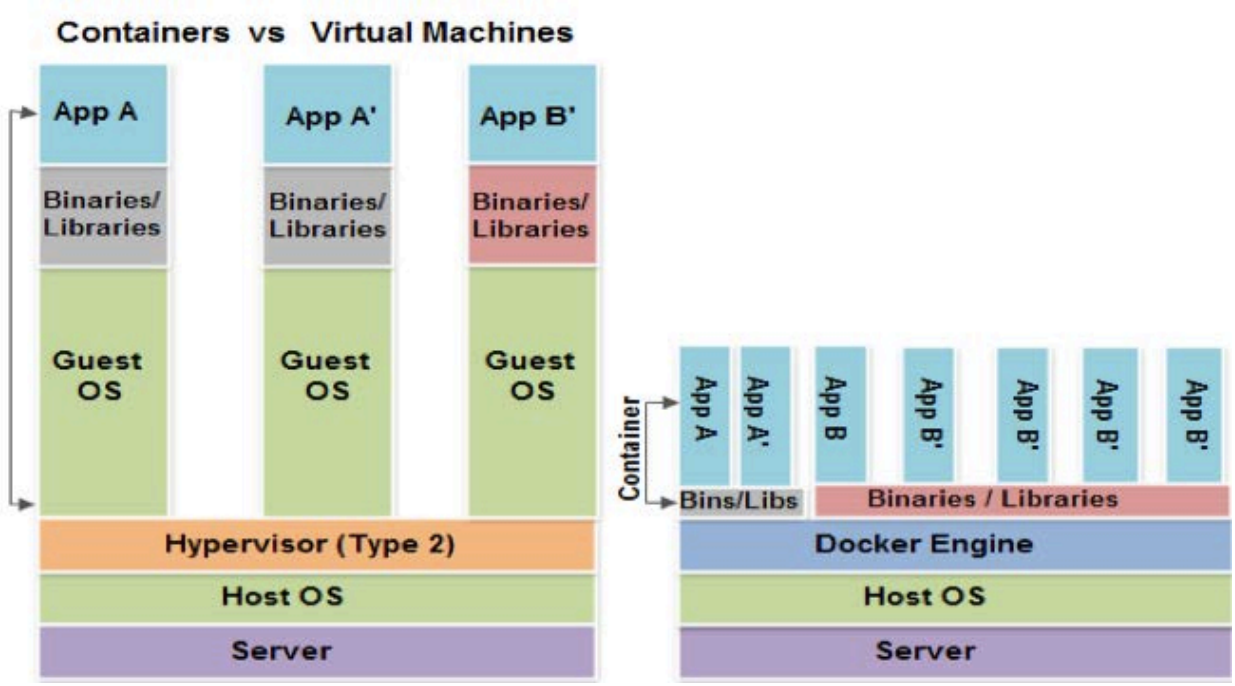
TP : Containers Docker




```
root@osboxes:~# apt-get update
root@osboxes:~# apt-get install apt-transport-https ca-certificates
root@osboxes:~# apt-key adv --keyserver hkp://p80.pool.sks-
keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D
……..
gpg: requesting key 2C52609D from hkp server p80.pool.sks-keyservers.net
gpg: key 2C52609D: public key "Docker Release Tool (releasedocker)
<docker@docker.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

Il faut ensuite éditer:**/etc/apt/sources.list.d/docker.list** (ajouter la ligne suivante) :
deb https://apt.dockerproject.org/repo debian-jessie main

```
root@osboxes:~# apt-get update
root@osboxes:~# apt-get install docker-engine
 ...
root@osboxes:~# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c04b14da8d14: Pull complete
Digest:
sha256:0256e8a36e2070f7bf2d0b0763dbabdd67798512411de4cdcf9431a1feb60fd9
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working
correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
```

```
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
 3. The Docker daemon created a new container from that image which runs
the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which
sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker Hub
account:
 https://hub.docker.com
For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

Nous allons maintenant télécharger un container plus intéressant que hello-world : le container officiel ubuntu.
Allez sur le hub docker **https://hub.docker.com/** en cliquant sur Explore en haut à gauche et trouvez les images officielles d'Ubuntu, debian, mysql, apache, etc.

```
root@osboxes:~# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
af49a5ceb2a5: Pull complete
8f9757b472e7: Pull complete
e931b117db38: Pull complete
47b5e16c0811: Pull complete
9332eaf1a55b: Pull complete
Digest:
sha256:3b64c309deae7ab0f7dbdd42b6b326261ccd6261da5d88396439353162703fb5
Status: Downloaded newer image for ubuntu:latest
```

