISO** ou le **CD** de **boot2docker**, on copie le contenu de **/dev/cdrom** sur le disque de la machine virtuelle.

```
root@boot2docker:~# dd if=/dev/cdrom of=/dev/sda

47104+0 records in
47104+0 records out
24117248 bytes (23.0MB) copied, 4.730351 seconds, 4.9MB/s
root@boot2docker:~#
```

#### 2.2. Création de la partition sda2

On crée une nouvelle partition primaire sur la partition # 2 qui s'étendra sur tout le reste du disque.

```
root@boot2docker:~# fdisk /dev/sda

The number of cylinders for this disk is set to 8192.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
 1) software that runs at boot time (e.g., old versions of LILO)
 2) booting and partitioning software from other OSS
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (24-8192, default 24): Using default value 24 [Entrée]
Last cylinder or +size or +sizeM or +sizeK (24-8192, default 8192): Using default value 8192
[Entrée]
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table
root@boot2docker:~#
```

## 2.3. Formatage de la partition sda2

On formate la nouvelle partition en lui donnant un Label de **boot2docker-data**.

```
root@boot2docker:~# mkfs.ext4 -L boot2docker-data /dev/sda2

mke2fs 1.42.7 (21-Jan-2013)
Filesystem label=boot2docker-data
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
523264 inodes, 2091264 blocks
104563 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2143289344
64 block groups
32768 blocks per group, 32768 fragments per group
8176 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

root@boot2docker:~#
```

## 2.4. Montage de la partition sda2

Création du point de montage.

```
root@boot2docker:~# mkdir -p /mnt/sda2

root@boot2docker:~#
```

On vérifie.

```
root@boot2docker:~# ls -alsd /mnt/sda2

    0 drwxr-xr-x    2 root    root          40 Mar  6 22:07 /mnt/sda2
root@boot2docker:~#
```

Montage.

```
root@boot2docker:~# mount /dev/sda2 /mnt/sda2

root@boot2docker:~#
```

On vérifie.

```
root@boot2docker:~# df -h

Filesystem      Size      Used Available Use% Mounted on
rootfs          1.8G    101.5M      1.7G   6% /
tmpfs           1.8G    101.5M      1.7G   6% /
tmpfs          1004.3M         0    1004.3M   0% /dev/shm
cgroup          1004.3M         0    1004.3M   0% /sys/fs/cgroup
tmpfs           1.8G    101.5M      1.7G   6% /var/lib/docker/aufs
/dev/sda2       7.7G    18.0M      7.3G   0% /mnt/sda2
root@boot2docker:~#
```



## 4. Vérification

On devient **root**.

```
docker@boot2docker:~$ sudo -s
root@boot2docker:/home/docker#
```

On se rends dans le répertoire personnel de **root**.

```
root@boot2docker:/home/docker# cd
root@boot2docker:~#
```

### 4.1. Partition sda2

```
root@boot2docker:~# df -h
Filesystem      Size      Used Available Use% Mounted on
rootfs          1.8G      87.0M    1.7G    5% /
tmpfs           1.8G      87.0M    1.7G    5% /
tmpfs           1004.3M    0      1004.3M   0% /dev/shm
/dev/sda2       7.7G      32.6M    7.3G    0% /mnt/sda2
cgroup          1004.3M    0      1004.3M   0% /sys/fs/cgroup
/dev/sda2       7.7G      32.6M    7.3G    0% /mnt/sda2/var/lib/docker/aufs
root@boot2docker:~#
```

### 4.2. Fichier toto

```
root@boot2docker:~# ls -als /mnt/sda2
total 28
 4 drwxr-xr-x  5 root   root           4096 Mar  6 22:14 .
 0 drwxr-xr-x  6 root   root           120 Mar  6 22:14 ..
16 drwx----- 2 root   root          16384 Mar  6 22:05 lost+found
 4 drwxrwxrwt  4 root   staff          4096 Mar  6 22:14 tmp
 0 -rw-r--r--  1 root   root            0 Mar  6 22:12 toto
 4 drwxr-xr-x  3 root   root           4096 Mar  6 22:14 var
root@boot2docker:~#
```

On remarque qu'une nouvelle arborescence **var** a été créée,. C'est celle-ci qui rend boot2docker permanent.

### 4.3. Images & conteneurs

Pour l'instant on n'a aucune image...

```
root@boot2docker:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
root@boot2docker:~#
```

... ni conteneur.

```
root@boot2docker:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS              NAMES
root@boot2docker:~#
```

## 5. Répertoire de stockage des conteneurs

Plus tard, boot2docker placera ses conteneurs dans le répertoire: `/mnt/sda2/var/lib/docker/containers/`.

```
root@boot2docker:~# ls -als /mnt/sda2/var/lib/docker/

total 52
 4 drwxr-xr-x 10 root   root   4096 Mar  6 22:15 .
 4 drwxr-xr-x  4 root   root   4096 Mar  6 22:14 ..
 4 drwxr-xr-x  5 root   root   4096 Mar  6 22:15 aufs
 4 drwx----- 2 root   root   4096 Mar  6 22:15 containers
 4 drwx----- 3 root   root   4096 Mar  6 22:15 execdriver
 4 drwx----- 2 root   root   4096 Mar  6 22:15 graph
 4 drwx----- 2 root   root   4096 Mar  6 22:15 init
 8 -rw-r--r--  1 root   root  5120 Mar  6 22:15 linkgraph.db
 4 -rw-----  1 root   root    19 Mar  6 22:15 repositories-aufs
 4 drwx----- 2 root   root   4096 Mar  6 22:15 tmp
 4 drwx----- 2 root   root   4096 Mar  6 22:15 trust
 4 drwx----- 2 root   root   4096 Mar  6 22:15 volumes

root@boot2docker:~#
```

## 6. DNS

Vous pouvez spécifier l'IP du serveur DNS de votre domaine à boot2docker. Lorsqu'on lancera une image, boot2docker lui passera directement un argument spécifiant cette adresse. Si boot2docker ne le fait pas, votre image docker pourra lancer un ping d'une adresse IP mais ne pourra pas faire la traduction pour un nom de domaine.

Nous allons spécifier à boot2docker l'IP du serveur DNS de notre domaine, **192.168.1.1**.

### 6.1. /mnt/sda2/var/lib/boot2docker/profile

Les différents paramètres et arguments à passer à boot2docker sont mis dans le fichier `/mnt/sda2/var/lib/boot2docker/profile`.



Dans ce fichier, il faut qu'il y ait seulement un seul `other_args=`. S'il y a en a plusieurs, boot2docker va prendre seulement le dernier et ignorera les autres. Si plusieurs arguments sont nécessaires, on les entre sur la même ligne et ils sont séparés par un espace. Il n'y a qu'une seule paire de guillemets: un guillemet avant le premier argument et un après le dernier.

#### 6.1.1. Argument `--dns`



On ajoute les lignes suivante à `/mnt/sda2/var/lib/boot2docker/profile` en lançant la commande ci-dessous.

```
cat >> /mnt/sda2/var/lib/boot2docker/profile << FIN
#
# MAR 2015-03-16_10h59 HAE
# Pour passer l'IP du serveur DNS aux images
#
# Reference: http://wiki.contribs.org/Docker
# other_args="--dns 208.67.220.220 --dns 208.67.220.222"
#
other_args="--dns 192.168.1.1"

FIN
```

On vérifie qu'il n'y a qu'un seul `other_args=`.

```
root@boot2docker:~# cat /mnt/sda2/var/lib/boot2docker/profile
#
# MAR 2015-03-16_10h59 HAE
# Pour passer l'IP du serveur DNS aux images
#
# Reference: http://wiki.contribs.org/Docker
# other_args="--dns 208.67.220.220 --dns 208.67.220.222"
#
other_args="--dns 192.168.1.1"
root@boot2docker:~#
```

# IV- Deuxième disque

## 1. Introduction

Nous allons créer un deuxième disque qui servira pour le stockage temporaire de fichiers de travail afin de minimiser la taille du disque principal. Dans un autre document, il servira pour la réception d'un volumineux fichier **tar.gz**.

## 2. Création du disque

On éteint la machine virtuelle, on crée un deuxième disque dans VirtualBox pour notre serveur boot2docker. Une grandeur de **4 GB** devrait être suffisante pour l'instant. On relance boot2docker, on se logue, on devient **root** et on se rend dans son répertoire personnel.

On vérifie si le deuxième disque a bien été créé.

```
root@boot2docker:~# cat /proc/partitions
major minor #blocks name
250      0    482292 zram0
 8       0   8388608 sda
 8       1    23552 sda1
 8       2   8365056 sda2
 8       8  4194304 sdb
11      0   1048575 sr0
root@boot2docker:~#
```

### 2.1. Partition primaire

On crée une partition primaire sur le deuxième disque.

```
root@boot2docker:~# fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI, OSF or GPT disklabel
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that the previous content
won't be recoverable.

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-522, default 1): Using default value 1 [Entrée]
Last cylinder or +size or +sizeM or +sizeK (1-522, default 522): Using default value 522
[Entrée]
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table
root@boot2docker:~#
```

Vérification.

```
root@boot2docker:~# cat /proc/partitions
major minor #blocks name
250      0    482292 zram0
  8       0   8388608 sda
  8       1    23552 sda1
  8       2   8365056 sda2
  8      16  4194304 sdb
  8      17  4192933 sdb1
 11       0   1048575 sr0
root@boot2docker:~#
```

## 2.2. Formatage

On formate la partition du nouveau disque en **ext4**.

```
root@boot2docker:~# mkfs.ext4 /dev/sdb1
mke2fs 1.42.7 (21-Jan-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
262144 inodes, 1048233 blocks
52411 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1073741824
32 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

root@boot2docker:~#
```

## 2.3. Point de montage

On crée un point de montage pour la nouvelle partition.

```
root@boot2docker:~# mkdir /mnt/sdb1
root@boot2docker:~#
```

On vérifie.

```
root@boot2docker:~# ls -alsd /mnt/sdb1
    0 drwxr-xr-x  2 root  root          40 Mar  6 22:48 /mnt/sdb1
root@boot2docker:~#
```

On monte la nouvelle partition.

```
root@boot2docker:~# mount /dev/sdb1 /mnt/sdb1
root@boot2docker:~#
```

On vérifie.

```
root@boot2docker:~# df -h
Filesystem      Size      Used Available Use% Mounted on
rootfs          1.8G      87.0M      1.7G   5% /
tmpfs           1.8G      87.0M      1.7G   5% /
tmpfs           1004.3M    0          1004.3M  0% /dev/shm
/dev/sda2       7.7G      32.6M      7.3G   0% /mnt/sda2
cgroup          1004.3M    0          1004.3M  0% /sys/fs/cgroup
/dev/sda2       7.7G      32.6M      7.3G   0% /mnt/sda2/var/lib/docker/aufs
/dev/sdb1       3.9G      8.0M      3.6G   0% /mnt/sdb1
root@boot2docker:~#
```

On y crée un fichier.

```
root@boot2docker:~# touch /mnt/sdb1/titi
root@boot2docker:~#
```

On vérifie.

```
root@boot2docker:~# ls -als /mnt/sdb1/titi
0 -rw-r--r-- 1 root root 0 Mar 6 22:51 /mnt/sdb1/titi
root@boot2docker:~#
```

### 3. Permanence avec bootlocal.sh

Si on réamorce, on verra que la partition `/dev/sdb1` n'est pas montée. On va y remédier en créant un script qui montera cette partition à la fin d'un amorçage.



Comme toutes les distributions Linux, boot2docker possède un fichier personnalisable qu'il exécute à la fin d'un amorçage. Ce fichier est `/var/lib/boot2docker/bootlocal.sh`.

#### 3.1. Création du fichier

Par défaut, ce fichier n'existe pas.

```
root@boot2docker:~# ls -als /var/lib/boot2docker/bootlocal.sh
ls: /var/lib/boot2docker/bootlocal.sh: No such file or directory
root@boot2docker:~#
```

On crée le fichier.

```
root@boot2docker:~# touch /var/lib/boot2docker/bootlocal.sh
root@boot2docker:~#
```

On vérifie.

```
root@boot2docker:~# ls -als /var/lib/boot2docker/bootlocal.sh
0 -rw-r--r-- 1 root root 0 Mar 6 23:01 /var/lib/boot2docker/bootlocal.sh
root@boot2docker:~#
```

On rend le fichier exécutable.

```
root@boot2docker:~# chmod +x /var/lib/boot2docker/bootlocal.sh
root@boot2docker:~#
```

On vérifie.

```
root@boot2docker:~# ls -als /var/lib/boot2docker/bootlocal.sh
0 -rwxr-xr-x 1 root root 0 Mar 6 23:01 /var/lib/boot2docker/bootlocal.sh
root@boot2docker:~#
```



Dans boot2docker, le shell bash exécutable est **/bin/sh**.

On dépose le script dans le fichier.



```
cat > /var/lib/boot2docker/bootlocal.sh << EOF
#!/bin/sh
mount /dev/sdb1 /mnt/sdb1
EOF
```

On vérifie.

```
root@boot2docker:~# cat /var/lib/boot2docker/bootlocal.sh
#!/bin/sh
mount /dev/sdb1 /mnt/sdb1
root@boot2docker:~#
```

On réamorce.

```
root@boot2docker:~# reboot
root@boot2docker:~#
```

On vérifie la permanence du montage de la partition.

```
root@boot2docker:~# df -h
Filesystem      Size      Used Available Use% Mounted on
rootfs          1.8G      87.0M    1.7G    5% /
tmpfs           1.8G      87.0M    1.7G    5% /
tmpfs           1004.3M    0      1004.3M  0% /dev/shm
/dev/sda2       7.7G      32.6M    7.3G    0% /mnt/sda2
cgroup          1004.3M    0      1004.3M  0% /sys/fs/cgroup
/dev/sdb1       3.9G      8.0M     3.6G    0% /mnt/sdb1
/dev/sda2       7.7G      32.6M    7.3G    0% /mnt/sda2/var/lib/docker/aufs
root@boot2docker:~#
```

La partition **/dev/sdb1** est effectivement montée.

## 4. Conclusion de ce chapitre

On peut maintenant expérimenter avec boot2docker.



Seul l'utilisateur **docker** peut installer des packages supplémentaires avec **tce-load**. Il peut installer **Midnight Commander** (le fameux **mc**). Il faut noter que cette installation ne sera pas permanente; elle disparaîtra lors d'un réamorçage.

# V- Login à distance

## 1. Adresse IP du serveur boot2docker

À la console du serveur boot2docker, on affiche les adresses utilisées.

```

root@boot2docker:~# ifconfig

docker0  Link encap:Ethernet  HWaddr 56:84:7A:FE:97:99
          inet addr:172.17.42.1  Bcast:0.0.0.0  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0     Link encap:Ethernet  HWaddr 08:00:27:5F:FB:FF
          inet addr:192.168.1.191  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe5f:fbff/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:852 errors:0 dropped:0 overruns:0 frame:0
          TX packets:93 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:56146 (54.8 KiB)  TX bytes:13288 (12.9 KiB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:48 errors:0 dropped:0 overruns:0 frame:0
          TX packets:48 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3456 (3.3 KiB)  TX bytes:3456 (3.3 KiB)

root@boot2docker:~#

```

## 2. Login à distance

Sur un poste de travail, on lance **PuTTY** pour se connecter à distance.

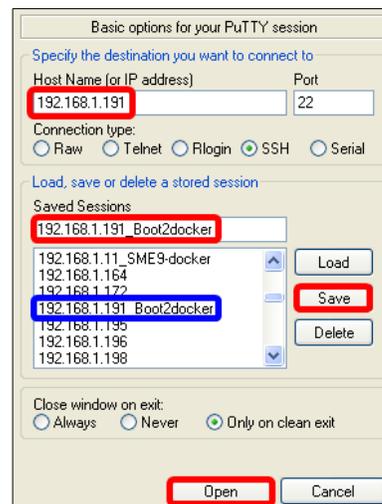
**Host Name for IP address:** *192.168.1.191*

**Port:** laisser le port 22 par défaut

**Saved Sessions:** On entre *192.168.1.191\_Boot2docker*

**Save** pour sauvegarder cette configuration.

**Open** pour se connecter.





# VI- Docker Hub

## 1. Utilisation de Docker Hub?

Référence: <http://docs.docker.com/userguide/dockerhub/>.

**Docker Hub** est la plate-forme centrale pour Docker. Il accueille les images publiques Docker et fournit des services pour aider à construire et gérer un environnement Docker.

Ce chapitre fournit une introduction à [Docker Hub](#) y compris la façon de créer un compte.

Docker Hub est une ressource centralisée pour travailler avec Docker et ses composantes. Il aide à collaborer avec des collègues et tirer le meilleur parti de Docker. Pour ce faire, Docker Hub fournit des services tels que:

- Hébergement d'image Docker.
- Authentification d'un utilisateur.
- Automatisation de la construction d'une image.
- Outils de flux de travail tels que les déclencheurs de construction et "hooks" Web.
- Intégration avec **GitHub** et **BitBucket**.

Pour utiliser Docker Hub, on doit d'abord s'inscrire et créer un compte. Ne vous inquiétez pas, la création d'un compte est simple et gratuite.

## 2. Création d'un compte Docker Hub

Il existe deux façons de créer un compte Docker:

- via le Web, ou
- via la ligne de commande.

### 2.1. Via le Web

Se rendre à l'adresse: <https://hub.docker.com/account/signup/>.

Remplir le formulaire d'inscription en choisissant un **nom d'utilisateur**, un **mot de passe** et une **adresse courriel valide**.

On peut également s'abonner à la liste de diffusion hebdomadaire Docker qui offre des informations intéressantes sur ce qui se passe dans le monde de Docker.

The screenshot shows the Docker Hub sign-up interface. At the top, there is a dark button labeled "Sign up with Github". Below it, the text "Or with Email" is centered. The form consists of three input fields: the first contains "michelandre", the second contains a masked password "\*\*\*\*\*", and the third contains "michelandre". Below the third field is a checkbox with a checkmark and the text "Yes! I want the weekly newsletter!". At the bottom right is an orange "Sign up" button.

## 2.2. Confirmer son adresse courriel

Après avoir rempli le formulaire, on vérifie ses courriels et un message de bienvenue demandera de confirmer son adresse courriel pour activer son compte.

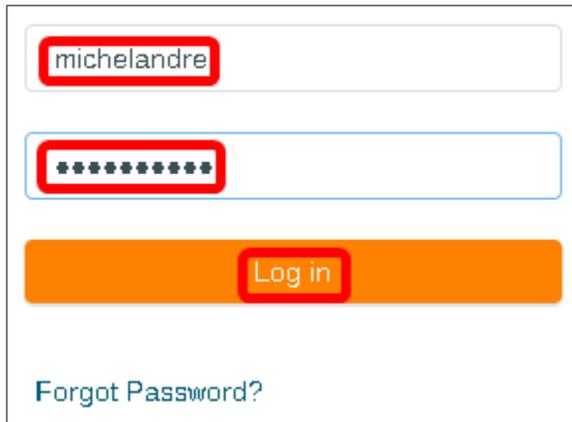
### Confirm you email.



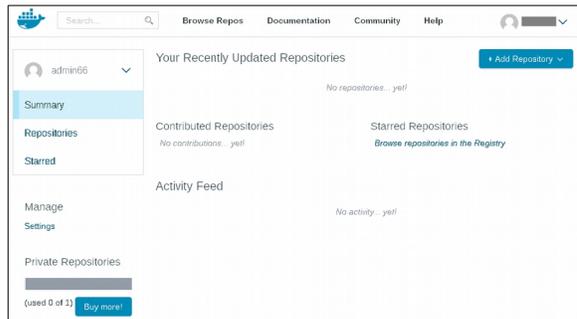
### Log In.



Enter son nom d'utilisateur et son mot de passe | **Log In.**



Voilà, on est logué chez **Docker Hub.**



## 2.3. S'inscrire via la ligne de commande

On peut également créer un compte Docker Hub via la ligne de commande en utilisant **docker login**.

```
root@boot2docker:~# docker login
Username: michelandre
Password: mot-de-passe
Email: michelandre arobas micronator.org
Account created. Please use the confirmation link we sent to your e-mail to activate it.
root@boot2docker:~#
```

Pareil à l'inscription via le Web, il confirmer son adresse pour activer son compte

Le compte **Docker Hub** est maintenant prêt à être utilisé.

# VII- Dockeriser des applications

## 1. Introduction

Si on n'utilise pas **boot2docker** comme le seul système d'exploitation sur notre serveur à distance, alors on tape **sudo** devant les commandes Docker figurant dans les exemples de cette documentation.



Dans nos exemples, on se logue toujours à distance avec l'utilisateur standard **docker** puis on change d'utilisateur pour devenir **root**. L'utilisateur **docker** nécessite **sudo** devant une commande Docker mais pas l'utilisateur **root**.

## 2. Images et conteneurs

Nous n'avons présentement aucune image ni conteneur.

Pour voir les images qui sont stockées sur notre serveur.

```
root@boot2docker:~# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
root@boot2docker:~#				

Pour voir les conteneurs.

```
root@boot2docker:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			

## 3. "Bonjour tout le monde"

Maintenant, essayons boot2docker. Nous lançons la commande suivante.

```
root@boot2docker:~# docker run ubuntu:14.04 /bin/echo 'Bonjour tout le monde'
```

**docker** le binaire qu'on veut utiliser.

**run** la commande de ce logiciel (*boot2docker*) qu'on veut exécuter.

La commande créera et lancera un conteneur qui contiendra l'image qu'on lui passe en paramètre.

**ubuntu:14.04** le nom de l'image qu'on veut utiliser.

En premier, **boot2docker** recherche toujours localement l'image à utiliser. Vu que le serveur ne contient pas cette image localement, boot2docker contactera le **dépôt public** de **Docker Hub**, recherchera une image **ubuntu** version **14.04** et, une fois trouvée, la téléchargera localement sur notre serveur, créera un nouveau conteneur pour cette image et le lancera.

**/bin/echo** la commande à exécuter à l'intérieur du conteneur.

**'Bonjour tout le monde'** le paramètre à passer à la commande **/bin/echo**.

Qu'est-il arrivé à notre conteneur après l'exécution de la commande? Eh bien les conteneurs Docker ne fonctionnent que tant que la(*les*) commande(s) qu'on leur spécifie est(*sont*) active(s). Ici, dès que **echo 'Bonjour tout le monde'** a été exécutée, le conteneur est arrêté.

### 3.1. Affichage à l'écran

```
root@boot2docker:~# docker run ubuntu:14.04 /bin/echo 'Bonjour tout le monde'
Unable to find image 'ubuntu:14.04' locally
511136ea3c5a: Pull complete
511136ea3c5a: Download complete
f3c84ac3a053: Download complete
a1a958a24818: Download complete
9fec74352904: Download complete
d0955f21bf24: Download complete
Status: Downloaded newer image for ubuntu:14.04
Bonjour tout le monde
root@boot2docker:~#
```

### 3.2. Détails de la commande

La commande complète.

```
docker run ubuntu:14.04 /bin/echo 'Bonjour tout le monde'
```

boot2docker n'a pas trouvé l'image localement.

```
Unable to find image 'ubuntu:14.04' locally
```

boot2docker a trouvé l'image dans le dépôt **Docker Hub**. L'image comprend plusieurs couches, il les a toutes téléchargées.

```
511136ea3c5a: Pull complete
511136ea3c5a: Download complete
f3c84ac3a053: Download complete
a1a958a24818: Download complete
9fec74352904: Download complete
d0955f21bf24: Download complete
```

boot2docker a combiné toutes les couches et en a fait qu'une seule image.

```
Status: Downloaded newer image for ubuntu:14.04
```

boot2docker a créé le conteneur, y a inséré l'image, lancé le conteneur et enfin exécuté la commande spécifiée. On voit le résultat affiché à l'écran.

```
Bonjour tout le monde
```

### 3.3. Conséquences de l'exécution de la commande

#### 3.3.1. Images

Les images ont subi plusieurs mises à jour. La dernière est **ubuntu:latest**.

```
root@boot2docker:~# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
ubuntu	trusty-20150320	d0955f21bf24	14 hours ago	188.3 MB
ubuntu	14.04	d0955f21bf24	14 hours ago	188.3 MB
ubuntu	14.04.2	d0955f21bf24	14 hours ago	188.3 MB
<b>ubuntu</b>	<b>latest</b>	<b>d0955f21bf24</b>	14 hours ago	188.3 MB
ubuntu	trusty	d0955f21bf24	14 hours ago	188.3 MB

```
root@boot2docker:~#
```

boot2docker a créé plusieurs conteneurs intermédiaires avant la construction de la dernière image mais il les a tous effacés. Le conteneur qui apparaît ici est le conteneur créé pour exécuter la commande **echo**.

```
root@boot2docker:~# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
2c9f697c0f2b	ubuntu:14.04	"/bin/echo 'Bonjour jolly_jones"	29 minutes ago	Exited

```
root@boot2docker:~#
```

Le dernier (*last*) conteneur qui a roulé sur le serveur.

```
root@boot2docker:~# docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
2c9f697c0f2b	ubuntu:14.04	"/bin/echo 'Bonjour jolly_jones"	26 minutes ago	Exited

```
root@boot2docker:~#
```

### 4. Historique d'une image

On peut toujours afficher l'historique d'une image qu'elle soit disponible localement ou sur le dépôt **Docker Hub**. L'historique donne les détails de la création et des mises à jour d'une image.

```
docker@boot2docker:~$ docker history ubuntu:latest
```

IMAGE	CREATED	CREATED BY	SIZE
d0955f21bf24	5 days ago	/bin/sh -c #(nop) CMD ["/bin/bash"]	0 B
9fec74352904	5 days ago	/bin/sh -c sed -i 's/^#\s*\ (deb.*universe\)\$/	1.895 kB
a1a958a24818	5 days ago	/bin/sh -c echo '#!/bin/sh' > /usr/sbin/polic	194.5 kB
f3c84ac3a053	5 days ago	/bin/sh -c #(nop) ADD file:777fad733fc954c0c1	188.1 MB
511136ea3c5a	21 months ago		0 B

```
docker@boot2docker:~$
```

## 5. Conteneur interactif

Lançons **docker run** de nouveau. Cette fois on spécifie une nouvelle commande i.e. **/bin/bash** à exécuter dans le conteneur.

```
docker run -t -i ubuntu:14.04 /bin/bash
```

Ici, nous allons à nouveau préciser la commande **docker run** et lancer l'image **ubuntu:14.04**. Mais nous allons également passer deux arguments supplémentaires à **docker run**: **-t** et **-i**.

- L'argument **-t** (*terminal*) attribue un **pseudo-tty** ou pseudo-terminal à l'intérieur de notre nouveau conteneur.
- L'argument **-i** (*interactif*) nous permet de faire une connexion interactive en saisissant **STDIN** (*l'entrée standard - le clavier*) du conteneur.
- **/bin/bash**, cette commande va lancer un **shell Bash** l'intérieur de notre nouveau conteneur.

Quand notre conteneur sera lancé, nous pourrons voir que nous avons une invite de commande à l'intérieur:

Avant de lancer une image, on peut afficher tous les arguments qu'on peut passer à **docker run**.

```
root@boot2docker:~# docker run --help
Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
Run a command in a new container
-a, --attach=[]          Attach to STDIN, STDOUT or STDERR.
--add-host=[]           Add a custom host-to-IP mapping (host:ip)
...
-i, --interactive=false  Keep STDIN open even if not attached
...
-t, --tty=false          Allocate a pseudo-TTY
...
-w, --workdir=""        Working directory inside the container
root@boot2docker:~#
```

On lance notre nouvelle commande.

```
root@boot2docker:~# docker run -t -i ubuntu:14.04 /bin/bash
root@921a97de6575:/#
```

Nous pouvons voir que nous avons une invite de commande **root@921a97de6575:/#** à l'intérieur du conteneur.

Essayons l'exécution de certaines commandes Linux à l'intérieur de notre conteneur.

```
root@921a97de6575:/# pwd
/
root@921a97de6575:/#
```

```
root@921a97de6575:/# ls /
bin  dev  home  lib64  mnt  proc  run  srv  tmp  var
boot  etc  lib  media  opt  root  sbin  sys  usr
root@921a97de6575:/#
```

Nous avons lancé **pwd** pour afficher le chemin de notre répertoire courant et vu que nous sommes dans le répertoire **/root**. Nous avons également fait une liste du contenu du répertoire racine **"/** qui montre que nous sommes bien sur un serveur **Linux**.

On peut expérimenter à l'intérieur de ce conteneur et lorsqu'on a terminé, on utilise la commande **exit** ou **[Ctrl] [D]** pour sortir du conteneur.

```
root@921a97de6575:~/# exit
exit
root@boot2docker:~#
```

Comme pour notre conteneur précédent, une fois le processus **shell Bash** terminé, le conteneur est arrêté.

### 5.1. Conséquences

Aucune nouvelle image n'a été créée.

```
root@boot2docker:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
ubuntu              14.04              d0955f21bf24       15 hours ago       188.3 MB
ubuntu              14.04.2            d0955f21bf24       15 hours ago       188.3 MB
ubuntu              latest             d0955f21bf24       15 hours ago       188.3 MB
ubuntu              trusty             d0955f21bf24       15 hours ago       188.3 MB
ubuntu              trusty-20150320    d0955f21bf24       15 hours ago       188.3 MB
root@boot2docker:~#
```

Un nouveau conteneur a été créé.

```
root@boot2docker:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS              NAMES
921a97de6575       ubuntu:14.04       "/bin/bash"        24 minutes ago     Exited
(0) About a minute ago                determined_colden
2c9f697c0f2b       ubuntu:14.04       "/bin/echo 'Hello wo About an hour ago     Exited
(0) About an hour ago                jolly_jones
root@boot2docker:~#
```

Le dernier conteneur roulé sur le serveur.

```
root@boot2docker:~# docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS              NAMES
921a97de6575       ubuntu:14.04       "/bin/bash"        24 minutes ago     Exited (0)
About a minute ago                determined_colden
root@boot2docker:~#
```

## VIII- "Bonjour tout le monde" daemonisé

### 1. Commande

Un conteneur qui exécute une commande et qui s'arrête est efficace mais n'est pas très utile. Créons un conteneur qui roule en tant que **daemon** comme la plupart des applications que nous allons probablement rouler avec Docker.

Encore une fois, nous allons utiliser la commande **docker run**:



```
root@boot2docker:~# docker run -d ubuntu:14.04 /bin/sh -c "while true; do echo Bonjour tout le monde; sleep 1; done"
d759edea4441d78c114787dea117aa44e6f1f22749639bb250b9aae3f5b4ead0
root@boot2docker:~#
```

Un instant! Où est passé notre "**Bonjour tout le monde**"? Voyons ce que nous avons lancé, la sortie devrait nous être familière.

- Nous avons lancé **docker run** mais cette fois nous avons spécifié un nouvel argument: **-d**. L'argument **-d** spécifie à Docker de lancer le conteneur mais de rouler celui-ci en arrière-plan, de le **daemoniser**.
- Nous avons également précisé d'utiliser la même image: **ubuntu:14.04**.
- Enfin, nous avons spécifié une commande à exécuter: `/bin/sh -c "while true; do echo Bonjour tout le monde; sleep 1; done"`.

C'est un **daemon** des plus bizarres; un script shell qui fait un écho de "**Bonjour tout le monde**", pour toujours.

Pourquoi ne voyons-nous pas "**Bonjour tout le monde**"? À la place, Docker a retourné une très longue chaîne de caractères: `d759edea4441d78c114787dea117aa44e6f1f22749639bb250b9aae3f5b4ead0`.

Cette très longue chaîne est appelée l'**IDentifiant du conteneur**. Il identifie de manière unique un conteneur afin que nous puissions travailler avec celui-ci.



L'**IDentifiant du conteneur** est très long. Plus tard, on verra un **IDentifiant** beaucoup plus court et les façons de nommer les conteneurs pour rendre leur manipulation plus facile.

Nous pouvons utiliser cet **IDentifiant du conteneur** pour voir ce qui se passe avec notre **daemon "Bonjour tout le monde"**.

Tout d'abord, assurons-nous que notre conteneur est en cours d'exécution. Nous pouvons le vérifier avec la commande **docker ps**. Cette commande interroge Docker pour afficher certaines informations.

```
root@boot2docker:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS              NAMES
d759edea4441       ubuntu:14.04       "/bin/sh -c 'while t
seconds           distracted_wilson   6 seconds ago      Up 5
root@boot2docker:~#
```

Ici, nous pouvons voir notre conteneur daemonisé. **docker ps** a retourné des informations utiles à ce sujet, une variante plus courte de son IDentifiant, **d759edea4441**. Nous pouvons aussi savoir quelle image a été utilisée pour le construire, **ubuntu:14.04**, la commande **"/bin/sh -c 'while t...** qui en cours d'exécution, son statut et un nom attribué automatiquement, **distracted\_wilson**.



Docker attribue automatiquement un nom à tous les conteneurs qu'il lance. Plus tard on verra comment on peut choisir ses propres noms.

Nous savons maintenant que notre conteneur roule mais est-il en train de faire ce que nous lui avons demandé? Pour le savoir, nous allons regarder à l'intérieur du conteneur en utilisant la commande **docker logs**.

On peut utiliser le nom du conteneur que Docker lui a attribué.

```
root@boot2docker:~# docker logs distracted_wilson

Bonjour tout le monde
Bonjour tout le monde
Bonjour tout le monde
Bonjour tout le monde
...
root@boot2docker:~#
```

**docker logs** regarde à l'intérieur du conteneur et retourne sa sortie standard [STDOUT] pour notre commande, **"Bonjour tout le monde"**.

- Merveilleux! Notre daemon travaille et nous venons de créer notre première application dockerisée.
- Nous avons établi que nous pouvions créer nos propres conteneurs.

On peut maintenant arrêter notre conteneur daemonisé. La commande **docker stop** demande à Docker d'arrêter, d'une manière conforme, le conteneur en cours d'exécution. S'il réussit, il retournera le nom du conteneur qu'il vient d'arrêter.

```
root@boot2docker:~# docker stop distracted_wilson

distracted_wilson
root@boot2docker:~#
```

Vérifions si l'arrêt a bien fonctionné avec la commande **docker ps**.

```
root@boot2docker:~# docker ps

CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS              NAMES
root@boot2docker:~#
```

Notre conteneur s'est arrêté de la bonne façon.

## 2. Conséquences

Aucune image n'a été créée.

```
root@boot2docker:~# docker images

REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
ubuntu              14.04              d0955f21bf24       16 hours ago       188.3 MB
ubuntu              14.04.2            d0955f21bf24       16 hours ago       188.3 MB
ubuntu              latest             d0955f21bf24       16 hours ago       188.3 MB
ubuntu              trusty             d0955f21bf24       16 hours ago       188.3 MB
ubuntu              trusty-20150320    d0955f21bf24       16 hours ago       188.3 MB
root@boot2docker:~#
```

Un nouveau conteneur a été créé.

