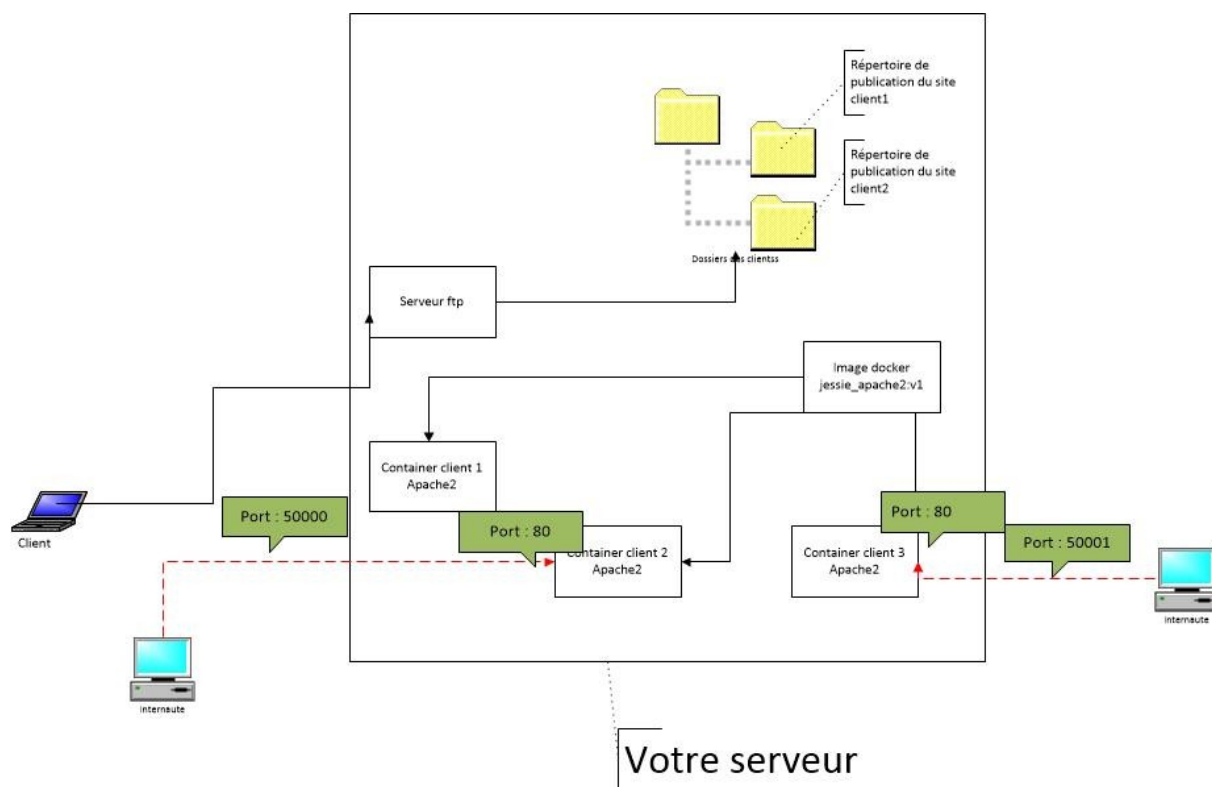


Cloud d'entreprises : docker

I CONTEXTE

Vous vous lancez dans le cloud. Vous décidez dans un premier temps de proposer comme service l'hébergement de sites web statiques à vos clients. Pour répondre rapidement à leurs attentes, vous décidez d'utiliser docker. Grâce à docker vous aurez une image docker jessie_apache2 que vous allez construire et que vous lancerez dans un nouveau container docker pour chaque client. Les clients eux déposeront grâce à un service ftp leur site web statique dans leur répertoire personnel créé sur la machine physique hébergeant les containers

Tout ceci peut être représenté de la manière suivante :



II INSTALLATION DE DOCKER

Avec docker les processus sont isolés dans des containers. Un container embarque l'application et ses dépendances, l'utilisation de l'espace de disque hébergeant les containers est donc optimale. Il n'y a pas d'émulation, l'exécution des processus est donc aussi rapide que si l'application était installée directement sur le system hôte. Seul inconvénient : système linux obligatoire. Vous allez quand même utiliser l'émulation puisque notre machine hébergeant les containers docker sera une machine virtuelle Ubuntu 14 LTS.