On va lister les images disponibles :

```
root@osboxes:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
ubuntu              latest              4ca3a192ff2a        8 days ago
128.2 MB
hello-world         latest              c54a2cc56cbb        5 months ago
1.848 kB
```

Démarrons notre premier container ubuntu.

```
root@osboxes:~# docker run ubuntu
```

Un container ne reste en vie que si un processus est actif. On peut lister les containers actifsavec la commande docker ps. On peut aussi lister tous les containers, actifs ou inactifs avec docker ps -a.

```
root@osboxes:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
root@osboxes:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS                      PORTS               NAMES
48564eb0d27c        ubuntu              "/bin/bash"         21 seconds
ago      Exited (0) 20 seconds ago                       clever_minsky
cb96c775cc8d        hello-world         "/hello"            9 minutes ago
Exited (0) 9 minutes ago                        naughty_allen
```

Nous allons maintenant rediriger l'entrée standard du container avec l'option -i et ouvrir un pseudo-terminal avec -t, le tout en exécutant le processus /bin/bash

```
root@osboxes:~# docker run -ti --name=ubuntu ubuntu /bin/bash
root@7bd8427382bb:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@7bd8427382bb:/# exit

root@osboxes:~# docker stop ubuntu
ubuntu
root@osboxes:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
root@osboxes:~# docker start ubuntu
ubuntu
root@osboxes:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
7bd8427382bb        ubuntu              "/bin/bash"         6 minutes ago
Up 2 seconds                            ubuntu
root@osboxes:~# docker attach Ubuntu #( parfois il faut appuyer sur une touche du clavier pour avoir le prompt)
root@7bd8427382bb:/# exit
exit
```

Pour tuer notre container :

```
root@osboxes:~# docker rm ubuntu
ubuntu

root@osboxes:~# docker run -ti --name=toto ubuntu /bin/bash
```

On peut inspecter ce qui se passe dans le container depuis la machine hôte avec la commande, essayer sur un autre terminal de taper :

```
root@osboxes:~# docker stats toto (sur un autre terminal)
root@osboxes:~# docker logs -f toto (sur un autre terminal)
root@osboxes:~# docker stats toto (sur un autre terminal)
```

Créons une machine Debian :

```
root@osboxes:~# docker pull debian
Using default tag: latest
latest: Pulling from library/debian
```

```
386a066cd84a: Pull complete
Digest:
sha256:c1ce85a0f7126a3b5cbf7c57676b01b37c755b9ff9e2f39ca88181c02b985724
Status: Downloaded newer image for debian:latest
```

Nous allons maintenant créer notre propre image qui va nous permettre de lancer un serveur Web apache. Pour cela, il faut définir la méthode de construction du container dans un fichier.

```
root@osboxes:~# mkdir -p  Docker/Apache
root@osboxes:~# cd Docker/Apache
root@osboxes:~# nano Dockerfile
```

```
FROM debian:latest
   MAINTAINER GRASSA
   RUN apt-get -yqq update && apt-get install -yqq apache2

   WORKDIR /var/www/html

   ENV APACHE_RUN_USER www-data
   ENV APACHE_RUN_GROUP www-data
   ENV APACHE_LOG_DIR /var/log/apache2
   ENV APACHE_PID_FILE /var/run/apache2.pid
   ENV APACHE_RUN_DIR /var/run/apache2
   ENV APACHE_LOCK_DIR /var/lock/apache2

   RUN mkdir -p $APACHE_RUN_DIR $APACHE_LOCK_DIR $APACHE_LOG_DIR

   ENTRYPOINT [ "/usr/sbin/apache2" ]
   CMD ["-D", "FOREGROUND"]
   EXPOSE 80
```

root@osboxes:~# **docker build -t="grassa/apache" .** #Attention à ne pas oublier le point '.' à la fin de la commande (qui indique que le fichier Dockerfile est dans le répertoire local)

```
Sending build context to Docker daemon 6.656 kB
Step 1 : FROM debian:latest
 ---> 73e72bf822ca
Step 2 : MAINTAINER GRASSA
 ---> Running in 81418305032e
 ---> 7c45588e5b07
Removing intermediate container 81418305032e
Step 3 : RUN apt-get -yqq update && apt-get install -yqq apache2
 ---> Running in f1f2126f8f33
debconf: delaying package configuration, since apt-utils is not installed
……
Successfully built cd5f19931229
```

```
root@osboxes:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
grassa/apache       latest              cd5f19931229        29 seconds
ago      191.5 MB
ubuntu              latest              4ca3a192ff2a        8 days ago
128.2 MB
```

```
debian              latest              73e72bf822ca        4 weeks ago
123 MB
hello-world         latest              c54a2cc56cbb        5 months ago
1.848 kB
```

```
root@osboxes:~# docker run -d --name=demon ubuntu /bin/bash
cf6f49b555f0982adfaefac363685721772ec651462edb0df4ca9a173f418928
```