```
root@boot2docker:~# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
<b>d759edea4441</b> (137) 28 seconds ago	ubuntu:14.04	"/bin/sh -c 'while t <b>distracted_wilson</b>	About a minute ago	Exited
921a97de6575 (0) 49 minutes ago	ubuntu:14.04	"/bin/bash" determined_colden	About an hour ago	Exited
2c9f697c0f2b (0) 2 hours ago	ubuntu:14.04	"/bin/echo 'Hello wo jolly_jones	2 hours ago	Exited

```
root@boot2docker:~#
```

Le dernier conteneur lancé.

```
root@boot2docker:~# docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
<b>d759edea4441</b> (137) 31 seconds ago	ubuntu:14.04	"/bin/sh -c 'while t <b>distracted_wilson</b>	About a minute ago	<b>Exited</b>

```
root@boot2docker:~#
```

# IX- Client Docker

## 1. Introduction

Référence: <http://docs.docker.com/userguide/usingdocker/>.

Dans le chapitre [Dockeriser des applications](#), nous avons lancé nos premiers conteneurs en utilisant **docker run**:

- un conteneur que nous avons lancé de manière interactive au premier plan et
- un conteneur **daemonisé** qui roulait en arrière-plan.

Dans ces processus, nous avons appris plusieurs commandes Docker:

- **docker ps** - pour lister les conteneurs.
- **docker logs** - pour afficher la sortie standard d'un conteneur.
- **docker stop** - pour arrêter un conteneurs en cours d'exécution.



Une autre façon d'apprendre les commandes de Docker est le [tutoriel interactif](#).

## 2. Client Docker

Le client Docker est simple. Chaque action que vous pouvez entreprendre avec Docker est une commande et chaque commande peut prendre une série d'arguments.

```
Usage: [sudo] docker [OPTIONS] COMMAND [arg...]
```

Exemple:

```
docker run -i -t ubuntu /bin/bash
```

Voyons une commande en action en utilisant **docker version** pour afficher des informations sur la version du client Docker actuellement installé ainsi que sur celle du daemon.

Cette commande n'affichera pas seulement la version du client Docker et du daemon que vous utilisez mais aussi la version de **Go** (*le langage de programmation de Docker*).

```
root@boot2docker:~# docker version
Client version: 1.5.0
Client API version: 1.17
Go version (client): go1.4.1
Git commit (client): a8a31ef
OS/Arch (client): linux/amd64
Server version: 1.5.0
Server API version: 1.17
Go version (server): go1.4.1
Git commit (server): a8a31ef
root@boot2docker:~#
```

## 2.1. Ce que le client Docker peut faire

Comme vu au paragraphe [Commandes de base de docker](#), nous pouvons afficher toutes les commandes disponibles du client Docker en exécutant docker sans aucune option.

```
root@boot2docker:~# docker -h

Usage: docker [OPTIONS] COMMAND [arg...]
...
Options:
  --api-enable-cors=false      Enable CORS headers in the remote API
  -b, --bridge=""             Attach containers to a pre-existing network bridge
                               use 'none' to disable container networking
...
Commands:
  attach      Attach to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
...
Run 'docker COMMAND --help' for more information on a command.
root@boot2docker:~#
```

## 2.2. Usage

Vous pouvez également examiner l'usage spécifique d'une commande.

Tapez **docker** suivi d' un [commande] pour voir son utilisation.

```
root@boot2docker:~# docker attach

docker: "attach" requires 1 argument. See 'docker attach --help'.
root@boot2docker:~#
```

Vous pouvez également passer l'argument **--help** qui permet d'afficher le texte d'aide et tous les arguments disponibles.

```
root@boot2docker:~# docker attach --help

Usage: docker attach [OPTIONS] CONTAINER

Attach to a running container

  --help=false          Print usage
  --no-stdin=false     Do not attach STDIN
  --sig-proxy=true     Proxy all received signals to the process (non-TTY mode only).
                       SIGCHLD, SIGKILL, and SIGSTOP are not proxied.
root@boot2docker:~#
```



Vous pouvez consulter la liste complète des commandes Docker:

<http://docs.docker.com/reference/commandline/cli/>.

# X- Une application Web

## 1. Exécution d'une application Web dans Docker

Jusqu'à présent, aucun des conteneurs que nous avons roulés n'a fait de tâches particulièrement utiles. Allons plus loin en lançant une application Web roulant sous Docker.

Pour notre expérimentation Web nous allons exécuter une application **Python Flask**. Commençons par une commande **docker run**.

```
root@boot2docker:~# docker run -d -P training/webapp python app.py

Unable to find image 'training/webapp:latest' locally
Pulling repository training/webapp
31fa814ba25a: Download complete
511136ea3c5a: Download complete
f10ebce2c0e1: Download complete
82cdea7ab5b5: Download complete
5dbd9cb5a02f: Download complete
74fe38d11401: Download complete
64523f641a05: Download complete
0e2afc9aad6e: Download complete
e8fc7643ceb1: Download complete
733b0e3dbcee: Download complete
a1feb043c441: Download complete
e12923494f6a: Download complete
a15f98c46748: Download complete
Status: Downloaded newer image for training/webapp:latest
2842be864bba6896ed040ab4df7d4af4db204f56b060b0ba6f70fb17dee30f4e
root@boot2docker:~#
```

Passons en revue ce que notre commande a fait.

Nous avons spécifié deux arguments: **-d** et **-P**. Nous avons déjà vu l'argument **-d** qui spécifie à Docker d'exécuter le conteneur en l'arrière-plan. L'argument **-P** est nouveau et demande à Docker de **mapper** tous les ports réseau utilisés par notre application Web, de l'intérieur du conteneur vers l'extérieur i.e. vers le serveur hôte.

Analysons notre application Web.

- Nous avons spécifié une image, **training/webapp**. Cette image est une image pré-construite qui contient une application Web simple, **Python Flask**. Vu que l'image n'était pas disponible localement, elle a été téléchargée depuis le dépôt **Docker Hub**.
- Enfin, nous avons spécifié la commande **python app.py** à être exécutée par notre conteneur. C'est cette commande qui lance notre application Web.



Pour de plus amples détails sur la commande **docker run** voir la [référence générale des commandes](#) et la [référence de la commande docker run](#).

## 2. Le conteneur de l'application Web

Affichons certaines informations sur le conteneur en utilisant la commande **docker ps**.

```
root@boot2docker:~# docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
2842be864bba	<b>PORTS</b> training/webapp:latest	NAMES "python app.py"	2 hours ago	Up 2
minutes	<b>0.0.0.0:49153-&gt;5000/tcp</b>	naughty_mestorf		

```
root@boot2docker:~#
root@boot2docker:~#
```

On peut voir que nous avons spécifié un nouvel argument, **-l** à la commande **docker ps**. Il indique à la commande **docker ps** de fournir les détails du dernier (*last*) conteneur lancé.



Par défaut, la commande **docker ps** ne fournit des informations que sur les conteneurs qui sont en exécution. Si on veut voir les conteneurs qui sont arrêtés, on utilise l'argument **-a**.

Nous pouvons voir les mêmes détails que ceux que nous avons vus dans la section [Conséquences](#) lors de notre premier conteneur dockerisé mais avec un ajout important dans la colonne des **PORTS**.

```
PORTS
0.0.0.0:49153->5000/tcp
```

Lorsque nous avons spécifié l'argument **-P** à la commande **docker run**, boot2docker a mappé les ports ouverts par l'image vers des ports du serveur hôte.



Nous en apprendrons davantage sur la façon de mapper les ports lorsque nous verrons comment construire une image.

Dans le cas présent, Docker a mappé le port **5000** (le port par défaut de *Python Flask*) vers le port **49153** du serveur hôte. Les liaisons de port réseau (*port bindings*) sont configurables dans Docker.

Dans notre exemple, l'argument **-P** est un raccourci pour **-p 5000** qui mappe le port **5000** à l'intérieur du conteneur vers un port élevé sur l'hôte Docker (de la gamme de **49153** à **65535**).

Nous pouvons également lier les ports d'un conteneur vers des ports spécifiques du serveur hôte en utilisant l'argument **-p**.

Exemple:

```
docker run -d -p 5000:5000 training/webapp python app.py
```

Ceci mapperait le port **5000** à l'intérieur de notre conteneur vers le port **5000** du serveur hôte.

Vous vous demandez peut-être pourquoi ne voudrions-nous pas tout simplement utiliser un mappage de ports dans un rapport 1:1 (*port-x-intérieur:port-x-serveur*) plutôt que le mappage vers les ports du haut? Un mappage 1:1 a la contrainte de ne pouvoir mapper qu'une seule fois chacun des ports de l'hôte local. Supposons que vous voulez tester simultanément deux applications **Python**. Chacune de ces applications serait liée au port **5000** à l'intérieur de son propre conteneur. Sans le mappage de port, vous ne pourriez accéder qu'à une seule application à la fois.

## Une application Web

Nous allons utiliser Firefox pour examiner le port **49153** du serveur hôte afin de nous connecter à l'application Web du conteneur.

On trouve l'adresse **IP** du serveur hôte.

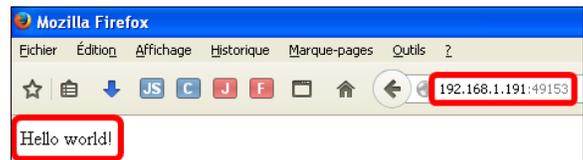
```
root@boot2docker:~# ifconfig eth0

eth0      Link encap:Ethernet  HWaddr 08:00:27:5F:FB:FF
          inet addr:192.168.1.191  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe5f:fbff/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:348475 errors:0 dropped:0 overruns:0 frame:0
          TX packets:111606 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:306219586 (292.0 MiB)  TX bytes:7861873 (7.4 MiB)

root@boot2docker:~#
```

On lance Firefox et on se rend à l'adresse **192.168.1.191:49153**.

Notre application **Python** fonctionne et le port **5000** du conteneur a bien été mappé vers le port **59153** du serveur hôte.



# XI- Commandes diverses

## 1. Mappage du port interne du conteneur

On peut utiliser la commande **docker ps** pour afficher le port interne qui est mappé vers le port externe.

```
root@boot2docker:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
2842be864bba	training/webapp:latest	"python app.py"	2 hours ago	Up 10 minutes
	0.0.0.0:49153->5000/tcp	naughty_mestorf		

```
root@boot2docker:~#
```

Docker possède un raccourci que nous pouvons aussi utiliser, **docker port**.

Pour utiliser **docker port**, nous spécifions l'Identifiant ou le nom de notre conteneur puis le port privé (*interne au conteneur*) pour lequel nous recherchons le port public (*externe au conteneur*) correspondant.

```
root@boot2docker:~# docker port 2842be864bba 5000
```

0.0.0.0:49153

```
root@boot2docker:~#
```

Aussi, sans spécifier aucun ports.

```
root@boot2docker:~# docker port naughty_mestorf
```

5000/tcp -> 0.0.0.0:49153

```
root@boot2docker:~#
```

La commande nous renseigne que le port privé **5000** est mappé sur le port public **49153** du serveur hôte.

## 2. Journaux du conteneur

Voyons ce qui se passe avec notre application Web en utilisant une autre commande que nous avons apprise précédemment, **docker logs** pour afficher les journaux (*logs*).

```
root@boot2docker:~# docker logs -f naughty_mestorf
```

```
* Running on http://0.0.0.0:5000/
192.168.1.129 - - [21/Mar/2015 21:10:26] "GET / HTTP/1.1" 200 -
192.168.1.129 - - [21/Mar/2015 21:10:26] "GET /favicon.ico HTTP/1.1" 404 -
```

Nous avons ajouté un nouvel argument, **-f**. Cet argument spécifie à **docker logs** d'agir comme la commande Linux **tail -f** et d'afficher la sortie standard [STDOUT] du conteneur. Nous pouvons alors voir les journaux de **Flask** montrant l'application fonctionnant avec le port **5000** de même que les entrées pour les accès **HTTP**.



[CTL] [c] pour sortir de la commande.

### 3. Processus du conteneur

En plus des journaux, nous pouvons également examiner les processus en cours à l'intérieur du conteneur, à l'aide de la commande **docker top**.

```
root@boot2docker:~# docker top naughty_mestorf

PID                USER              COMMAND
16506              root              python app.py
root@boot2docker:~#
```

Nous pouvons voir notre commande **python app.py** qui est le seul processus en cours à l'intérieur du conteneur.

### 4. Inspection du conteneur

Enfin, nous pouvons aller encore plus en profondeur dans notre conteneur en utilisant **docker inspect**. Cette commande retourne un [hachage JSON](#) (*JavaScript Object Notation*) de la configuration de même que d'autres informations sur l'état d'un conteneur Docker.

```
root@boot2docker:~# docker inspect naughty_mestorf

[{"AppArmorProfile": "",
  "Args": [
    "app.py"
  ],
  "Config": {
    "AttachStderr": false,
    "AttachStdin": false,
    "AttachStdout": false,
    "Cmd": [
      "python",
      "app.py"
    ],
    "CpuShares": 0,
    ...
  }
}
root@boot2docker:~#
```

Nous pouvons également raffiner l'information que nous voulons en demandant un élément spécifique, l'adresse **IP** du conteneur.

```
root@boot2docker:~# docker inspect -f '{{ .NetworkSettings.IPAddress }}' naughty_mestorf

172.17.0.6
root@boot2docker:~#
```

### 5. Arrêt du conteneur

Nous avons vu l'application Web au travail. Arrêtons-la à l'aide de la commande d'arrêt **docker stop** suivi du nom de notre container, **naughty\_mestorf**.

```
root@boot2docker:~# docker stop naughty_mestorf

naughty_mestorf
root@boot2docker:~#
```

On peut utiliser la commande **docker ps** pour vérifier si le conteneur a été arrêté.

```
root@boot2docker:~# docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
2842be864bba	training/webapp:latest	"python app.py"	9 hours ago	Exited

```
(137) 2 minutes ago
naughty_mestorf
root@boot2docker:~#
```

## 6. Redémarrage du conteneur

Juste après avoir arrêté le conteneur, vous recevez un appel qu'un autre développeur en a encore besoin.

Vous avez deux choix: créer un nouveau conteneur ou redémarrer l'ancien. Voyons le redémarrage de notre conteneur.

On lance le conteneur.

```
root@boot2docker:~# docker start naughty_mestorf
```

```
naughty_mestorf
root@boot2docker:~#
```

À nouveau, on exécute **docker ps -l** pour savoir si le conteneur est en cours d'exécution

```
root@boot2docker:~# docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
2842be864bba	training/webapp:latest	"python app.py"	9 hours ago	Up 9

```
seconds 0.0.0.0:49154->5000/tcp naughty_mestorf
root@boot2docker:~#
```

Le conteneur est bien en exécution mais on remarque que le port public a changé et est maintenant **49154**. Le port privé est toujours **5000**.

On navigue jusqu'à l'URL du conteneur pour voir si l'application répond.

L'application répond au nouveau port.



On consulte le journal du conteneur.

```
root@boot2docker:~# docker logs -f naughty_mestorf
```

```
* Running on http://0.0.0.0:5000/
192.168.1.129 - - [21/Mar/2015 21:10:26] "GET / HTTP/1.1" 200 -
192.168.1.129 - - [21/Mar/2015 21:10:26] "GET /favicon.ico HTTP/1.1" 404 -
* Running on http://0.0.0.0:5000/
192.168.1.129 - - [22/Mar/2015 01:48:14] "GET / HTTP/1.1" 200 -
192.168.1.129 - - [22/Mar/2015 01:48:15] "GET /favicon.ico HTTP/1.1" 404 -
```

Le journal du conteneur indique qu'il a bien reçu la demande du fureteur et qu'il lui a répondu, **200**.



Docker dispose aussi d'une commande de redémarrage, **docker restart** qui exécute un arrêt puis redémarre le conteneur.

## 7. Suppression du conteneur

Votre collègue vient vous dire qu'ils ont maintenant terminé avec le conteneur et qu'ils n'en auront plus besoin. Nous allons donc le supprimer en utilisant la commande **docker rm**.

```
root@boot2docker:~# docker rm naughty_mestorf
Error response from daemon: Conflict, You cannot remove a running container. Stop the
container before attempting removal or use -f
FATA[0000] Error: failed to remove one or more containers
root@boot2docker:~#
```

Que se passe-t-il?

Nous ne pouvons pas supprimer un conteneur qui est en cours d'exécution. Ceci vous empêche de supprimer accidentellement un contenant en cours d'exécution et dont vous pourriez avoir encore besoin.

Essayons encore mais cette fois, après avoir arrêter le conteneur.

Arrêt du conteneur.

```
root@boot2docker:~# docker stop naughty_mestorf
naughty_mestorf
root@boot2docker:~#
```

Vérification de l'arrêt.

```
root@boot2docker:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS              NAMES
root@boot2docker:~#
```

Le conteneur est arrêté, on peut maintenant les supprimer.

```
root@boot2docker:~# docker rm naughty_mestorf
naughty_mestorf
root@boot2docker:~#
```

Vérification.

```
root@boot2docker:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS              NAMES
d759edea4441       ubuntu:14.04       "/bin/sh -c 'while t
(137) 27 hours ago   distracted_wilson   27 hours ago       Exited
921a97de6575       ubuntu:14.04       "/bin/bash"        29 hours ago       Exited
(0) 28 hours ago   determined_colden
2c9f697c0f2b       ubuntu:14.04       "/bin/echo 'Hello wo
(0) 30 hours ago   jolly_jones        30 hours ago       Exited
root@boot2docker:~#
```

Notre conteneur a été arrêté et supprimé. Il ne reste que les conteneurs utilisées au chapitre ["Bonjour tout le monde" daemonisé](#).



Rappelez-vous que la suppression d'un conteneur est définitive!

Jusqu'à présent, nous n'utilisons seulement que des images **que nous avons téléchargées depuis Docker Hub**.

Dans le prochain chapitre, nous explorerons la création de nos propres images.

# XII- Les images

## 1. Introduction

Référence: <http://docs.docker.com/userguide/dockerimages/>.

Depuis le début de notre apprentissage de **boot2docker**, nous avons découvert que les images Docker sont la base des conteneurs. Dans les chapitres précédents, nous avons utilisé des images Docker qui existaient déjà: l'image **ubuntu:14.04** et l'image **training/webapp**.

Nous avons aussi appris que Docker entrepose les images téléchargées sur notre serveur hôte boot2docker. Si une image n'est pas déjà présente sur l'hôte, elle va être téléchargée depuis un registre, par défaut le [registre Docker Hub](#).

Dans les prochains chapitres, nous allons explorer un peu plus les images Docker.

- Gérer et travailler localement avec des images sur notre serveur hôte.
- Créer des images de base.
- Téléverser des images vers le registre **Docker Hub**.

## 2. Liste des images sur l'hôte

Commençons par énumérer les images que nous avons localement sur notre hôte en utilisant la commande **docker images**.

```
docker@boot2docker:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
ubuntu	14.04	d0955f21bf24	2 days ago	188.3 MB
ubuntu	14.04.2	d0955f21bf24	2 days ago	188.3 MB
ubuntu	latest	d0955f21bf24	2 days ago	188.3 MB
ubuntu	trusty	d0955f21bf24	2 days ago	188.3 MB
ubuntu	trusty-20150320	d0955f21bf24	2 days ago	188.3 MB
training/webapp	latest	31fa814ba25a	9 months ago	278.8 MB

```
docker@boot2docker:~$
```

Nous pouvons voir les images que nous avons utilisées précédemment. Chacune a été téléchargée depuis le registre **Docker Hub** lorsque nous avons lancé un conteneur qui utilisait cette image.

Dans la liste des images, nous pouvons voir trois éléments essentiels de nos images.

- De quel registre (*REPOSITORY*) elles proviennent: **ubuntu** et **training/webapp**.
- Les étiquettes (*TAG*) pour chacune des images: **14.04**, **14.04.2**, **latest**...
- L'Identifiant (*IMAGE ID*) de chaque image.

Un registre détient potentiellement de multiples variantes d'une image. Dans le cas de notre image **ubuntu**, nous pouvons voir de multiples variantes: **14.04**, **14.04.2**, **latest**... Chaque variante est identifiée par une étiquette et vous pouvez vous référer à une image étiquetée comme ci-dessous:

```
ubuntu:14.04
```

## Les images

---

Quand nous lançons un **run** d'une image, nous nous référons à une image étiquetée.

```
docker run -t -i ubuntu:14.04 /bin/bash
```

Si nous voulions exécuter l'image **Ubuntu 14.04.2**.

```
docker run -t -i ubuntu:14.04.2 /bin/bash
```

Si vous ne spécifiez pas de variante i.e. juste **ubuntu**, Docker utilisera par défaut l'image **ubuntu:latest**.



Nous vous recommandons de toujours utiliser une image et son étiquette: **ubuntu:12.04**. De cette façon, vous savez toujours exactement quelle variante d'une image est utilisée.

## XIII- Nouvelle image

### 1. Obtenir une nouvelle image

Comment pouvons-nous obtenir de nouvelles images? Docker téléchargera automatiquement une image qu'on veut utiliser et qui n'est pas présente localement sur l'hôte. Ce téléchargement peut potentiellement ajouter un certain temps avant le lancement du conteneur.

Si nous voulons télécharger une image, nous pouvons le faire à l'aide de la commande **docker pull**. À titre d'exemple, nous allons télécharger une image de **CentOS**.

```
docker@boot2docker:~$ docker pull centos

5b12ef8fd570: Pull complete
88f9454e60dd: Pull complete
511136ea3c5a: Already exists
centos:latest: The image you are pulling has been verified. Important: image verification is a tech preview feature and should not be relied on to provide security.
Status: Downloaded newer image for centos:latest
docker@boot2docker:~$
```

Nous pouvons voir que chaque couche de l'image a été téléchargée. Nous pouvons exécuter un conteneur de cette image et nous n'aurons plus à attendre le téléchargement de celle-ci avant de la lancer.

On vérifie l'image pour vérifier que toutes les couches ont été assemblées en une seule.

```
docker@boot2docker:~$ docker images

REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
ubuntu              14.04              d0955f21bf24       2 days ago         188.3 MB
ubuntu              14.04.2            d0955f21bf24       2 days ago         188.3 MB
ubuntu              latest             d0955f21bf24       2 days ago         188.3 MB
ubuntu              trusty             d0955f21bf24       2 days ago         188.3 MB
ubuntu              trusty-20150320    d0955f21bf24       2 days ago         188.3 MB
centos              latest             88f9454e60dd       2 weeks ago        210 MB
training/webapp     latest             31fa814ba25a       9 months ago       278.8 MB
docker@boot2docker:~$
```

On lance l'image.

```
docker@boot2docker:~$ docker run -t -i centos:latest /bin/bash

[root@c5b866897f5a /]#
```

L'invite contient l'**ID**entifiant du nouveau conteneur. On vérifie la version de **CentOS**.

```
[root@c5b866897f5a /]# cat /etc/centos-release

CentOS Linux release 7.0.1406 (Core)
[root@c5b866897f5a /]#
```

## Nouvelle image

On sort de l'image.

```
[root@c5b866897f5a /]# exit
exit
docker@boot2docker:~$
```

La sortie et la fermeture du conteneur se sont bien passées.

```
docker@boot2docker:~$ docker ps -l
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS              NAMES
c5b866897f5a       centos:latest      "/bin/bash"        19 seconds ago     Exited (0) 6
seconds ago        cranky_payne
docker@boot2docker:~$
```

Plus rien ne roule?

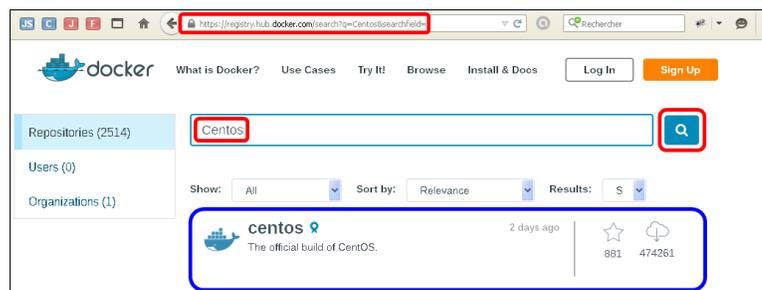
```
docker@boot2docker:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS              NAMES
docker@boot2docker:~$
```

Un nouveau conteneur a été créé suite au lancement de l'image. L'**IDentifiant** du conteneur est bien celui créé après le lancement de l'image.

```
docker@boot2docker:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS              NAMES
c5b866897f5a       centos:latest      "/bin/bash"        About a minute ago Exited
(0) About a minute ago        cranky_payne
d759edea4441       ubuntu:14.04       "/bin/sh -c 'while t 42 hours ago     Exited
(137) 42 hours ago        distracted_wilson
921a97de6575       ubuntu:14.04       "/bin/bash"        43 hours ago     Exited
(0) 43 hours ago        determined_colden
2c9f697c0f2b       ubuntu:14.04       "/bin/echo 'Hello wo 44 hours ago     Exited
(0) 44 hours ago        jolly_jones
docker@boot2docker:~$
```

## 2. Trouver des images

Une des caractéristiques de Docker est que plusieurs personnes ont créé des images pour une variété de raisons et qu'un grand nombre de ces images (*des milliers*) ont été téléversées sur **Docker Hub**. Nous pouvons lancer une recherche parmi ces images sur le site **Docker Hub** à l'aide d'un navigateur Web.



Nous pouvons également rechercher des images depuis la ligne de commande en utilisant **docker search**.

Supposons que notre équipe veut une image avec **Ruby** et **Sinatra** installés et avec laquelle elle pourra faire du développement d'applications Web.

Nous pouvons rechercher une image appropriée en utilisant **docker search** pour trouver toutes les images qui contiennent le terme **sinatra**.

```
docker@boot2docker:~$ docker search sinatra
NAME                                DESCRIPTION          STARS     OFFICIAL   AUTOMATED
training/sinatra                    5
gwjjeff/sinatra                     0           [OK]
dcarley/example-ruby-sinatra        0           [OK]
larmar/sinatra-puppet               0           [OK]
zoomix/sinatra-galleria             0           [OK]
smashwilson/minimal-sinatra         0           [OK]
andyshinn/sinatra-echo              0           [OK]
synctree/sinatra-echo               0           [OK]
...
docker@boot2docker:~$
```

Nous pouvons voir que **search** a trouvé un grand nombre d'images qui utilisent le terme **sinatra**. Il affiche une liste d'images avec descriptions, étoiles (*qui mesurent la popularité des images - si un utilisateur aime une image il peut l'étoiler*) et le statut de leur construction: **Officielle** ou **Automatisée**. Les images Officielles sont construites et entretenues par le projet [Stackbrew](#). Les Automatisées sont de construction automatisée et vous permettent de valider la source et le contenu d'une image.

Nous avons passé en revue les images disponibles et décidé d'utiliser l'image **training/sinatra**.

Jusqu'à présent, nous avons vu deux types de registres d'images. Les images telle que **ubuntu** sont appelées de **base** ou images **root**. Ces images de base sont fournies par **Docker Inc.** et sont construites, validées et prises en charge par Stackbrew. Celles-ci peuvent être identifiées par leur nom d'un seul mot.

Nous avons également vu des images d'utilisateurs telle que l'image **training/sinatra** que nous avons choisie. Une image d'utilisateur appartient à un membre de la communauté Docker et est construite et entretenue par la communauté entière. Vous pouvez facilement identifier une image d'utilisateur car elle est toujours préfixée du nom de celui qui l'a créée. Pour notre image, l'utilisateur/créateur est **training**.

### 3. Téléchargement (*pull*)

Nous avons identifié une image, **training/sinatra** et nous pouvons la télécharger en utilisant **docker pull**.

```
docker@boot2docker:~$ docker pull training/sinatra
Pulling repository training/sinatra
f0f4ab557f95: Download complete
511136ea3c5a: Download complete
3e76c0a80540: Download complete
be88c4c27e80: Download complete
bfab314f3b76: Download complete
e809f156dc98: Download complete
ce80548340bb: Download complete
79e6bf39f993: Download complete
Status: Downloaded newer image for training/sinatra:latest
docker@boot2docker:~$
```

## Nouvelle image

On vérifie que l'image a bien été téléchargée.

```
docker@boot2docker:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
ubuntu	14.04	d0955f21bf24	2 days ago	188.3 MB
ubuntu	14.04.2	d0955f21bf24	2 days ago	188.3 MB
ubuntu	latest	d0955f21bf24	2 days ago	188.3 MB
ubuntu	trusty	d0955f21bf24	2 days ago	188.3 MB
ubuntu	trusty-20150320	d0955f21bf24	2 days ago	188.3 MB
centos	latest	88f9454e60dd	2 weeks ago	210 MB
<b>training/sinatra</b>	<b>latest</b>	<b>f0f4ab557f95</b>	<b>9 months ago</b>	<b>447 MB</b>
training/webapp	latest	31fa814ba25a	9 months ago	278.8 MB

```
docker@boot2docker:~$
```

Tous les membres de l'équipe peut maintenant utiliser cette image en lançant leur propre conteneur.

```
docker@boot2docker:~$ docker run -t -i training/sinatra:latest /bin/bash
```

```
root@222c580654b1:/#
```

À l'intérieur du conteneur, on liste les répertoires de "/".

```
root@222c580654b1:/# ls /
```

```
bin  dev  home  lib64  mnt  proc  run  srv  tmp  var
boot  etc  lib  media  opt  root  sbin  sys  usr
```

```
root@222c580654b1:/#
```

On affiche la version du système.

```
root@222c580654b1:/# cat /etc/os-release
```

```
NAME="Ubuntu"
VERSION="14.04, Trusty Tahr"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 14.04 LTS"
VERSION_ID="14.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
root@222c580654b1:/#
```

On sort de l'image.

```
root@222c580654b1:/# exit
```

```
exit
```

```
docker@boot2docker:~$
```

On affiche les conteneurs pour en connaître le nouveau.

```
docker@boot2docker:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
<b>222c580654b1</b>	<b>training/sinatra:latest</b>	<b>"/bin/bash"</b>	<b>About a minute ago</b>
<b>Exited (0) 12 seconds ago</b>		<b>sharp_goldstine</b>	
81fa24f6dba2	centos:latest	"/bin/bash"	2 hours ago
Exited (0) 2 hours ago		grave_pare	
...			

```
docker@boot2docker:~$
```

## XIV- Création d'une image

### 1. Introduction

L'équipe a trouvé **training/sinatra** très utile mais ce n'est pas tout à fait ce dont ils ont besoin et nous devons faire quelques changements à l'image. Il existe deux façons de procéder: mettre à jour l'image ou en créer une toute nouvelle.

- 1) Nous pouvons mettre à jour un conteneur créé à partir d'une image puis exécuter un **commit** pour créer une nouvelle image.
- 2) Nous pouvons utiliser un fichier **Dockerfile** afin de spécifier les instructions pour la création d'une toute nouvelle image.

### 2. Mise à jour et commit d'une image

Pour mettre à jour une image, il nous faut d'abord créer un conteneur à partir de celle-ci.

```
docker@boot2docker:~$ sudo docker run -t -i training/sinatra:latest /bin/bash
root@a180bba9ce55:/#
```



Prenez note de l'**IDentifiant** du conteneur qui a été créé, **a180bba9ce55**, nous en aurons besoin dans un moment.

À l'intérieur de notre conteneur, ajoutons la gemme<sup>1</sup> **json**.

```
root@a180bba9ce55:/# gem install json

Fetching: json-1.8.2.gem (100%)
Building native extensions. This could take a while...
Successfully installed json-1.8.2
1 gem installed
Installing ri documentation for json-1.8.2...
Installing RDoc documentation for json-1.8.2...
root@a180bba9ce55:/#
```

Une fois l'installation terminée nous quittons notre conteneur en utilisant **exit**.

```
root@a180bba9ce55:/# exit
exit
docker@boot2docker:~$
```

---

1) **Référence:** <http://fr.wikipedia.org/>. Le **GEM** (*Graphical Environment Manager*) est un environnement de bureau créé par Digital Research, l'inventeur du système d'exploitation CP/M (ancêtre de MS-DOS). Le logo du GEM est une gemme (*pietre précieuse*) en référence au mot anglais.

## Création d'une image

Maintenant nous avons un conteneur avec les changements que nous y avons apportés. Nous pouvons exécuter un **commit** de ce conteneur pour en faire une image.



```
docker@boot2docker:~$ docker commit -m "Ajout de la gemme json" \
-a "Michel-André" \
  a180bba9ce55 \
  michelandre/sinatra:v2
99dddcf64cfb91e194cc4f6e961a033dd88f5c78d0147a1bd8bc4bfcc5f23767
docker@boot2docker:~$
```

Nous avons utilisé la commande **docker commit**. Nous avons spécifié deux arguments: **-m** et **-a**. L'argument **-m** nous permet de spécifier un **m**essage de validation, un peu comme vous le feriez avec un **commit** sur un système de contrôle de version. L'argument **-a** nous permet de spécifier un **a**uteur pour notre mise à jour.

Nous avons également précisé le conteneur **a180bba9ce55** (*c'est l'IDentifiant que nous avons noté plus tôt*) à partir duquel nous voulons créer cette nouvelle image et enfin nous avons spécifié une cible **michelandre/sinatra:v2** pour l'image.

Décomposons cette cible. Elle se compose d'un nouvel utilisateur, notre-usager, pour lequel nous créons cette image. Nous avons également précisé le nom de l'image; nous avons conservé le nom **sinatra** de l'image originale. Enfin nous avons spécifié l'étiquette **v2** pour la nouvelle image.

Nous pouvons regarder notre nouvelle image **michelandre/sinatra:v2** à l'aide de **docker images**.

```
docker@boot2docker:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL
SIZE
michelandre/sinatra v2                 99dddcf64cfb      15 seconds ago    452 MB
ubuntu              14.04             d0955f21bf24      2 days ago        188.3 MB
ubuntu              14.04.2           d0955f21bf24      2 days ago        188.3 MB
ubuntu              latest            d0955f21bf24      2 days ago        188.3 MB
ubuntu              trusty            d0955f21bf24      2 days ago        188.3 MB
ubuntu              trusty-20150320  d0955f21bf24      2 days ago        188.3 MB
centos              latest            88f9454e60dd      2 weeks ago       210 MB
training/sinatra    latest            f0f4ab557f95      9 months ago      447 MB
training/webapp     latest            31fa814ba25a      9 months ago      278.8 MB
docker@boot2docker:~$
```

On lance notre nouvelle image et ainsi créer un nouveau conteneur.

```
docker@boot2docker:~$ docker run -i -t michelandre/sinatra:v2 /bin/bash
root@8be113a1a38b:/#
```

Nous voyons que l'IDentifiant du nouveau conteneur est **8be113a1a38b**.