II.1 INSTALLATION DE DOCKER ET TÉLÉCHARGEMENT D'UNE IMAGE DEBIAN VIDE DE TOUTE APPLICATION

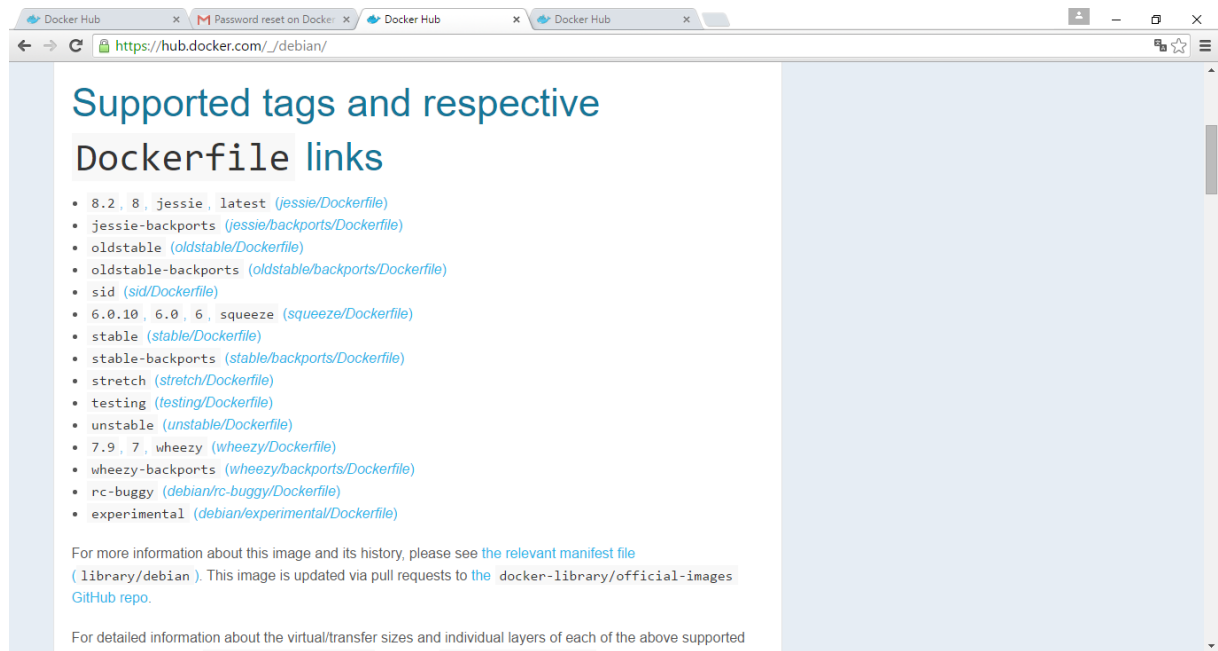
Installation : **sudo apt-get install docker.io**

Ajout de votre user au groupe docker : **sudo addgroup votre-login docker**

Déconnectez-vous et reconnectez-vous sous votre compte.

Docker télécharge ses images sur un dépôt <https://registry.hub.docker.com> (site officiel).

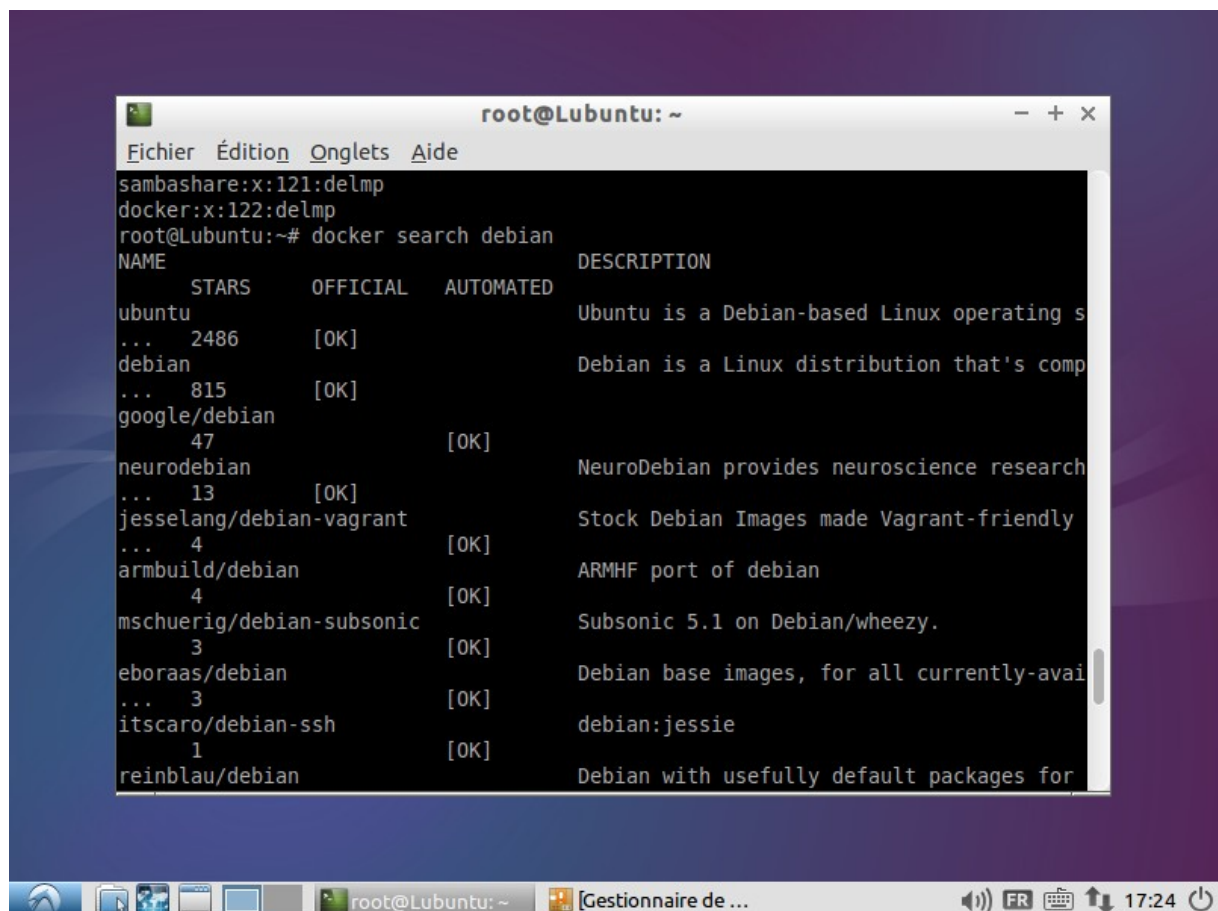
Connectez-vous à ce site, en vous créant un compte et vérifiez que l'image debian : jessie existe.



En local, pour chercher si une image existe vous utilisez la commande search : exemple **docker search WordPress** (pour une image WordPress), **docker search debian** (pour une image debian).