Que se passe-t-il ? On va faire un **docker ps** pour comprendre

```
root@osboxes:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
root@osboxes:~# docker rm –f demon
```

Nous allons corriger le problème précédent en faisant quelque chose dans le container

```
root@osboxes:~# docker run –d –name=demon debian sh -c 'while true;do
echo "hello Nordine";sleep 1;done'
```

```
root@osboxes:~# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS               NAMES
94f61cf9c0ba        debian              "sh -c 'while true;do"   7
seconds ago       Up 6 seconds                                 zen_yalow
```

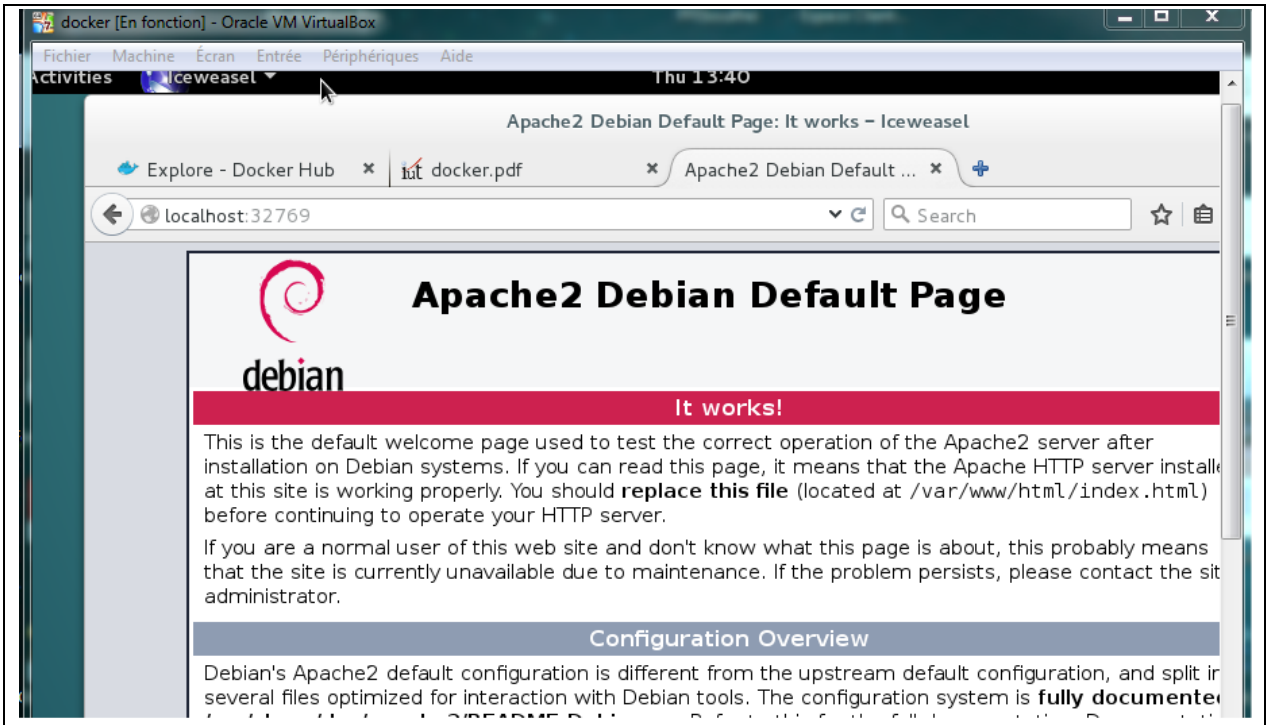
Démarrons un container **grassa/apache** en mode démon en exposant le port 80

```
root@osboxes:~# docker run -d -p 80 --name=apache grassa/apache
5b9318028a16ce59b3d48e631d4160684fbb90416d8e257d8c86500ba4f84c0e
root@osboxes:~# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                   NAMES
5b9318028a16        grassa/apache       "/usr/sbin/apache2 -D"   5
seconds ago       Up 3 seconds        0.0.0.0:32769->80/tcp   apache
94f61cf9c0ba        debian              "sh -c 'while true;do"   7
minutes ago       Up 7 minutes                                 zen_yalow
```

Pour trouver quel est le port choisi par Docker :