On sort de l'image.

```
root@8be113a1a38b:/# exit
exit
docker@boot2docker:~$
```

Le conteneur a été créé.

```
docker@boot2docker:~$ docker ps -a
```

CONTAINER ID	IMAGE	PORTS	COMMAND	CREATED
8be113a1a38b	micelandre/sinatra:v2		"/bin/bash"	30 seconds ago
Exited (0) 13 seconds ago			modest_tesla	
ce35beeff7f9	training/sinatra:latest		"/bin/bash"	About an hour ago
Exited (0) 55 minutes ago			loving_fermi	
0fe8dca2ca43	training/sinatra:latest		"/bin/bash"	About an hour ago
Exited (0) About an hour ago			berserk_jang	
81fa24f6dba2	centos:latest		"/bin/bash"	4 hours ago
Exited (0) 4 hours ago			grave_pare	
d759edea4441	ubuntu:14.04		"/bin/sh -c 'while t	46 hours ago
Exited (137) 46 hours ago			distracted_wilson	
921a97de6575	ubuntu:14.04		"/bin/bash"	47 hours ago
Exited (0) 47 hours ago			determined_colden	
2c9f697c0f2b	ubuntu:14.04		"/bin/echo 'Hello wo	2 days ago
Exited (0) 2 days ago			jolly_jones	

```
docker@boot2docker:~$
```

# XV- Dockerfile

## 1. Introduction

Utiliser **docker commit** est un moyen simple d'étendre une image mais cette commande est un peu inconmode et il n'est pas facile de partager un processus de développement d'images entre tous les membres d'une même équipe. En lieu et place, nous pouvons utiliser une nouvelle commande, **docker build** pour construire de nouvelles images à partir de zéro.

Pour ce faire, nous créons un fichier **Dockerfile** qui contient un ensemble d'instructions qui indique à Docker comment construire notre nouvelle image.

## 2. Répertoire et fichier Dockerfile

Les meilleures pratiques: [https://docs.docker.com/articles/dockerfile\\_best-practices/](https://docs.docker.com/articles/dockerfile_best-practices/).

Référence: <https://docs.docker.com/reference/builder/>.

Docker peut construire automatiquement des images en lisant les instructions d'un fichier **Dockerfile**. Un fichier Dockerfile est un document texte qui contient toutes les commandes que vous auriez normalement exécuter manuellement afin de construire une image Docker. En appelant **docker build** depuis votre terminal, vous pouvez signifier à Docker qu'il construise votre image étape par étape en exécutant des instructions successivement.

### 2.1. Création du répertoire

Chaque fichier Dockerfile a son propre répertoire.

On change d'utilisateur pour devenir **root**.

```
docker@boot2docker:~$ sudo -s
root@boot2docker:/home/docker#
```

On crée le fichier pour la construction des images.

```
root@boot2docker:/home/docker# mkdir /construction
root@boot2docker:/home/docker#
```

On se rend dans le répertoire de création d'images.

```
root@boot2docker:/home/docker# cd /construction
root@boot2docker:/construction#
```

```
root@boot2docker:/construction# pwd
/construction
root@boot2docker:/construction#
```

## 2.2. Fichier Dockerfile

Chaque instruction crée une nouvelle couche de l'image. Examinons un exemple simple pour construire notre propre image **sinatra** pour notre équipe de développement.

```
# Ceci est un commentaire
#
FROM ubuntu:14.04
MAINTAINER Le grand général Toto <general-toto@micronator.org>
RUN apt-get update && apt-get install -y ruby ruby-dev
RUN gem install sinatra
```

Examinons ce que fait notre fichier **Dockerfile**. Chaque instruction est en majuscule et placée au début d'une ligne et elle est suivie d'une déclaration.

```
INSTRUCTION déclaration
```



On utilise un dièse "#" pour indiquer un commentaire.

- La première instruction **FROM** indique à Docker quelle est la source de notre image. Dans le cas présent, nous nous basons sur une image **Ubuntu 14.04**.
- L'instruction **MAINTAINER** précise le responsable de la maintenance de l'image.



Une instruction **RUN** exécute une commande à l'intérieur de l'image; par exemple l'installation d'un paquetage

- Nous avons spécifié deux instructions **RUN**. Avec la première, nous effectuons une mise à jour du système puis nous installons **Ruby** et **RubyGems**. Avec la deuxième **RUN**, nous installons la gemme **Sinatra**.
- L'argument **-y** spécifie à **apt-get install** de ne pas demander de confirmation à l'utilisateur pour l'installation des paquetage.



Il existe [beaucoup plus d'instructions](#) pour un fichier **Dockerfile**.

### 2.2.1. Création du fichier Dockerfile

Nous créons le fichier.



```
cat > Dockerfile << FIN
# Ceci est un commentaire
#
FROM ubuntu:14.04
MAINTAINER Le grand général Toto <general-toto@micronator.org>
RUN apt-get update && apt-get install -y ruby ruby-dev
RUN gem install sinatra
FIN
```

On vérifie la création du fichier **Dockerfile**.

```
root@boot2docker:/construction# ls -als
total 4
 0 drwxr-xr-x  2 root   root           60 Mar 23 01:46 .
 0 drwxr-xr-x 17 root   root          400 Mar 23 00:42 ..
 4 -rw-r--r--  1 root   root          190 Mar 23 01:46 Dockerfile
root@boot2docker:/construction#
```

On vérifie le contenu du fichier.

```
root@boot2docker:/construction# cat Dockerfile
# Ceci est un commentaire
#
FROM ubuntu:14.04
MAINTAINER Le grand général Toto <general-toto@micronator.org>
RUN apt-get update && apt-get install -y ruby ruby-dev
RUN gem install sinatra
root@boot2docker:/construction#
```

Fichier d'aide de la commande docker build.

```
root@boot2docker:/construction# docker build -h
Usage: docker build [OPTIONS] PATH | URL | -
Build a new image from the source code at PATH
-f, --file=""          Name of the Dockerfile (Default is 'Dockerfile' at context root)
--force-rm=false      Always remove intermediate containers, even after unsuccessful builds
--help=false         Print usage
--no-cache=false     Do not use cache when building the image
--pull=false         Always attempt to pull a newer version of the image
-q, --quiet=false    Suppress the verbose output generated by the containers
--rm=true           Remove intermediate containers after a successful build
-t, --tag=""        Repository name (and optionally a tag) to be applied to the resulting
                    image in case of success
root@boot2docker:/construction#
```

```
docker build -t general-toto/sinatra:v1 .
| | | | |
| | | | | chemin du fichier Dockerfile (répertoire de
| | | | | construction i.e. le répertoire courant)
| | | | | l'étiquette (ici, nous spécifions la version de
| | | | | l'image
| | | | | le nom de l'image
| | | | | le nom de l'usager
option -t indique qu'on spécifie le nom du registre et l'étiquette
```

Nous allons spécifier notre fichier **Dockerfile** à la commande **docker build** afin qu'elle construise notre nouvelle image.

```
root@boot2docker:/construction# docker build -t general-toto/sinatra:v1 .
Sending build context to Docker daemon 2.048 kB
Sending build context to Docker daemon
Step 0 : FROM ubuntu:14.04
---> d0955f21bf24
Step 1 : MAINTAINER Le grand général Toto <general-toto@micronator.org>
---> Running in 34b88e466a8e
---> 137e1ceecf83
Removing intermediate container 34b88e466a8e
Step 2 : RUN apt-get update && apt-get install -y ruby ruby-dev
---> Running in f01a1439dcfe
Ign http://archive.ubuntu.com trusty InRelease
Ign http://archive.ubuntu.com trusty-updates InRelease
Ign http://archive.ubuntu.com trusty-security InRelease
Hit http://archive.ubuntu.com trusty Release.gpg
Get:1 http://archive.ubuntu.com trusty-updates Release.gpg [933 B]
Get:2 http://archive.ubuntu.com trusty-security Release.gpg [933 B]
Hit http://archive.ubuntu.com trusty Release
Get:3 http://archive.ubuntu.com trusty-updates Release [62.0 kB]
```

```
Get:4 http://archive.ubuntu.com trusty-security Release [62.0 kB]
Get:5 http://archive.ubuntu.com trusty/main Sources [1335 kB]
Get:6 http://archive.ubuntu.com trusty/restricted Sources [5335 B]
Get:7 http://archive.ubuntu.com trusty/universe Sources [7926 kB]
Get:8 http://archive.ubuntu.com trusty/main amd64 Packages [1743 kB]
Get:9 http://archive.ubuntu.com trusty/restricted amd64 Packages [16.0 kB]
Get:10 http://archive.ubuntu.com trusty/universe amd64 Packages [7589 kB]
Get:11 http://archive.ubuntu.com trusty-updates/main Sources [235 kB]
Get:12 http://archive.ubuntu.com trusty-updates/restricted Sources [2310 B]
Get:13 http://archive.ubuntu.com trusty-updates/universe Sources [133 kB]
Get:14 http://archive.ubuntu.com trusty-updates/main amd64 Packages [576 kB]
Get:15 http://archive.ubuntu.com trusty-updates/restricted amd64 Packages [15.1 kB]
Get:16 http://archive.ubuntu.com trusty-updates/universe amd64 Packages [338 kB]
Get:17 http://archive.ubuntu.com trusty-security/main Sources [89.6 kB]
Get:18 http://archive.ubuntu.com trusty-security/restricted Sources [1874 B]
Get:19 http://archive.ubuntu.com trusty-security/universe Sources [19.6 kB]
Get:20 http://archive.ubuntu.com trusty-security/main amd64 Packages [280 kB]
Get:21 http://archive.ubuntu.com trusty-security/restricted amd64 Packages [14.8 kB]
Get:22 http://archive.ubuntu.com trusty-security/universe amd64 Packages [114 kB]
Fetched 20.6 MB in 38s (529 kB/s)
Reading package lists...
Reading package lists...
Building dependency tree...
Reading state information...
The following extra packages will be installed:
  binutils ca-certificates cpp cpp-4.8 gcc gcc-4.8 libasan0 libatomic1
  libc-dev-bin libc6-dev libcloog-isl4 libgcc-4.8-dev libgmp10 libgomp1
  libisl10 libitm1 libmpc3 libmpfr4 libquadmath0 libruby1.9.1 libtsan0
  libyaml-0-2 linux-libc-dev manpages manpages-dev openssl ruby1.9.1
  ruby1.9.1-dev
Suggested packages:
  binutils-doc cpp-doc gcc-4.8-locales gcc-multilib make autoconf automake1.9
  libtool flex bison gdb gcc-doc gcc-4.8-multilib gcc-4.8-doc libgcc1-dbg
  libgomp1-dbg libitm1-dbg libatomic1-dbg libasan0-dbg libtsan0-dbg
  libbacktrace1-dbg libquadmath0-dbg binutils-gold glibc-doc man-browser ri
  ruby1.9.1-examples ril.9.1 graphviz ruby-switch
The following NEW packages will be installed:
  binutils ca-certificates cpp cpp-4.8 gcc gcc-4.8 libasan0 libatomic1
  libc-dev-bin libc6-dev libcloog-isl4 libgcc-4.8-dev libgmp10 libgomp1
  libisl10 libitm1 libmpc3 libmpfr4 libquadmath0 libruby1.9.1 libtsan0
  libyaml-0-2 linux-libc-dev manpages manpages-dev openssl ruby ruby-dev
  ruby1.9.1 ruby1.9.1-dev
0 upgraded, 30 newly installed, 0 to remove and 1 not upgraded.
Need to get 24.0 MB of archives.
After this operation, 89.9 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu/ trusty/main libasan0 amd64 4.8.2-19ubuntu1 [63.0 kB]
Get:2 http://archive.ubuntu.com/ubuntu/ trusty/main libatomic1 amd64 4.8.2-19ubuntu1 [8626
B]
Get:3 http://archive.ubuntu.com/ubuntu/ trusty/main libgmp10 amd64 2:5.1.3+dfsg-1ubuntu1
[218 kB]
Get:4 http://archive.ubuntu.com/ubuntu/ trusty/main libisl10 amd64 0.12.2-1 [419 kB]
Get:5 http://archive.ubuntu.com/ubuntu/ trusty/main libcloog-isl4 amd64 0.18.2-1 [57.5 kB]
Get:6 http://archive.ubuntu.com/ubuntu/ trusty/main libgomp1 amd64 4.8.2-19ubuntu1 [23.2 kB]
Get:7 http://archive.ubuntu.com/ubuntu/ trusty/main libitm1 amd64 4.8.2-19ubuntu1 [28.5 kB]
Get:8 http://archive.ubuntu.com/ubuntu/ trusty/main libmpfr4 amd64 3.1.2-1 [203 kB]
Get:9 http://archive.ubuntu.com/ubuntu/ trusty/main libquadmath0 amd64 4.8.2-19ubuntu1 [126
kB]
Get:10 http://archive.ubuntu.com/ubuntu/ trusty/main libtsan0 amd64 4.8.2-19ubuntu1 [94.7
kB]
Get:11 http://archive.ubuntu.com/ubuntu/ trusty-updates/main libyaml-0-2 amd64 0.1.4-
3ubuntu3.1 [48.1 kB]
Get:12 http://archive.ubuntu.com/ubuntu/ trusty/main libmpc3 amd64 1.0.1-1ubuntu1 [38.4 kB]
Get:13 http://archive.ubuntu.com/ubuntu/ trusty-updates/main openssl amd64 1.0.1f-
1ubuntu2.11 [488 kB]
Get:14 http://archive.ubuntu.com/ubuntu/ trusty-updates/main ca-certificates all
20141019ubuntu0.14.04.1 [189 kB]
Get:15 http://archive.ubuntu.com/ubuntu/ trusty/main manpages all 3.54-1ubuntu1 [627 kB]
Get:16 http://archive.ubuntu.com/ubuntu/ trusty-updates/main binutils amd64 2.24-5ubuntu3.1
```

## Dockerfile

```
[2076 kB]
Get:17 http://archive.ubuntu.com/ubuntu/ trusty/main cpp-4.8 amd64 4.8.2-19ubuntu1 [4439 kB]
Get:18 http://archive.ubuntu.com/ubuntu/ trusty/main cpp amd64 4:4.8.2-1ubuntu6 [27.5 kB]
Get:19 http://archive.ubuntu.com/ubuntu/ trusty/main libgcc-4.8-dev amd64 4.8.2-19ubuntu1
[1688 kB]
Get:20 http://archive.ubuntu.com/ubuntu/ trusty/main gcc-4.8 amd64 4.8.2-19ubuntu1 [5012 kB]
Get:21 http://archive.ubuntu.com/ubuntu/ trusty/main gcc amd64 4:4.8.2-1ubuntu6 [5098 B]
Get:22 http://archive.ubuntu.com/ubuntu/ trusty-updates/main libc-dev-bin amd64 2.19-
0ubuntu6.6 [68.9 kB]
Get:23 http://archive.ubuntu.com/ubuntu/ trusty-updates/main linux-libc-dev amd64 3.13.0-
46.79 [779 kB]
Get:24 http://archive.ubuntu.com/ubuntu/ trusty-updates/main libc6-dev amd64 2.19-0ubuntu6.6
[1910 kB]
Get:25 http://archive.ubuntu.com/ubuntu/ trusty/main ruby all 1:1.9.3.4 [5334 B]
Get:26 http://archive.ubuntu.com/ubuntu/ trusty-updates/main ruby1.9.1 amd64 1.9.3.484-
2ubuntu1.2 [35.6 kB]
Get:27 http://archive.ubuntu.com/ubuntu/ trusty-updates/main libruby1.9.1 amd64 1.9.3.484-
2ubuntu1.2 [2645 kB]
Get:28 http://archive.ubuntu.com/ubuntu/ trusty/main manpages-dev all 3.54-1ubuntu1 [1820
kB]
Get:29 http://archive.ubuntu.com/ubuntu/ trusty-updates/main ruby1.9.1-dev amd64 1.9.3.484-
2ubuntu1.2 [871 kB]
Get:30 http://archive.ubuntu.com/ubuntu/ trusty/main ruby-dev all 1:1.9.3.4 [4660 B]
debconf: unable to initialize frontend: Dialog
debconf: (TERM is not set, so the dialog frontend is not usable.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Fetched 24.0 MB in 41s (575 kB/s)
Selecting previously unselected package libasan0:amd64.
(Reading database ... 11527 files and directories currently installed.)
Preparing to unpack .../libasan0_4.8.2-19ubuntu1_amd64.deb ...
Unpacking libasan0:amd64 (4.8.2-19ubuntu1) ...
Selecting previously unselected package libatomic1:amd64.
Preparing to unpack .../libatomic1_4.8.2-19ubuntu1_amd64.deb ...
Unpacking libatomic1:amd64 (4.8.2-19ubuntu1) ...
Selecting previously unselected package libgmp10:amd64.
Preparing to unpack .../libgmp10_2%3a5.1.3+dfsg-1ubuntu1_amd64.deb ...
Unpacking libgmp10:amd64 (2:5.1.3+dfsg-1ubuntu1) ...
Selecting previously unselected package libisl10:amd64.
Preparing to unpack .../libisl10_0.12.2-1_amd64.deb ...
Unpacking libisl10:amd64 (0.12.2-1) ...
Selecting previously unselected package libcloog-isl4:amd64.
Preparing to unpack .../libcloog-isl4_0.18.2-1_amd64.deb ...
Unpacking libcloog-isl4:amd64 (0.18.2-1) ...
Selecting previously unselected package libgomp1:amd64.
Preparing to unpack .../libgomp1_4.8.2-19ubuntu1_amd64.deb ...
Unpacking libgomp1:amd64 (4.8.2-19ubuntu1) ...
Selecting previously unselected package libitm1:amd64.
Preparing to unpack .../libitm1_4.8.2-19ubuntu1_amd64.deb ...
Unpacking libitm1:amd64 (4.8.2-19ubuntu1) ...
Selecting previously unselected package libmpfr4:amd64.
Preparing to unpack .../libmpfr4_3.1.2-1_amd64.deb ...
Unpacking libmpfr4:amd64 (3.1.2-1) ...
Selecting previously unselected package libquadmath0:amd64.
Preparing to unpack .../libquadmath0_4.8.2-19ubuntu1_amd64.deb ...
Unpacking libquadmath0:amd64 (4.8.2-19ubuntu1) ...
Selecting previously unselected package libtsan0:amd64.
Preparing to unpack .../libtsan0_4.8.2-19ubuntu1_amd64.deb ...
Unpacking libtsan0:amd64 (4.8.2-19ubuntu1) ...
Selecting previously unselected package libyaml-0-2:amd64.
Preparing to unpack .../libyaml-0-2_0.1.4-3ubuntu3.1_amd64.deb ...
Unpacking libyaml-0-2:amd64 (0.1.4-3ubuntu3.1) ...
Selecting previously unselected package libmpc3:amd64.
Preparing to unpack .../libmpc3_1.0.1-1ubuntu1_amd64.deb ...
Unpacking libmpc3:amd64 (1.0.1-1ubuntu1) ...
```

## Tutoriel - boot2docker

```
Selecting previously unselected package openssl.
Preparing to unpack ../openssl_1.0.1f-1ubuntu2.11_amd64.deb ...
Unpacking openssl (1.0.1f-1ubuntu2.11) ...
Selecting previously unselected package ca-certificates.
Preparing to unpack ../ca-certificates_20141019ubuntu0.14.04.1_all.deb ...
Unpacking ca-certificates (20141019ubuntu0.14.04.1) ...
Selecting previously unselected package manpages.
Preparing to unpack ../manpages_3.54-1ubuntu1_all.deb ...
Unpacking manpages (3.54-1ubuntu1) ...
Selecting previously unselected package binutils.
Preparing to unpack ../binutils_2.24-5ubuntu3.1_amd64.deb ...
Unpacking binutils (2.24-5ubuntu3.1) ...
Selecting previously unselected package cpp-4.8.
Preparing to unpack ../cpp-4.8_4.8.2-19ubuntu1_amd64.deb ...
Unpacking cpp-4.8 (4.8.2-19ubuntu1) ...
Selecting previously unselected package cpp.
Preparing to unpack ../cpp_4%3a4.8.2-1ubuntu6_amd64.deb ...
Unpacking cpp (4:4.8.2-1ubuntu6) ...
Selecting previously unselected package libgcc-4.8-dev:amd64.
Preparing to unpack ../libgcc-4.8-dev_4.8.2-19ubuntu1_amd64.deb ...
Unpacking libgcc-4.8-dev:amd64 (4.8.2-19ubuntu1) ...
Selecting previously unselected package gcc-4.8.
Preparing to unpack ../gcc-4.8_4.8.2-19ubuntu1_amd64.deb ...
Unpacking gcc-4.8 (4.8.2-19ubuntu1) ...
Selecting previously unselected package gcc.
Preparing to unpack ../gcc_4%3a4.8.2-1ubuntu6_amd64.deb ...
Unpacking gcc (4:4.8.2-1ubuntu6) ...
Selecting previously unselected package libc-dev-bin.
Preparing to unpack ../libc-dev-bin_2.19-0ubuntu6.6_amd64.deb ...
Unpacking libc-dev-bin (2.19-0ubuntu6.6) ...
Selecting previously unselected package linux-libc-dev:amd64.
Preparing to unpack ../linux-libc-dev_3.13.0-46.79_amd64.deb ...
Unpacking linux-libc-dev:amd64 (3.13.0-46.79) ...
Selecting previously unselected package libc6-dev:amd64.
Preparing to unpack ../libc6-dev_2.19-0ubuntu6.6_amd64.deb ...
Unpacking libc6-dev:amd64 (2.19-0ubuntu6.6) ...
Selecting previously unselected package ruby.
Preparing to unpack ../ruby_1%3a1.9.3.4_all.deb ...
Unpacking ruby (1:1.9.3.4) ...
Selecting previously unselected package ruby1.9.1.
Preparing to unpack ../ruby1.9.1_1.9.3.484-2ubuntu1.2_amd64.deb ...
Unpacking ruby1.9.1 (1.9.3.484-2ubuntu1.2) ...
Selecting previously unselected package libruby1.9.1.
Preparing to unpack ../libruby1.9.1_1.9.3.484-2ubuntu1.2_amd64.deb ...
Unpacking libruby1.9.1 (1.9.3.484-2ubuntu1.2) ...
Selecting previously unselected package manpages-dev.
Preparing to unpack ../manpages-dev_3.54-1ubuntu1_all.deb ...
Unpacking manpages-dev (3.54-1ubuntu1) ...
Selecting previously unselected package ruby1.9.1-dev.
Preparing to unpack ../ruby1.9.1-dev_1.9.3.484-2ubuntu1.2_amd64.deb ...
Unpacking ruby1.9.1-dev (1.9.3.484-2ubuntu1.2) ...
Selecting previously unselected package ruby-dev.
Preparing to unpack ../ruby-dev_1%3a1.9.3.4_all.deb ...
Unpacking ruby-dev (1:1.9.3.4) ...
Setting up libasan0:amd64 (4.8.2-19ubuntu1) ...
Setting up libatomic1:amd64 (4.8.2-19ubuntu1) ...
Setting up libgmp10:amd64 (2:5.1.3+dfsg-1ubuntu1) ...
Setting up libisl10:amd64 (0.12.2-1) ...
Setting up libcloog-isl4:amd64 (0.18.2-1) ...
Setting up libgomp1:amd64 (4.8.2-19ubuntu1) ...
Setting up libitm1:amd64 (4.8.2-19ubuntu1) ...
Setting up libmpfr4:amd64 (3.1.2-1) ...
Setting up libquadmath0:amd64 (4.8.2-19ubuntu1) ...
Setting up libtsan0:amd64 (4.8.2-19ubuntu1) ...
Setting up libyaml-0-2:amd64 (0.1.4-3ubuntu3.1) ...
Setting up libmpc3:amd64 (1.0.1-1ubuntu1) ...
Setting up openssl (1.0.1f-1ubuntu2.11) ...
Setting up ca-certificates (20141019ubuntu0.14.04.1) ...
```

## Dockerfile

```
debconf: unable to initialize frontend: Dialog
debconf: (TERM is not set, so the dialog frontend is not usable.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
Setting up manpages (3.54-1ubuntu1) ...
Setting up binutils (2.24-5ubuntu3.1) ...
Setting up cpp-4.8 (4.8.2-19ubuntu1) ...
Setting up cpp (4:4.8.2-1ubuntu6) ...
Setting up libgcc-4.8-dev:amd64 (4.8.2-19ubuntu1) ...
Setting up gcc-4.8 (4.8.2-19ubuntu1) ...
Setting up gcc (4:4.8.2-1ubuntu6) ...
Setting up libc-dev-bin (2.19-0ubuntu6.6) ...
Setting up linux-libc-dev:amd64 (3.13.0-46.79) ...
Setting up libc6-dev:amd64 (2.19-0ubuntu6.6) ...
Setting up manpages-dev (3.54-1ubuntu1) ...
Setting up libruby1.9.1 (1.9.3.484-2ubuntu1.2) ...
Setting up ruby1.9.1-dev (1.9.3.484-2ubuntu1.2) ...
Setting up ruby-dev (1:1.9.3.4) ...
Setting up ruby (1:1.9.3.4) ...
Setting up ruby1.9.1 (1.9.3.484-2ubuntu1.2) ...
Processing triggers for libc-bin (2.19-0ubuntu6.6) ...
Processing triggers for ca-certificates (20141019ubuntu0.14.04.1) ...
Updating certificates in /etc/ssl/certs... 173 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d....done.
--> 39c09d6d4f9e
Removing intermediate container f01a1439dcfe
Step 3 : RUN gem install sinatra
--> Running in 491c56304bd3
unable to convert "\xc3" to UTF-8 in conversion from ASCII-8BIT to UTF-8 to US-ASCII for
README.rdoc, skipping
unable to convert "\xc3" to UTF-8 in conversion from ASCII-8BIT to UTF-8 to US-ASCII for
README.rdoc, skipping
Successfully installed rack-1.6.0
Successfully installed tilt-1.4.1
Successfully installed rack-protection-1.5.3
Successfully installed sinatra-1.4.5
4 gems installed
Installing ri documentation for rack-1.6.0...
Installing ri documentation for tilt-1.4.1...
Installing ri documentation for rack-protection-1.5.3...
Installing ri documentation for sinatra-1.4.5...
Installing RDoc documentation for rack-1.6.0...
Installing RDoc documentation for tilt-1.4.1...
Installing RDoc documentation for rack-protection-1.5.3...
Installing RDoc documentation for sinatra-1.4.5...
--> 47c01091dc50
Removing intermediate container 491c56304bd3
Successfully built 47c01091dc50
root@boot2docker:/construction#
```

La nouvelle image **47c01091dc50** a été créée avec succès.

```
root@boot2docker:/construction# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
<b>general-toto/sinatra</b>	<b>v1</b>	<b>47c01091dc50</b>	4 minutes ago	317.2 MB
michelandre/sinatra	v2	99dddcf64cfb	11 minutes ago	452 MB
ubuntu	14.04	d0955f21bf24	2 days ago	188.3 MB
ubuntu	14.04.2	d0955f21bf24	2 days ago	188.3 MB
ubuntu	latest	d0955f21bf24	2 days ago	188.3 MB
...				

```
root@boot2docker:/construction#
```

Aucun nouveau conteneur n'a été créé.

```
root@boot2docker:/construction# docker ps -a

CONTAINER ID        IMAGE                      COMMAND                  CREATED
STATUS             PORTS                    NAMES
a180bba9ce55      training/sinatra:latest  "/bin/bash"            20 minutes ago
Exited (0) 19 minutes ago                    cocky_sammet
...
root@boot2docker:/construction#
```

Nous avons vu le processus de construction au travail. La première étape de Docker a été de télécharger le contexte de construction, essentiellement le contenu du répertoire dans lequel vous construisez l'image. Ce téléchargement est exécuté parce que le daemon Docker fait la construction actuelle de l'image et il a besoin du contexte local pour la faire.

Nous avons vu, étape par étape, l'exécution de chaque instruction de Dockerfile. Chaque étape a créé un nouveau conteneur, exécuté l'instruction à l'intérieur de ce conteneur puis lancé un **commit** de ce changement - tout comme le flux de travail de **docker commit** que nous avons vu précédemment. Lorsque toutes les instructions ont été exécutées, nous nous sommes retrouvés avec l'image **47c01091dc50** (aussi nommée **general-toto/sinatra:v1**). Tous les conteneurs intermédiaires ont été supprimés.



Une image ne peut avoir plus de 127 couches. Cette limitation est définie globalement pour encourager l'optimisation de la taille totale des images.

Nous pouvons maintenant créer un conteneur de notre nouvelle image.

```
root@boot2docker:/construction# docker run -i -t general-toto/sinatra:v1 /bin/bash
root@3b4170aead35:/#
```

On sort du conteneur

```
root@3b4170aead35:/# exit
exit
root@boot2docker:/construction#
```

Un conteneur a été créé avec notre nouvelle image.

```
root@boot2docker:/construction# docker ps -a

CONTAINER ID        IMAGE                      COMMAND                  CREATED
STATUS             PORTS                    NAMES
3b4170aead35      general-toto/sinatra:v1  "/bin/bash"            21 seconds ago
Exited (0) 5 seconds ago                    condensing_poitras
a180bba9ce55      training/sinatra:latest  "/bin/bash"            20 minutes ago
Exited (0) 19 minutes ago                    cocky_sammet
...
root@boot2docker:/construction#
```

Ceci n'est qu'une brève introduction à la création d'images. Nous n'avons vu que quelques instructions, il en existe de nombreuses autres. Nous allons en voir quelques unes dans les prochains chapitres.

Vous pouvez consulter la [documentation de Dockerfile](#) pour une description détaillée et des exemples de chaque instruction.

Pour vous aider à écrire un fichier Dockerfile clair, lisible et d'une maintenance aisée, consultez le [guide des meilleures pratiques](#).



Pour en apprendre un peu plus, complétez ce [tutoriel Dockerfile](#).

### 3. Étiquette d'une image

Vous pouvez ajouter une étiquette à une image existante en utilisant **docker tag**.

Aide sur l'usage de **docker tag**.

```
root@boot2docker:/construction# docker tag -h

Usage: docker tag [OPTIONS] IMAGE[:TAG] [REGISTRYHOST/] [USERNAME/]NAME[:TAG]

Tag an image into a repository

  -f, --force=false      Force
  --help=false           Print usage
root@boot2docker:/construction#
```

Voyons les images actuelles du grand général Toto.

```
root@boot2docker:/construction# docker images | grep general

general-toto/sinatra v1          47c01091dc50      13 minutes ago    317.2 MB
root@boot2docker:/construction#
```

Ajoutons une nouvelle **étiquette** à notre image **general-toto/sinatra:v1**.

```
root@boot2docker:/construction# docker tag 47c01091dc50 general-toto/sinatra:devel
root@boot2docker:/construction#
```



La commande **docker tag** utilise: l'IDentifiant de l'image i.e. **47c01091dc50**, le nom de l'utilisateur/registre et la **nouvelle étiquette**.

On vérifie notre nouvelle étiquette.

```
root@boot2docker:/construction# docker images

REPOSITORY          TAG          IMAGE ID          CREATED          VIRTUAL
SIZE
general-toto/sinatra devel        47c01091dc50     15 minutes ago  317.2 MB
general-toto/sinatra v1          47c01091dc50     115 minutes ago  317.2 MB
...
root@boot2docker:/construction#
```

### 4. Téléversement d'une image

Après avoir construit/créé une nouvelle image, vous pouvez la téléverser sur **Docker Hub** avec **docker push**. Ceci vous permet de partager une image avec d'autres internautes, que ce soit publiquement ou téléversée dans un registre privé.

Pour ce téléversement on crée une nouvelle image.

```
root@boot2docker:/construction# docker build -t michelandre/documentation .

Sending build context to Docker daemon 2.048 kB
Sending build context to Docker daemon
Step 0 : FROM ubuntu:14.04
---> d0955f21bf24
Step 1 : MAINTAINER Le grand général Toto <general-toto@micronator.org>
---> Using cache
---> 1b27ef978d44
Step 2 : RUN apt-get update && apt-get install -y ruby ruby-dev
---> Using cache
---> 6dd17e0853b4
```

## Tutoriel - boot2docker

```
Step 3 : RUN gem install sinatra
----> Using cache
----> 47c01091dc50
Successfully built 47c01091dc50
root@boot2docker:/construction#
```

Cette nouvelle image ne prend que quelques secondes à être construite car elle l'a déjà été pour le grand général Toto. Elle a d'ailleurs le même IDentifiant **47c01091dc50**.

IDentifiant de l'image **michelandre/documentation:latest**.

```
root@boot2docker:/construction# docker images | grep documentation
michelandre/documentation latest 47c01091dc50 11 hours ago 317.2 MB
root@boot2docker:/construction#
```

IDentifiant des images du grand général: **general-toto/sinatra:latest** et **general-toto/sinatra:v1**.

```
root@boot2docker:/construction# docker images | grep general
general-toto/sinatra devel 47c01091dc50 11 hours ago 317.2 MB
general-toto/sinatra v1 47c01091dc50 11 hours ago 317.2 MB
root@boot2docker:/construction#
```

On se logue chez Docker Hub.

```
root@boot2docker:/construction# docker login

Username: nom-de-l'usager
Password: mot-de-passe
Email: adresse-courriel
Login Succeeded
root@boot2docker:/construction#
```

On y téléverse l'image.

```
root@boot2docker:/construction# docker push michelandre/documentation:latest

The push refers to a repository [michelandre/documentation] (len: 1)
Sending image list
Pushing repository michelandre/documentation (1 tags)
511136ea3c5a: Image successfully pushed
f3c84ac3a053: Image successfully pushed
a1a958a24818: Image successfully pushed
9fec74352904: Image successfully pushed
d0955f21bf24: Image successfully pushed
1b27ef978d44: Image successfully pushed
6dd17e0853b4: Image successfully pushed
47c01091dc50: Image successfully pushed
Pushing tag for rev [47c01091dc50] on {https://cdn-registry-
1.docker.io/v1/repositories/michelandre/documentation/tags/latest}
root@boot2docker:/construction#
```

On se rend chez Docker Hub, on se logue et on affiche les étiquettes contenues dans notre registre. Ce registre est public et n'importe qui peut alors télécharger cette images en utilisant **docker pull**.

```
docker pull michelandre/documentation
```

Updated 43 minutes ago

michelandre / documentation

Pull this repository | docker pull michelandre/documentation

Registre pour les téléversements lors de la documentation de la série boot2docker

☆ 0 🗨 0 📄 0

Information | Tags | Properties

latest

🕒 2015-03-23 11:01:26  
👤 michelandre

## 5. Suppression d'une image

Vous pouvez supprimer des images de votre hôte boot2docker en utilisant **docker rmi**.

Supprimons l'image **training/sinatra** dont nous n'avons plus besoin.

Cherchons l'IDentifiant de **training/sinatra**.

```
root@boot2docker:/construction# docker images | grep training
training/sinatra      latest      f0f4ab557f95      9 months ago      447 MB
training/webapp      latest      31fa814ba25a      9 months ago      278.8 MB
root@boot2docker:/construction#
```

On supprime **f0f4ab557f95**.

```
root@boot2docker:/construction# docker rmi f0f4ab557f95
Error response from daemon: Conflict, cannot delete f0f4ab557f95 because the container a180bba9ce55 is using it, use -f to force
FATA[0000] Error: failed to remove one or more images
root@boot2docker:/construction#
```

Le conteneur **a180bba9ce55** utilise l'image. Il nous faut supprimer le conteneur avant de pouvoir supprimer l'image.

On supprime donc le conteneur.