Vérifiez l'existence des images debian :

docker search debian



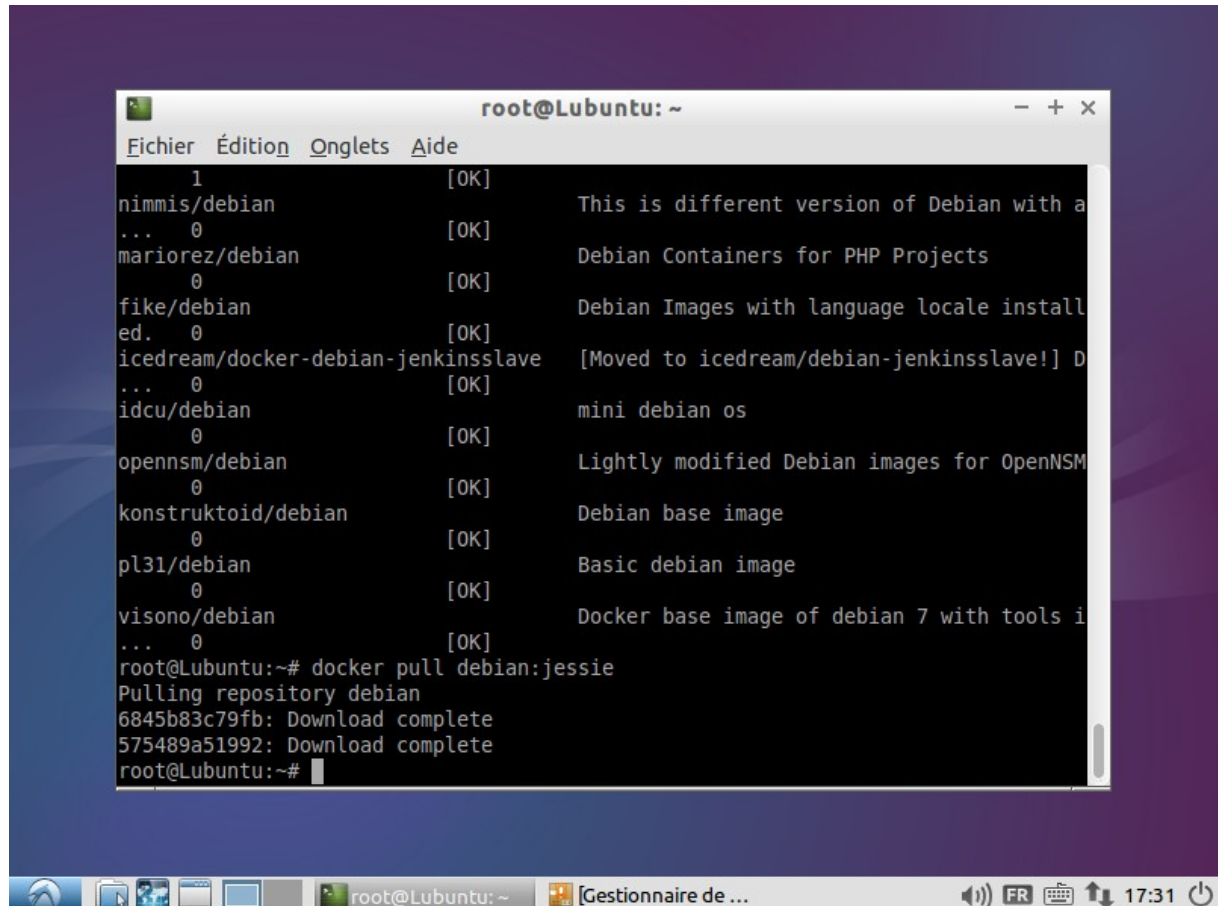
Stars : indique la popularité de l'image, Official indique si l'image a été produite par une source officielle reconnue, Automated indique si l'image a été créée automatiquement depuis un dépôt GitHub ou BitBucket.

Pour télécharger une image : **docker pull nom_image :tag_image**.

Exemple pour télécharger une image debian, jessie : **docker pull debian :jessie**.

