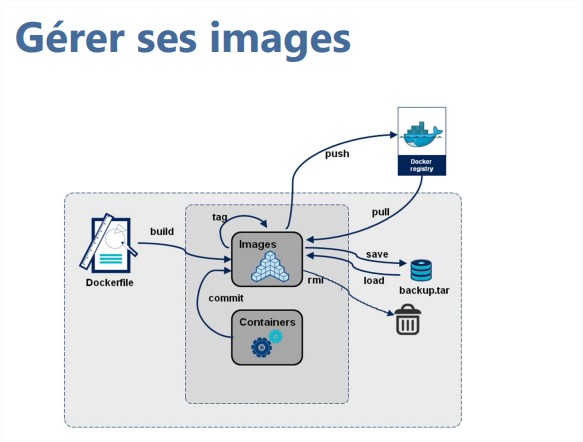
# Help

De l’aide sur l’utilisation ou les options d’une commande docker :

root@docker:~# docker <commande> --help

# Fonctionnement



# Les images Docker

Voir les commandes pour l’administration des images :

root@docker:~# docker | grep --color image

Rechercher une image (on peut savoir si c’est une officielle ou pas) sur le dockerhub :

root@docker:~# docker search debian

--filter "is-official=true" : Rechercher que les images officielles

Récupérer une image

root@docker:~# docker pull debian

Vérifier les images disponibles :

root@docker:~# docker images

Supprimer une image (option - -force pour forcer la suppression) :

root@docker:~# docker rmi <imageID>

Sauvegarde une image :

root@docker:~# docker save <imageID> -o /tmp/debian.tar

Charger une image depuis un tar :

root@docker:~# docker load -i /tmp/debian.tar

# Docker registry (à voir) ?

# Les conteneurs

Créer un conteneur :

root@docker:~# docker run -ti debian

-i : interagir avec le conteneur

-t : garder le même terminal ouvert

--name : nom du conteneur

--rm : Supprime le conteneur après l’avoir quitter

Visualiser des conteneurs :

root@docker:~# docker ps -a

-a : afficher tous les conteneurs

Raccourcis pour sortir du conteneur sans le détruire :

**Ctrl + P + Q**

Exécuter une commande dans un conteneur :

root@docker:~# docker exec <conteneurID> /bin/bash -c 'ls /var/www/html'

Astuce se connecter sur le terminal :

root@docker:~# docker exec -ti <conteneurID> /bin/bash

Retourner à un conteneur déjà crée :

root@docker:~# docker attach <conteneurID>

Stopper et supprimer un conteneur :

root@docker:~# docker stop <conteneurID>

root@docker:~# docker rm <conteneurID>

Démarrer un conteneur (de 0) qui a été stop :

root@docker:~# docker start <conteneurID>

Mettre en pause et récupérer un conteneur :

root@docker:~# docker pause <conteneurID>

root@docker:~# docker unpause <conteneurID>

Commiter un conteneur (passer d’un conteneur à une image docker) :

root@docker:~# docker commit <conteneurID> <imageNAME>

## Mappage de ports :

Mapper le port 80 du conteneur en port 8080 :

root@docker:~# docker run --name apache\_kinos -d -p 8080:80 httpd

-d : Exécute le conteneur en arrière-plan et imprimer l'ID du conteneur

<http://IP:8080>

Création de ma propre image + conteneur :

# Dockerfile (voir un tuto + poussé) :

FROM debian:stable-slim

ARG VERSION="1.0.3"

ARG APT\_FLAGS="-q -y"

ENV LANG="en\_US.UTF-8"

LABEL maintainer="NOM Prenom <test@gmail.com>" \

version="$VERSION-debian-testing" \

description="ROS2 for java" \

# il y'a d'autres option pour le maintainer

RUN apt-get update && \

apt-get install ${APT\_FLAGS} \

curl \

wget \

apt-utils \

apache2 \

&& apt-get autoclean

COPY index.html /var/www/html

EXPOSE 80

EXPOSE 443

# La prochaine commande utilisera ce répertoire de travail.

WORKDIR /etc/apache

CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]

* **FROM** permet de définir depuis quelle base votre image ve être créée
* **MAINTAINER** indique la personne qui a créé ou maintient ce Dockerfile
* **ARG** : variable qu'on peut utiliser dans un Dockerfile
* **RUN** permet d'exécuter une commande lors du build (création) de l'image. Chaque instruction RUN va créer un Layer qui sera utilisé dans le cas de modification ultérieur de votre Dockerfile.
* **COPY** permet de copier des fichiers depuis la machine de contrôle ou une url vers la machine hôte
* **EXPOSE** permet d'exposer un port du container vers l'extérieur
* **CMD** détermine la commande qui sera exécutée lorsque le container démarrera cette clé est unique et s'il y en a plusieurs seule la dernière sera utilisée. Cette option peut être surchargé par la commande docker run.
* **ENTRYPOINT** permet d'ajouter une commande qui sera exécutée lors de la création du conteneur. Contrairement à CMD ENTRYPOINT ne peut pas être surchargé par la commande “docker run”.
* **WORKDIR** : définit le répertoire de travail qui sera utilisé pour le lancement des commandes ENTRYPOINT et/ou CMD
* **ENV** permet de définir des variables d'environnements dans notre conteneur (taper la commande **env** dans le conteneur pour la voir) mais elles peuvent aussi être utilisé dans le Dockerfile
* **VOLUMES** permet de créer un point de montage qui permettra de persister les données. On pourra alors choisir de monter ce volume dans un dossier spécifique en utilisant la commande `run -v :
* **USER :** désigne quel utilisateur lancera les prochaines instructions RUN ou ENTRYPOINT

Création de l’image depuis le dockerfile :

root@docker:~# docker build -t hatim\_apache .