```
root@boot2docker:/construction# docker rm a180bba9ce55
Error response from daemon: Conflict, cannot delete f0f4ab557f95 because the container 222c580654b1 is using it, use -f to force
FATA[0000] Error: failed to remove one or more images
root@boot2docker:/construction#
```

Un autre conteneur, **222c580654b1** utilise aussi l'image et il nous empêche de la supprimer.

On recherche tous les conteneurs qui utilise notre image.

```
root@boot2docker:/construction# docker ps -a | grep training/sinatra
222c580654b1      training/sinatra:latest      "/bin/bash"          18 hours ago
Exited (0) 18 hours ago      berserk_jang
root@boot2docker:/construction#
```

Il n'y a plus que le conteneur **222c580654b1** qui utilise encore notre image. On supprime ce conteneur.

```
root@boot2docker:/construction# docker rm 222c580654b1
222c580654b1
root@boot2docker:/construction#
```

Il ne devrait plus y avoir d'obstacles à la suppression de notre image.

```
root@boot2docker:/construction# docker images | grep training
training/webapp      latest      31fa814ba25a      9 months ago
278.8 MB
root@boot2docker:/construction#
```

Notre image est disparue. La seule image restante de l'utilisateur **training** est celle utilisée dans le chapitre précédent.



Testez vos connaissances avec le [tutoriel Dockerfile](#).



Jusqu'à présent, nous avons vu comment créer des applications individuelles à l'intérieur d'un conteneur Docker. Dans le prochain chapitre, nous allons apprendre à construire des piles entières d'application en reliant plusieurs conteneurs Docker entre-eux.

# XVI- Liaisons inter-conteneurs

## 1. Introduction

Référence: <http://docs.docker.com/userguide/dockerlinks/>.

Dans le chapitre [Client Docker](#), nous avons vu comment on pouvait se connecter à un service roulant à l'intérieur d'un conteneur Docker via un port réseau. Une connexion par port réseau n'est qu'une des façons parmi d'autres d'interagir avec des services et des applications en cours d'exécution à l'intérieur de conteneurs Docker. Dans ce document, nous allons brièvement revoir la connexion via un port réseau puis nous allons présenter une autre méthode d'accès, les **liaisons inter-conteneur**.

## 2. Connexion utilisant le mappage de port réseau

Dans le chapitre [Client Docker](#), nous avons créé un conteneur qui exécutait une application **Python Flask**.

```
root@boot2docker:~# docker run -d -P training/webapp python app.py
3c582ac6ad3d1fd5df1c8f9759d6739f4fc3d0675791846c8113e51d300830f3
root@boot2docker:~#
```

Un conteneur possède un réseau interne et une adresse **IP** (*nous avons utilisé la commande [docker inspect](#) pour afficher l'adresse IP du conteneur*). Docker peut avoir une variété de configurations réseau. Vous pouvez en apprendre beaucoup plus sur les réseaux Docker [ici](#).

Lorsque nous avons créé le conteneur, l'option **-P** a été utilisé pour mapper automatiquement le port réseau interne vers un port aléatoire de notre hôte boot2docker. Lorsqu'on exécute **docker ps**, nous voyons que le port **5000** à l'intérieur du conteneur a été mappé au port **49155** de l'hôte.

```
root@boot2docker:~# docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
3c582ac6ad3d	training/webapp	latest "python app.py"	18 minutes ago	Up 18 minutes
	0.0.0.0:49155->5000/tcp	distracted_wright		

```
root@boot2docker:~#
```

On arrête le conteneur.

```
root@boot2docker:~# docker stop 3c582ac6ad3d
3c582ac6ad3d
root@boot2docker:~#
```

On vérifie si un conteneur roule.

```
root@boot2docker:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			

```
root@boot2docker:~#
```

On a également vu comment on pouvait lier les ports d'un conteneur à un port spécifique en utilisant l'option **-p**.

```
root@boot2docker:~# docker run -d -p 5000:5000 training/webapp python app.py
```

```
8cbd48a96b6e0f7223575874d43774ff91d38679015237e290712efe333ac5d2
```

```
root@boot2docker:~#
```

On vérifie le mappage des ports puis on arrête le conteneur.

```
root@boot2docker:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			
8cbd48a96b6e	training/webapp:latest	"python app.py"	9 seconds ago	Up 8
seconds	0.0.0.0:5000->5000/tcp	focused_rosalind		

```
root@boot2docker:~#
```

On a aussi vu pourquoi ce n'était pas la meilleure façon de lier les ports de cette manière à cause de la contrainte de ne pouvoir utiliser qu'un seul conteneur sur ce port spécifique.

Il existe aussi quelques autres façons de configurer l'option **-p**.

Par défaut, l'option **-p** va lier le port spécifié à toutes les interfaces sur la machine hôte.

On peut également spécifier une liaison à une interface spécifique i.e. localhost (**127.0.0.1**).

```
root@boot2docker:~# docker run -d -p 127.0.0.1:5000:5000 training/webapp python app.py
```

```
304c5a3d8cc04663d30c3e9eea63c3513fe68b976b22979872431599312a9512
```

```
root@boot2docker:~#
```

Cette commande va lier le port **5000** à l'intérieur du conteneur au port **5000** de l'interface localhost ou **127.0.0.1** sur la machine hôte.

On vérifie le mappage des ports puis on arrête le conteneur.

```
root@boot2docker:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			
304c5a3d8cc0	training/webapp:latest	"python app.py"	14 seconds ago	Up 13
seconds	127.0.0.1:5000->5000/tcp	fervent_wright		

```
root@boot2docker:~#
```

```
root@boot2docker:~#
```

On peut aussi lier le port **5000** du conteneur à un port dynamique et seulement à localhost et non plus à toutes les interfaces de l'hôte.

```
root@boot2docker:~# docker run -d -p 127.0.0.1::5000 training/webapp python app.py
```

```
707395a76d55131e2d1e2a3b200a8b501b33c0077e45c37b435d0846f1e01c06
```

```
root@boot2docker:~#
```

On vérifie le mappage des ports puis on arrête le conteneur.

```
root@boot2docker:~# docker ps
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS
707395a76d55	training/webapp	latest	"python app.py"	11 seconds ago	Up 10
seconds	127.0.0.1:49153->5000	/tcp	elegant_davinci		

```
root@boot2docker:~#
```

On peut également lier les ports UDP en ajoutant le suffixe `"/udp"` au port de l'argument.

```
root@boot2docker:~# docker run -d -p 127.0.0.1:5000:5000/udp training/webapp python app.py
```

```
fabc48ce7eeb76a460f51895689bfb1a72c31446e15ff41c5af58d8e45aa452c
```

```
root@boot2docker:~#
```

On vérifie le mappage des ports et on arrête le conteneur.

```
root@boot2docker:~# docker ps
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS
fabc48ce7eeb	training/webapp	latest	"python app.py"	13 seconds ago	Up 13
seconds	5000/tcp, 127.0.0.1:5000->5000	/udp	desperate_mcclintock		

```
root@boot2docker:~#
```

## 2.1. docker port

On a également appris le raccourci `docker port` qui nous affiche le mappage des port actuels.

Affichage de l'aide de l'argument `port`.

```
root@boot2docker:~# docker port -h
```

```
Usage: docker port [OPTIONS] CONTAINER [PRIVATE_PORT[/PROTO]]
```

List port mappings for the CONTAINER, or lookup the public-facing port that is NAT-ed to the PRIVATE\_PORT

```
--help=false      Print usage
```

```
root@boot2docker:~#
```

On lance `training/webapp`.

```
root@boot2docker:~# docker run -d -p 127.0.0.1:5000:5000 training/webapp python app.py
```

```
cd0270e467f971565a16534d2d571cf15919f955fa65ff3d769fa8d95e47c21d
```

```
root@boot2docker:~#
```

On affiche le mappage des ports.

```
root@boot2docker:~# docker port cd0270e467f9
```

```
5000/tcp -> 127.0.0.1:5000
```

```
root@boot2docker:~#
```

La commande **docker port** est également utile pour afficher les configurations d'un port spécifique.

Si vous avez mappé le port du conteneur à **localhost** sur la machine hôte, alors la commande **docker port** peut afficher le mappage du port spécifié.

```
root@boot2docker:~# docker port cd0270e467f9 5000
127.0.0.1:5000
root@boot2docker:~#
```

On arrête le conteneur.



L'option **-p** peut être utilisée plusieurs fois pour configurer plusieurs ports.



```
root@boot2docker:~# docker run -d -p 127.0.0.1:6000:5000 \
                                -p 127.0.0.1:6000:5000/udp \
                                training/webapp python app.py
b943cad08c737e979ff0abedaa119654811a3841f390ba2dac0af173370ff919
root@boot2docker:~#
```

Référence: <https://docs.docker.com/reference/commandline/cli/>.



Vous pouvez trouver tous les ports mappés en ne spécifiant pas de **PRIVATE\_PORT** ou en spécifiant un port et un protocole.

```
root@boot2docker:~# docker port b943cad08c73
5000/tcp -> 127.0.0.1:6000
5000/udp -> 127.0.0.1:6000
root@boot2docker:~#
```

Ne pas oublier l'espace entre **5000** et **/tcp**.

```
root@boot2docker:~# docker port b943cad08c73 5000 /tcp
5000/tcp -> 127.0.0.1:6000
5000/udp -> 127.0.0.1:6000
root@boot2docker:~#
```

```
root@boot2docker:~# docker port b943cad08c73 5000 /udp
5000/tcp -> 127.0.0.1:6000
5000/udp -> 127.0.0.1:6000
root@boot2docker:~#
```

La commande **docker port** nous montre bien que les protocoles **TCP** et **UDP** du port interne **5000** du conteneur sont mappés aux protocoles **TCP** et **UDP** du port **6000** de l'hôte.

On peut aussi vérifier les ports avec la commande **docker ps**.

```
root@boot2docker:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS
b943cad08c73       training/webapp:latest  "python app.py"    9 minutes ago      Up 9
minutes
127.0.0.1:6000->5000/tcp, 127.0.0.1:6000->5000/udp    agitated_wozniak
root@boot2docker:~#
```

On arrête le(s) conteneur(s).

# XVII- Connexion par liens

## 1. Introduction

Le mappage de ports réseau n'est pas la seule façon pour les conteneurs Docker de se connecter les uns aux autres. Docker possède également un système de liaison qui permet de lier plusieurs conteneurs ensemble et d'envoyer des informations de connexion de l'un à l'autre. Lorsque des conteneurs sont reliés, les informations sur un conteneur source peuvent être envoyées à un conteneur destinataire. Ceci permet au destinataire de voir les informations sélectionnées décrivant les aspects du conteneur source.

## 2. L'importance de donner des noms

Pour établir des liens, Docker s'appuie sur le nom des conteneurs. On a déjà vu que chaque conteneur qu'on crée reçoit un nom qui lui est automatiquement attribué. Ci-dessus, vous avez sans doute remarqué les noms tels que **desperate\_mcclintock**, **elegant\_davinci**, **fervent\_wright**, **focused\_rosalind** etc. Vous aussi, vous pouvez donner des noms à vos conteneurs.

Ces noms ont deux fonctions utiles.

- Il peut être utile de nommer des conteneurs qui exécutent des fonctions spécifiques d'une manière qui facilite la mémorisation de leur nom.. Nommer **web** un conteneur d'applications Web est des plus significatif.
- Fournir à Docker un critère qui lui permet de faire référence aux conteneurs. Vous pouvez très bien spécifier à Docker de lier le conteneur **web** au conteneur **db** qui roule la base de données.

On peut nommer un conteneur en utilisant l'argument **--name**.

```
root@boot2docker:~# docker run -d -P --name web training/webapp python app.py
5dfad81b67b8f7780b05365d8436e0290f0112786f47a225ebca7f9057173512
root@boot2docker:~#
```

Cette commande a lancé un nouveau conteneur en utilisant l'argument **--name** pour le nommer **web**. Vous pouvez voir le nom du conteneur en utilisant **docker ps**.

```
root@boot2docker:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
5dfad81b67b8      training/webapp:latest  "python app.py"    2 minutes ago      Up 2
minutes           0.0.0.0:49156->5000/tcp    web
```

Vous pouvez également utiliser **docker inspect** pour retourner le nom du conteneur.

```
root@boot2docker:~# docker inspect -f "{{ .Name }}" 5dfad81b67b8
/web
root@boot2docker:~#
```

Les noms des conteneurs doivent être uniques. Vous ne pouvez nommer **web** qu'un seul conteneur. Si vous voulez réutiliser le nom d'un conteneur, vous devez supprimer l'ancien avant de pouvoir en créer un nouveau avec le même nom. Comme alternative, vous pouvez utiliser l'argument **--rm** avec la commande **docker run**. Cet argument supprime le conteneur immédiatement après son arrêt.

On arrête le conteneur **web**.

```
root@boot2docker:~# docker stop web
web
root@boot2docker:~#
```

On doit supprimer le conteneur **web** que nous avons créé précédemment de sorte qu'on pourra le remplacer par un nouveau conteneur lié portant le même nom.

```
root@boot2docker:~# docker rm -f web
web
root@boot2docker:~#
```

# XVIII- Communication par liens

## 1. Introduction

Les liens permettent aux conteneurs de se découvrir les uns les autres et de transférer, en toute sécurité, des informations d'un conteneur à l'autre.

## 2. Création d'un lien

Lorsqu'on configure un lien, on crée un canal entre un conteneur source et un conteneur de destination. Le destinataire peut alors accéder à des données sur la source. Pour créer un lien, on utilise l'argument `--link`. D'abord, nous allons créer un nouveau conteneur roulant une base de données et le nommer `db`.

```
root@boot2docker:~# docker run -d --name db training/postgres

Unable to find image 'training/postgres:latest' locally
258105bea10d: Download complete
511136ea3c5a: Download complete
35f6dd4dd141: Download complete
7baf0ef6f14a: Download complete
e497c7c1bfbb: Download complete
5cf8fd909c6c: Download complete
8726e050fbc9: Download complete
043c01407567: Download complete
65a89e6a06f8: Download complete
6af9ddfafbd3: Download complete
316f4525806b: Download complete
bfb096044e3: Download complete
444db2eae2c3: Download complete
e06e512105c3: Download complete
Status: Downloaded newer image for training/postgres:latest
6f043a6aab4e1efc96bde07f65c415e6d7dce5959f06c144ba1fdde1deee9d49
root@boot2docker:~#
```

Comme on peut le voir, cette image se compose de plusieurs couches. La commande `docker run` a créé un nouveau conteneur avec l'image `training/postgres`. L'argument `--name` a nommé ce conteneur `db` vu qu'il roule une base de données PostgreSQL.

On crée un nouveau conteneur `web` et on le lie au conteneur `db`.

```
root@boot2docker:~# docker run -d -P --name web --link db:lien_db training/webapp python
app.py

9a5ffe12a0d9f1f73b0cb8ad6415e043fd2646ee26d0d2d3411d8e8d866fd00
root@boot2docker:~#
```

Cette commande a créé un nouveau conteneur `web` et l'a relié au conteneur `db` qu'on a créé précédemment.

L'argument `--link` prend la forme:

```
--link <name or id>:alias
```

- **name** est le nom du conteneur auquel nous nous relierons.

- **alias** est un nom pour le lien.

Exemple.

```
--link db:lien_db
```

On verra bientôt comment est utilisé cet alias.



Le conteneur **db** de l'image **training/postgres** est le **conteneur source**.

Le conteneur **web** de l'image **training/webapp** est le **conteneur client** ou **conteneur destinataire**.

Examinons le conteneur **web** avec la commande **docker inspect**.

```
root@boot2docker:~# docker inspect -f "{{ .HostConfig.Links }}" web
[/db:/web/lien_db]
root@boot2docker:~#
```



**[/conteneur-source:/conteneur-destinataire/alias-du-lien]**.

Le conteneur source **db** et le conteneur destinataire **web** sont reliés par le lien **lien\_db**. Ce lien permet au conteneur **web** d'accéder aux informations et donnée du conteneur **db**.

On liste les conteneurs présentement actifs.

```
root@boot2docker:~# docker ps
CONTAINER ID        IMAGE                               COMMAND                  CREATED
STATUS            PORTS                               NAMES
9a5ffe12a0d9      training/webapp:latest            "python app.py"        4 minutes ago    Up
4 minutes        0.0.0.0:49158->5000/tcp            web
6f043a6aab4e      training/postgres:latest          "su postgres -c '/us  About an hour ago Up
About an hour     5432/tcp                            db
root@boot2docker:~#
```

- Le port interne **5000** du conteneur **web** est mappé au port **49158** du serveur hôte.
- **PostgreSQL** offre ses données seulement sur le port interne 5432 du conteneur **db**.

### 3. Avantages

On peut voir que le conteneur **web** est maintenant lié au conteneur **db**, ce qui lui permet d'accéder aux informations de la base de données **PostgreSQL**.

Alors quel est le but réel de la procédure de relier deux conteneurs ensemble? On a appris qu'un lien permet à un conteneur source de fournir des informations sur lui-même à un conteneur destinataire. Dans notre exemple, le destinataire **web** peut accéder aux informations et aux données du conteneur source **db**. Pour ce faire, Docker crée un tunnel sécurisé entre les conteneurs qui n'ont alors pas besoin d'exposer leurs ports à l'extérieur de leur conteneur respectif. Il est à remarquer que nous n'avons pas utiliser les arguments **-P** ou **-p** lorsqu'on a lancé le conteneur **db**. C'est un précieux avantage de la liaison. Nous n'avons pas à exposer sur le réseau les ports du conteneur source, ici la base de données **PostgreSQL**.

Docker expose au conteneur source les informations de connectivité vers le conteneur destinataire de deux façons:

- À travers des variables d'environnement.
- Par la mise à jour des fichiers **/etc/hosts** des conteneurs.

## 4. Variables d'environnement

Docker crée plusieurs variables d'environnement lorsqu'on relie des conteneurs. Il crée automatiquement des variables d'environnement dans le conteneur destinataire sur la base de l'argument `--link`. Il y exposera également toutes les variables d'environnement Docker provenant du conteneur source. Celles-ci incluent des variables à partir:

- des commandes **ENV** contenues dans le fichier **Dockerfile** du conteneur source
- des options `-e`, `--env` et `--env-file` de la commande **docker run** lors du lancement du conteneur source.

Ces variables d'environnement permettent la découverte programmatique, depuis l'intérieur même du conteneur destinataire, des informations liées au conteneur source.



**Attention:** Il est important de comprendre que toutes les variables d'environnement parvenant à Docker depuis un conteneur source, sont mises à la disposition de tout autre conteneur relié à cette source. L'accès à ces variables pourrait avoir des conséquences désastreuses pour la sécurité si des données sensibles sont stockées dans ces dernières.

Docker crée une variable d'environnement `<alias>_NAME` pour chaque conteneur destinataire à partir des paramètres de l'argument `--link`. Par exemple, si un nouveau conteneur appelé **web** est lié à un conteneur appelé **db** via `--link db:lien_webdb`, alors Docker crée une variable `LIEN_WEBDB_NAME=/web/lien_webdb` dans le conteneur **web**.

Docker définit également un ensemble de variables d'environnement pour chaque port exposé par le conteneur source. Chaque variable a un préfixe unique sous la forme:

```
<name>_PORT_<port>_<protocol>
```

Les composantes de ce préfixe sont:

`<name>` .....l'alias spécifié avec l'argument `--link` (par exemple, **lien\_webdb**)

`<port>` .....le numéro du port exposé (par exemple, **8080**)

`<protocole>`..le protocole **TCP** ou **UDP**

Docker utilise ce format de préfixe pour définir trois variables d'environnement distinctes:

- La variable **prefix\_ADDR** contient l'adresse IP de l'URL  
exemple `LIEN_WEBDB_PORT_8080_TCP_ADDR=172.17.0.82`.
- La variable **prefix\_PORT** contient seulement le numéro de port de l'URL  
exemple `LIEN_WEBDB_PORT_8080_TCP_PORT=8080`.
- La variable **prefix\_PROTO** contient seulement le protocole de l'URL  
exemple `LIEN_WEBDB_PORT_8080_TCP_PROTO=tcp`.

Si le conteneur expose plusieurs ports, une variable d'environnement est définie pour chacun d'eux. Si un conteneur expose **quatre ports** Docker crée **12 variables** d'environnement, **trois** pour chaque port.

En outre, Docker crée une variable d'environnement appelée `<alias>_PORT`. Cette variable contient l'URL du premier port exposé par le conteneur source. Le 'premier' port est défini comme le port exposée avec le plus petit nombre. Par exemple, considérez la variable `LIEN_WEBDB_PORT=tcp://172.17.0.82:8080`, si ce port est utilisé à la fois pour TCP et UDP, alors celui de TCP est spécifié.

Enfin, Docker expose également chaque variable d'environnement Docker provenant du conteneur source, comme une variable d'environnement dans le destinataire. Pour chacune de ces variables, Docker crée une variable `<alias>_ENV_<name>` dans le conteneur destinataire. La valeur de cette dernière variable est définie par la valeur que Docker a utilisée lorsqu'il a lancé le conteneur source.



## 6. Mise à jour du fichier /etc/hosts

En plus des variables d'environnement, Docker ajoute une entrée spécifiant le **conteneur source** dans le fichier **/etc/hosts** du conteneur destinataire.

Pour voir les entrées d'un nouveau conteneur destinataire qu'on relie au conteneur source **db**.

```
root@boot2docker:~# docker run -t -i --rm --link db:lien_db training/webapp /bin/bash
root@1e775c411953:/opt/webapp#
```

Une fois à l'intérieur du conteneur, on affiche le fichier **/etc/hosts**.

```
root@1e775c411953:/opt/webapp# cat /etc/hosts
172.17.0.45      1e775c411953
127.0.0.1       localhost
::1            localhost ip6-localhost ip6-loopback
fe00::0        ip6-localnet
ff00::0        ip6-mcastprefix
ff02::1        ip6-allnodes
ff02::2        ip6-allrouters
172.17.0.40     lien_db
root@1e775c411953:/opt/webapp#
```

On peut y voir deux entrées intéressantes. La première **172.17.0.45 1e775c411953**, est une entrée pour le conteneur **web** et qui utilise l'IDentifiant du conteneur en tant que nom d'hôte. La seconde entrée **172.17.0.40 lien\_db**, utilise l'alias de liaison pour référencer l'adresse IP du **conteneur source db**. Vous pouvez faire un ping du **conteneur source** en utilisant ce nom d'hôte.

On doit installer l'utilitaire **ping** car il n'est pas inclus dans l'image du conteneur.

```
root@1e775c411953:/opt/webapp# apt-get install -yqq inetutils-ping

debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package netbase.
(Reading database ... 8961 files and directories currently installed.)
Unpacking netbase (from ../netbase_4.47ubuntu1_all.deb) ...
Selecting previously unselected package inetutils-ping.
Unpacking inetutils-ping (from ../inetutils-ping_2%3a1.8-6_amd64.deb) ...
Setting up netbase (4.47ubuntu1) ...
Setting up inetutils-ping (2:1.8-6) ...
root@1e775c411953:/opt/webapp#
```

On lance 3 ping du conteneur source **db**.

```
root@1e775c411953:/opt/webapp# ping -c3 lien_db

PING lien-db (172.17.0.40): 48 data bytes
56 bytes from 172.17.0.40: icmp_seq=0 ttl=64 time=0.260 ms
56 bytes from 172.17.0.40: icmp_seq=1 ttl=64 time=0.147 ms
56 bytes from 172.17.0.40: icmp_seq=2 ttl=64 time=0.215 ms
--- lien-db ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.147/0.207/0.260/0.046 ms
root@1e775c411953:/opt/webapp#
```

Ici, on ping le conteneur source **db** en utilisant son entrée de nom d'hôte, **lien\_db** dont la résolution de nom est **172.17.0.40**. On peut utiliser ce nom d'hôte pour configurer une application qui utilise le conteneur **db**.



On peut lier plusieurs conteneurs destinataires à une source unique. Par exemple, on peut avoir plusieurs conteneurs destinataires **web** (avec des noms différents) connectés à votre conteneur source **db**.

On sort du conteneur. Il sera automatiquement détruit à cause de l'argument `--rm` de la commande.

Si vous redémarrez le conteneur source, les fichiers `/etc/hosts` des conteneurs liés seront automatiquement mis à jour avec la nouvelle adresse IP du **conteneur source** et permettra à la communication liée de continuer à fonctionner.

On redémarre le conteneur **db**.

```
root@boot2docker:~# docker restart db
db
root@boot2docker:~#
```

On lance un nouveau conteneur qu'on relie au conteneur source **db**. Il affichera le fichier `/etc/hosts`. Une fois l'affichage effectué, le nouveau conteneur se fermera et il sera automatiquement détruit par l'argument `--rm` de la ligne de commande.

```
root@boot2docker:~# docker run -t -i --rm --link db:lien_db training/webapp cat /etc/hosts
172.17.0.50      a9f25e0d955e
127.0.0.1       localhost
::1            localhost ip6-localhost ip6-loopback
fe00::0        ip6-localnet
ff00::0        ip6-mcastprefix
ff02::1        ip6-allnodes
ff02::2        ip6-allrouters
172.17.0.46     lien_db
root@1a4878aa75bc:/opt/webapp#
```

L'adresse **IP** pour le conteneur source **db** a bien été mis à jour lors du redémarrage de celui-ci mais **lien\_db** est encore là et pointe sur la nouvelle adresse du conteneur source **db**.

Maintenant qu'on sait lier des conteneurs Docker, la prochaine étape est d'apprendre à gérer les données, les volumes et les montages (*mount*) à l'intérieur de nos conteneurs.

C'est ce qu'on verra dans le prochain chapitre.

# XIX- Gestion des données dans les conteneurs

## 1. Introduction

Jusqu'ici, nous avons été introduits à certains concepts de base de Docker: les images Docker, l'adressage réseau et les liaisons inter-conteneurs. Dans ce chapitre, nous allons discuter de la gestion des données à l'intérieur et entre les conteneurs Docker.

Nous allons examiner les deux principales gestions de données de Docker.

- La gestion des volumes de données.
- La gestion des conteneurs-volume-de-données.

## 2. Volumes de données

Un volume de données est un répertoire spécialement assigné à un ou plusieurs conteneurs et qui contourne Union File System. Les volumes de données offrent plusieurs fonctionnalités pour les données persistantes ou partagées.

- Les volumes sont initialisés lorsqu'un conteneur est créé. Si l'image de base du conteneur contient des données au point de montage spécifié, les données sont copiées dans le nouveau volume.
- Les volumes de données peuvent être partagés et réutilisés entre conteneurs.
- Les modifications apportées à un volume de données sont directement effectuées.
- Ces modifications ne seront pas incluses lors d'une mise à jour d'une image.
- Les volumes de données persistent même si le conteneur lui-même est supprimé.

Les volumes de données sont conçus pour garder en permanence les données, indépendamment du cycle de vie du conteneur. Ainsi, Docker ne supprime jamais automatiquement les volumes lorsqu'on supprime un conteneur ni n'exécute de [ramasse-miettes](#) sur les volumes qui ne sont plus référencés par un conteneur.

## 3. Création d'un volume de données

Vous pouvez ajouter un volume de données à un conteneur en utilisant l'argument `-v` dans les commandes `docker create` et `docker run`.

Vous pouvez utiliser plusieurs fois l'argument `-v` pour monter plusieurs volumes de données.

Montons un seul volume dans notre conteneur d'application Web.

```
root@boot2docker:~# docker run -d -P --name web -v /webapp training/webapp python app.py
FATA[0000] Error response from daemon: Conflict. The name "web" is already in use by container 9a5ffe12a0d9. You have to delete (or rename) that container to be able to reuse that name.
root@boot2docker:~#
```

Le nom **web** est déjà utilisé par le conteneur **9a5ffe12a0d9**. Il nous faut donc arrêter et détruire ce conteneur avant d'en créer un nouveau utilisant le même nom.

```
root@boot2docker:~# docker stop web
web
root@boot2docker:~#
```

```
root@boot2docker:~# docker rm web
web
root@boot2docker:~#
```

On essaie encore la création du conteneur.

```
root@boot2docker:~# docker run -d -P --name web -v /webapp training/webapp python app.py
fd9c749fc7060a21a67e6748e113575b34033f5fa7a5815a388d4e04b7dac158
root@boot2docker:~#
```

La commande a créé un nouveau volume: **/webapp** à l'intérieur du conteneur **web**.



Vous pouvez également utiliser l'instruction de volume dans un **fichier Dockerfile** afin d'ajouter un ou plusieurs nouveaux volumes à n'importe quel conteneur créé à partir de cette image.

#### 4. Monter un répertoire hôte en tant que volume de données

En plus de créer un volume en utilisant l'option **-v** on peut aussi monter un répertoire de l'hôte dans un conteneur.



Si on utilise boot2docker (*notre cas*), le daemon Docker n'a qu'un accès limité au système de fichiers OSX/Windows (*ce qui n'est pas notre cas vu que tous nos systèmes sont des Linux*). Docker essaie d'auto-partager les répertoires **/Users** (OSX) ou **C:\Users** (Windows). On peut monter des fichiers de ces répertoires ou ces répertoires eux-même à l'aide de **docker run -v /Users/<path>:/<container path> ...** (OSX) ou **docker run -v /c/Users/<path>:/<container path> ...** (Windows). Tous les autres chemins proviennent du système de fichiers de la machine virtuelle qui roule Docker.



Pour la même raison que précédemment, si on veut utiliser le nom **web** pour un nouveau conteneur, il nous faut arrêter et supprimer le conteneur qui utilise présentement ce nom.

On arrête le conteneur nommé **web** lancé auparavant.

```
root@boot2docker:~# docker stop web
web
root@boot2docker:~#
```

Vu que le dernier conteneur **web** référençait un volume, il nous faut utiliser un argument supplémentaire **"-v"** pour le détruire. Pour plus de détails, voir l'explication [Mise en garde importante](#) à la page [87](#).



```
root@boot2docker:~# docker rm -v web
web
root@boot2docker:~#
```

On peut maintenant lancer la commande ci-dessous qui utilise encore **web** pour nommer le nouveau conteneur.



```
root@boot2docker:~# docker run -d \
-P \
--name web \
-v /src/webapp:/opt/webapp \
training/webapp python app.py

8bfa677dd9bd6632a0700f5d65e45bf9155a36c1007644518a76ff677568e49d
root@boot2docker:~#
```

Cette commande montera le répertoire source **/src/webapp** du serveur hôte boot2docker dans le répertoire de destination **/opt/webapp** du conteneur **web**.

- Si le point de montage **/opt/webapp** existe déjà à l'intérieur de l'image **training/webapp** du conteneur **web**, son contenu sera remplacé par le contenu du répertoire source **/src/webapp** de l'hôte boot2docker afin de rester cohérent avec le comportement conventionnel de **mount**.
- Le **répertoire** de l'hôte doit être spécifié avec un chemin absolu.
- Si le répertoire n'existe pas sur l'hôte, Docker le créera automatiquement pour nous.
- Ce genre de montage d'un volume est très utile pour le développement logiciel car on peut monter notre code source à l'intérieur du conteneur de destination et voir notre application au travail lorsqu'on change le contenu de la source.



Cette fonctionnalité de montage de volume n'est pas disponible à partir d'un **fichier Dockerfile** en raison de la portabilité et de la finalité du partage d'images construites. Le répertoire de l'hôte est, de par sa nature, dépendant de l'hôte; un répertoire d'hôte spécifié dans un fichier Dockerfile ne fonctionnerait probablement pas sur tous les hôtes.



Un volume Docker est par défaut en **mode lecture-écriture**. Par contre, on peut monter un volume en mode lecture seule, "**ro**" (*read only*).

Encore une fois, on arrête et on supprime le conteneur **web** lancé précédemment.

On lance la commande suivante.



```
root@boot2docker:~# docker run -d \
-P \
--name web \
-v /src/webapp:/opt/webapp:ro \
training/webapp \
python app.py

646f59bca91ccfcd6e6f7ca75457742bc7ec09940f4d006f67d66bccf28a30c1
root@boot2docker:~#
```

La commande a monté le même répertoire **/src/webapp** mais on a ajouté l'argument **ro** pour préciser que le **mount** doit être en lecture seule.

On arrête et on supprime le conteneur **web** qu'on vient de lancer.

## 5. Monter un fichier de l'hôte en tant que volume de données

L'argument `-v` peut également être utilisé pour monter un fichier au lieu d'un répertoire du serveur hôte.

Le fichier de l'historique du shell `ash` sur le serveur `boot2docker` est: `/root/.ash_history`.

Le fichier de l'historique du shell `bash` sur le serveur `ubuntu:latest` est: `/root/.bash_history`.

On lance la commande.

```
root@boot2docker:~# docker run -i \
-t \
-v /root/.ash_history:/root/.bash_history \
ubuntu:latest \
/bin/bash
root@0b9333aa9e75:/#
```

Cette commande va:

- `-i`, lancer le nouveau conteneur en mode interactive
- `-t`, offrir un terminale
- `-v`, mapper le fichier source `/root/.ash_history` du serveur `boot2docker` sur le fichier de destination `/root/.bash_history` du conteneur
- `ubuntu:latest`, créer un conteneur avec l'image la plus récente d'Ubuntu
- `/bin/bash`, lancer un **shell bash** à l'intérieur du nouveau conteneur
- ajouter au fichier source de l'hôte i.e. `/root/.ash_history` toutes les commandes lancées à l'intérieur du conteneur, en passant par le fichier `/bin/bash` du conteneur i.e. `/root/.bash_history`.

### Conséquences:

- L'historique de l'hôte sera disponible à l'intérieur du conteneur.
- Après la sortie, l'historique du conteneur pourra être consultée dans l'historique de l'hôte.



De nombreux outils utilisés pour modifier des fichiers tels `vi` et `sed` peuvent entraîner une modification de l'**inode**. Depuis **Docker v1.1.0**, ces modifications de fichiers vont produire: **erreur: "sed: ne peut pas renommer ./sedKdJ9Dy: Périphérique ou ressource occupé"**. Au cas où vous souhaiteriez modifier le fichier monté, il est souvent préférable de monter le répertoire parent du fichier.

Nous vous laissons l'expérimentation de cette procédure.

Avant de passer au prochain chapitre, on arrête et on supprime tous les conteneurs qui roulent présentement.

# XX- Conteneur-volume-de-données

## 1. Introduction

Si vous avez des données persistantes que vous souhaitez partager avec des conteneurs ou si vous voulez utiliser des données de conteneurs non persistants, il est préférable de créer et nommer un conteneur-volume-de-données, puis de monter les données qu'il contient.

## 2. Mise en garde importante

-  Pour supprimer les volumes d'un disque, vous devez lancer explicitement **docker rm -v** lors de la suppression du dernier conteneur, y compris le conteneur initial, ayant une référence au volume.
-  Lors de la suppression du dernier conteneur, l'argument **-v** permet de mettre à jour ou de migrer efficacement des volumes de données entre conteneurs.
-  Docker ne vous avertira pas lorsque vous supprimez un conteneur sans utiliser l'argument **-v**. Si vous supprimez des conteneurs sans utiliser l'argument **-v**, vous pouvez vous retrouver avec des "**dangling volumes**" qui ne sont plus référencés par un conteneur. Les "**dangling volumes**" sont difficiles à supprimer et peuvent prendre une grande quantité d'espace disque. Les développeurs de Docker travaillent à l'amélioration de la gestion de volumes et vous pouvez vérifier les progrès à ce sujet en vous référant à: [requête #8484](#).

## 3. Création d'un conteneur-volume-de-données

Créons un nouveau conteneur nommé **conteneurdata** ayant un volume **voldata** à partager.

-  Bien que ce conteneur ne roule aucune application, nous allons ré-utiliser l'image **training/postgres**. De la sorte, tous les conteneurs utiliseront des couches communes qui résultera en une économie de l'espace disque.