Télécharger une image debian : jessie

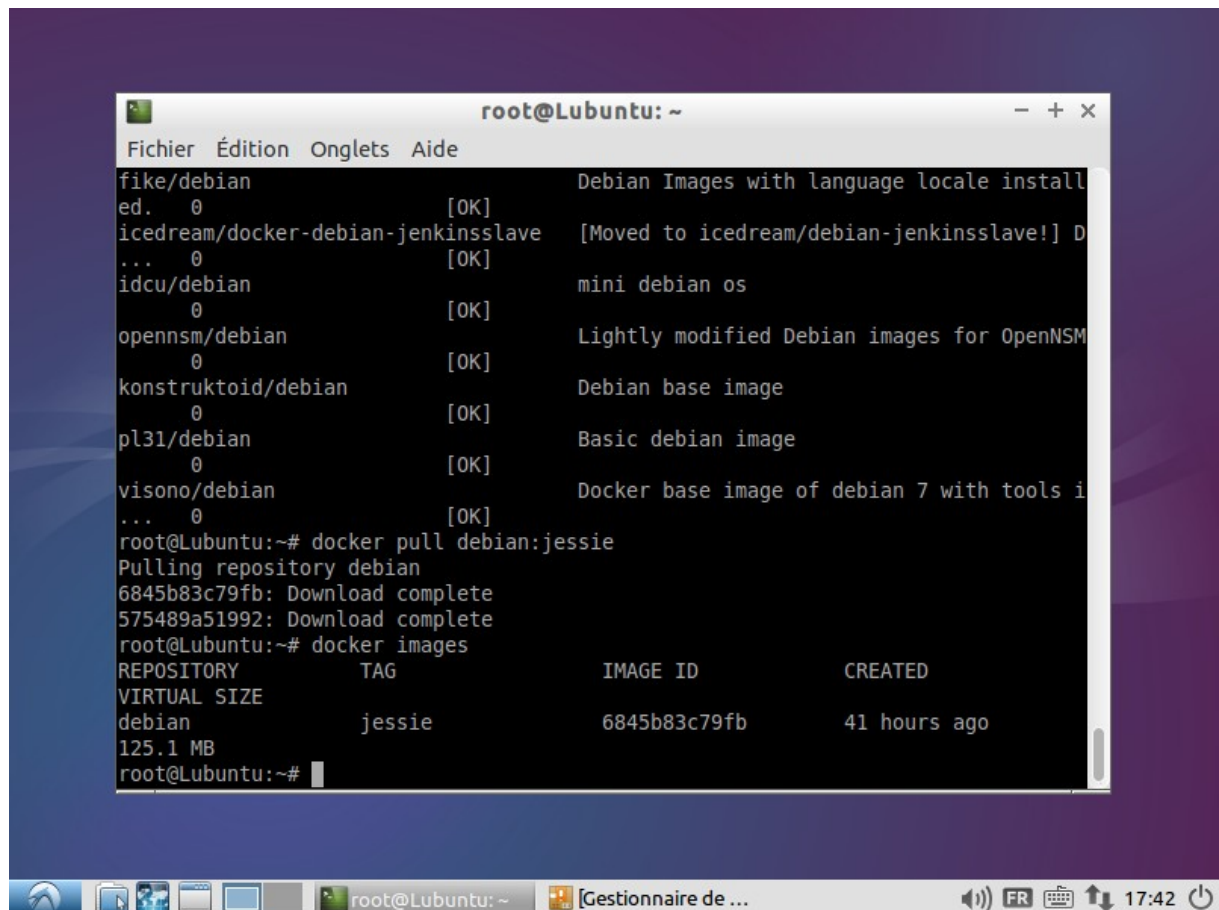
docker pull debian :jessie



```
root@Lubuntu: ~  
Fichier  Édition  Onglets  Aide  
1 [OK]  
nimmis/debian This is different version of Debian with a  
... 0 [OK]  
mariores/debian Debian Containers for PHP Projects  
0 [OK]  
fike/debian Debian Images with language locale install  
ed. 0 [OK]  
icedream/docker-debian-jenkinslave [Moved to icedream/debian-jenkinslave!] D  
... 0 [OK]  
idcu/debian mini debian os  
0 [OK]  
opennsn/debian Lightly modified Debian images for OpenNSM  
0 [OK]  
konstruktoid/debian Debian base image  
0 [OK]  
pl31/debian Basic debian image  
0 [OK]  
visono/debian Docker base image of debian 7 with tools i  
... 0 [OK]  
root@Lubuntu:~# docker pull debian:jessie  
Pulling repository debian  
6845b83c79fb: Download complete  
575489a51992: Download complete  
root@Lubuntu:~#
```

Deux téléchargements ont eu lieu, car les images sont gérées par version (dans la succession des versions seules les différences sont retenues, la mise en place de la version finale demandant le rétablissement de toutes les différences depuis la version initiale.

Pour vérifier que l'image est maintenant sur votre poste : **docker images**

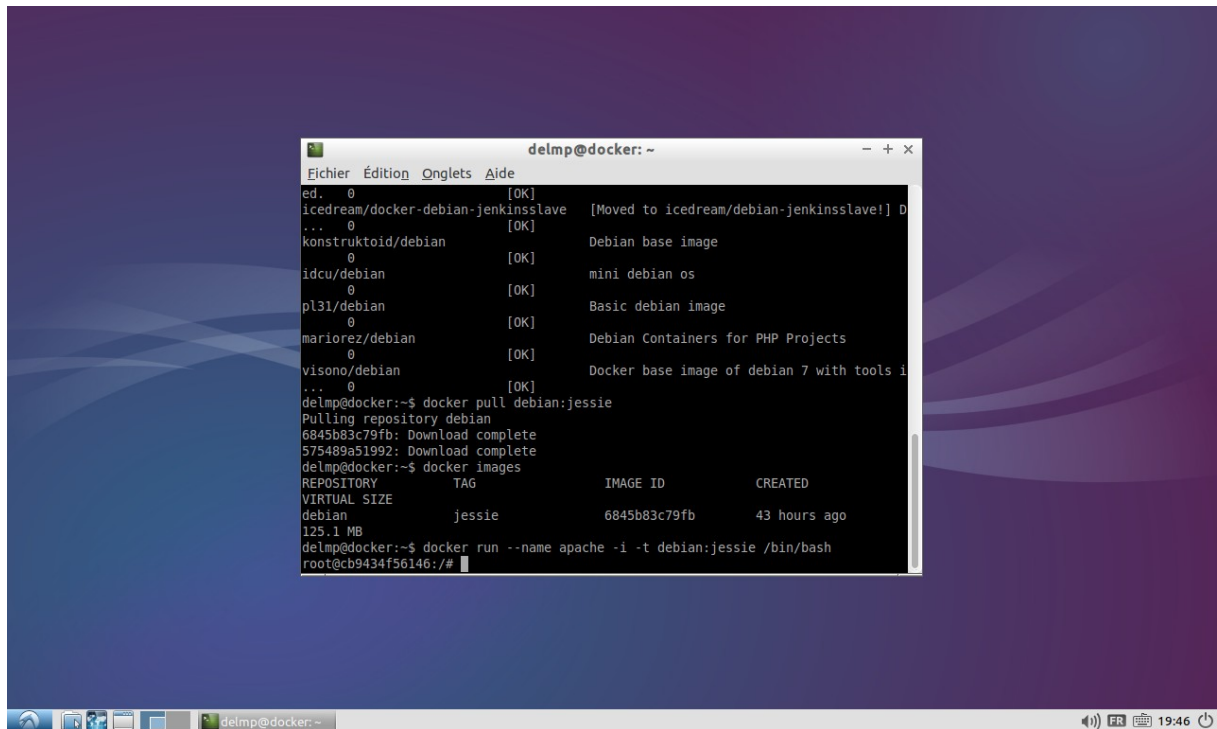


Une image est un fichier en lecture seule, contenant lui-même un système de fichiers avec tout le nécessaire pour faire fonctionner de préférence une seule application. Le container est le processus qui utilise l'image pour faire tourner l'application.

