Les Commandes Docker

Pour afficher l'aide
$ **docker** help

Pour afficher la version de docker installée
$ **docker** –version

Pour afficher des informations sur le fonctionnement en cours de docker
$ **docker** info

Pour lister les images sur la machine
$ **docker** image ls
Ou
$ **docker** images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
Le titre REPOSITORY peut porter à confusion, c'est essentiellement le nom de l'image.	un tag ici est une façon de faire référence à votre image, ils sont utilisés principalement pour affecter une version à une image	L'identifiant de l'image (unique pour chaque image téléchargée)	Date de la dernière modification de l'image	Taille de l'image

Pour supprimer une image sur la machine
$ **docker** rmi <*nom_image*>
Ou
$ **docker** rmi <*id_image*>
Ou
$ **docker** rmi -f <*nom_image*> // Force la suppression de l'image et tous les conteneurs qui l'utilisent
Ou
$ **docker** rmi -f $(docker images -q) // Supprime TOUTES les images disponibles sur la machine

Pour rechercher une image sur le hub docker
$ **docker** search <*nom_image*>
Filtrer les images officielles
$ **docker** search ubuntu --filter "is-official=true" // Afficher que les images officielles

Pour télécharger une image à partir du hub
$ **docker** pull <*nom_image*>
Exemple
$ **docker** pull ubuntu
Avec un tag
$ **docker** pull ubuntu:16.04
Dernière version
$ **docker** pull ubuntu:latest

Pour exécuter une image
$ **docker** run [OPTIONS] <*nom_image ou ID*>

liste les conteneurs actifs
$ **docker** ps
ou
$ **docker** container ls

liste les conteneurs même à l'arrêt //all
$ **docker** ps -a
ou
$ **docker** container ls -a

Démarrer un conteneur
$ **docker** start <*nom conteneur*>

Créer un conteneur
$ **docker** run [OPTIONS] <*nom_image*> // **docker** run -it --name=container_web_server debian bash

*** **Quelques options** ***
-t : Allouer un terminal tty (terminal virtuel)
-i : Garder un STDIN ouvert (l'entrée standard, plus précisément l'entrée clavier)
-d : Exécuter le conteneur en arrière-plan
-p : Exposer un ou plusieurs ports (mapping)
--name : donner un nom au container
--expose : Exposer un port ou une plage de ports (on demande au firewall du conteneur de nous ouvrir un port ou une plage de port)
-p ou **–publish** : Mapper un port déjà exposé, plus simplement ça permet de faire une redirection de port
--nework : Pour connecter un conteneur au moment de la création

*** **Options** Linux***
-y => Pour accepter automatiquement les questions.

Pour inspecter un conteneur
$ docker inspect <nom_conteneur>

Pour stopper un conteneur Actif
$ **docker** stop <*container_name* ou *ID*>

Pour supprimer un conteneur :
$ **docker** rm <*container_id*> ou <*container_name*>

Pour exécuter une commande dans un conteneur
$ **docker** exec [OPTIONS] <*container_id* ou *container_name*> command

Avoir les logs d'un conteneur
$ **docker** logs -ft <*container_id*> ou <*container_name*>

Pour convertir un conteneur en Image
$ **docker** commit <*container_id* ou *container_name*> <*image_name*>

Pour créer un volume, on utilise la commande
$ **docker** volume create <*nom_volume*>

Création d'une image à partir d'un Dockerfile
$ **docker** build -t <*nom_image*> . // Ne pas oublier le point

Pour lister des volumes
$ **docker** volume ls

Avoir les infos sur un volume
$ **docker** volume inspect <nom_volume>

Pour supprimer un volume
$ **docker** volume rm <*nom_volume*>

Créer un conteneur avec volume, on utilise l'option -v
$ **docker** run –v <*nom_volume*>**:**<*dossier_conteneur*>

Monter un volume local sur un conteneur
$ **docker** run –v <*volume_local*>:<*volume_conteneur*>

Supprimer un conteneur Docker avec le/les volumes associés
$ **docker** rm -v <CONTAINER_ID ou CONTAINER_NAME>
-f ou --force : forcer la suppression
-v ou --volume : supprime les volumes associés au conteneur

Pour créer un réseau docker
$ **docker** network create --driver <*DRIVER TYPE*> <*NETWORK NAME*>

Pour lister les réseaux docker
$ **docker** network ls

Pour inspecter un réseau
$ **docker** network inspect <*network_name*>

Exécuter une image avec mapping port
$ docker run -d –name <*nom_conteneur*> -p 8080 :80 <*image_source*>

Récupérer les adresse ip des conteneurs
$ docker inspect -f '{{.Name}} - {{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $(docker ps -aq)

###
DOCKER-COMPOSE
###

# Démarrer notre application multi-conteneurs
$ docker-compose up –d

# Lister l'ensemble des conteneurs de notre app
$ docker-compose ps

# Voir les logs
$ docker-compose logs

# Tuer l'ensemble des conteneurs
$ docker-compose kill

# Stopper l'ensemble des conteneurs de notre app
$ docker-compose stop // -t pour un timeout

# Redémarrer l'ensemble des conteneurs de notre app
$ docker-compose start

# Arrêtez les conteneurs et supprimer les conteneurs, réseaux, volumes, et les images
$ docker-compose down

# Supprimer les conteneurs stoppés du docker-compose
$ docker-compose rm

# Lister les images utilisées dans le docker-compose
$ docker-compose images