```
root@boot2docker:~# docker create -v /voldata --name conteneurdata training/postgres
ee47972d16fe36cfc51da512a8cddd2a85abb076a34496ca35e6a350e17a9409
root@boot2docker:~#
```

Le volume **/voldata** du nouveau conteneur **conteneurdata** est maintenant offert en partage.

On peut maintenant utiliser l'argument **--volumes-from** pour monter le volume offert en partage dans un premier conteneur: **conteneurdata01**.

```
docker@boot2docker:~$ docker run -d \
  --volumes-from conteneurdata \
  --name conteneurdata01 \
  training/postgres
6338265b4710b85fa61c6725e404766fe265d7183e5957ebb549a3b376480800
docker@boot2docker:~$
```

On monte aussi le volume `/voldata` dans un deuxième conteneur: `conteneurdata02`.



```
docker@boot2docker:~$ docker run -d \
  --volumes-from conteneurdata \
  --name conteneurdata02 \
  training/postgres
73e53374c63db4ef75f2c07e7c02dcce652f99fcdd742098f12cf9a21d1379e5
docker@boot2docker:~$
```



Dans ces deux derniers cas, si l'image originale `training/postgres` (celle utilisée pour créer tous nos conteneur) contenait déjà un répertoire appelé `/voldata` et qu'on monte `/voldata` depuis `conteneurdata`, `/voldata` dissimulera les fichiers `/voldata` de l'image `training/postgres` originale. Le résultat est que les fichiers de `/voldata` à partir du conteneur `conteneurdata` sont visibles.



Vous pouvez utiliser plusieurs fois l'argument `--volumes-from` pour réunir plusieurs volumes de données à partir de plusieurs conteneurs.

On peut également étendre la chaîne en montant le volume partagé du conteneur `conteneurdata` sur un troisième conteneur `conteneurdata03` via l'intermédiaire de `conteneurdata01` ou `conteneurdata02`.



```
docker@boot2docker:~$ docker run -d \
  --volumes-from conteneurdata02 \
  --name conteneurdata03 \
  training/postgres
fe06f22ba049b1491494e3eff90a2947b99d7c5ab7719052f5b9b62cf8280bd9
docker@boot2docker:~$
```



Si vous supprimez des conteneurs qui montent des volumes, y compris le conteneur initial `conteneurdata` ou les conteneurs subséquents `conteneurdata01` et `conteneurdata02`, les volumes ne seront pas supprimés. Se référer au paragraphe [Mise en garde importante](#) ci-dessus.

## 4. Suppression de conteneurs avec volume

Nous allons arrêter et supprimer tous les conteneurs.

On arrête d'abord les trois conteneurs.

```
root@boot2docker:~# docker stop conteneurdata01 conteneurdata02
conteneurdata01
conteneurdata02
conteneurdata03
root@boot2docker:~#
```

On supprime seulement les deux premiers conteneurs de manière conventionnelle.

```
root@boot2docker:~# docker rm conteneurdata01 conteneurdata02
conteneurdata01
conteneurdata02
root@boot2docker:~#
```



On supprime, sans oublier l'argument `-v`, le dernier conteneur i.e. `conteneurdata03` ayant monté le volume.

```
root@boot2docker:~# docker rm -v conteneurdata03
conteneurdata03
root@boot2docker:~#
```

On conserve `conteneurdata` pour le prochain chapitre.

# XXI- Sauvegarder, restaurer et migrer des volumes

## 1. Rappel de la mise en garde



Pour supprimer les volumes d'un disque, vous devez lancer explicitement **docker rm -v** lors de la suppression du dernier conteneur, y compris le conteneur initial, ayant une référence au volume.

## 2. Contenu du volume

Nous allons examiner le contenu du volume **voldata** afin de pouvoir le comparer après sa modification et lors de sa restauration.



```
root@boot2docker:~# docker run -i -t \
    --volumes-from conteneurdata \
    --name conteneurdata04 \
    training/postgres \
    /bin/bash
root@7306b28b82db:/#
```

Une fois à l'intérieur du nouveau conteneur, nous affichons le contenu de **/voldata** du conteneur nouvellement créé.

```
root@7306b28b82db:/# ls -als /voldata/
total 8
4 drwxr-xr-x  2 root root 4096 Mar 28 14:26 .
4 drwxr-xr-x 61 root root 4096 Mar 28 14:27 ..
root@7306b28b82db:/#
```

Le répertoire est vide.

### 2.1. Modification

Nous allons y créer un fichier pour notre comparaison ultérieure.

```
root@7306b28b82db:/# touch /voldata/fichier_original_dans_voldata
root@7306b28b82db:/#
```

On vérifie.

```
root@7306b28b82db:/# ls -als /voldata/
total 8
4 drwxr-xr-x  2 root root 4096 Mar 28 14:28 .
4 drwxr-xr-x 61 root root 4096 Mar 28 14:27 ..
0 -rw-r--r--  1 root root    0 Mar 28 14:28 fichier_original_dans_voldata
root@7306b28b82db:/#
```

On sort du conteneur.

```
root@7306b28b82db:/# exit
exit
root@boot2docker:~#
```

### 2.1.1. Vérification

On lance un nouveau conteneur pour vérifier si le fichier est toujours là.



```
root@boot2docker:~# docker run -i -t
--volumes-from conteneurdata \
--name conteneurdata05 \
training/postgres \
/bin/bash
root@boot2docker:~#
```

```
root@boot2docker:~# ls -als /voldata/
total 8
4 drwxr-xr-x  2 root root 4096 Mar 28 14:28 .
4 drwxr-xr-x 61 root root 4096 Mar 28 14:30 ..
0 -rw-r--r--  1 root root    0 Mar 28 14:28 fichier_original_dans_voldata
root@0486d1d8e311:/#
```

Il y est.

On sort du conteneur.

```
root@0486d1d8e311:/# exit
exit
root@boot2docker:~#
```

### 3. Sauvegarde d'un volume

D'autres fonctions très utiles que nous pouvons effectuer avec les volumes est de les utiliser pour des sauvegardes, restaurations ou migrations.

#### 3.1. Création d'un conteneur qui sauvegarde un volume

Pour ce faire, on crée un nouveau conteneur qui exécute la sauvegarde du volume.

```
root@boot2docker:~# docker run --volumes-from conteneurdata \
-v $(pwd) :/sauvegarde \
--name conteneur06 \
ubuntu \
tar -cvf /sauvegarde/sauvegarde.tar /voldata

tar: Removing leading `/' from member names
/voldata/
/voldata/fichier_original_dans_voldata
root@boot2docker:~# ls -als
```

- volumes-from conteneurdata** Le nouveau conteneur montera le volume **/voldata** de **conteneurdata**.
- v \$(pwd):/sauvegarde** Il montera aussi le répertoire actuel **\$(pwd)** de l'hôte dans le répertoire **/sauvegarde** du nouveau conteneur.
- name conteneur06** Le nouveau conteneur se nommera conteneur **conteneur06**.
- ubuntu** Il utilisera l'image ubuntu.
- tar -cvf /sauvegarde/sauvegarde.tar /voldata**  
À l'intérieur du conteneur, on exécutera un **tar**:
  - c** - créer un fichier **tar**
  - v** - en mode **verbose** i.e. afficher tout ce que fait le **tar**.
  - f** - le **f**ichier de sortie sera **/sauvegarde/sauvegarde.tar**faire un **tar** du répertoire **/voldata**.

Lorsque ces instructions seront terminées, le conteneur s'arrêtera vu qu'il n'aura plus d'autres instructions à exécuter.

Après la sortie, à l'invite du serveur hôte, on affiche le contenu du répertoire courant.

```
root@boot2docker:~# ls -ls

total 12
  12 -rw-r--r--  1 root   staff   10240 Mar 28 15:46 sauvegarde.tar
root@boot2docker:~#
```

Nous voyons que le **tar** a bien créé le fichier **sauvegarde.tar** dans le répertoire courant.

#### 3.2. Vérification

Pour vérifier le fichier **sauvegarde.tar**, on crée un répertoire temporaire **/temp** on y extrait le fichier.

```
root@boot2docker:~# mkdir /temp

root@boot2docker:~#
```

```
root@boot2docker:~# tar -xvf sauvegarde.tar -C /temp
voldata/
voldata/fichier_original_dans_voldata
root@boot2docker:~#
```

On vérifie.

```
root@boot2docker:~# ls -als /temp/

total 0
 0 drwxr-xr-x  3 root   root    60 Mar 28 16:55 .
 0 drwxr-xr-x 17 root   root   400 Mar 28 16:52 ..
 0 drwxr-xr-x  2 root   root    60 Mar 28 14:28 voldata
root@boot2docker:~#
```

```
root@boot2docker:~# ls -als /temp/voldata/

total 0
 0 drwxr-xr-x  2 root   root    60 Mar 28 14:28 .
 0 drwxr-xr-x  3 root   root    60 Mar 28 16:55 ..
 0 -rw-r--r--  1 root   root     0 Mar 28 14:28 fichier_original_dans_voldata
root@boot2docker:~#
```

### 4. Ajout au volume

On modifie encore une fois le contenu du volume en y ajoutant un nouveau fichier pour simuler un travail d'ajout.

```
root@boot2docker:~# docker run -i -t \
    --volumes-from conteneurdata \
    --name conteneur07 \
    training/postgres \
    touch /voldata/fichier_d-ajout_dans_voldata
root@boot2docker:~#
```

On vérifie le contenu du volume.

```
root@boot2docker:~# docker run -i -t \
    --volumes-from conteneurdata \
    --name conteneur08 \
    training/postgres \
    ls -als /voldata/

total 8
4 drwxr-xr-x  2 root  root 4096 Mar 28 17:09 .
4 drwxr-xr-x 61 root  root 4096 Mar 28 17:14 ..
0 -rw-r--r--  1 root  root   0 Mar 28 17:09 fichier_d-ajout_dans_voldata
0 -rw-r--r--  1 root  root   0 Mar 28 14:28 fichier_original_dans_voldata
root@boot2docker:~#
```

Le fichier d'ajout a bien été créé.

## 5. Restauration d'un volume à son emplacement original

Nous nous apercevons que la dernière modification est une erreur. Nous devons restaurer notre sauvegarde.

Nous allons créer un **script bash** qui roulera à l'intérieur d'un nouveau conteneur qu'on créera.

Ce script:

- effacera le contenu du volume
- affichera le volume qui doit maintenant être vide
- fera la restauration de la sauvegarde
- enfin, il affichera le contenu restauré du volume.

On crée le script.



```
cat > /root/restaure.sh << FIN
#!/bin/bash

# Nous effaçons complètement le contenu du répertoire /voldata
/bin/rm -rf /voldata/*

# On vérifie.
/bin/ls -als /voldata

# Le fichier de sauvegarde est: /sauvegarde/sauvegarde.tar
# On l'extrait dans / car les fichiers dans le tar contiennent leur chemin
/bin/tar -xvf /sauvegarde/sauvegarde.tar -C /

# On affiche le répertoire restauré
/bin/ls -als /voldata/

FIN
```

On rend le script exécutable par le propriétaire du script seulement.

```
root@boot2docker:~# chmod u+x restaure.sh
root@boot2docker:~#
```

On vérifie les droits et privilèges du script.

```
root@boot2docker:~# ls -ls restaure.sh
    4 -rwxr--r--  1 root    root          343 Mar 28 19:25 restaure.sh
root@boot2docker:~#
```

On vérifie le contenu du script.



La première ligne du script n'est pas vide. Ici la ligne vide sert seulement à faciliter la copie de la commande **cat restaure.sh** depuis le fichier PDF vers l'invite du serveur. La première ligne d'un script bash doit toujours débuter par #!/bin/bash.



```
root@boot2docker:~# cat restaure.sh
#!/bin/bash

# Nous effaçons complètement le contenu du répertoire /voldata
/bin/rm -rf /voldata/*

# On vérifie.
/bin/ls -als /voldata

# Le fichier de sauvegarde est: /sauvegarde/sauvegarde.tar
# On l'extrait dans / car les fichiers dans le tar contiennent leur chemin
/bin/tar -xvf /sauvegarde/sauvegarde.tar -C /

# On affiche le répertoire restauré
/bin/ls -als /voldata/

root@boot2docker:~#
```

Nous allons monter le répertoire courant de l'hôte dans le répertoire **/sauvegarde** du conteneur. De cette façon, le shell du conteneur aura accès au script **restaure.sh** et au fichier de sauvegarde **sauvegarde.tar**.

Nous allons utiliser les arguments **-i** et **-t** de même que **--rm** qui lui, effacera le conteneur à la sortie.



```
root@boot2docker:~# docker run -i -t \
                                --rm \
                                --volumes-from conteneurdata \
                                -v $(pwd):/sauvegarde \
                                --name conteneur09 \
                                ubuntu:latest \
                                /bin/bash /sauvegarde/restaure.sh

total 8
4 drwxr-xr-x  2 root root 4096 Mar 28 19:35 .
4 drwxr-xr-x 34 root root 4096 Mar 28 19:35 ..
voldata/
voldata/fichier_original_dans_voldata
voldata/fichier_d-ajout_dans_voldata
total 8
4 drwxr-xr-x  2 root root 4096 Mar 28 19:32 .
4 drwxr-xr-x 34 root root 4096 Mar 28 19:35 ..
0 -rw-r--r--  1 root root    0 Mar 28 19:32 fichier_original_dans_voldata
root@boot2docker:~#
```

Le script **restaure.sh** a été exécuté à l'intérieur du conteneur.

On peut vérifier le contenu du volume **voldata**.



```
root@boot2docker:~# docker run -i -t \
                                --rm \
                                --volumes-from conteneurdata \
                                --name conteneur09 \
                                ubuntu:latest \
                                ls -als /voldata

total 8
4 drwxr-xr-x  2 root root 4096 Mar 28 19:32 .
4 drwxr-xr-x 33 root root 4096 Mar 28 19:44 ..
0 -rw-r--r--  1 root root    0 Mar 28 19:32 fichier_original_dans_voldata
root@boot2docker:~#
```

On peut conclure que la restauration a réussi. Le fichier ajouté par erreur est disparu.

## 6. Restauration d'un volume à un nouvel emplacement

Pour restaurer une sauvegarde dans un nouveau conteneur:

- on crée un nouveau conteneur
- on y monte le répertoire qui contient la sauvegarde en tant que volume.
- une fois à l'intérieur du nouveau conteneur on restaure la sauvegarde en lançant le script de restauration..



Il faut absolument être dans le répertoire contenant la sauvegarde avant de créer le nouveau conteneur.



```
root@boot2docker:~# docker run -i -t \
-v $(pwd) :/sauvegarde \
--name conteneur09 \
ubuntu:latest \
/bin/bash

root@35b44001840a:/#
```

On vérifie que le répertoire contenant la sauvegarde a bien été monté.

```
root@35b44001840a:/# ls -ls sauvegarde/

total 16
12 -rw-r--r-- 1 root root 10240 Mar 28 20:01 sauvegarde.tar
 4 -rwxr--r-- 1 root root   343 Mar 28 20:05 restaure.sh

root@35b44001840a:/#
```

On restaure la sauvegarde en spécifiant à l'aide de l'argument **-C** du **tar**, le répertoire parent du répertoire de destination de la restauration. Ici, on restaure dans le répertoire racine du conteneur.

```
root@35b44001840a:/# tar -xvf /sauvegarde/sauvegarde.tar -C /

voldata/
voldata/fichier_original_dans_voldata
root@35b44001840a:/#
```

On vérifie le répertoire de restauration.

```
root@35b44001840a:/# ls -alsd /voldata

4 drwxr-xr-x 2 root root 4096 Mar 28 20:00 /voldata

root@35b44001840a:/#
```

On vérifie son contenu.

```
root@35b44001840a:/# ls -ls /voldata

total 0
0 -rw-r--r-- 1 root root 0 Mar 28 19:32 fichier_original_dans_voldata

root@35b44001840a:/#
```

La restauration a fonctionné.

On sort du conteneur.

```
root@35b44001840a:/# exit

exit
root@boot2docker:~#
```

## XXII- Nettoyage du serveur

### 1. Rappel de la mise en garde

-  Pour supprimer les volumes d'un disque, vous devez lancer explicitement **docker rm -v** lors de la suppression du dernier conteneur, y compris le conteneur initial, ayant une référence au volume.
-  Lors de la suppression du dernier conteneur, l'argument **-v** permet de mettre à jour ou de migrer efficacement des volumes de données entre conteneurs.
-  Docker ne vous avertira pas lorsque vous supprimez un conteneur sans utiliser l'argument **-v**. Si vous supprimez des conteneurs sans utiliser l'argument **-v**, vous pouvez vous retrouver avec des **"dangling volumes"** qui ne sont plus référencés par un conteneur. Les **"dangling volumes"** sont difficiles à supprimer et peuvent prendre une grande quantité d'espace disque. Les développeurs de Docker travaillent à l'amélioration de la gestion de volumes et vous pouvez vérifier les progrès à ce sujet en vous référant à: [requête #8484](#).

### 2. Nettoyage

On vérifie qu'aucun conteneur ne roule, sinon on les arrête avec la commande **docker stop**.

```
root@boot2docker:~# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			

```
root@boot2docker:~#
```

Tous nos conteneurs se nommaient **conteneurdata0X** ou **conteneur0X**.

On affiche ces conteneurs.

```
root@boot2docker:~# docker ps -a | grep conteneur*
```

35b44001840a	ubuntu:14.04	"/bin/bash"	22 minutes ago	<b>conteneur09</b>
Exited (130) 7 minutes ago				
add770190172	training/postgres:latest	"ls -als /voldata/"	39 minutes ago	<b>conteneur08</b>
Exited (0) 39 minutes ago				
3690e93cf0bc	training/postgres:latest	"touch /voldata/fich	40 minutes ago	<b>conteneur07</b>
Exited (0) 40 minutes ago				
3f6f050023a0	ubuntu:14.04	"tar -cvf /backup/ba	42 minutes ago	<b>conteneur06</b>
Exited (0) 42 minutes ago				
efd5c0609f28	training/postgres:latest	"/bin/bash"	42 minutes ago	<b>conteneurdata05</b>
Exited (0) 42 minutes ago				
a7ee14970408	training/postgres:latest	"/bin/bash"	44 minutes ago	<b>conteneurdata04</b>
Exited (0) 43 minutes ago				
ae828d46e64c	training/postgres:latest	"su postgres -c '/us	45 minutes ago	<b>conteneurdata</b>

```
root@boot2docker:~#
```

## Nettoyage du serveur

---

Tous nos conteneurs qui montait un volume, utilisaient **voldata** de **conteneurdata**. Nous allons donc supprimer tous ces conteneurs sauf **conteneur09** et **conteneurdata** qu'on supprimera tous deux en dernier avec l'argument **-v** afin d'éviter les "dangling volumes".



```
root@boot2docker:~# docker rm conteneur08 conteneur07 conteneur06 conteneurdata05
conteneurdata04

conteneur08
conteneur07
conteneur06
conteneurdata05
conteneurdata04
root@boot2docker:~#
```

On supprime **conteneur09**, le dernier conteneur référençant **voldata**.

```
root@boot2docker:~# docker rm -v conteneur09

conteneur09
root@boot2docker:~#
```

On supprime **conteneurdata** le conteneur source offrant **voldata** en partage.

```
root@boot2docker:~# docker rm -v conteneurdata

conteneurdata
root@boot2docker:~#
```

On vérifie.

```
root@boot2docker:~# docker ps -a | grep conteneur*

root@boot2docker:~#
```

### 3. Prochaine étape

Dans ce chapitre, nous avons approfondi la façon d'utiliser Docker. Dans le suivant, nous allons voir comment combiner Docker avec les services disponibles sur **Docker Hub** y compris les **constructions automatiques** et les **registres privés**.

# XXIII- Docker Hub

## 1. Introduction



Ce chapitre contient une répétition de quelques sections précédentes relatives à **Docker Hub**. Elles sont regroupées ici pour permettre une vue plus approfondie de ce service essentiel de Docker.

Référence: <http://docs.docker.com/userguide/dockerhub/>.

Jusqu'ici, nous avons appris à utiliser la ligne de commande pour exécuter Docker sur notre hôte local, à faire des **docker pull** d'images pour construire des conteneurs à partir d'images existantes et comment créer nos propres images.

Dans ce chapitre, nous apprendrons comment utiliser **Docker Hub** pour simplifier et améliorer nos flux de travaux Docker.

## 2. Utilisation de Docker Hub

**Docker Hub** est un registre public tenu par **Docker Inc.** et est sa plate-forme centrale. Il contient plus de 15 000 images que vous pouvez télécharger et utiliser pour construire des conteneurs.

### 2.1. Commandes boot2docker et Docker Hub

Le système d'exploitation boot2docker fournit l'accès aux services **Docker Hub** via les commandes **docker: search, pull, login, et push**. Ce chapitre va nous en apprendre un peu plus sur le fonctionnement de ces commandes.

### 2.2. Services offerts

**Docker Hub** est une ressource centralisée pour travailler avec Docker et ses composantes; il facilite la collaboration entre collègues en tirant le meilleur parti de Docker. Pour ce faire, **Docker Hub** fournit des services tels que:

- L'hébergement d'image Docker.
- L'authentification d'un utilisateur.
- L'automatisation de la construction d'une image.
- Des outils de flux de travail tels que les déclencheurs de construction et les points Web d'accueil logiciel (*webhook*).
- L'intégration avec **GitHub** et **BitBucket**.

Pour utiliser **Docker Hub**, on doit d'abord s'inscrire afin de créer un compte; procédure simple et gratuite.

### 3. Création d'un compte Docker Hub

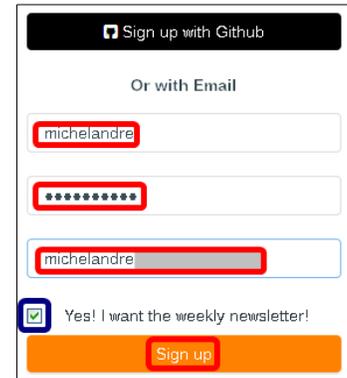
Il existe deux façons de créer un compte:

- via le Web, ou
- via la ligne de commande boot2docker.

#### 3.1. Inscription via le Web

Se rendre à l'adresse: <https://hub.docker.com/account/signup/>.

Remplir le formulaire d'inscription en choisissant un nom d'utilisateur, un mot de passe et une adresse courriel valide. On peut également s'inscrire à la liste de diffusion hebdomadaire Docker qui offre des informations intéressantes sur ce qui se passe dans le monde Docker.



#### 3.2. Confirmer son adresse courriel

Après avoir rempli le formulaire, on vérifie ses courriels et un message demandera de confirmer son adresse pour l'activation du compte.

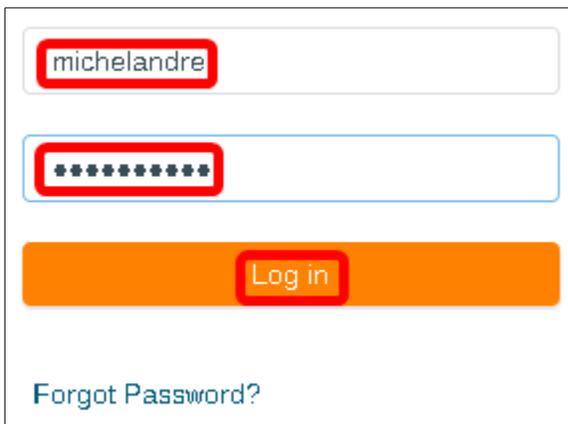
Confirm you email.



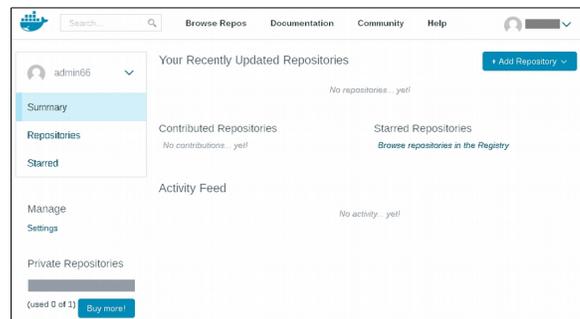
Log In.



Entrer son nom d'utilisateur et son mot de passe | Log In.



Voilà, on est logué chez Docker Hub.



### 3.3. Inscription via la ligne de commande

On peut également créer un compte **Docker Hub** via la ligne de commande en utilisant **docker login**.

```
root@boot2docker:~# docker login

Username: legrandgeneral
Password: mot-de-passe
Email: legrandgeneral@toto.com
Account created. Please use the confirmation link we sent to your e-mail to activate it.
root@boot2docker:~#
```

Comme ci-dessus au paragraphe [Inscription via le Web](#), il faut confirmer son adresse pour activer son compte

Le compte **Docker Hub** est maintenant prêt à être utilisé.

Vous pouvez maintenant faire un **docker commit** de vos images et les téléverser dans vos registres chez **Docker Hub** avec la commande **docker push**.

## 4. Login, fichier `.dockercfg` et logout

Une fois le compte confirmé, on peut se connecter chez **Docker Hub**.

```
root@boot2docker:~# docker login

Username: legrandgeneral
Password: mot-de-passe
Email: legrandgeneral@toto.com
Login Succeeded
root@boot2docker:~#
```



Vos références d'authentification seront stockées dans le fichier `.dockercfg` dans le répertoire personnel de celui qui a lancé la commande **docker login**.

```
root@boot2docker:~# ls -als ~

total 16
 0 drwxrwxr-x  2 root  staff    120 Mar 30 02:30 .
 0 drwxr-xr-x 16 root   root    380 Mar 30 01:42 ..
 4 -rw-rw-r--  1 root  staff    248 Feb 10 23:31 .Xdefaults
 4 -rw-----  1 root   root    143 Mar 30 02:34 .ash_history
 4 -rw-----  1 root   root    126 Mar 30 02:30 .dockercfg
 4 -rw-rw-r--  1 root  staff    278 Feb 10 23:31 .profile
root@boot2docker:~#
```

Ici, c'est l'utilisateur **root** qui a lancé la commande et qui s'est connecté avec le nom du Grand Général. C'est pourquoi les références du login sont stockées dans le répertoire personnel de **root**.

On affiche le fichier `.dockercfg`.

```
root@boot2docker:~# cat ~/.dockercfg

{
  "https://index.docker.io/v1/": {
    "auth": "0123456789abcdef0123456789abcdef",
    "email": "legrandgeneral@toto.com"
  }
}
root@boot2docker:~#
```

Logout.

```
root@boot2docker:~# docker logout  
  
Remove login credentials for https://index.docker.io/v1/  
root@boot2docker:~#
```

Le fichier de références est supprimé lors d'un logout.

```
root@boot2docker:~# ls -als ~  
  
total 12  
  0 drwxrwxr-x   2 root   staff   100 Mar 30 02:44 .  
  0 drwxr-xr-x  16 root   root    380 Mar 30 01:42 ..  
  4 -rw-rw-r--   1 root   staff   248 Feb 10 23:31 .Xdefaults  
  4 -rw-----   1 root   root    210 Mar 30 02:45 .ash_history  
  4 -rw-rw-r--   1 root   staff   278 Feb 10 23:31 .profile  
root@boot2docker:~#
```

## XXIV- Recherche d'images

### 1. Introduction

Vous pouvez lancer une recherche dans le registre **Docker Hub** via son interface de recherche sur la page Web ou en utilisant l'interface de la ligne de commande. La recherche peut trouver des images par le nom de l'image, le nom d'utilisateur ou la description.

```
root@boot2docker:~# docker search centos
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
<b>centos</b>	The official build of CentOS.	896	[OK]	
<b>ansible/centos7-ansible</b>	Ansible on Centos7	36	[OK]	
tutum/centos	Centos image with SSH access.	13	[OK]	
blalor/centos	Bare-bones base CentOS 6.5 image	9	[OK]	
...				

```
root@boot2docker:~#
```

On peut voir les résultats de la recherche: **centos** et **ansible/centos7-ansible**. Le second résultat montre qu'il provient d'un utilisateur, nommé **ansible/**, tandis que le premier résultat, **centos**, n'énumère pas explicitement un registre et qui signifie qu'il provient de l'espace de noms d'un haut niveau de confiance. Le caractère "/" sépare le nom du créateur de celui de l'image.

Une fois l'image trouvée, vous pouvez la télécharger avec la commande **docker pull <imagename>**.

```
docker@boot2docker:~$ docker pull centos
```

```
5b12ef8fd570: Pull complete
88f9454e60dd: Pull complete
511136ea3c5a: Already exists
centos:latest: The image you are pulling has been verified. Important: image verification is a tech preview feature and should not be relied on to provide security.
Status: Downloaded newer image for centos:latest
docker@boot2docker:~$
```

Vous avez maintenant une image à partir de laquelle vous pouvez rouler des conteneurs.

### 2. Contribuer à Docker Hub

N'importe qui peut exécuter un **docker pull** d'une image publique du registre Docker Hub.

Si vous souhaitez partager vos propres images, vous devez d'abord vous inscrire comme nous l'avons vu plus haut à la section [Création d'un compte Docker Hub](#).

### 3. Pousser une image vers Docker Hub

Afin de pousser un registre vers Docker Hub, vous devez avoir nommé une image ou exécuté un **docker commit** de votre conteneur vers une **image nommée** comme nous l'avons vu aux sections [L'importance de donner des noms](#) et [Mise à jour et commit d'une image](#).

Après avoir exécuter une de ces deux manipulations, vous pouvez faire un **push** de ce registre vers le registre désigné par son **nom** ou **étiquette**.

```
docker push votre-nom/nouvelle-image
```

# XXV- Fonctionnalités de Docker Hub

## 1. Introduction

Jetons un coup d'oeil de plus près à quelques-unes des caractéristiques de **Docker Hub**.

- Registres privés
- Organisations et équipes
- Construction automatisée
- Webhooks

Vous pouvez trouver plus d'informations [ici](#).

## 2. Registres privés

Parfois, vous avez les images que vous ne souhaitez pas rendre public ni les partager avec tout le monde. **Docker Hub** vous permet d'avoir des **registres privés**. Le premier est gratuit mais vous pouvez vous inscrire à un plan d'abonnement [ici](#).

## 3. Organisations et équipes

Un des aspects le plus utile des registres privés est que vous pouvez les partager avec tous ou seulement avec les membres de votre organisation ou avec une équipe particulière. **Docker Hub** vous permet de créer des organisations où vous pouvez collaborer avec vos collègues et gérer les registres privés. Vous pouvez apprendre à créer et gérer une organisation [ici](#), (*on demandera de vous loguer*).

## 4. Construction automatisée

La construction automatisée permet la construction et la mise à jour des images de **GitHub** ou **BitBucket** directement chez **Docker Hub**. Elle fonctionne en ajoutant **un point d'accueil logiciel** (*hook*) au registre GitHub ou BitBucket que vous avez choisi et qui déclenche une construction et une mise à jour lorsque vous faites un **push** d'un nouveau **commit**.

### 4.1. Configuration de la construction automatisé

- 1) Créer un compte [Docker Hub](#) et connectez-vous.
- 2) Lier votre compte GitHub ou BitBucket avec le menu "[Link Accounts](#)".
- 3) [Configurer une construction automatique](#).
- 4) Choisir un projet GitHub ou BitBucket qui possède un **fichier Dockerfile** qu'on veut utiliser pour la construction.
- 5) Choisir la branche qu'on veut construire (*la valeur par défaut est la **branche master***).
- 6) Donner un nom à la construction automatisée.
- 7) Attribuer facultativement une étiquette Docker à la construction.
- 8) Indiquer où le **fichier Dockerfile** est localisé. La valeur par défaut est `"/`.

Une fois la construction automatisée configurée, elle va déclencher automatiquement un **build** et en quelques minutes vous devriez voir votre nouvelle construction dans le **Registre Docker Hub**. Elle restera en synchronisation avec votre registre GitHub ou BitBucket jusqu'à ce que vous la désactiviez.

Si vous voulez voir l'état de votre construction, allez à votre page de construction automatisée chez **Docker Hub**; vous y verrez son état de même que son historique. Après avoir créé une construction automatique, vous pouvez la désactiver ou le supprimer mais vous ne pourrez plus y faire un **push**. Toutefois, vous pourrez en faire la gestion en téléversant un nouveau **commit** chez GitHub ou BitBucket.

Vous pouvez créer plusieurs constructions automatisées dans le même registre et les configurer pour pointer vers un fichier Dockerfile spécifique ou des branches **Git**.

Pour en savoir plus, voir: [Automated Builds on Docker Hub](#).

### 5. Déclencheur de construction

Une construction automatisée peut aussi être déclenchée via un **URL** sur **Docker Hub**. Ceci permet de reconstruire, à la demande, une image d'une construction automatisée.

### 6. Point Web d'accueil logiciel (*webhook*)

Les points Web d'accueil logiciel sont attachés à vos registres et vous permettent de déclencher un événement quand une image ou sa mise à jour est poussée dans le registre. Avec un point Web d'accueil logiciel, vous pouvez spécifier un URL cible et une charge de données **JSON** qui sera livrée lorsque l'image est poussé.

Pour en savoir plus, voir: [Webhooks](#).

### 7. Prochaine étape

**Docker Compose** qui permet de définir les composantes d'une application - leurs conteneurs, configuration, liens et volumes - dans un seul fichier. Puis une seule commande mettra tout en place et démarrera cette application.

# XXVI- Docker Compose

## 1. Remarque importante



Bien que **boot2docker** ne roule que sur des machine **64 bits**, il est par contre lui-même une application compilée en **32 bits**. Pour cette raison, si on utilise une machine sans aucun autre système d'exploitation que **boot2docker**, on ne pourra utiliser **Compose** qui lui, est compilé pour s'exécuter sur un système **64 bits**.

Pour cette raison, on a décidé d'utiliser un serveur **SME-9/64** pour notre hôte et sur lequel on installe **Docker**. Ainsi **Compose** pourra exécuter ses commandes **64 bits** sans problème. On peut aussi utiliser toute autre distribution **Linux/64** pour y installer **Docker**.

### 1.1. Installation de VirtualBox

VirtualBox sous Windows-7: [http://www.micronator.org/?page\\_id=1318](http://www.micronator.org/?page_id=1318).

### 1.2. Installation de SME-9/64

Installation de SME-9 sous VirtualBox: [http://www.micronator.org/?page\\_id=1327](http://www.micronator.org/?page_id=1327).

### 1.3. Installation de Docker sur un SME-9/64

Voir le chapitre II dans le document: [http://www.micronator.org/?page\\_id=1837](http://www.micronator.org/?page_id=1837).

### 1.4. Forum de discussion

Pour en savoir plus sur la compilation **boot2docker/64**:

<https://forums.docker.com/t/32-bits-64-bits-and-tiny-core-linux-6-0-as-a-base-for-boot2docker-iso/722>.

## 2. Introduction

Référence: <http://docs.docker.com/compose/>.

**Compose** est un outil pour définir et exécuter des applications complexes sous **Docker**. Avec **Compose**, vous définissez une application multi-conteneurs dans un seul fichier puis, vous lancez votre application à l'aide d'une seule commande qui assemble tout ce qui est nécessaire pour la faire fonctionner.

**Compose** est idéal pour les environnements de développement, les serveurs de tests et l'amélioration continue de ceux-ci. Présentement, **Compose** n'est pas recommandé dans un environnement de production.

### 3. Utilisation

L'utilisation de **Compose** est fondamentalement un processus comprenant trois étapes.

1) Tout d'abord vous définissez l'environnement de votre application avec un fichier **Dockerfile** de sorte qu'elle puissent être reproduites ailleurs.

Exemple:

```
FROM python:2.7
WORKDIR /code
ADD requirements.txt /code/
RUN pip install -r requirements.txt
ADD . /code
CMD python app.py
```

2) Vous définissez les services qui composent votre application dans un fichier **docker-compose.yml** afin qu'ils aient la capacité de fonctionner tous ensembles dans un environnement isolé.

Exemple:

```
web:
  build: .
  links:
    - db
  ports:
    - "8000:8000"
db:
  image: postgres
```

3) Enfin, vous exécutez **docker compose up** et **Compose** va démarrer et gérer l'ensemble de votre application.

#### 3.1. Commandes de Compose

**Compose** comprend une gamme complète de commandes pour la gestion intégrale du cycle de vie de votre application.

- Démarrer, arrêter et reconstruire les services.
- Visualiser l'état des services en cours d'exécution.
- Afficher la sortie du journal des services en cours d'exécution.
- Exécuter une commande ponctuelle sur un service.

### 4. Documentation

[Installation de Compose](#)

[Référence des commandes à la ligne](#)

[Référence pour le fichier Yaml](#)

[Variables d'environnement de Compose](#)

### 5. Installation de Compose

Commençons avec une procédure afin d'obtenir une application Web élémentaire en **Python** roulant sous **Compose**. On présume une connaissance modeste de **Python** mais les concepts démontrés ci-dessous sont compréhensibles même si vous n'êtes pas familier avec **Python**.

### 5.1. Compose

Référence: <http://docs.docker.com/compose/install/>.



Nous utilisons le serveur **SME-9/64** pour la démonstration de **Compose**.

Vu que nous avons déjà installé **Docker** sur le serveur **SME-9/64** dans un [précédent document](#), il ne reste qu'à procéder à l'installation de **Compose** avec la commande **curl**.