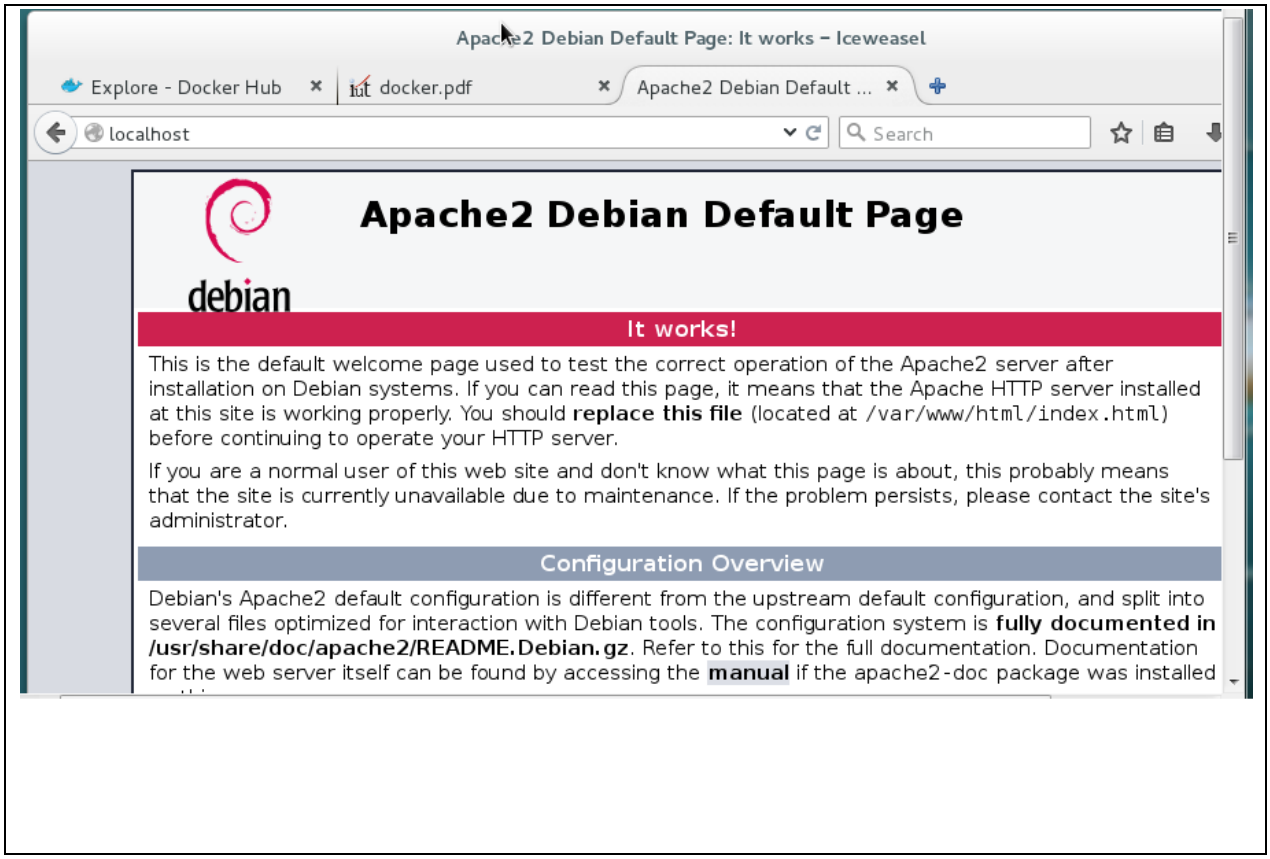
```
root@osboxes:~# docker port apache 80
0.0.0.0:32769
```

Faisons un test en ouvrant votre navigateur et en se connectant sur notre machine locale (Hôte) et sur le port docker.



On peut mieux contrôler le port choisi. Si par exemple on veut que cela soit le port 80, il suffit de modifier la commande précédente :

```
root@osboxes:~# docker rm apache
root@osboxes:~# docker run -d -p 80:80 --name=apache grassa/apache
077f87115365cd3aee583bcc0d665b587cce418724d4f13bd6a9e5ad622b0b37
root@osboxes:~# docker port apache 80
0.0.0.0:80
```



On veut maintenant en plus contrôler le contenu du serveur Web depuis l'hôte. Pour cela on va :
(a) créer un répertoire **website** dans le répertoire **Apache** crée précédemment.
(b) mettre dans ce répertoire un fichier **index.html** avec le contenu suivant :