III UTILISATION DES CONTAINERS

Pour lancer votre image debian : jessie dans un container

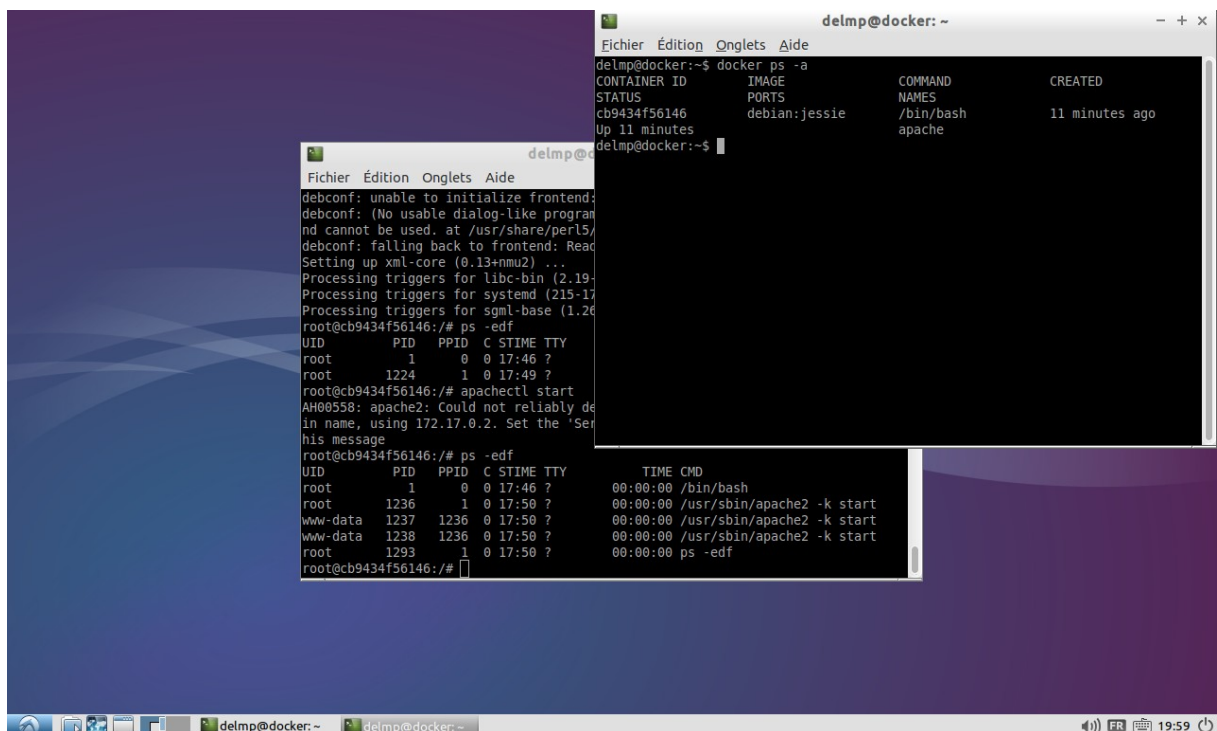
docker run --name apache -i -t debian /bin/bash (--name sera le nom de notre container, -i -t permettent d'avoir un shell accessible dans le container, debian:jessie est le nom de l'image à lancer, /bin/bash est le nom de la commande à lancer dans le container ce qui nous permet d'avoir un shell au sein du container.



```
delmp@docker: ~  
Fichier Édition Onglets Aide  
ed. 0 [OK]  
icedream/docker-debian-jenkinslave [Moved to icedream/debian-jenkinslave!] D  
... 0 [OK]  
konstruktoid/debian Debian base image  
idcu/debian mini debian os  
pl31/debian Basic debian image  
marioz/debian Debian Containers for PHP Projects  
visono/debian Docker base image of debian 7 with tools i  
... 0 [OK]  
delmp@docker:~$ docker pull debian:jessie  
Pulling repository debian  
6845b83c79fb: Download complete  
575489a51992: Download complete  
delmp@docker:~$ docker images  
REPOSITORY TAG IMAGE ID CREATED  
VIRTUAL SIZE  
debian jessie 6845b83c79fb 43 hours ago  
125.1 MB  
delmp@docker:~$ docker run --name apache -i -t debian:jessie /bin/bash  
root@cb9434f56146:/#
```

Le prompt a changé vous êtes maintenant dans votre container. Il faut maintenant installer apache2 et le lancer. Je vous laisse faire.