```
[root@sme9-docker ~]# curl -L
https://github.com/docker/compose/releases/download/1.1.0/docker-compose-`uname -s`-`uname
-m` > /usr/local/bin/docker-compose

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 5134k  100 5134k    0     0  425k      0  0:00:12  0:00:12  --:--:--  627k
[root@sme9-docker ~]#
```

Vérification.

```
[root@sme9-docker ~]# ls -als /usr/local/bin/docker-compose

5136 -rw-r--r-- 1 root root 5257430 31 mars  14:16 /usr/local/bin/docker-compose
[root@sme9-docker ~]#
```

Rendre le fichier exécutable.

```
[root@sme9-docker ~]# chmod +x /usr/local/bin/docker-compose

[root@sme9-docker ~]#
```

Vérification.

```
[root@sme9-docker ~]# ls -als /usr/local/bin/docker-compose

5136 -rwxr-xr-x 1 root root 5257430 31 mars  14:16 /usr/local/bin/docker-compose
[root@sme9-docker ~]#
```

Aucune mesure supplémentaire n'est nécessaire; **Compose** devrait maintenant être installé avec succès.



**Compose** est disponible pour **OS X** et **Linux/64**. Si vous êtes sur une autre plate-forme, **Compose** peut également être installé comme un **paquetage de Python**:

```
Sudo install PIP -U-docker composer
```

#### Version de Compose

Vous pouvez tester l'installation en exécutant **docker-compose --version**.

```
[root@sme9-docker ~]# docker-compose --version

docker-compose 1.1.0
[root@sme9-docker ~]#
```



Il n'est pas nécessaire d'exécuter les commandes **signal-event post-upgrade** et **signal-event reboot** après l'installation de **Compose** sur le serveur **SME-9/64**.

## 6. Commandes docker-compose disponibles

On affiche les commande disponibles.

```
[root@sme9-docker composetest]# docker-compose -h

Fast, isolated development environments using Docker.

Usage:
  docker-compose [options] [COMMAND] [ARGS...]
  docker-compose -h|--help

Options:
  --verbose                Show more output
  --version                Print version and exit
  -f, --file FILE         Specify an alternate compose file (default: docker-compose.yml)
  -p, --project-name NAME Specify an alternate project name (default: directory name)

Commands:
  build      Build or rebuild services
  help      Get help on a command
  kill      Kill containers
  logs      View output from containers
  port      Print the public port for a port binding
  ps        List containers
  pull      Pulls service images
  rm        Remove stopped containers
  run       Run a one-off command
  scale     Set number of containers for a service
  start     Start services
  stop      Stop services
  restart   Restart services
  up        Create and start containers

[root@sme9-docker composetest]#
```

# XXVII- Démarrage rapide

## 1. Répertoire de travail

Dans quel répertoire sommes-nous?

```
[root@sme9-docker ~]# pwd  
  
/root  
[root@sme9-docker ~]#
```

On crée un répertoire pour notre projet.

```
[root@sme9-docker ~]# mkdir composetest  
[root@sme9-docker ~]#
```

On se rend dans le répertoire.

```
[root@sme9-docker ~]# cd composetest  
[root@sme9-docker composetest]#
```

## 2. Fichier app.py

On crée un fichier **app.py**, une application Web simple qui utilise **Flask** et qui incrémente une valeur **Redis**.



```
cat > app.py << FIN  
from flask import Flask  
from redis import Redis  
import os  
app = Flask(__name__)  
redis = Redis(host='redis', port=6379)  
  
@app.route('/')  
def hello():  
    redis.incr('hits')  
    return 'Hello World! I have been seen %s times.' % redis.get('hits')  
  
if __name__ == "__main__":  
    app.run(host="0.0.0.0", debug=True)  
FIN
```

On vérifie.

```
[root@sme9-docker composetest]# cat app.py

from flask import Flask
from redis import Redis
import os
app = Flask(__name__)
redis = Redis(host='redis', port=6379)

@app.route('/')
def hello():
    redis.incr('hits')
    return 'Hello World! I have been seen %s times.' % redis.get('hits')

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True)
[root@sme9-docker composetest]#
```

### 3. Dépendances Python

On définit les dépendances **Python** dans un fichier appelé **requirements.txt**.



```
cat > requirements.txt << FIN
flask
redis
FIN
```

On vérifie.

```
[root@sme9-docker composetest]# cat requirements.txt

flask
redis
[root@sme9-docker composetest]#
```

# XXVIII- Création d'une image Docker

## 1. Définition de l'environnement avec le fichier Dockerfile

Tout d'abord, on définit l'environnement de notre application avec un fichier **Dockerfile** de sorte qu'elle puissent être reproduite ailleurs.

```
cat > Dockerfile << FIN
FROM python:2.7
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
FIN
```

On vérifie.

```
[root@sme9-docker composetest]# cat Dockerfile

FROM python:2.7
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
[root@sme9-docker composetest]#
```

Ce fichier indique à **Docker** d'inclure **Python**, notre code et nos dépendances **Python** dans une image **Docker**.

Pour plus d'informations sur la façon d'écrire un fichier **Dockerfile**, voir [Docker user guide](#) et [Dockerfile reference](#).

## 2. Définition des services avec le fichier docker-compose.yml

On définit les services qui composent notre application dans un fichier **docker-compose.yml** afin qu'ils aient la capacité de fonctionner ensembles dans un environnement isolé.

```
cat > docker-compose.yml << FIN
web:
  build: .
  command: python app.py
  ports:
    - "5000:5000"
  volumes:
    - ./code
  links:
    - redis
redis:
  image: redis
FIN
```

On vérifie.

```
[root@sme9-docker composetest]# cat docker-compose.yml
web:
  build: .
  command: python app.py
  ports:
    - "5000:5000"
  volumes:
    - ./code
  links:
    - redis
redis:
  image: redis
[root@sme9-docker composetest]#
```

Dans ce fichier, on définit deux services:

- **web**, un service construit à partir du fichier **Dockerfile** dans le répertoire courant. Il spécifie aussi d'exécuter la commande **python app.py** à l'intérieur de l'image, de rediriger le **port 5000** exposé par le conteneur vers le **port 5000 de la machine hôte**, de connecter le service **redis** et finalement de monter le répertoire courant à l'intérieur du conteneur afin que nous puissions travailler sur notre code sans avoir à reconstruire l'image.
- **redis**, un service qui utilise l'image publique [redis](#) qui est obtenue par un **pull** du registre **Docker Hub**.

### 3. Construction et exécution de l'application à l'aide de Compose

Lorsque qu'on exécutera la commande **run docker-compose up**, **Compose** va faire un **pull** d'une image **Redis**, construire une image pour notre code et démarrer le tout.



Il faut absolument être dans le répertoire du projet qu'on veut construire. Certaines variables utilisées par **Compose** pointent sur le répertoire courant.

```
[root@sme9-docker composetest]# docker-compose up
Creating composetest_redis_1...
Pulling image redis:latest...
redis:latest: The image you are pulling has been verified
511136ea3c5a: Pull complete
e977d53b9210: Pull complete
c9fa20ecce88: Pull complete
d3b4674dd536: Pull complete
8615bfb42a84: Pull complete
d1fae67f7a3f: Pull complete
51b576738805: Pull complete
748a0c3bc585: Pull complete
b92fb9da44c7: Pull complete
e050e4489700: Pull complete
815faf24012e: Pull complete
1e75af93ef31: Pull complete
d52169358d63: Pull complete
e0ae4a4caac8: Pull complete
87083227c8ed: Pull complete
e1935d9fea6d: Pull complete
2d23d535ec5e: Pull complete
e03cdbcf651a: Pull complete
Status: Downloaded newer image for redis:latest
Creating composetest_web_1...
Building web...
Step 0 : FROM python:2.7
python:2.7: The image you are pulling has been verified
3b3a4796eef1: Pull complete
50ec2d202fe8: Pull complete
1073b544a1cb: Pull complete
```



```

redis_1 | http://redis.io
redis_1 |
redis_1 | [1] 01 Apr 19:05:05.222 # Server started, Redis version 2.8.19
redis_1 | [1] 01 Apr 19:05:05.222 # WARNING overcommit_memory is set to 0! Background save
may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to
/etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this
to take effect.
redis_1 | [1] 01 Apr 19:05:05.222 # WARNING you have Transparent Huge Pages (THP) support
enabled in your kernel. This will create latency and memory usage issues with Redis. To fix
this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as
root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis
must be restarted after THP is disabled.
redis_1 | [1] 01 Apr 19:05:05.223 # WARNING: The TCP backlog setting of 511 cannot be
enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
redis_1 | [1] 01 Apr 19:05:05.223 * The server is now ready to accept connections on port
6379
web_1 | * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
web_1 | * Restarting with stat
    
```

L'application **web** devrait maintenant être à l'écoute sur le **port 5000** de l'hôte **Docker** (*ipconfig* vous donnera son adresse).

L'adresse de notre hôte est **192.168.1.193**. On se rend donc à l'adresse **192.168.1.193:5000**. L'application roule bien. On rafraîchit quelques fois pour voir le compteur s'accroître.



À la console de l'hôte, on peut voir l'effet de notre visite et du rafraîchissement de la page Web.

```

...
web_1 | * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
web_1 | * Restarting with stat
web_1 | 192.168.1.129 - - [01/Apr/2015 19:33:07] "GET / HTTP/1.1" 200 -
web_1 | 192.168.1.129 - - [01/Apr/2015 19:33:11] "GET / HTTP/1.1" 200 -
web_1 | 192.168.1.129 - - [01/Apr/2015 19:33:12] "GET / HTTP/1.1" 200 -
web_1 | 192.168.1.129 - - [01/Apr/2015 19:33:14] "GET / HTTP/1.1" 200 -
web_1 | 192.168.1.129 - - [01/Apr/2015 19:34:35] "GET / HTTP/1.1" 200 -
...
    
```

### 3.1. Arrêt des applications

On arrête l'application avec **[CTL] + [c]**.

```

^CGracefully stopping... (press Ctrl+C again to force)
Stopping composetest_web_1...
Stopping composetest_redis_1...

Aborting.
[root@sme9-docker composetest]#
    
```

## Création d'une image Docker



Si nous ouvrons une autre console vers l'hôte et si nous nous rendons dans le répertoire du projet en cours, on pourra utiliser la commande **docker-compose stop** pour arrêter l'application.

Après l'arrêt, l'invite de l'hôte redevient disponible.

On vérifie si tout est bien arrêté.

```
[root@sme9-docker composetest]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			

```
[root@sme9-docker composetest]#
```

### 3.2. Images créées

Compose a créé trois images.

```
[root@sme9-docker composetest]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
composetest_web	latest	b8c17cefc9e5	8 minutes ago	753.4 MB
python	2.7	38960da60ac6	40 hours ago	746.4 MB
redis	latest	afb26bcb6434	45 hours ago	110.9 MB

```
[root@sme9-docker composetest]#
```

### 3.3. Conteneurs créés

```
[root@sme9-docker composetest]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
bbef61017906	composetest_web:latest	"python app.py"	9 minutes ago
Exited (0) 4 minutes ago		composetest_web_1	
1ce9471628a0	redis:latest	"/entrypoint.sh redi	34 minutes ago
Exited (0) 4 minutes ago		composetest_redis_1	

```
[root@sme9-docker composetest]#
```

## 4. Mode daemon

Si on souhaite exécuter nos services en **arrière-plan**, on peut utiliser l'argument **-d** (pour le mode *daemon*) à la suite de la commande **docker-compose** et ainsi on pourra utiliser **docker-compose ps** à la console de l'hôte pour voir ce qui est actuellement en cours d'exécution.

```
[root@sme9-docker composetest]# docker-compose up -d
```

```
Recreating composetest_redis_1...
Recreating composetest_web_1...
[root@sme9-docker composetest]#
```

On affiche les conteneurs qui roulent avec la commande **docker-compose**.

```
[root@sme9-docker composetest]# docker-compose ps
```

Name	Command	State	Ports
composetest_redis_1	/entrypoint.sh redis-server	Up	6379/tcp
composetest_web_1	python app.py	Up	0.0.0.0:5000->5000/tcp

```
[root@sme9-docker composetest]#
```

On affiche les conteneurs qui roulent mais cette fois, avec la commande **docker**.

```
[root@sme9-docker composetest]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	
<b>f903c8ca97a8</b>	composetest_web:latest	"python app.py"	36 seconds ago	Up
34 seconds	0.0.0.0:5000->5000/tcp	<b>composetest_web_1</b>		
<b>226df640ef86</b>	redis:latest	"/entrypoint.sh redi	46 seconds ago	Up
44 seconds	6379/tcp	<b>composetest_redis_1</b>		

```
[root@sme9-docker composetest]#
```

On voit que **Compose** a bien recréer les conteneurs. Ils ont toujours le même **nom** mais leurs **IDentifiants** ont changés.

Il n'y a pas eu d'autres images de créer.

```
[root@sme9-docker composetest]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
<b>composetest_web</b>	<b>latest</b>	b8c17cefc9e5	12 minutes ago	753.4 MB
<b>python</b>	<b>2.7</b>	38960da60ac6	40 hours ago	746.4 MB
<b>redis</b>	<b>latest</b>	afb26bcb6434	45 hours ago	110.9 MB

```
[root@sme9-docker composetest]#
```

On affiche le dernier conteneur lancé. On voit que c'est **composetest\_web\_1**. Ceci implique que le conteneur **composetest\_redis\_1** a été le premier à être lancé.

```
[root@sme9-docker composetest]# docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
<b>f903c8ca97a8</b>	composetest_web:latest	"python app.py"	2 minutes ago	Up 2
minutes	0.0.0.0:5000->5000/tcp	<b>composetest_web_1</b>		

```
[root@sme9-docker composetest]#
```

On revisite notre page Web.



Il est à remarquer que le compteur n'a pas redémarré mais qu'il a continué où il était rendu.



## 5. Variables d'environnement

La commande **docker-compose run** permet d'exécuter certaines commandes pour vos services.

Pour voir quelles variables d'environnement sont disponibles pour le service **web**, utiliser la commande ci-dessous.

```
[root@sme9-docker composetest]# docker-compose run web env
```

```
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=d032721a6745
TERM=xterm
REDIS_PORT=tcp://172.17.0.6:6379
REDIS_PORT_6379_TCP=tcp://172.17.0.6:6379
REDIS_PORT_6379_TCP_ADDR=172.17.0.6
REDIS_PORT_6379_TCP_PORT=6379
REDIS_PORT_6379_TCP_PROTO=tcp
REDIS_NAME=/composetest_web_run_1/redis
```

```
REDIS_ENV_REDIS_VERSION=2.8.19
REDIS_ENV_REDIS_DOWNLOAD_URL=http://download.redis.io/releases/redis-2.8.19.tar.gz
REDIS_ENV_REDIS_DOWNLOAD_SHA1=3e362f4770ac2fdbdce58a5aa951c1967e0facc8
REDIS_1_PORT=tcp://172.17.0.6:6379
REDIS_1_PORT_6379_TCP=tcp://172.17.0.6:6379
REDIS_1_PORT_6379_TCP_ADDR=172.17.0.6
REDIS_1_PORT_6379_TCP_PORT=6379
REDIS_1_PORT_6379_TCP_PROTO=tcp
REDIS_1_NAME=/composetest_web_run_1/redis_1
REDIS_1_ENV_REDIS_VERSION=2.8.19
REDIS_1_ENV_REDIS_DOWNLOAD_URL=http://download.redis.io/releases/redis-2.8.19.tar.gz
REDIS_1_ENV_REDIS_DOWNLOAD_SHA1=3e362f4770ac2fdbdce58a5aa951c1967e0facc8
WEB_PORT=tcp://172.17.0.8:5000
WEB_PORT_5000_TCP=tcp://172.17.0.8:5000
WEB_PORT_5000_TCP_ADDR=172.17.0.8
WEB_PORT_5000_TCP_PORT=5000
WEB_PORT_5000_TCP_PROTO=tcp
WEB_NAME=/composetest_web_run_1/web
WEB_ENV_LANG=C.UTF-8
WEB_ENV_PYTHON_VERSION=2.7.9
WEB_1_PORT=tcp://172.17.0.8:5000
WEB_1_PORT_5000_TCP=tcp://172.17.0.8:5000
WEB_1_PORT_5000_TCP_ADDR=172.17.0.8
WEB_1_PORT_5000_TCP_PORT=5000
WEB_1_PORT_5000_TCP_PROTO=tcp
WEB_1_NAME=/composetest_web_run_1/web_1
WEB_1_ENV_LANG=C.UTF-8
WEB_1_ENV_PYTHON_VERSION=2.7.9
COMPOSETEST_REDIS_1_PORT=tcp://172.17.0.6:6379
COMPOSETEST_REDIS_1_PORT_6379_TCP=tcp://172.17.0.6:6379
COMPOSETEST_REDIS_1_PORT_6379_TCP_ADDR=172.17.0.6
COMPOSETEST_REDIS_1_PORT_6379_TCP_PORT=6379
COMPOSETEST_REDIS_1_PORT_6379_TCP_PROTO=tcp
COMPOSETEST_REDIS_1_NAME=/composetest_web_run_1/composetest_redis_1
COMPOSETEST_REDIS_1_ENV_REDIS_VERSION=2.8.19
COMPOSETEST_REDIS_1_ENV_REDIS_DOWNLOAD_URL=http://download.redis.io/releases/redis-2.8.19.tar.gz
COMPOSETEST_REDIS_1_ENV_REDIS_DOWNLOAD_SHA1=3e362f4770ac2fdbdce58a5aa951c1967e0facc8
COMPOSETEST_WEB_1_PORT=tcp://172.17.0.8:5000
COMPOSETEST_WEB_1_PORT_5000_TCP=tcp://172.17.0.8:5000
COMPOSETEST_WEB_1_PORT_5000_TCP_ADDR=172.17.0.8
COMPOSETEST_WEB_1_PORT_5000_TCP_PORT=5000
COMPOSETEST_WEB_1_PORT_5000_TCP_PROTO=tcp
COMPOSETEST_WEB_1_NAME=/composetest_web_run_1/composetest_web_1
COMPOSETEST_WEB_1_ENV_LANG=C.UTF-8
COMPOSETEST_WEB_1_ENV_PYTHON_VERSION=2.7.9
LANG=C.UTF-8
PYTHON_VERSION=2.7.9
HOME=/root
[root@sme9-docker composetest]#
```

## 6. Arrêt de Compose

Si on a démarré **Compose** avec l'argument **-d**, on peut arrêter nos services lorsqu'on en a terminé en utilisant la commande ci-dessous.



Il faut être dans le répertoire du projet pour exécuter la commande **docker-compose stop**.

```
[root@sme9-docker composetest]# docker-compose stop
Stopping composetest_web_1...
Stopping composetest_redis_1...
[root@sme9-docker composetest]#
```

On affiche les conteneurs **qui ont roulé**.

```
[root@sme9-docker composetest]# docker-compose ps
```

Name	Command	State	Ports
-----	-----	-----	-----
composetest_redis_1	/entrypoint.sh redis-server	Exit 0	
composetest_web_1	python app.py	Exit 0	

```
[root@sme9-docker composetest]#
```

La commande **docker ps** affiche les conteneurs **qui roulent présentement**. Ils sont tous arrêtés.

```
[root@sme9-docker composetest]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
-----	-----	-----	-----	-----
PORTS	NAMES			

```
[root@sme9-docker composetest]#
```

On affiche le dernier conteneur qui a roulé.

```
[root@sme9-docker composetest]# docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
-----	-----	-----	-----	-----
3b5e3b229568	composetest_web:latest	"env"	About a minute ago	Exited
(0) About a minute ago		composetest_web_run_1		

```
[root@sme9-docker composetest]#
```

On peut voir que la commande **docker-compose run web env** a créé un conteneur au nom de **env**.

```
[root@sme9-docker composetest]# docker ps -a
```

3b5e3b229568	composetest_web:latest	"env"	About a minute ago	Exited
(0) About a minute ago		composetest_web_run_1		

```
[root@sme9-docker composetest]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
-----	-----	-----	-----	-----
STATUS	PORTS	NAMES		
3b5e3b229568	composetest_web:latest	"env"	2 minutes ago	
Exited (0) 2 minutes ago		composetest_web_run_1		
f903c8ca97a8	composetest_web:latest	"python app.py"	5 minutes ago	
Exited (0) About a minute ago		composetest_web_1		
226df640ef86	redis:latest	"/entrypoint.sh redi	5 minutes ago	
Exited (0) About a minute ago		composetest_redis_1		

```
[root@sme9-docker composetest]#
```

# XXIX- Compose et Wordpress

## 1. Introduction

Référence: <http://docs.docker.com/compose/wordpress/>.

Vous pouvez utiliser **Compose** pour exécuter **Wordpress** dans un environnement isolé construit avec des conteneurs **Docker**.

## 2. Téléchargement de Wordpress

On se rend tout de même dans le répertoire personnel de root afin de ne pas mélanger les applications.

```
[root@sme9-docker composetest]# cd
[root@sme9-docker ~]#
```

On vérifie.

```
[root@sme9-docker ~]# pwd
/root
[root@sme9-docker ~]#
```



Il n'est pas nécessaire de créer un répertoire pour le projet **Wordpress** car le **tar** va en créer un.

On télécharge **Wordpress** à l'aide de **curl**.

```
[root@sme9-docker ~]# curl https://wordpress.org/latest.tar.gz | tar -xvzf -

% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
  0     0     0     0     0     0      0     0  --:--:--  --:--:--  --:--:--    0wordpress/
wordpress/wp-settings.php
wordpress/wp-cron.php
wordpress/wp-comments-post.php
wordpress/wp-activate.php
...
wordpress/wp-includes/Text/Diff.php
100 6041k  100 6041k    0     0  518k      0  0:00:11  0:00:11  --:--:--  601k
wordpress/wp-includes/update.php
wordpress/wp-includes/comment.php
wordpress/wp-config-sample.php
[root@sme9-docker ~]#
```

On vérifie que le **tar** a bien créé un répertoire **wordpress**.

```
[root@sme9-docker ~]# ls -alsd wordpress/
4 drwxr-xr-x 5 nobody nfsnobody 4096 18 févr. 18:10 wordpress/
[root@sme9-docker ~]#
```

On se rend dans le répertoire **wordpress**.

```
[root@sme9-docker ~]# cd wordpress/  
[root@sme9-docker wordpress]#
```

### 3. Fichier Dockerfile

Dans ce répertoire, on crée un fichier **Dockerfile** qui définit l'environnement dans lequel notre application va fonctionner et ce que contiendra le conteneur: **PHP** et notre **code** dans le répertoire courant i.e. **WordPress**.



```
cat > Dockerfile << FIN  
FROM orchardup/php5  
ADD . /code  
FIN
```

On vérifie.

```
[root@sme9-docker wordpress]# cat Dockerfile  
  
FROM orchardup/php5  
ADD . /code  
[root@sme9-docker wordpress]#
```

### 4. Fichier docker-compose.yml

On crée un fichier **docker-compose.yml** qui va démarrer nos services: un service **web** et un service **db** qui est une instance de **MySQL** qui roulera dans un autre conteneur.



```
cat > docker-compose.yml << FIN  
web:  
  build: .  
  command: php -S 0.0.0.0:8000 -t /code  
  ports:  
    - "8000:8000"  
  links:  
    - db  
  volumes:  
    - ./code  
db:  
  image: orchardup/mysql  
  environment:  
    MYSQL_DATABASE: wordpress  
FIN
```

On vérifie.

```
[root@sme9-docker wordpress]# cat docker-compose.yml
web:
  build: .
  command: php -S 0.0.0.0:8000 -t /code
  ports:
    - "8000:8000"
  links:
    - db
  volumes:
    - ./code
db:
  image: orchardup/mysql
  environment:
    MYSQL_DATABASE: wordpress
[root@sme9-docker wordpress]#
```

## 5. Fichier de configuration de Wordpress

Deux fichiers de support sont nécessaires pour la configuration de **WordPress**.

### 5.1. Fichier wp-config.php

C'est le fichier standard de configuration **WordPress** avec un seul changement spécifiant au conteneur **db** les données d'authentification et à quelle base de données on veut se connecter. (*On ne peut changer ni le nom de la **BD**, ni l'utilisateur ou le mot de passe. Ces paramètres ont déjà été spécifiés par le créateur de l'image orchardup/mysql de la base de données MySQL.*)

Ce fichier contient la variable **\$table\_prefix** qui commence par **\$** et fait en sorte qu'on ne peut utiliser **cat >** pour créer le fichier. Il faut utiliser l'éditeur **vi**, entrer manuellement le texte, sauvegarder et quitter.

On lance **vi**.

```
vi wp-config.php
```

On entre le texte ci-dessous, on sauvegarde et on quitte.



```
<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'root');
define('DB_PASSWORD', '');
define('DB_HOST', 'db:3306');
define('DB_CHARSET', 'utf8');
define('DB_COLLATE', '');

define('AUTH_KEY',          'put your unique phrase here');
define('SECURE_AUTH_KEY',  'put your unique phrase here');
define('LOGGED_IN_KEY',    'put your unique phrase here');
define('NONCE_KEY',        'put your unique phrase here');
define('AUTH_SALT',        'put your unique phrase here');
define('SECURE_AUTH_SALT', 'put your unique phrase here');
define('LOGGED_IN_SALT',   'put your unique phrase here');
define('NONCE_SALT',       'put your unique phrase here');

$table_prefix = 'wp_';
define('WPLANG', '');
define('WP_DEBUG', false);

if ( !defined('ABSPATH') )
    define('ABSPATH', dirname(__FILE__) . '/');

require_once(ABSPATH . 'wp-settings.php');
```

On vérifie.

```
[root@sme9-docker wordpress]# cat wp-config.php

<?php
define('DB_NAME', 'wordpress');
define('DB_USER', 'root');
define('DB_PASSWORD', '');
define('DB_HOST', "db:3306");
define('DB_CHARSET', 'utf8');
define('DB_COLLATE', '');

define('AUTH_KEY',         'put your unique phrase here');
define('SECURE_AUTH_KEY', 'put your unique phrase here');
define('LOGGED_IN_KEY',   'put your unique phrase here');
define('NONCE_KEY',       'put your unique phrase here');
define('AUTH_SALT',       'put your unique phrase here');
define('SECURE_AUTH_SALT', 'put your unique phrase here');
define('LOGGED_IN_SALT',  'put your unique phrase here');
define('NONCE_SALT',     'put your unique phrase here');

$table_prefix = 'wp_';
define('WPLANG', '');
define('WP_DEBUG', false);

if ( !defined('ABSPATH') )
    define('ABSPATH', dirname(__FILE__) . '/');

require_once(ABSPATH . 'wp-settings.php');
[root@sme9-docker wordpress]#
```

## 5.2. Fichier router.php

Le fichier **router.php** indique au serveur **web** intégré comment exécuter **Wordpress**

Ce fichier contient la variable **\$root** qui commence par **\$** et fait en sorte qu'on ne peut utiliser **cat >** pour créer le fichier. Il faut utiliser l'éditeur **vi**, entrer manuellement le texte, sauvegarder et quitter.

On lance **vi**.

```
vi router.php
```

On entre le texte ci-dessous, on sauvegarde et on quitte.



```
<?php
$root = $_SERVER['DOCUMENT_ROOT'];
chdir($root);
$path = '/' . ltrim(parse_url($_SERVER['REQUEST_URI'])['path'], '/');
set_include_path(get_include_path() . ':' . __DIR__);
if (file_exists($root . $path))
{
    if (is_dir($root . $path) && substr($path, strlen($path) - 1, 1) !== '/')
        $path = rtrim($path, '/') . '/index.php';
    if (strpos($path, '.php') === false) return false;
    else {
        chdir(dirname($root . $path));
        require_once $root . $path;
    }
}
else include_once 'index.php';
```

On vérifie.

```
[root@sme9-docker wordpress]# cat router.php

<?php

$root = $_SERVER['DOCUMENT_ROOT'];
chdir($root);
$path = '/'.ltrim(parse_url($_SERVER['REQUEST_URI'])['path'],'/');
set_include_path(get_include_path().':'.__DIR__);
if(file_exists($root.$path))
{
    if(is_dir($root.$path) && substr($path,strlen($path) - 1, 1) != '/')
        $path = rtrim($path,'/').'/index.php';
    if(strpos($path,'.php') === false) return false;
    else {
        chdir(dirname($root.$path));
        require_once $root.$path;
    }
}
else include_once 'index.php';

[root@sme9-docker wordpress]#
```

## 6. Génération du projet

Avec ces quatre fichiers en place, on va exécuter **docker-compose up** à l'intérieur du répertoire **wordpress**.

**Compose** va lancer les commandes **docker pull** nécessaires, construire les images et démarrer les conteneurs **web** et **db** qu'il va créer.

On pourra ensuite visiter notre site Web sur le **port 8000** de l'hôte i.e. **192.168.1.193:8000**.

On lance la construction des conteneurs.

```
[root@sme9-docker wordpress]# docker-compose up

Creating wordpress_db_1...
Pulling image orchardup/mysql:latest...
Pulling repository orchardup/mysql
ab3b99429ab1: Download complete
B11136ea3c5a: Download complete
2758ea31b20b: Download complete
25c55ec6c1ab: Download complete
dae3606452c3: Download complete
fa5288457082: Download complete
b9e56c8f2cf5: Download complete
db9e8e4146a4: Download complete
0b6a7c9a0a7f: Download complete
a020988ba8ca: Download complete
d89ab5cf0150: Download complete
222dbd4c9c5a: Download complete
c245bc9bf11f: Download complete
0e15b00347c9: Download complete
3c2d5a5db636: Download complete
061b756f7e0d: Download complete
Status: Downloaded newer image for orchardup/mysql:latest
Creating wordpress_web_1...
Building web...
Step 0 : FROM orchardup/php5
Pulling repository orchardup/php5
7113324d9d9e: Download complete
22 MB 0s:sssa: Download complete
e2aa6665d371: Download complete
f0ee64c4df74: Download complete
2209cbf9dcd3: Download complete
5e019ab7bf6d: Download complete
```

```
Status: Downloaded newer image for orchardup/php5:latest
---> 7113324d9d9e
Step 1 : ADD . /code
---> bd1de2d63904
Removing intermediate container 37811a7d2b7f
Successfully built bd1de2d63904
Attaching to wordpress_db_1, wordpress_web_1
...
```

## 7. Config de Wordpress

On se rend à l'adresse de l'hôte: **192.168.1.193:8000**.

On entre les informations demandées | **Install Wordpress**.

The screenshot shows the WordPress installation 'Information needed' form. The fields are filled with the following information:

- Site Title: **boot2docker & Wordpress**
- Username: **michelandre**
- Password, twice: [masked with dots]
- Your E-mail: [masked]
- Privacy:  Allow search engines to index this site.