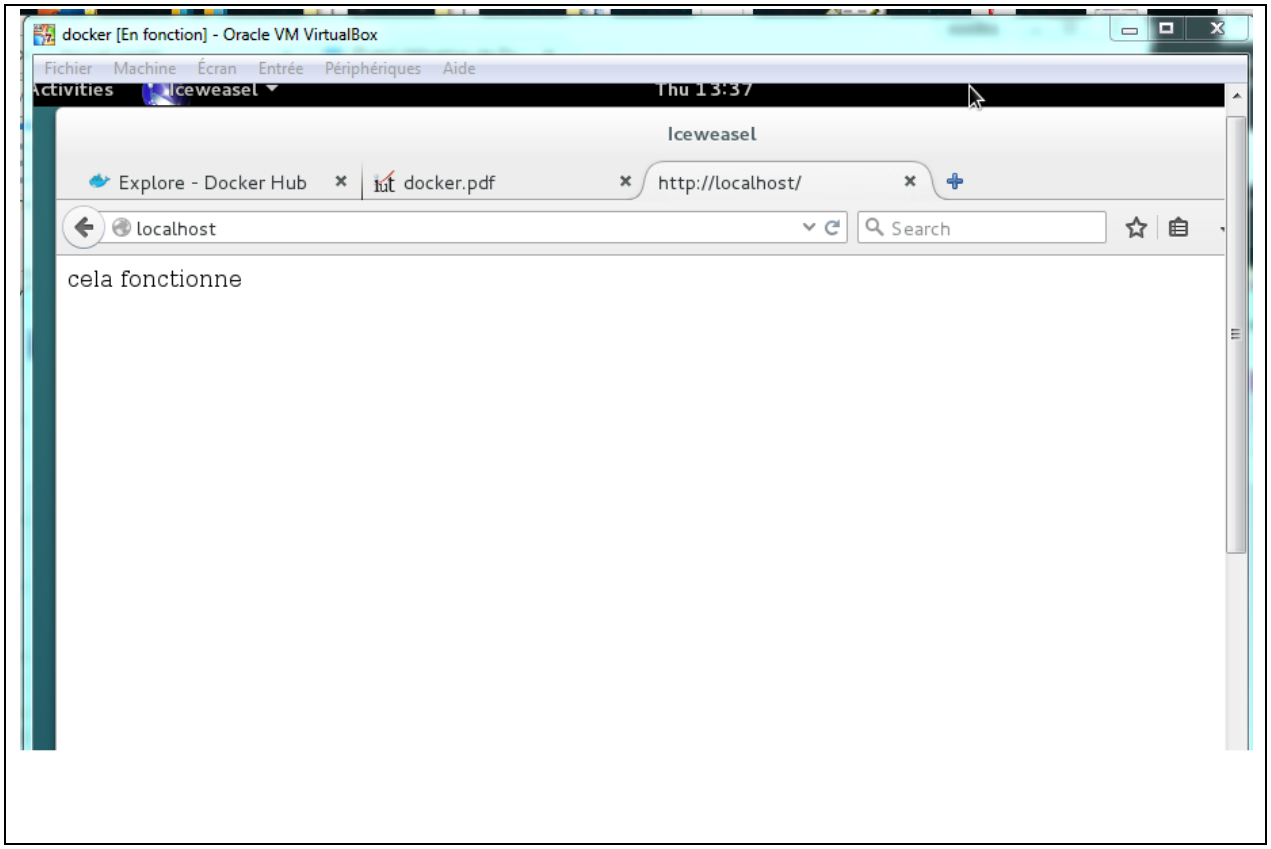
```
root@osboxes:~/Docker/Apache# mkdir website
root@osboxes:~/Docker/Apache# cd website/
root@osboxes:~/Docker/Apache/website# ls
root@osboxes:~/Docker/Apache/website# nano index.html
<html>
cela fonctionne
</html>
root@osboxes:~# docker rm  -f apache
apache
```

Démarrons votre container en montant le répertoire apache la ou le serveur Apache dans le container va chercher ses données :

```
root@osboxes:~# docker run -d -p 80:80 –v
~/Docker/Apache/website:/var/www/html --name=apache grassa/apache
ff4e790c724e7a39efe1497655b1c4433d9bd8429e13da21680e699cb8914b11
```

Pour trouver l'adresse ip du container **grassa/apache** voici la commande à taper.
On peut accéder à la page web via son adresse ip aussi.

```
root@osboxes:~# docker inspect –format="{{range
.NetworkSetting.Networks}}{{.IPAddress}}{{end}}" 5b9318028a16
```



REFERENCE :

M2102 - TP 6: Containers Docker V0.1 { Guillaume Urvoy-Keller }

https://docs.docker.com/engine/installation/linux/debian/

https://mondedie.fr/d/7164-Tuto-Utilisation-de-Docker/7