Laissez cette console ouverte, lancez en une deuxième et tapez la commande **docker ps -a** (cette commande vous donne la liste de tous les containers en cours d'exécution).



```
delmp@docker: ~  
Fichier Édition Onglets Aide  
delmp@docker:~$ docker ps -a  
CONTAINER ID IMAGE COMMAND CREATED  
STATUS PORTS NAMES  
cb9434f56146 debian:jessie /bin/bash 11 minutes ago  
Up 11 minutes  
delmp@docker:~$  
  
delmp@docker: ~  
Fichier Édition Onglets Aide  
debconf: unable to initialize frontend:  
debconf: (No usable dialog-like program  
nd cannot be used, at /usr/share/perl5/  
debconf: falling back to frontend: Read  
Setting up xml-core (0.13+mmu2) ...  
Processing triggers for libc-bin (2.19-  
Processing triggers for systemd (215-17  
Processing triggers for sgml-base (1.26  
root@cb9434f56146:/# ps -edf  
UID PID PPID C STIME TTY  
root 1 0 0 17:46 ?  
root 1224 1 0 17:49 ?  
root@cb9434f56146:/# apache2ctl start  
AH00558: apache2: Could not reliably de  
in name, using 172.17.0.2. Set the 'Ser  
his message  
root@cb9434f56146:/# ps -edf  
UID PID PPID C STIME TTY TIME CMD  
root 1 0 0 17:46 ? 00:00:00 /bin/bash  
root 1236 1 0 17:50 ? 00:00:00 /usr/sbin/apache2 -k start  
www-data 1237 1236 0 17:50 ? 00:00:00 /usr/sbin/apache2 -k start  
www-data 1238 1236 0 17:50 ? 00:00:00 /usr/sbin/apache2 -k start  
root 1293 1 0 17:50 ? 00:00:00 ps -edf  
root@cb9434f56146:/#
```

Il faut maintenant enregistrer les modifications faites à notre nouvelle image que nous appellerons **jessie_apache2**. Pour cela dans la deuxième console, il faut sauvegarder les changements de notre image par la commande « **docker commit** »

docker commit -m « TP cloud avec docker » -a « delamare » apache docker/jessie_apache2 :v1

-m pour mettre un commentaire, -a pour indiquer l'auteur, apache est le nom du container et docker/jessie_apache2 :v1 est le nom de la nouvelle image.

```

delmp@docker:~$ docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED
STATUS        PORTS     NAMES
cb9434f56146   debian:jessie /bin/bash               11 minutes ago
Up 11 minutes
delmp@docker:~$ docker commit -m "Tp cloud avec docker" -a "delamare" apache2 docker/jessie_apache2:v1
f609e9f489e3e80cd911265660671241eb1b339c1aa72600533ec4586837e029
delmp@docker:~$

```

```

root@cb9434f56146:/# ps -edf
UID        PID     PPID    C   STIME TTY          TIME CMD
root         1         0    0   17:46 ?        00:00:00 /bin/bash
root       1224         1    0   17:49 ?        00:00:00 /usr/sbin/apache2 -k start
root@cb9434f56146:/# ps -edf
UID        PID     PPID    C   STIME TTY          TIME CMD
root         1         0    0   17:46 ?        00:00:00 /bin/bash
root       1236         1    0   17:50 ?        00:00:00 /usr/sbin/apache2 -k start
www-data   1237     1236    0   17:50 ?        00:00:00 /usr/sbin/apache2 -k start
www-data   1238     1236    0   17:50 ?        00:00:00 /usr/sbin/apache2 -k start
root       1293         1    0   17:50 ?        00:00:00 ps -edf

```

Vérifiez avec la commande permettant de voir les images disponibles en local :

docker images

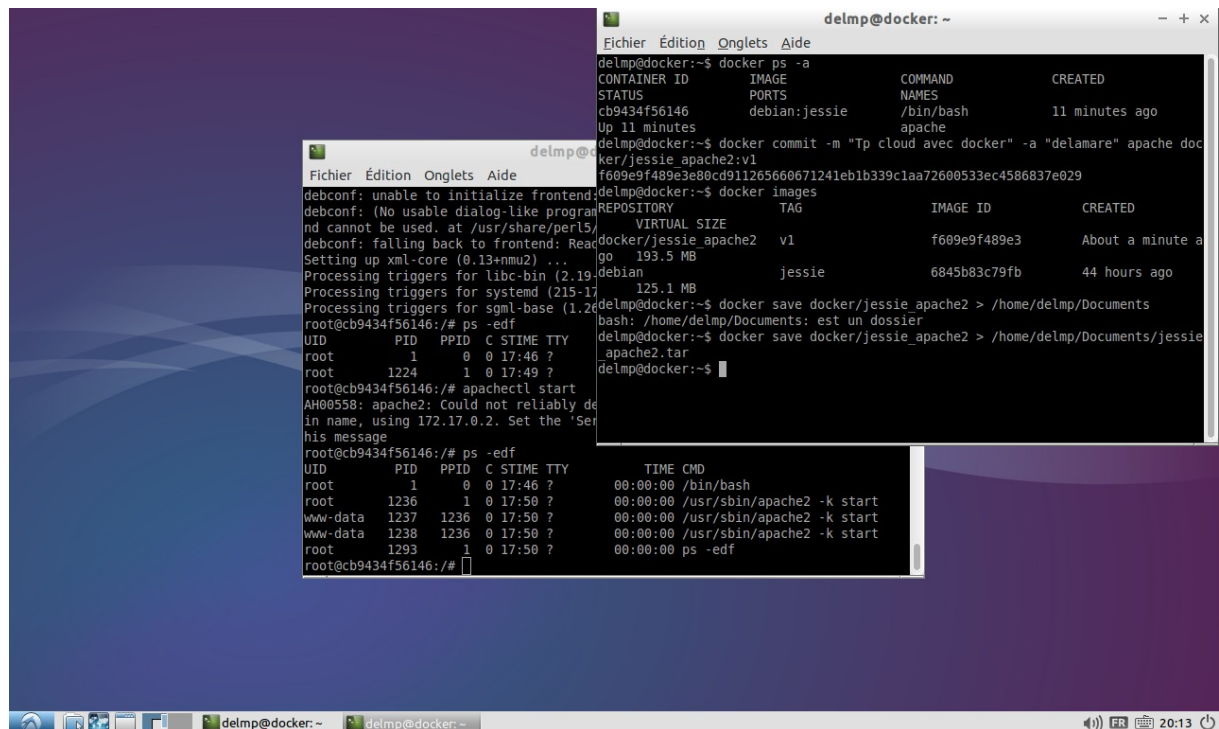
```

delmp@docker:~$ docker images
REPOSITORY      TAG              IMAGE ID          CREATED
docker/jessie   apache2          v1                f609e9f489e3     About a minute ago
go              193.5 MB
debian          jessie           6845b83c79fb     44 hours ago