Création du conteneur :

root@docker:~# docker run --name web\_server\_hatim -d -p 8088:80 hatim\_apache

# Les volumes

## Mappage des volumes :

Les données dans les containeurs sont éphémères, une des solutions serait de mapper un dossier(volume) d’un conteneur vers un dossier(volume) de notre machine local (voir un SAN)

root@docker:~# docker run -d --name < conteneurNAME > -p 8080:80 -v ~/data/:/usr/share/nginx/html/ nginx

Sur mon dossier en local **/data** j’ai un fichier **index.html** et chaque changement de ce fichier affectera les containeurs qui map ce volume.

# Les links :

Permet de lier plusieurs conteneurs entre eux, ici par exemple le linkage entre wordpress + mysql.

Création du conteneur mysql

root@docker:~# docker run --name db -d -e MYSQL\_ROOT\_PASSWORD=password -d mysql

Création du conteneur wordpress + linkage :

root@docker:~# docker run --name <conteneurNAME> --link db db -p 8080:80 -d wordpress

--link : Ajouter un lien à un autre conteneur

# Debuguer :

Avoir des informations sur un conteneur ou une image :

root@docker:~# docker inspect <conteneurID> /<imageID>

Filtrer l’output de la commande inspect : <https://docs.docker.com/engine/reference/commandline/inspect/#get-an-instances-mac-address>

Afficher en live les ressources consommées par le conteneur :

root@docker:~# docker stats <conteneurID>

# Docker compose (voir tuto) :

Vérifier et valider notre config :

root@docker:~# docker-compose config

# Docker-Machine



Installation : <https://docs.docker.com/machine/install-machine/#where-to-go-next>

**Condition pour créer une docker-Machine :**

* Activer la virtualisation sur son CPU
* Si VirtualBox alors avoir VirtualBox de préinstallé même chose pour Hyper-V VMWare etc. …

Les drivers permettent de créer une docker-machine, voici la liste et la doc des différents drivers disponible :

<https://docs.docker.com/machine/drivers/>

Création d’un hôte docker VirtualBox :

root@docker:~# docker-machine create -d "virtualbox" <hostNAME>

--virtualbox-disk-size "10000"

--virtualbox-memory "512"

Supprimer un hôte :

root@docker:~# docker-machine rm <hostNAME>

Lister les hôtes avec certaines informations :

root@docker:~# docker-machine ls

Démarrer/stopper un hôte :

root@docker:~# docker-machine start/stop <hostNAME>

Avoir des informations sur un hôte :

root@docker:~# docker-machine inspect <hostNAME>

Upgrader le moteur de DOCKER sur l’hôte (stop la machine temporairement) :

root@docker:~# docker-machine upgrade <hostNAME>

Se connecter à un hôte :

Façon 1 :

* Etape 1 (Récupérer la commande pour se connecter à la machine) :

root@docker:~# docker-machine env <hostNAME>

* Etape 2 (Runner la commande afficher sur l’output)

Façon 2 :

root@docker:~# docker-machine ssh <hostNAME>

Coper des fichiers depuis des machines :

root@docker:~# docker-machine scp <path> <hostNAME>:<path>

root@docker:~# docker-machine scp <hostNAME>:<path> <path>

**Une fois qu’on est connecté sur la machine alors nous avons déjà le docker-engine installé ou on peut manipuler des images et des conteneurs.**

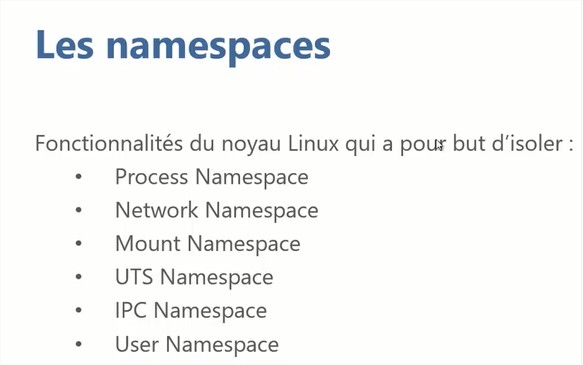
# Culture G :

Docker utilise le mode bridge mode (**docker0**) pour attribuer des ips et Patter (translation de port) à ses conteneurs.

Pour vérifier la configuration réseau de mon bridge docker il suffit de taper la commande :

root@docker:~# docker network inspect bridge

documentation network : <https://docs.docker.com/v17.09/engine/userguide/networking/#bridge-networks>





<https://www.yggtorrent.is/torrent/application/formation/168423-alphorm+lxc+-+maitrisez+la+virtualisation+par+conteneurs+2015+french+mp4>