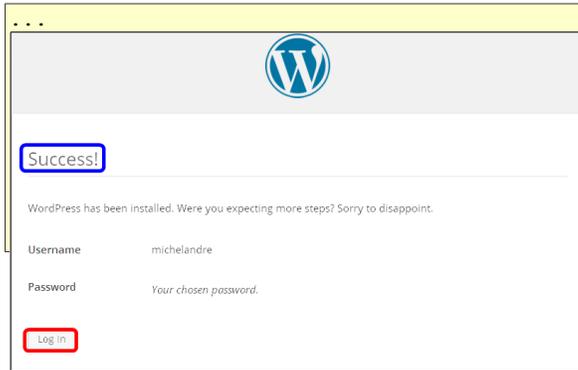
The 'Install WordPress' button is highlighted in red. To the right of the form, a terminal window shows the installation process with various file paths and status codes:

```
... could not be resolved: Name or
[200]: /
[200]: /wp-admin/install.php
[200]: /wp-
[200]: /wp-admin/css/install.min.css?
[200]: /wp-includes/js/zxcvbn-
[200]: /wp-
[200]: /wp-includes/js/jquery/jquery-
[200]: /wp-admin/js/password-strength-
[200]: /wp-
[200]: /wp-includes/js/wp-util.min.js?
[200]: /wp-admin/js/user-
[200]: /wp-admin/js/language-
[200]: /wp-admin/images/wordpress-
[200]: /wp-includes/js/zxcvbn.min.js
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1186 [200]: /wp-admin/install.php?step=2
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1187 [200]: /wp-
includes/css/buttons.min.css?ver=4.1.1
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1188 [200]: /wp-admin/css/install.min.css?
ver=4.1.1
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1189 [200]: /wp-
includes/js/jquery/jquery.js?ver=1.11.1
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1190 [200]: /wp-includes/js/jquery/jquery-
migrate.min.js?ver=1.2.1
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1191 [200]: /wp-includes/js/zxcvbn-
async.min.js?ver=1.0
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1192 [200]: /wp-admin/js/password-strength-
meter.min.js?ver=4.1.1
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1193 [200]: /wp-
includes/js/underscore.min.js?ver=1.6.0
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1194 [200]: /wp-includes/js/wp-util.min.js?
ver=4.1.1
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1197 [200]: /wp-admin/js/user-
profile.min.js?ver=4.1.1
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1198 [200]: /wp-admin/js/language-
chooser.min.js?ver=4.1.1
```

## Compose et Wordpress

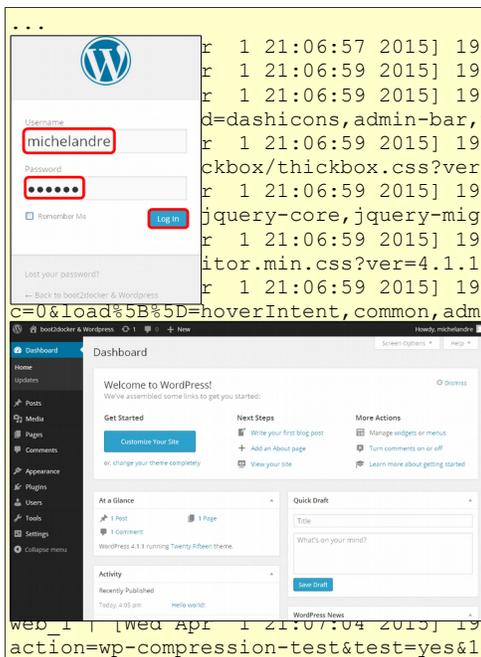
```
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1199 [200]: /wp-admin/images/wordpress-  
logo.svg?ver=20131107  
web_1 | [Wed Apr 1 21:00:51 2015] 192.168.1.129:1200 [200]: /wp-includes/js/zxcvbn.min.js
```

### Login.



```
129:1202 [200]: /wp-login.php  
129:1203 [200]: /wp-  
129:1204 [200]: /wp-  
129:1205 [200]: /wp-admin/css/login.min.css?  
129:1206 [200]: /wp-admin/images/wordpress-
```

On entre le nom d'utilisateur donné à l'écran d'installation | **Log In**. L'écran du **Tableau bord** de Wordpress apparaît.



```
1 21:06:57 2015] 192.168.1.129:1209 [302]: /wp-login.php  
1 21:06:59 2015] 192.168.1.129:1210 [200]: /wp-admin/  
1 21:06:59 2015] 192.168.1.129:1211 [200]: /wp-admin/load-styles.php?  
d=dashicons,admin-bar,wp-admin,buttons,wp-auth-check&ver=4.1.1  
1 21:06:59 2015] 192.168.1.129:1213 [200]: /wp-  
checkbox/thickbox.css?ver=4.1.1  
1 21:06:59 2015] 192.168.1.129:1215 [200]: /wp-admin/load-scripts.php?  
jquery-core,jquery-migrate,utils,json2&ver=4.1.1  
1 21:06:59 2015] 192.168.1.129:1216 [200]: /wp-  
itor.min.css?ver=4.1.1  
1 21:06:59 2015] 192.168.1.129:1217 [200]: /wp-admin/load-scripts.php?  
c=0&load%5B%5D=hoverIntent,common,admin-bar,wp-ajax-response,jquery-color,wp-  
ments,jquery-ui-core,jquery-&load%5B%5D=ui-  
ble,postbox,dashboard,underscore,customize-  
instal&load%5B%5D=1,shortcode,media-upload,svg-  
count,wplink&ver=4.1.1  
.168.1.129:1227 [200]: /wp-admin/admin-ajax.php?  
922449278  
.168.1.129:1228 [200]: /wp-  
gif  
.168.1.129:1229 [200]: /wp-admin/admin-ajax.php?  
ard_primary&pagenow=dashboard  
.168.1.129:1231 [200]: /wp-admin/admin-ajax.php?  
922449458  
web_1 | [Wed Apr 1 21:07:04 2015] 192.168.1.129:1232 [200]: /wp-admin/admin-ajax.php?  
action=wp-compression-test&test=yes&1427922453490
```

## 8. Arrêt des conteneurs

On se connecte au serveur avec une autre connexion et on se rend dans le répertoire du projet **WordPress**.

```
[root@sme9-docker ~]# cd wordpress/  
[root@sme9-docker wordpress]#
```

On arrête les conteneurs.

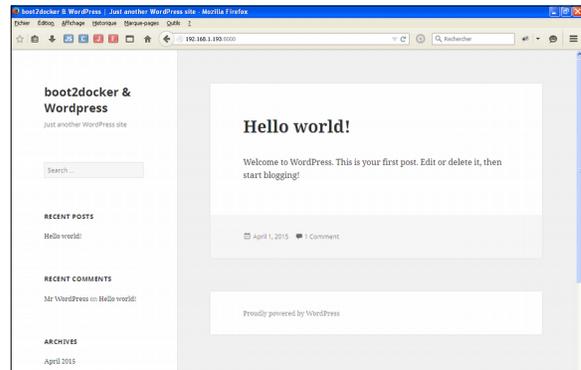
```
[root@sme9-docker wordpress]# docker-compose stop  
Stopping wordpress_web_1...  
Stopping wordpress_db_1...  
[root@sme9-docker wordpress]#
```

On redémarre le projet **WordPress**.

```
[root@sme9-docker wordpress]# docker-compose up  
Recreating wordpress_db_1...  
Recreating wordpress_web_1...  
Attaching to wordpress_db_1, wordpress_web_1  
...
```

On revisite la page **192.168.1.193:8000** et la page d'accueil s'affiche.

**Tout est fonctionnel.**



À ce stade, nous avons vu les rudiments de **Compose**.

On peut consulter les guides de démarrage rapide pour [Django](#) ou [Rails](#) et les guides de référence pour les détails complets sur les [commandes](#), le [fichier de configuration](#) et les [variables d'environnement](#).

## 9. Documentation de Compose

Pour en savoir plus sur **Compose**, consultez les documentations ci-dessous.

[Installing Compose.](#)

[User guide.](#)

[Command line reference.](#)

[Yaml file reference.](#)

[Compose environment variables.](#)

[Compose command line completion.](#)



**Victoire totale.**

## Crédits

© 2015 RF-232

Auteur: **Michel-André Robillard CLP**

Remerciement: **Tous les contributeurs GNU/GPL.**

Intégré par: **Michel-André Robillard CLP**

Contact: **michelandre at micronator.org**

Répertoire de ce document: E:\000\_DocPourRF232\_general\RF-232\_Docker\RF-232\_boot2docker\_Tutoriel\_2015-04-04\_14h07.odt

Historique des modifications:

<i>Version</i>	<i>Date</i>	<i>Commentaire</i>	<i>Auteur</i>
RC-1	2015-03-06	Début de Présentation.	M.-A. Robillard
RC-2	2015-03-16	Ajout pour --dns.	M.-A. Robillard
RC-3	2015-03-20	Changement du titre du document, Premiers pas.	M.-A. Robillard
RC-4	2015-03-21	Modifications mineures et changement du titre du document. Les conteneurs.	M.-A. Robillard
RC-5	2015-03-22	Modifications préliminaires. Les images.	M.-A. Robillard
RC-6	2015-03-23	Liaisons inter-conteneurs.	M.-A. Robillard
RC-7	2015-03-25	Assemblage de tous les chapitres de la série boot2docker dans un seul document.	M.-A. Robillard
RC-8	2015-03-25	Ajout de la gestion des volumes.	M.-A. Robillard
RC-9	2015-03-30	Correction dans docker login - ligne d'adresse de courriel en trop. Ajout pour la convention des couleurs. Correction d'erreurs typographiques. Intégration de Docker Hub.	M.-A. Robillard
RC-10	2015-04-01	Ajout de Compose.	M.-A. Robillard
0.0.1	2015-04-02	Vérification complète et corrections.	M.-A. Robillard

## Index

0		
0.0.0.0:8000.....	120	
1		
127.0.0.1.....	72	
14.04, 14.04.2, latest.....	50	
15 000 images.....	98	
172.17.0.40.....	81	
172.17.0.46.....	82	
192.168.1.1.....	23	
192.168.1.191.....	29, 45	
192.168.1.193:5000.....	114	
192.168.1.193:8000.....	123	
192.168.1.200.....	13	
2		
22.....	29	
4		
4.47ubuntu1.....	81	
49153.....	44	
49154.....	48	
49155.....	71	
49155->5000.....	71	
49158.....	78	
5		
5000.....	117	
5000 /tcp.....	74	
5000->5000/udp.....	73	
5000:5000.....	72	
5000:5000/udp.....	73	
5000/tcp.....	44	
5000/tcp ->.....	74	
5000/udp ->.....	74	
5432.....	78	
6		
6000->5000/tcp.....	74	
6000->5000/udp.....	74	
6379.....	116	
64 bits.....	9	
8		
8000:8000.....	106, 120	
8080.....	79	
A		
à la demande.....	104	
Aborting.....	114	
Account created.....	100	
ADD.....	106	
adressage réseau.....	83	
adresse courriel.....	32, 99	
adresse IP.....	12	
adresse-courriel.....	68	
Affichage à l'écran.....	34	
Afficher la sortie du journal.....	106	
alias.....	78	
ancêtre de MS-DOS.....	56	
ansible/centos7-ansible.....	102	
Apache Cassandra.....	9	
API.....	9	
app.py.....	43, 71, 83, 109	
apt-get install.....	81	
apt-get install -y.....	60	
Argument --dns.....	23	
argument -d.....	38	
Arrêt de Compose.....	117	
Arrêt du conteneur.....	47	
arrière-plan.....	41, 115	
ASCII.....	10	
ash.....	86	
astuce.....	10	
Attaching to.....	124	
Attention.....	79	
Authentification.....	31	
AUTOMATED.....	102	
Automated Builds on Docker Hub		
.....	104	
Automatisées.....	54	
Avantages.....	78	
Avertissement.....	2	
B		
base ou images root.....	54	
bash.....	28, 86	
Bash.....	36	
BitBucket.....	31, 98	
bleu.....	10	
boot2docker permanent.....	19	
boot2docker/64.....	105	
bootlocal.sh.....	27	
branche master.....	103	
Brancher les aînés.....	11	
branches Git.....	104	
build.....	104, 108	
build master : a66bce5.....	12	
build:.....	106	
C		
C:\Users.....	84	
cat /etc/hosts.....	81, 82	
cat >.....	93, 109	
cat > Dockerfile.....	60	
cd.....	14, 30	
centos.....	102	
CentOS.....	52	
centos:latest.....	52, 102	
centos7-ansible.....	102	
Certificat.....	21	
egroups.....	9	
charge de données JSON.....	104	
chemin.....	93	
chemin absolu.....	85	
chemins.....	84	
chmod +x.....	27	
chmod u+x restaure.sh.....	93	
Client Docker.....	41	
Client version.....	41	
Cloudlets.....	9	
collaboration entre collègues.....	98	
COMMAND.....	41	
commande ponctuelle.....	106	
Commandes boot2docker.....	98	
Commandes de base de docker.....	14	
Commandes de Compose.....	106	
Commandes diverses.....	46	
Commandes docker-compose.....	108	
Commandes Linux.....	16	
Commands.....	42	
Commands:.....	108	
Commentaire.....	127	
Commentaires et suggestions.....	11	
commit.....	56, 57	
Communication par liens.....	77	
Compose.....	107	
Compose et Wordpress.....	119	
composetest.....	109	
composetest_redis_1.....	113	

- composetest\_web\_1.....113  
 compte Docker Hub.....31, 99, 100  
 Conclusion de ce chapitre.....28  
 Config de Wordpress.....124  
 Configuration de la construction  
 automatisé.....103  
 configuration de Wordpress.....121  
 Confirm you email.....32, 99  
 confirmation link.....100  
 Conflict.....83  
 Connexion à distance.....12  
 Connexion par liens.....75  
 Conséquences de l'exécution.....35  
 Construction et exécution.....112  
 constructions automatiques.....97  
 containers.....23  
 conteneur client.....78  
 conteneur daemonisé.....41  
 conteneur destinataire.....78  
 Conteneur interactif.....36  
 conteneur source.....78  
 Conteneur-volume-de-données.....87  
 conteneur06.....91  
 conteneur07.....92  
 conteneur08.....92  
 conteneur09.....94, 95, 97  
 conteneur0X.....96  
 conteneurdata.....87  
 conteneurdata01.....87  
 conteneurdata02.....88  
 conteneurdata03.....88  
 conteneurdata04.....89  
 conteneurdata05.....90  
 conteneurdata0X.....96  
 Conteneurs créés.....115  
 conteneurs-volume-de-données.....83  
 contenu du script.....94  
 Contenu du volume.....89  
 Contribuer à Docker Hub.....102  
 Conventions.....10  
 CP/M.....56  
 CR.....10  
 Création d'un compte.....31, 99  
 Création d'un lien.....77  
 Création d'un volume de données.....83  
 Création d'une image.....56  
 Création du disque.....25  
 Création du fichier Dockerfile.....60  
 Création du répertoire.....59  
 Crédits.....127  
 CTRL+C to quit.....114  
 curl.....107  
 curl -L.....107
- D**  
 daemon.....41, 115  
 dangling volumes.....87, 96  
 DB\_NAME.....121  
 DB\_PASSWORD.....121  
 DB\_USER.....121  
 db:3306.....121  
 db:lien\_db.....80  
 db1.....80  
 DB1.....80  
 dd if=/dev/cdrom of=/dev/sda.....19  
 de données.....104  
 déclenchée via un URL.....104  
 Déclencheur de construction.....104  
 déclencheurs.....31, 98  
 défaut est "/".....103  
 Définition de l'environnement.....111  
 Définition des services.....111  
 Démarrage rapide.....109  
 Démarrer.....106  
 Dépendances Python.....110  
 desperate\_mcclintock.....75  
 determined\_colden.....37  
 Deuxième disque.....25  
 df -h.....20  
 diffusion du savoir.....11  
 digestIT 2004.....10  
 disque principa.....12  
 disque secondaire.....12  
 distracted\_wilson.....38  
 Django.....126  
 Docker.....9  
 docker --help.....14  
 docker attach.....42  
 docker attach --help.....42  
 docker build.....59  
 docker build -h.....61  
 docker build -t.....67  
 docker commit.....57, 100  
 Docker Compose.....104, 105  
 docker compose up.....106  
 docker create.....83  
 Docker Hub.....31, 97, 98, 112  
 docker images.....16, 22, 33, 115  
 Docker Inc.....54, 98  
 docker info.....18  
 docker inspect.....47, 78  
 docker inspect -f.....47  
 docker login.....32, 100  
 docker logout.....101  
 docker logs.....39, 41, 46  
 docker logs -f.....46  
 docker port.....73
- docker port -h.....73  
 docker ps.....33, 41, 71, 96, 115  
 docker ps -a.....22, 35  
 docker ps -l.....35, 44, 48, 71  
 docker pull.....69, 123  
 docker pull <imagename>.....102  
 docker pull centos.....52, 102  
 docker pull training/sinatra.....54  
 docker push.....67, 100  
 docker push votre-nom/nouvelle-  
 image.....102  
 docker restart.....48  
 docker restart db.....82  
 docker rm.....49, 70, 84  
 docker rm -f web.....76  
 docker rm -v.....87, 96  
 docker rm -v web.....84  
 docker rmi.....69  
 docker run.....41, 83  
 docker run -i -t.....94  
 docker run --help.....36  
 docker run -d.....85  
 docker run -t -i.....51  
 docker search.....54  
 docker search centos.....102  
 docker search sinatra.....54  
 docker start.....48  
 docker stop.....39, 41, 47, 49, 84  
 docker stop web.....84  
 Docker sur un SME-9/64.....105  
 docker tag.....67  
 docker top.....47  
 Docker user guide.....111  
 docker version.....41  
 Docker version 1.5.0.....30  
 docker-compose run web env.....116  
 docker-compose stop.....115  
 docker-compose up.....112  
 docker-compose up -d.....115  
 docker-compose.yml.....106, 111, 120  
 Dockerfile.....59, 79, 106, 111  
 Dockerfile reference.....111  
 Dockeriser des applications.....33  
 Documentation.....106  
 Documentation de Compose.....126  
 done.....38  
 données persistantes.....83  
 donner des noms.....75  
 dotCloud.....9  
 Downloaded newer image.....77
- E**  
 echo.....34

## Index

echo \$PATH.....	16	Go.....	41	JSON.....	47
elegant_davinci.....	75	Go version.....	41	L	
Email:.....	100	Gracefully stopping.....	114	L'authentification.....	98
emplacement original.....	93	Graphical Environment Manager.....	56	L'hébergement d'image.....	98
env.....	118	grep.....	67	Label.....	20
ENV.....	79	grep conteneur*.....	96	last.....	35, 44
EOF.....	28	H		legrandgeneral.....	100
erreur.....	86	hachage.....	47	legrandgeneral@toto.com.....	100
étape.....	10	Hébergement d'image.....	31	Les images.....	50
étiquette.....	102	hooks.....	31	LF.....	10
Étiquette d'une image.....	67	hook.....	103	liaisons inter-conteneurs.....	83
exit.....	37, 90, 95	Host Name for IP address.....	29	Liaisons inter-conteneurs.....	71
F		HOSTNAME=.....	116	lien_db.....	81
FATA[0000] Error.....	83	http://redis.io.....	114	LIEN_DB_ENV.....	80
FATA[0000] Error:.....	49	I		LIEN_DB_NAME.....	80
favicon.ico.....	46	IDentifiant du conteneur.....	38	LIEN_DB_PORT.....	80
fdisk /dev/sda.....	19	IDentifiants.....	116	lien_webdb.....	79
fdisk /dev/sdb.....	25	ifconfig.....	12, 29	LIEN_WEBDB_NAME=.....	79
fervent_wright.....	75	ifconfig eth0.....	45	LIEN_WEBDB_PORT=.....	79
fichier .dockercfg.....	100	image Docker.....	111	Link Accounts.....	103
fichier /etc/hosts.....	81	image nommée.....	102	links:.....	106
Fichier app.py.....	109	image verification.....	102	linux/64.....	12
fichier de références.....	101	Images & conteneurs.....	22	liste de diffusion.....	31, 99
fichier Dockerfile.....	59, 85, 103, 111	images Docker.....	83	Liste des images.....	50
Fichier Dockerfile.....	60, 120	images root.....	54	Log In.....	32, 99, 125
Fichier ISO.....	12	inetutils-ping.....	81	Logiciels recommandés.....	10
fichier source.....	86	Informatique Libre.....	11	login.....	13
Fichier toto.....	22	Infos de boot2docker.....	18	Login.....	100
fichier_d-ajout_dans_voldata.....	92	Inscription.....	100	Login à distance.....	29
fichier_original_dans_voldata.....	89, 95	Inscription via la ligne de commande.....	100	Login Succeeded.....	68
Flask.....	46, 109	Inscription via le Web.....	99	logout.....	100
flux de travail.....	31, 98	Inspection du conteneur.....	47	ls -als /mnt/sdb1/titi.....	27
flux de travaux.....	98	Install Wordpress.....	124	ls -alsd /mnt/sda2.....	20
focused_rosalind.....	75	Installation.....	12	ls /.....	36
Fonctionnalités de Docker Hub.....	103	Installation de Compose.....	106	ls /usr/bin/.....	17
forks.....	9	intermediate container.....	113	LXC.....	9
Formatage de la partition sda2.....	20	IP du serveur boot2docker.....	29	M	
formulaire d'inscription.....	31, 99	IP du serveur DNS.....	23	Machine virtuelle.....	12
Forum de discussion.....	105	IP du serveur hôte.....	45	magenta.....	10
FROM.....	60, 106	ipconfig.....	114	MAINTAINER.....	60
G		itsdangerous.....	113	Manipulation.....	10
gcc.....	113	J		mappage 1:1.....	44
GEM.....	56	JavaScript Object Notation.....	47	mappage de port.....	44
gem install json.....	56	Jinja2.....	113	mappage de ports.....	75
gemme.....	56	jolly_jones.....	37	mappage des ports.....	73
gemme json.....	57	Journaux du conteneur.....	46	Mappage du port interne.....	46
general-toto/sinatra:v1.....	61	json.....	56	MarkupSafe.....	113
Gestion des données dans les conteneurs.....	83			mc.....	28
GitHub.....	9, 31, 98			meilleures pratiques.....	59
				message.....	57

## Index

- micelandre/documentation.....67  
micelandre/documentation/.....68  
micelandre/sinatra:v2.....57  
micronator.org.....11  
Midnight Commander.....28  
migrer.....89  
Mise à jour.....56, 81  
Mise en garde importante.....87  
mkdir -p /mnt/sda2.....20  
mkdir /construction.....59  
mkdir /mnt/sdb1.....26  
mkdir /temp.....91  
mkfs.ext4 -L boot2docker-data  
/dev/sda2.....20  
mkfs.ext4 /dev/sdb1.....26  
Mode daemon.....115  
Montage de la partition sda2.....20  
Monter un répertoire hôte.....84  
mot de passe par défaut.....30  
mot de passe tcuser.....13  
mot-de-passe.....100  
mot-de-passe.....68  
mount.....85  
mount /dev/sda2 /mnt/sda2.....20  
mount /dev/sdb1 /mnt/sdb1.....26  
MySQL.....120
- N**
- name.....77, 79  
naughty\_mestorf.....46  
Nettoyage.....96  
Nettoyage du serveur.....96  
nom.....102  
nom-de-l'usager.....68  
noms différents.....81  
non vérifié.....10  
NON-RESPONSABILITÉ.....2  
note.....10  
Notepad++.....10  
Notes au lecteur.....10  
Notes importantes.....80  
nouvel emplacement.....95  
Nouvelle image.....52  
noyau Linux.....9
- O**
- Obtenir une nouvelle image.....52  
OFFICIAL.....102  
Officielles.....54  
Open.....29  
Options.....42  
Options:.....108
- orange.....10  
orchardup/mysql.....120  
Organisations et équipes.....103  
OSX.....84  
OSX/Windows.....84
- P**
- Paas.....9  
paquetage de Python.....107  
Partition primaire.....25  
partition sda2.....19  
Partition sda2.....22  
passwd.....13  
Password:.....100  
PATH=.....116  
PDF.....10, 94  
Permanence avec bootlocal.sh.....27  
Permanence de boot2docker.....19  
PHP.....120  
ping.....81  
ping -c3 lien\_db.....81  
pip install.....106  
Point de montage.....26  
points Web d'accueil logiciel.....98  
port.....79  
port 22.....13  
port 5000.....71  
port 5000 de l'hôte.....114  
port 5000 de la machine hôte.....112  
port 5000 exposé.....112  
port 6000 de l'hôte.....74  
port interne 5000.....74  
port spécifique.....72  
port SSH.....13  
port-x-intérieur.....44  
port-x-serveur.....44  
PORTS.....44  
ports:.....106  
PostgreSQL.....77, 78  
Pousser une image.....102  
prefix\_ADDR.....79  
prefix\_PORT.....79  
prefix\_PROTO.....79  
premier plan.....41  
PRETTY\_NAME.....55  
PRIVATE\_PORT.....74  
procédure.....10  
Processus du conteneur.....47  
Prochaine étape.....97, 104  
projet Stackbrew.....54  
protocole.....79  
pseudo-TTY.....36  
pull.....54, 108, 112
- Pull complete.....34, 52, 102  
pulling has been verified.....112  
PuTTY.....10, 12, 29  
pwd.....14, 30, 36, 59, 109  
python.....83  
Python.....44, 106  
python app.py.....43, 47  
Python Flask.....43, 71
- R**
- Rails.....126  
ramasse-miettes.....83  
read only.....85  
reboot.....21, 28  
Recherche d'images.....102  
recherche dans le registre.....102  
recherche sur la page Web.....102  
recommandation.....10  
reconstruire.....106  
Redémarrage du conteneur.....48  
Redis.....109  
Redis 2.8.19.....113  
Référence des commandes à la ligne  
.....106  
référence internet.....10  
Référence pour le fichier Yaml...106  
registre Docker Hub.....50  
Registre Docker Hub.....104  
registres privés.....97  
Registres privés.....103  
Remarque importante.....105  
Remove login credentials.....101  
répertoire contenant la sauvegard.95  
Répertoire de stockage des  
conteneurs.....23  
Répertoire de travail.....109  
répertoire du projet en cours.....115  
Répertoire et fichier Dockerfile...59  
répertoire personnel.....100  
repositories/micelandre.....68  
requête #8484.....87, 96  
requirements.txt.....110  
réseau interne.....71  
ressource centralisée.....98  
ressource occup.....86  
restart.....108  
Restauration.....93  
Restauration d'un volume.....95  
restaurer.....89  
RF-232.....11  
Riak.....9  
rm.....70, 108  
rmi.....69

## Index

- ro.....85  
root.....14, 30  
rouge.....10  
router.php.....122  
ruby.....60  
Ruby.....54, 60  
ruby-dev.....60  
RubyGems.....60  
run.....108  
RUN.....60, 106
- S
- S'inscrire via la ligne de commande.....32  
Sauvegarde d'un volume.....91  
Sauvegarder.....89  
Save.....29  
Saved Sessions.....29  
script bash.....93  
sdb.....25  
sdb1.....26  
sed.....86  
Server version.....41  
service redis.....112  
Services offerts.....98  
shell bash.....28, 86  
shell Bash.....36  
signal-event post-upgrade.....107  
signal-event reboot.....107  
Sinatra.....54, 60  
sleep 1;.....38  
SME-9/64.....10, 105  
SSH.....10, 13  
sshd.....80  
Stackbrew.....54  
stand alone mode.....113  
STARS.....102  
start.....108  
STDIN.....36  
STDOUT.....39, 46  
Step 1 : ADD . /code.....113  
Step 1 : MAINTAINER.....61  
Step 2 : RUN.....61  
Step 2 : WORKDIR /code.....113  
Step 3 : RUN.....65  
Step 3 : RUN pip install.....113  
stop.....108  
Stopping composetest\_web\_1...117  
Successfully built.....65, 113  
Successfully installed.....65  
sudo.....33, 41  
sudo -s.....14, 22, 30, 59  
super-usagers.....17
- Suppression d'une image.....69  
Suppression du conteneur.....49  
système de liaison.....75
- T
- Tableau bord.....125  
tag -h.....67  
tar.....119  
tar -cvf.....91  
tce-load.....17, 28  
TCP.....79  
TCP et UDP.....74  
TCP\_PORT=5000.....117  
tcuser.....13, 30  
Téléchargement (pull).....54  
Téléchargement de Wordpress...119  
Téléversement d'une image.....67  
titi.....27  
toto.....21  
touch /mnt/sda2/toto.....21  
touch /mnt/sdb1/titi.....27  
tous les ports mappés.....74  
training/postgres.....77  
training/sinatra.....54, 69  
training/sinatra:latest.....54  
training/webapp.....43, 50, 71, 83  
Transfert de l'ISO vers le disque..19  
Trouver des images.....53  
tunnel sécurisé.....78  
tutoriel Dockerfile.....70  
tutoriel interactif.....41
- U
- u+x.....93  
ubuntu.....91  
ubuntu /bin/bash.....41  
ubuntu:14.04.....33, 49, 50  
ubuntu:latest.....51, 86  
UDP.....73, 79  
Unable to find image.....34  
Une application Web.....43  
Union File System.....83  
up.....108  
Up 5 seconds.....38  
URL.....48  
Usage.....41, 42  
Usage:.....108  
usager par défaut.....30  
usager training.....70  
Username:.....100  
utilisateur docker.....13  
Utilisation.....106
- Utilisation de Docker Hub.....98
- V
- Variables d'environnement...79, 116  
Variables d'environnement de  
Compose.....106  
variables d'environnement Docker  
.....80  
Vérification.....22  
version 1.5.0.....12  
Version de Compose.....107  
vi.....121  
Via le Web.....31  
Victoire.....126  
VirtualBox.....10, 12, 105  
voldata.....87  
volumes.....23  
volumes de données.....83
- W
- webhook.....98, 104  
Webhooks.....103  
Werkzeug.....113  
whoami.....14, 30  
Windows.....84  
WinSCP.....10  
Wordpress.....119  
WORKDIR.....106  
wp-config.php.....121
- 
- env.....79  
--env-file.....79  
--help.....42  
--interactive=false.....36  
--link.....77  
--link <name or id>:alias.....77  
--link db:lien\_db.....77, 78  
--name.....75  
--name conteneur06.....91  
--name web.....75, 83  
--name web2.....80  
--rm.....76, 82, 94  
--tty=false.....36  
--volumes-from.....91  
--volumes-from conteneurdata. .91,  
94  
-a.....44, 57  
-C /.....93  
-C /temp.....92  
-cvf.....91

## Index

-d.....	38, 43, 44, 71, 117
-e.....	79
-f.....	46
-i.....	36, 86
-l.....	44
-L boot2docker-data.....	20
-m.....	57
-p.....	72, 78
-P.....	43, 44, 71, 78, 85
-p 5000.....	44
-rwxr--r--.....	93
-t.....	36, 67, 86
-v.....	83, 86
-v /webapp.....	83
-v \$(pwd):/s.....	95
-v \$(pwd):/sauvegarde.....	91
-xvzf.....	119
-yqq.....	81
:	
.ro.....	85
.	
./sedKdJ9Dy.....	86
.ash_history.....	86
.bash_history.....	86
.dockercfg.....	100
,	
'Bonjour tout le monde'.....	33
"	
"_ ".....	80
"-".....	80
"auth":.....	100
"Bonjour tout le monde".....	33
"dangling volumes.....	87
"email":.....	100
"env".....	118
"GET / HTTP/1.1" 200.....	114
[	
[/db:/web/lien_db].....	78
[arg...].....	41
[commande].....	42
[CTL] [c].....	46
[Ctrl] [D].....	37
[Entrée].....	19, 25
[OPTIONS].....	41
[STDOUT].....	39, 46
{	
{{ HostConfig.Links }}.....	78
{{ Name }}.....	75
/	
</container path>.....	84
/bin/.....	16
/bin/bash.....	36, 86
/bin/echo.....	33
/bin/ls -als /voldata.....	93
/bin/rm -rf /voldata/*.....	93
/bin/sh.....	28
/bin/sh -c.....	38
/bin/tar.....	93
/c/Users.....	84
/construction.....	59
/etc/hosts.....	78, 80
/etc/os-release.....	55
/mnt/sda2/var/lib/docker/containers/ .....	23
/mnt/sdb1.....	26
/mnt/sdb1/titi.....	27
/opt/webapp#.....	81
/root/.ash_history.....	86
/root/.bash_history.....	86
/sauvegarde/sauvegarde.tar.....	91
/sbin/.....	17
/src/webapp/opt/webapp.....	85
/Users.....	84
/Users/<path>.....	84
/voldata.....	87
/web/lien_webdb.....	79
/web2/lien_db.....	80
/webapp.....	84
#	
#.....	60
#!/bin/bash.....	94
©	
© RF-232.....	2
<	
<< FIN.....	93, 109
<alias>_ENV_<name>.....	79
<alias>_NAME.....	79
<name>_PORT_<port>_<protocol> .....	79
\$	
\$root.....	122
\$table_prefix.....	121

## LICENCE PUBLIQUE GÉNÉRALE GNU

Version 3, du 29 juin 2007.

Copyright (C) 2007 Free Software Foundation, Inc.  
<<http://fsf.org/>>

Chacun est autorisé à copier et distribuer des copies conformes de ce document de licence, mais toute modification en est proscrite.

-----  
**Traduction française** par Philippe Verdy <verdy\_p (à wanadoo) (point) fr>, le 30 juin 2007 (dernière correction du 4 janvier 2011).**Avertissement important au sujet de cette traduction française.**

Ceci est une traduction en français de la licence "GNU General Public License" (GPL). Cette traduction est fournie ici dans l'espoir qu'elle facilitera sa compréhension, mais elle ne constitue pas une traduction officielle ou approuvée d'un point de vue juridique.

La Free Software Foundation (FSF) ne publie pas cette traduction et ne l'a pas approuvée en tant que substitut valide au plan légal pour la licence authentique "GNU General Public License". Cette traduction n'a pas encore été passée en revue attentivement par un juriste et donc le traducteur ne peut garantir avec certitude qu'elle représente avec exactitude la signification légale des termes de la licence authentique "GNU General Public License" publiée en anglais. Cette traduction n'établit donc légalement aucun des termes et conditions d'utilisation d'un logiciel sous licence GNU GPL — seul le texte original en anglais le fait. Si vous souhaitez être sûr que les activités que vous projetez seront autorisées par la GNU General Public License, veuillez vous référer à sa seule version anglaise authentique.

La FSF vous recommande fermement de ne pas utiliser cette traduction en tant que termes officiels pour vos propres programmes; veuillez plutôt utiliser la version anglaise authentique telle que publiée par la FSF. Si vous choisissez d'achever cette traduction en même temps qu'un Programme sous licence GNU GPL, cela ne vous dispense pas de l'obligation d'achever en même temps une copie de la licence authentique en anglais, et de conserver dans la traduction cet avertissement important en français et son équivalent en anglais ci-dessous.

**Important Warning About This French Translation.**

This is a translation of the GNU General Public License (GPL) into French. This translation is distributed in the hope that it will facilitate understanding, but it is not an official or legally approved translation.

The Free Software Foundation (FSF) is not the publisher of this translation and has not approved it as a legal substitute for the authentic GNU General Public License. The translation has not been reviewed carefully by lawyers, and therefore the translator cannot be sure that it exactly represents the legal meaning of the authentic GNU General Public License published in English. This translation does not legally state the terms and conditions of use of any Program licensed under GNU GPL — only the original English text of the GNU LGPL does that. If you wish to be sure whether your planned activities are permitted by the GNU General Public License, please refer to its sole authentic English version.

The FSF strongly urges you not to use this translation as the official distribution terms for your programs; instead, please use the authentic English version published by the FSF. If you choose to convey this translation along with a Program covered by the GPL License, this does not remove your obligation to convey at the same time a copy of the authentic GNU GPL License in English, and you must keep in this translation this important warning in English and its equivalent in French above.

**Préambule**

La Licence Publique Générale GNU ("GNU General Public License") est une licence libre, en "copyleft", destinée aux œuvres logicielles et d'autres types d'œuvres.

Les licences de la plupart des œuvres logicielles et autres œuvres de la pratique sont conçues pour vous ôter votre liberté de partager et modifier ces œuvres. À l'inverse, la Licence Publique Générale GNU a pour but de garantir votre liberté de partager et changer toutes les versions d'un programme — afin d'assurer qu'il restera libre pour tous les utilisateurs. Nous, la **Free Software Foundation**, utilisons la Licence Publique Générale GNU pour la plupart de nos logiciels; cela s'applique aussi à toute autre œuvre éditée de cette façon par ses auteurs. Vous pouvez, vous aussi, l'appliquer à vos propres programmes.