```

Sauvegardez votre image pour pouvoir éventuellement l'utiliser sur une autre machine

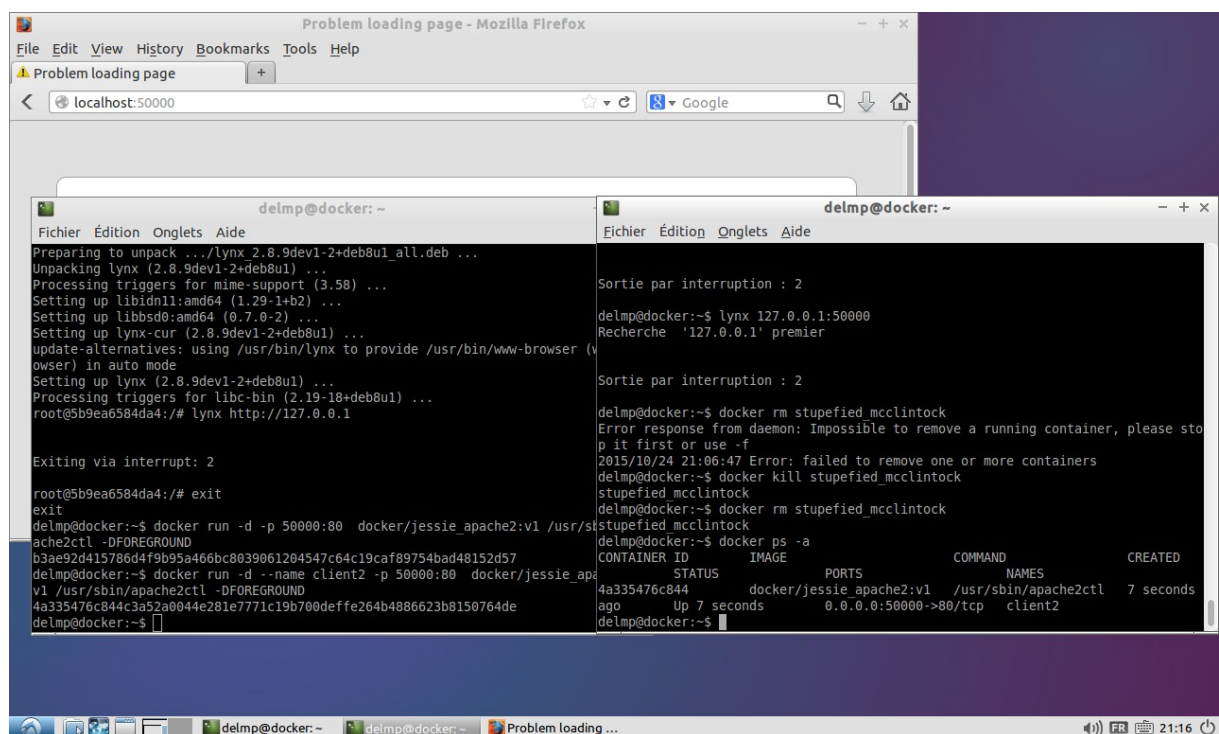
docker save docker/jessie_apache2 > jessie_apache2.tar (pour la réutiliser ailleurs vous utiliserez **docker load < jessie_apache2.tar**)



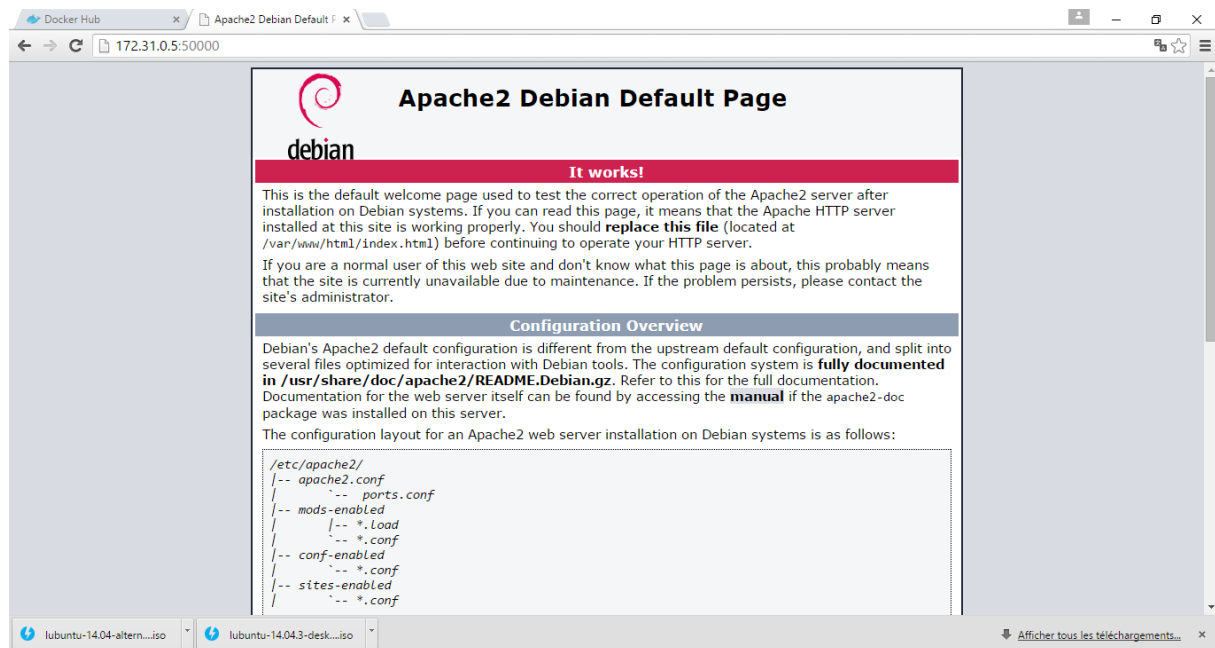
Sortez maintenant de votre container apache avec la commande **exit**. Lancez une commande **docker ps -a** pour vérifier qu'il est toujours présent. Détruisez-le avec la commande **docker rm apache** après l'avoir tué avec la commande **docker kill apache** si nécessaire.

Relancez-le avec la commande suivante :

docker run -d --name client2 -p 50000:80 jessie_apache2:v1 /usr/sbin/apache2ctl -DFOREGROUND (-d lance le container en arrière plan donc sans console, -p map le port 50000 vu de l'internaute vers le port 80 du container, /usr/sbin/apache2ctl est la commande à lancer dans le container avec le paramètre DFOREGROUND pour démarrer apache dans le container en premier plan).



Vérifiez avec un navigateur.



IV PUBLICATION DES SITES WEB DES CLIENTS

Pour ne pas modifier notre image et la laisser générique, il faut déposer les pages des clients dans des répertoires du serveur hôte. Il faut donc créer ces utilisateurs sur le serveur hôte, installer un serveur ftp sur le serveur hôte, « chrooter » les répertoires racines des utilisateurs ftp pour les placer dans leurs répertoires personnels, puis lancer un container par client en mappant le répertoire /var/www/html du container vers le répertoire du client sur le serveur hôte.

IV.1 INSTALLATION D'UN SERVEUR FTP SUR LE SERVEUR HÔTE

apt-get install vsftpd

Il faut modifier le fichier de configuration pour permettre aux utilisateurs linux de se connecter en ftp et d'être enfermés via la commande chroot dans leur répertoire personnel dans lesquels ils publieront leurs pages web.

Pour cela modifiez les paramètres suivants du fichier /etc/vsftpd.conf

```
local_enable=YES
write_enable=YES
local_umask=022
```

Pour limiter les utilisateurs à leur répertoire personnel, il faut créer un répertoire dédié à la configuration de vsftpd et y mettre le fichier liste.

```
sudo mkdir /etc/vsftpd
sudo nano /etc/vsftpd/chroot.list
```

Les identifiants des utilisateurs concernés doivent être renseignés dans /etc/vsftpd/chroot.list sous la forme d'une simple liste (il faudra penser à créer ces utilisateurs sous linux).

```
client1
client2
client3
...
```

Il faut ensuite modifier la configuration générale (/etc/vsftpd.conf) et décommenter voir corriger ces lignes:

```
chroot_local_user=YES
```



```
chroot_list_enable=YES  
chroot_list_file=/etc/vsftpd/chroot.list
```

Démarrez le service ftp avec la commande **sudo service vsftpd restart** et testez en local avec la commande **ftp**.

IV.2 DÉPOSER DES PAGES WEB DANS LES RÉPERTOIRES DES CLIENTS

Récupérez une page index.html, copiez la trois fois et changez le titre pour les différencier (client1, client2, client3) puis déposez-les avec un client ftp dans chacun des répertoires des clients (utilisateurs) sur votre serveur hôte.

IV.3 PUBLIER LES PAGES WEB DES CLIENTS DE VOTRE CLOUD

Il reste pour cela à lancer trois containers en les faisant pointer vers les répertoires de vos clients. Pour cela on va faire un mappage de répertoires entre les répertoires du serveur hôte et les répertoires des containers par la commande :

```
docker run -d --name client1 -p 49999:80 -v /home/client1:/var/www/html  
jessie_apache2:v1 /usr/sbin/apache2ctl -DFOREGROUND
```

```
docker run -d --name client2 -p 50000:80 -v /home/client2:/var/www/html  
jessie_apache2:v1 /usr/sbin/apache2ctl -DFOREGROUND
```

```
docker run -d --name client3 -p 50001:80 -v /home/client3:/var/www/html  
jessie_apache2:v1 /usr/sbin/apache2ctl -DFOREGROUND
```

Testez avec un navigateur.

Voilà votre Cloud est prêt.

Il faudrait ensuite intégrer cela dans Haproxy pour rediriger en fonction de l'URL demandée vers le serveur hôte sur un port précis (backend vers le serveur hôte et le port redirigé lui permettrait de trouver le bon container au sein de ce serveur hôte).

Comme cela vos clients interrogeraient leur site sur le port 80 mais avec leur nom de domaine et Haproxy renverrait vers le backend adéquate en fonction de l'URL demandée.

Annexe 1 : Quelques commandes bien utiles

GESTION DES IMAGES :

Détruire une image : **docker rmi nom_image:tag**

Publier une image sur le dépôt : **docker login** : se connecter avec son compte au dépôt

docker push nom_image

GESTION DES CONTAINERS :

Reprendre la main sur un container quitté par exit : **docker restart nom_container**

docker attach nom_container