Quand nous parlons de logiciel libre ("free"), nous nous référons à la liberté ("freedom"), pas au prix. Nos Licences Publiques Générales sont conçues pour assurer que vous ayez la liberté de distribuer des copies de logiciel libre (et le facturer si vous le souhaitez), que vous receviez le code source ou puissiez l'obtenir si vous le voulez, que vous puissiez modifier le logiciel ou en utiliser toute partie dans de nouveaux logiciels libres, et que vous sachiez que vous avez le droit de faire tout ceci.

Pour protéger vos droits, nous avons besoin d'empêcher que d'autres vous restreignent ces droits ou vous de-

mande de leur abandonner ces droits. En conséquence, vous avez certaines responsabilités si vous distribuez des copies d'un tel programme ou si vous le modifiez : les responsabilités de respecter la liberté des autres. Par exemple, si vous distribuez des copies d'un tel programme, que ce soit gratuit ou contre un paiement, vous devez accorder aux Destinataires les mêmes libertés que vous avez reçues. Vous devez aussi vous assurer qu'eux aussi reçoivent ou peuvent recevoir son code source. Et vous devez leur montrer les termes de cette licence afin qu'ils connaissent leurs droits.

Les développeurs qui utilisent la GPL GNU protègent vos droits en deux étapes : (1) ils affirment leur droits d'auteur ("copyright") sur le logiciel, et (2) vous accordent cette Licence qui vous donne la permission légale de le copier, le distribuer et/ou le modifier. Pour la protection des développeurs et auteurs, la GPL stipule clairement qu'il n'y a pas de garantie pour ce logiciel libre. Aux fins à la fois des utilisateurs et auteurs, la GPL requière que les versions modifiées soient marquées comme changées, afin que leurs problèmes ne soient pas attribués de façon erronée aux auteurs des versions précédentes.

Certains dispositifs sont conçus pour empêcher l'accès des utilisateurs à l'installation ou l'exécution de versions modifiées du logiciel à l'intérieur de ces dispositifs, alors que les fabricants le peuvent. Ceci est fondamentalement incompatible avec le but de protéger la liberté des utilisateurs de modifier le logiciel. L'aspect systématique de tels abus se produit dans le secteur des produits destinés aux utilisateurs individuels, ce qui est précisément ce qui est le plus inacceptable. Aussi, nous avons conçu cette version de la GPL pour prohiber cette pratique pour ces produits. Si de tels problèmes surviennent dans d'autres domaines, nous nous tenons prêt à étendre cette restriction à ces domaines dans de futures versions de la GPL, autant qu'il sera nécessaire pour protéger la liberté des utilisateurs.

Finalement, chaque programme est constamment menacé par les brevets logiciels. Les États ne devraient pas autoriser de tels brevets à restreindre le développement et l'utilisation de logiciels libres sur des ordinateurs d'usage général; mais dans ceux qui le font, nous voulons spécialement éviter le danger que les brevets appliqués à un programme libre puisse le rendre effectivement propriétaire. Pour empêcher ceci, la GPL assure que les brevets ne peuvent être utilisés pour rendre le programme non-libre.

Les termes précis et conditions concernant la copie, la distribution et la modification suivent.

**TERMES ET CONDITIONS****Article 0. Définitions.**

"Cette Licence" se réfère à la version 3 de la "GNU General Public License" (le texte original en anglais).

"Droit d'Auteur" signifie aussi les droits du "copyright" ou voisins qui s'appliquent à d'autres types d'œuvres, tels que celles sur les masques de semi-conducteurs.

"Le Programme" se réfère à toute œuvre qui peut être soumise au Droit d'Auteur ("copyright") et dont les droits d'utilisation sont concédés en vertu de cette Licence. Chacun des Licenciés, à qui cette Licence est concédée, est désigné par "vous." Les "Licenciés" et les "Destinataires" peuvent être des personnes physiques ou morales (individus ou organisations).

"Modifier" une œuvre signifie en obtenir une copie et adapter tout ou partie de l'œuvre d'une façon qui nécessite une autorisation d'un titulaire de Droit d'Auteur, autre que celle permettant d'en produire une copie conforme. L'œuvre résultante est appelée une "version modifiée" de la précédente œuvre, ou une œuvre "basée sur" la précédente œuvre.

Une "Œuvre Couverte" signifie soit le Programme non modifié soit une œuvre basée sur le Programme.

"Propager" une œuvre signifie faire quoi que ce soit avec elle qui, sans permission, vous rendrait directement ou indirectement responsable d'un délit de contrefaçon suivant les lois relatives au Droit d'Auteur, à l'exception de son exécution sur un ordinateur ou de la modification d'une copie privée. La propagation inclut la copie, la distribution (avec ou sans modification), la mise à disposition envers le public, et aussi d'autres activités dans certains pays.

"Acheminer" une œuvre signifie tout moyen de propagation de celle-ci qui permet à d'autres parties d'en réaliser ou recevoir des copies. La simple interaction d'un utilisateur à travers un réseau informatique, sans transfert effectif d'une copie, ne constitue pas un acheminement.

Une interface utilisateur interactive affiche des "Notices Légales Approuvées" quand elle comprend un dispositif convenable, bien visible et évident qui (1) affiche une notice appropriée sur les droits d'auteur et (2) informe l'utilisateur qu'il n'y a pas de garantie pour l'œuvre (sauf si des garanties ont été fournies hors du cadre de cette Licence), que les licenciés peuvent acheminer l'œuvre sous cette Licence, et comment consulter une copie de cette Licence. Si l'interface présente une liste de commandes utilisateur ou d'options, tel qu'un menu, un élément évident dans la liste présentée remplit ce critère.

**Article 1. Code source.**

Le "code source" d'une œuvre signifie la forme préférée

de l'œuvre qui permet ou facilite les modifications de celle-ci. Le "code objet" d'une œuvre signifie toute forme de l'œuvre qui n'en est pas le code source.

Une "Interface Standard" signifie une interface qui est soit celle d'une norme officielle définie par un organisme de normalisation reconnu ou, dans le cas des interfaces spécifiées pour un langage de programmation particulier, une interface largement utilisée parmi les développeurs qui travaillent dans ce langage.

Les "Bibliothèques Système" d'une œuvre exécutable incluent tout ce qui, en dehors de l'œuvre dans son ensemble, (a) est inclus dans la forme usuelle de paquetage d'un Composant Majeur mais ne fait pas partie de ce Composant Majeur et (b) sert seulement à permettre l'utilisation de l'œuvre avec ce Composant Majeur ou à mettre en œuvre une Interface Standard pour laquelle une mise en œuvre est disponible au public sous forme de code source; un "Composant Majeur" signifie, dans ce contexte, un composant majeur essentiel (noyau, système de fenêtre, etc.) du système d'exploitation (le cas échéant) d'un système sur lequel l'œuvre exécutable fonctionne, ou bien un compilateur utilisé pour produire le code objet de l'œuvre, ou un interprète de code objet utilisé pour exécuter celui-ci.

Le "Source Correspondant" d'une œuvre sous forme de code objet signifie l'ensemble des codes sources nécessaires pour générer, installer et (dans le cas d'une œuvre exécutable) exécuter le code objet et modifier l'œuvre, y compris les scripts pour contrôler ces activités. Cependant, cela n'inclut pas les Bibliothèques Système de l'œuvre, ni les outils d'usage général ou les programmes libres généralement disponibles qui peuvent être utilisés sans modification pour achever ces activités mais ne sont pas partie de cette œuvre. Par exemple le Source Correspondant inclut les fichiers de définition d'interfaces associés aux fichiers sources de l'œuvre, et le code source des bibliothèques partagées et des sous-routines liées dynamiquement, pour lesquelles l'œuvre est spécifiquement conçue pour les requérir via, par exemple, des communications de données ou contrôles de flux internes entre ces sous-programmes et d'autres parties de l'œuvre.

Le Source Correspondant n'a pas besoin d'inclure tout ce que les utilisateurs peuvent régénérer automatiquement à partir d'autres parties du Source Correspondant.

Le Source Correspondant pour une œuvre sous forme de code source est cette même œuvre.

**Article 2. Permissions de base.**

Tous les droits accordés suivant cette Licence le sont jusqu'au terme des Droits d'Auteur ("copyright") sur le Programme, et sont irrévocables pourvu que les conditions établies soient remplies. Cette Licence affirme explicitement votre permission illimitée d'exécuter le Programme non modifié. La sortie produite par l'exécution d'une Œuvre Couverte n'est couverte par cette Licence que si cette sortie, étant donné leur contenu, constitue une Œuvre Couverte. Cette Licence reconnaît vos propres droits d'usage raisonnable ("fair use" en législation des États-Unis d'Amérique) ou autres équivalents, tels qu'ils sont pourvus par la loi applicable sur le Droit d'Auteur ("copyright").

Vous pouvez créer, exécuter et propager sans condition des Œuvres Couvertes que vous n'acheminiez pas, aussi longtemps que votre licence demeure en vigueur. Vous pouvez acheminer des Œuvres Couvertes à d'autres personnes dans le seul but de leur faire réaliser des modifications à votre usage exclusif, ou pour qu'ils vous fournissent des facilités vous permettant d'exécuter ces œuvres, pourvu que vous vous conformiez aux termes de cette Licence lors de l'acheminement de tout matériel dont vous ne contrôlez pas le Droit d'Auteur ("copyright"). Ceux qui, dès lors, réalisent ou exécutent pour vous les Œuvres Couvertes ne doivent avoir le faire qu'exclusivement pour votre propre compte, sous votre direction et votre contrôle, suivant des termes qui leur interdisent de réaliser, en dehors de leurs relations avec vous, toute copie de votre matériel soumis au Droit d'Auteur.

L'acheminement dans toutes les autres circonstances n'est permis que selon les conditions établies ci-dessous. La concession de sous-licences n'est pas autorisée; l'article 10 rend cet usage non nécessaire.

**Article 3. Protection des droits légaux des utilisateurs envers les lois anti-contournement.**

Aucune Œuvre Couverte ne doit être vue comme faisant partie d'une mesure technologique effective selon toute loi applicable remplissant les obligations prévues à l'article 11 du traité international sur le droit d'auteur adopté à l'OMPI le 20 décembre 1996, ou toutes lois similaires qui prohibent ou restreignent le contournement de telles mesures.

Si vous acheminez une Œuvre Couverte, vous renoncez à tout pouvoir légal d'interdire le contournement des mesures technologiques dans tous les cas où un tel contournement serait effectué en exerçant les droits prévus dans cette Licence pour cette Œuvre Couverte, et vous déclarez rejeter toute intention de limiter l'opération ou la modification de l'Œuvre, en tant que moyens pour renforcer, à l'encontre des utilisateurs de cette Œuvre, vos droits légaux ou ceux de tierces parties d'interdire le contournement desdites mesures technologiques.

**Article 4. Acheminement des copies conformes.**

Vous pouvez acheminer des copies conformes du code source du Programme tel que vous l'avez reçu, sur n'importe quel support, pourvu que vous publiiez scrupuleusement et de façon appropriée sur chaque copie une notice de Droit d'Auteur appropriée; gardez intactes toutes les notices établissant que cette Licence et tous les termes additionnels non permisifs ajoutés en accord avec l'article 7 s'appliquent à ce code; et donnez à chacun des Destinataires une copie de cette Licence en même temps que le Programme.

Vous pouvez facturer un prix quelconque, y compris gratuit, pour chacune des copies que vous acheminez, et vous pouvez offrir une protection additionnelle de support ou de garantie en échange d'un paiement.

**Article 5. Acheminement des versions sources modifiées.**

Vous pouvez acheminer une œuvre basée sur le Programme, ou bien les modifications pour le produire à partir du Programme, sous la forme de code source suivant les termes de l'article 4, pourvu que vous satisfaisiez aussi à chacune des conditions requises suivantes :

- a) L'œuvre doit comporter des notices évidentes établissant que vous l'avez modifiée et donnant la date correspondante.
- b) L'œuvre doit comporter des notices évidentes établissant qu'elle est éditée selon cette Licence et les conditions ajoutées d'après l'article 7. Cette obligation vient modifier l'obligation de l'article 4 de "garder intactes toutes les notices."
- c) Vous devez licencier l'œuvre entière, comme un tout, suivant cette Licence à quiconque entre en possession d'une copie. Cette Licence s'appliquera en conséquence, avec les termes additionnels applicables prévus par l'article 7, à la totalité de l'œuvre et chacune de ses parties, indépendamment de la façon dont elles sont empaquetées. Cette licence ne donne aucune permission de licencier l'œuvre d'une autre façon, mais elle n'invalide pas une telle permission que vous auriez reçue séparément.
- d) Si l'œuvre a des interfaces utilisateurs interactives, chacune doit afficher les Notices Légales Approuvées; cependant si le Programme a des interfaces qui n'affichent pas les Notices Légales Approuvées, votre œuvre n'a pas à les modifier pour qu'elles les affichent.

Une compilation d'une Œuvre Couverte avec d'autres œuvres séparées et indépendantes, qui ne sont pas par leur nature des extensions de l'Œuvre Couverte, et qui ne sont pas combinés avec elle de façon à former un programme plus large, dans ou sur un volume de stockage ou un support de distribution, est appelé un "agrégat" si la compilation et son Droit d'Auteur résultant ne sont pas utilisés pour limiter l'accès ou les droits légaux des utilisateurs de la compilation en deçà de ce que permettent les œuvres individuelles. L'inclusion d'une Œuvre Couverte dans un agrégat ne cause pas l'application de cette Licence aux autres parties de l'agrégat.

**Article 6. Acheminement des formes non sources.**

Vous pouvez acheminer sous forme de code objet une Œuvre Couverte suivant les termes des articles 4 et 5, pourvu que vous acheminez également suivant les termes de cette Licence le Source Correspondant lisible par une machine, d'une des façons suivantes :

- a) Acheminer le code objet sur, ou inclus dans, un produit physique (y compris un support de distribution physique), accompagné par le Source Correspondant fixé sur un support physique durable habituellement utilisé pour les échanges de logiciels.
- b) Acheminer le code objet sur, ou inclus dans, un produit physique (y compris un support de distribution physique), accompagné d'une offre écrite, valide pour au moins trois années et valide pour aussi longtemps que vous fournissez des pièces de rechange ou un support client pour ce modèle de produit, afin de donner à quiconque possède le code objet soit (1) une copie du Source Correspondant à tout logiciel dans ce produit qui est couvert par cette Licence, sur un support physique durable habituellement utilisé pour les échanges de logiciels, pour un prix non supérieur au coût raisonnable de la réalisation physique de l'acheminement de la source, ou soit (2) un accès permettant de copier le Source Correspondant depuis un serveur réseau sans frais.
- c) Acheminer des copies individuelles du code objet avec une copie de l'offre écrite de fournir le Source Correspondant. Cette alternative est permise seulement occasionnellement et non-commercialement, et seulement si vous avez reçu le code objet avec une telle offre, en accord avec l'article 6 alinéa b.
- d) Acheminer le code objet en offrant un accès depuis un emplacement désigné (gratuit ou contre facturation) et offrir un accès équivalent au Source Correspondant de la même façon via le même emplacement et sans facturation supplémentaire. Vous n'avez pas besoin d'obliger les Destinataires à copier le Source Correspondant en même temps que le code objet. Si l'emplacement pour copier le code objet est un serveur réseau, le Source Correspondant peut être sur un serveur différent (opéré par vous ou par un tiers) qui supporte des facilités équivalentes de copie, pourvu que vous mainteniez des directions claires à proximité du code objet indiquant où trouver le Source Correspondant. Indépendamment de quel serveur héberge le Source Correspondant, vous res-

te obligé de vous assurer qu'il reste disponible aussi longtemps que nécessaire pour satisfaire à ces obligations.

e) Acheminer le code objet en utilisant une transmission d'égal-à-égal, pourvu que vous informiez les autres participants sur l'endroit où le code objet et le Source Correspondant de l'œuvre sont offerts sans frais au public général suivant l'article 6 alinéa d.

Une portion séparable du code objet, dont le code source est exclu du Source Correspondant en tant que Bibliothèque Système, n'a pas besoin d'être incluse dans l'acheminement de l'œuvre sous forme de code objet.

Un "Produit Utilisateur" est soit (1) un "Produit de Consommation", ce qui signifie toute propriété personnelle tangible normalement utilisée à des fins personnelles, familiales ou relatives au foyer, soit (2) toute chose conçue ou vendue pour l'incorporation dans un lieu d'habitation. Pour déterminer si un produit constitue un Produit de Consommation, les cas ambigus sont résolus en fonction de la couverture. Pour un produit particulier reçu par un utilisateur particulier, l'expression "normalement utilisée" ci-avant se réfère à une utilisation typique ou l'usage commun de produits de même catégorie, indépendamment du statut de cet utilisateur particulier ou de la façon spécifique dont cet utilisateur particulier utilise effectivement ou s'attend lui-même ou est attendu à utiliser ce produit. Un produit est un Produit de Consommation indépendamment du fait que ce produit a ou n'a pas d'utilisations substantielles commerciales, industrielles ou hors Consommation, à moins que de telles utilisations représentent le seul mode significatif d'utilisation du produit.

Les "Informations d'Installation" d'un Produit Utilisateur signifient toutes les méthodes, procédures, clés d'autorisation ou autres informations requises pour installer et exécuter des versions modifiées d'une Œuvre Couverte dans ce Produit Utilisateur à partir d'une version modifiée de son Source Correspondant. Les informations qui suffisent à assurer la continuité de fonctionnement du code objet modifié ne doivent en aucun cas être empêchées ou interférées du seul fait qu'une modification a été effectuée.

Si vous acheminez le code objet d'une Œuvre Couverte dans, ou avec, ou spécifiquement pour l'utilisation dans, un Produit Utilisateur et si l'acheminement se produit en tant qu'élément d'une transaction dans laquelle le droit de possession et d'utilisation du Produit Utilisateur est transféré au Destinataire définitivement ou pour un terme fixé (indépendamment de la façon dont la transaction est caractérisée), le Source Correspondant acheminé selon cet article-ci doit être accompagné des Informations d'Installation. Mais cette obligation ne s'applique pas si ni vous ni aucune tierce partie ne détient la possibilité d'installer un code objet modifié sur le Produit Utilisateur (par exemple, l'œuvre a été installée en mémoire morte).

L'obligation de fournir les Informations d'Installation n'inclut pas celle de continuer à fournir un service de support, une garantie ou des mises à jour pour une œuvre qui a été modifiée ou installée par le Destinataire, ou pour le Produit Utilisateur dans lequel elle a été modifiée ou installée. L'accès à un réseau peut être rejeté quand la modification elle-même affecte matériellement et défavorablement les opérations du réseau ou viole les règles et protocoles de communication au travers du réseau.

Le Source Correspondant acheminé et les Informations d'Installation fournies, en accord avec cet article, doivent être dans un format publiquement documenté (et dont une implémentation est disponible auprès du public sous forme de code source) et ne doit nécessiter aucune clé ou mot de passe spécial pour le dépaquetage, la lecture ou la copie.

### Article 7. Termes additionnels.

Les « permissions additionnelles » désignent les termes qui supplémentent ceux de cette Licence en émettant des exceptions à l'une ou plusieurs de ses conditions. Les permissions additionnelles qui sont applicables au Programme entier doivent être traitées comme si elles étaient incluses dans cette Licence, dans les limites de leur validité suivant la loi applicable. Si des permissions additionnelles s'appliquent seulement à une partie du Programme, cette partie peut être utilisée séparément suivant ces permissions, mais le Programme tout entier reste gouverné par cette Licence sans regard aux permissions additionnelles.

Quand vous acheminez une copie d'une Œuvre Couverte, vous pouvez à votre convenance ôter toute permission additionnelle de cette copie, ou de n'importe quelle partie de celui-ci. (Des permissions additionnelles peuvent être rédigées de façon à requérir leur propre suppression dans certains cas où vous modifiez l'œuvre.) Vous pouvez placer les permissions additionnelles sur le matériel acheminé, ajoutées par vous à une Œuvre Couverte pour laquelle vous avez ou pouvez donner les permissions de Droit d'Auteur ("copyright") appropriées. Nonobstant toute autre clause de cette Licence, pour tout constituant que vous ajoutez à une Œuvre Couverte, vous pouvez (si autorisé par les titulaires de Droit d'Auteur pour ce constituant) supplémenter les termes de cette Licence avec des termes :

a) qui rejettent la garantie ou limitent la responsabilité

de façon différente des termes des articles 15 et 16 de cette Licence; ou

b) qui requièrent la préservation de notices légales raisonnables spécifiées ou les attributions d'auteur dans ce constituant ou dans les Notices Légales Appropriées affichées par les œuvres qui le contiennent; ou

c) qui prohibent la représentation incorrecte de l'origine de ce constituant, ou qui requièrent que les versions modifiées d'un tel constituant soient marquées par des moyens raisonnables comme différentes de la version originale; ou

d) qui limitent l'usage à but publicitaire des noms des concédants de licence et des auteurs du constituant; ou

e) qui refusent à accorder des droits selon la législation relative aux marques commerciales, pour l'utilisation dans des noms commerciaux, marques commerciales ou marques de services; ou

f) qui requièrent l'indemnisation des concédants de licences et auteurs du constituant par quiconque achemine ce constituant (ou des versions modifiées de celui-ci) en assumant contractuellement la responsabilité envers le Destinataire, pour toute responsabilité que ces engagements contractuels imposent directement à ces octroyants de licences et auteurs.

Tous les autres termes additionnels non permisifs sont considérés comme des « restrictions avancées » dans le sens de l'article 10. Si le Programme tel que vous l'avez reçu, ou toute partie de celui-ci, contient une notice établissant qu'il est gouverné par cette Licence en même temps qu'un terme qui est une restriction avancée, vous pouvez ôter ce terme. Si un document de licence contient une restriction avancée mais permet la reconcession de licence ou l'acheminement suivant cette Licence, vous pouvez ajouter une Œuvre Couverte constituante gouvernée par les termes de ce document de licence, pourvu que la restriction avancée ne survit pas à une telle cession de licence ou un tel acheminement.

Si vous ajoutez des termes à une Œuvre Couverte en accord avec cet article, vous devez placer, dans les fichiers sources appropriés, une déclaration des termes additionnels qui s'appliquent à ces fichiers, ou une notice indiquant où trouver les termes applicables.

Les termes additionnels, qu'ils soient permisifs ou non permisifs, peuvent être établis sous la forme d'une licence écrite séparément, ou établis comme des exceptions; les obligations ci-dessus s'appliquent dans chacun de ces cas.

### Article 8. Termination.

Vous ne pouvez ni modifier ni acheminer une Œuvre Couverte autrement que suivant les termes de cette Licence. Toute autre tentative de le propager ou le modifier est nulle et terminera automatiquement vos droits selon cette Licence (y compris toute licence de brevet accordée selon le troisième paragraphe de l'article 11).

Cependant, si vous cessez toute violation de cette Licence, alors votre licence depuis un titulaire de Droit d'Auteur ("copyright") est réinstaurée (a) à titre provisoire à moins que et jusqu'à ce que le titulaire de Droit d'Auteur termine finalement et explicitement votre licence, et (b) de façon permanente si le titulaire de Droit d'Auteur ne parvient pas à vous notifier de la violation par quelque moyen raisonnable dans les soixante (60) jours après la cessation.

De plus, votre licence depuis un titulaire particulier de Droit d'Auteur est réinstaurée de façon permanente si ce titulaire vous a notifié de la violation par quelque moyen raisonnable, et si c'est la première fois que vous avez reçu une notification de violation de cette Licence (pour une œuvre quelconque) depuis ce titulaire de Droit d'Auteur, et si vous résolvez la violation dans les trente (30) jours qui suivent votre réception de la notification.

La terminaison de vos droits suivant cette section ne terminera pas les licences des parties qui ont reçu des copies ou droits de votre part suivant cette Licence. Si vos droits ont été terminés et non réinstaurés de façon permanente, vous n'êtes plus qualifié à recevoir de nouvelles licences pour les mêmes constituants selon l'article 10.

### Article 9. Acceptation non requise pour obtenir des copies.

Vous n'êtes pas obligé d'accepter cette licence afin de recevoir ou exécuter une copie du Programme. La propagation asservie d'une Œuvre Couverte qui se produit simplement en conséquence d'une transmission d'égal-à-égal pour recevoir une copie ne nécessite pas l'acceptation. Cependant, rien d'autre que cette Licence ne vous accorde la permission de propager ou modifier une quelconque Œuvre Couverte. Ces actions enfreignent le Droit d'Auteur si vous n'acceptez pas cette Licence. Par conséquent, en modifiant ou propageant une Œuvre Couverte, vous indiquez votre acceptation de cette Licence pour agir ainsi.

### Article 10. Cession automatique de Licence aux Destinataires et intermédiaires.

Chaque fois que vous acheminez une Œuvre Couverte, le Destinataire reçoit automatiquement une licence de la part des concédants originaux, pour exécuter, modifier et propager cette œuvre, suivant les termes de cette Licence. Vous n'êtes pas responsable du renforcement de la conformation des tierces parties aux termes de cette Licence.

Une "transaction d'entité" désigne une transaction qui

transfère le contrôle d'une organisation, ou de substantiellement tous ses actifs, ou la subdivision d'une organisation, ou la fusion de plusieurs organisations. Si la propagation d'une Œuvre Couverte résulte d'une transaction d'entité, chaque partie à cette transaction qui reçoit une copie de l'œuvre reçoit aussi les licences pour l'œuvre que le prédecesseur intéressé à cette partie avait ou pourrait donner selon le paragraphe précédent, plus un droit de possession du Source Correspondant de cette œuvre depuis le prédecesseur intéressé si ce prédecesseur en dispose ou peut l'obtenir par des efforts raisonnables.

Vous ne pouvez imposer aucune restriction avancée dans l'exercice des droits accordés ou affirmés selon cette Licence. Par exemple, vous ne pouvez imposer aucun paiement pour la licence, aucune royauté, ni aucune autre charge pour l'exercice des droits accordés selon cette Licence; et vous ne pouvez amorcer aucun litige judiciaire (y compris une réclamation croisée ou contre-réclamation dans un procès) sur l'allégation qu'une revendication de brevet est enfreinte par la réalisation, l'utilisation, la vente, l'offre de vente, ou l'importation du Programme ou d'une quelconque portion de celui-ci.

### Article 11. Brevets.

Un « contributeur » est un titulaire de Droit d'Auteur ("copyright") qui autorise l'utilisation selon cette Licence du Programme ou de l'œuvre sur laquelle le Programme est basé. L'œuvre ainsi soumise à licence est appelée la "version contributive" de ce contributeur. Les "revendications de brevets essentielles" sont toutes les revendications de brevets détenues ou contrôlées par le contributeur, qu'elles soient déjà acquises par lui ou acquises subseqüemment, qui pourraient être enfreintes de quelque manière, permises par cette Licence, sur la réalisation, l'utilisation ou la vente de la version contributive de celui-ci. Aux fins de cette définition, le "contrôle" inclut le droit de concéder des sous-licences de brevets d'une manière consistante, nécessaire et suffisante, avec les obligations de cette Licence.

Chaque contributeur vous accorde une licence de brevet non exclusive, mondiale et libre de toute royauté, selon les revendications de brevet essentielles, pour réaliser, utiliser, vendre, offrir à la vente, importer et autrement exécuter, modifier et propager les contenus de sa version contributive.

Dans les trois paragraphes suivants, une "licence de brevet" désigne tous les accords ou engagements exprimés, quel que soit le nom que vous lui donnez, de ne pas mettre en vigueur un brevet (telle qu'une permission explicite pour mettre en pratique un brevet, ou un accord pour ne pas poursuivre un Destinataire pour cause de violation de brevet). "Accorder" une telle licence de brevet à une partie signifie conclure un tel accord ou engagement à ne pas faire appliquer le brevet à cette partie.

Si vous acheminez une Œuvre Couverte, dépendant en connaissance d'une licence de brevet, et si le Source Correspondant de l'œuvre n'est pas disponible à quiconque copie, sans frais et suivant les termes de cette Licence, à travers un serveur réseau publiquement accessible ou tout autre moyen immédiatement accessible, alors vous devez soit (1) rendre la Source Correspondante ainsi disponible, soit (2) vous engager à vous priver pour vous-même du bénéfice de la licence de brevet pour cette œuvre particulière, soit (3) vous engager, d'une façon consistante avec les obligations de cette Licence, à étendre la licence de brevet aux Destinataires de cette œuvre. "Dépendant en connaissance" signifie que vous avez effectivement connaissance que, selon la licence de brevet, votre acheminement de l'Œuvre Couverte dans un pays, ou l'utilisation de l'Œuvre Couverte par votre Destinataire dans un pays, enfreindrait un ou plusieurs brevets identifiables dans ce pays où vous avez des raisons de penser qu'ils sont valides.

Si, conformément à ou en liaison avec une même transaction ou un même arrangement, vous acheminez, ou propagez en procurant un acheminement de, une Œuvre Couverte et si accordez une licence de brevet à l'une des parties recevant l'Œuvre Couverte pour lui permettre d'utiliser, propager, modifier ou acheminer une copie spécifique de l'Œuvre Couverte, alors votre accord est automatiquement étendu à tous les Destinataires de l'Œuvre Couverte et des œuvres basées sur celle-ci.

Une licence de brevet est "discriminatoire" si, dans le champ de sa couverture, elle n'inclut pas un ou plusieurs des droits qui sont spécifiquement accordés selon cette Licence, ou en prohibe l'exercice, ou est conditionnée par le non-exercice d'un ou plusieurs de ces droits. Vous ne pouvez pas acheminer une Œuvre Couverte si vous êtes partie à un arrangement, selon lequel une partie tierce exerçant son activité dans la distribution de logiciels et à laquelle vous effectuez un paiement fondé sur l'étendue de votre activité d'acheminement de l'œuvre, et selon lequel la partie tierce accorde, à une quelconque partie qui recevrait depuis vous l'Œuvre Couverte, une licence de brevet discriminatoire (a) en relation avec les copies de l'Œuvre Couverte acheminées par vous (ou les copies réalisées à partir de ces copies), ou (b) avant tout destinée à et en relation avec des produits spécifiques ou compilations contenant l'Œuvre Couverte, à moins que vous ayez conclu cet arrangement ou que la licence de brevet ait été accordée avant le 28 mars 2007.

Rien dans cette Licence ne devrait être interprété comme

devant exclure ou limiter toute licence implicite ou d'autres moyens de défense à une infraction qui vous seraient autrement disponible selon la loi applicable relative aux brevets.

### Article 12. Non abandon de la liberté des autres.

Si des conditions vous sont imposées (que ce soit par décision judiciaire, par un accord ou autrement) qui contredisent les conditions de cette Licence, elles ne vous excluent pas des conditions de cette Licence. Si vous ne pouvez pas acheminer une Œuvre Couverte de façon à satisfaire simultanément vos obligations suivant cette Licence et toutes autres obligations pertinentes, alors en conséquence vous ne pouvez pas du tout l'acheminer. Par exemple, si vous avez un accord sur des termes qui vous obligent à collecter pour le rachat de royalties devant ceux à qui vous acheminez le Programme, la seule façon qui puisse vous permettre de satisfaire à la fois à ces termes et ceux de cette Licence sera de vous abstenir entièrement d'acheminer le Programme.

### Article 13. Utilisation avec la Licence Générale Publique Affero GNU.

Nonobstant toute autre clause de cette Licence, vous avez la permission de lier ou combiner toute Œuvre Couverte avec une œuvre placée sous la version 3 de la Licence Générale Publique GNU Affero ("GNU Affero General Public License") en une seule œuvre combinée, et d'acheminer l'œuvre résultante. Les termes de cette Licence continueront à s'appliquer à la partie formant une Œuvre Couverte, mais les obligations spéciales de la Licence Générale Publique GNU Affero, article 13, concernant l'interaction à travers un réseau, s'appliqueront à la combinaison en tant que telle.

### Article 14. Versions révisées de cette Licence.

La Free Software Foundation peut publier des versions révisées et/ou nouvelles de la Licence Générale Publique GNU ("GNU General Public License") de temps en temps. De telles versions nouvelles resteront similaires dans l'esprit avec la présente version, mais peuvent différer dans le détail afin de traiter de nouveaux problèmes ou préoccupations.

Chaque version reçoit un numéro de version distinctif. Si le Programme indique qu'une version spécifique de la Licence Générale Publique GNU "ou toute version ultérieure" ("or any later version") s'applique à celui-ci, vous avez le choix de suivre soit les termes et conditions de cette version numérotée, soit ceux de n'importe quelle version publiée ultérieurement par la Free Software Foundation. Si le Programme n'indique pas une version spécifique de la Licence Générale Publique GNU, vous pouvez choisir l'une quelconque des versions qui ont été publiées par la Free Software Foundation.

Si le Programme spécifie qu'un intermédiaire peut décider quelles versions futures de la Licence Générale Publique GNU peut être utilisée, la déclaration publique d'acceptation d'une version par cet intermédiaire vous autorise à choisir cette version pour le Programme.

Des versions ultérieures de la licence peuvent vous donner des permissions additionnelles ou différentes. Cependant aucune obligation additionnelle n'est imposée à l'un des auteurs ou titulaires de Droit d'Auteur du fait de votre choix de suivre une version ultérieure.

### Article 15. Déclaration d'absence de garantie.

Il n'y a aucune garantie pour le programme, dans les limites permises par la loi applicable. À moins que cela ne soit établi différemment par écrit, les propriétaires de droits et/ou les autres parties fournissent le programme "en l'état" sans garantie d'aucune sorte, qu'elle soit exprimée ou implicite, ceci comprenant, sans se limiter à celles-ci, les garanties implicites de commercialisabilité et d'adéquation à un objectif particulier. Vous assumez le risque entier concernant la qualité et les performances du programme. Dans l'éventualité où le programme s'avérerait défectueux, vous assumez les coûts de tous les services, réparations ou corrections nécessaires.

### Article 16. Limitation de responsabilité.

En aucune autre circonstance que celles requises par la loi applicable ou accordées par écrit, un titulaire de droits sur le programme, ou tout autre partie qui modifie ou achemine le programme comme permis ci-dessus, ne peut être tenu pour responsable envers vous pour les dommages, incluant tout dommage général, spécial, accidentel ou induit survenant par suite de l'utilisation ou de l'incapacité d'utiliser le programme (y compris, sans se limiter à celles-ci, la perte de données ou l'inexactitude des données retournées ou les pertes subies par vous ou des parties tierces ou l'incapacité du programme à fonctionner avec tout autre programme), même si un tel titulaire ou toute autre partie a été avisé de la possibilité de tels dommages.

### Article 17. Interprétation des sections 15 et 16.

Si la déclaration d'absence de garantie et la limitation de responsabilité fournies ci-dessus ne peuvent prendre effet localement selon leurs termes, les cours de justice qui les examinent doivent appliquer la législation locale qui s'approche au plus près possible une levée absolue de toute responsabilité civile liée au Programme, à moins qu'une garantie ou assumption de responsabilité accompagne une copie du Programme en échange d'un paiement.