

DOCKER ET DOCKER COMPOSE – RÉCAPITULATIF DES COMMANDES DE BASE

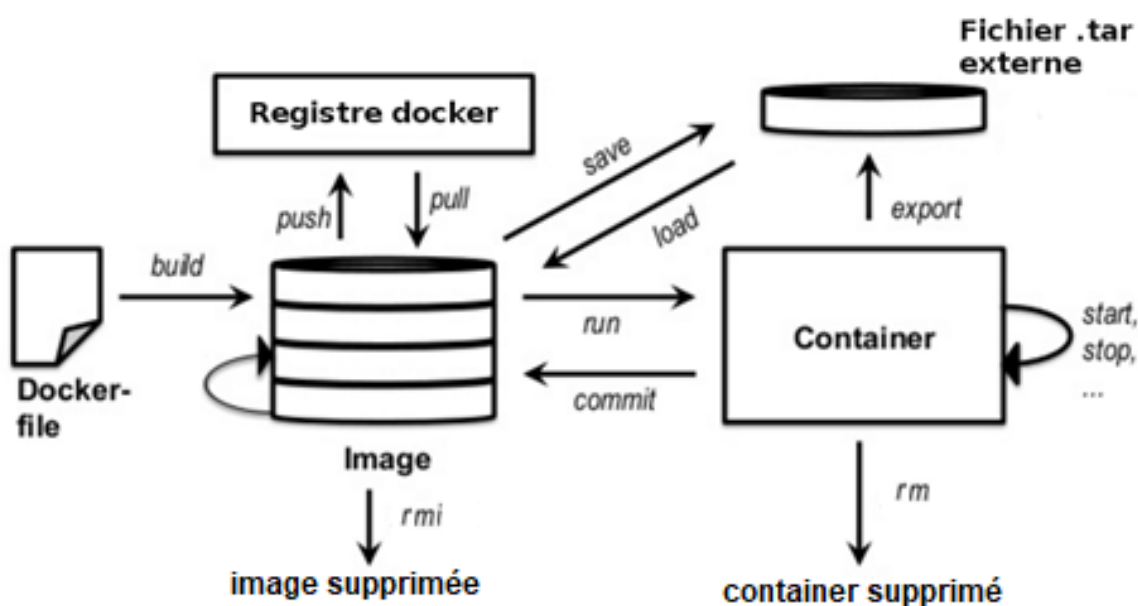
TABLE DES MATIÈRES

A. Récapitulatif des commandes de Docker sous forme de schéma.....	1
B. Gestion des images.....	2
C. Gestion des conteneurs.....	3
D. Gestion des volumes.....	4
E. Gestion du réseau.....	5
F. Nettoyage de Docker.....	6

A. RÉCAPITULATIF DES COMMANDES DE DOCKER SOUS FORME DE SCHÉMA

Un ensemble de commandes existe pour effectuer toutes les manipulations sur les conteneurs. La commande docker --help permet de voir les nombreuses commandes disponibles.

Les commandes Docker



Une image de conteneur peut être téléchargée depuis un serveur appelé registre Docker (pull). Le conteneur, créé à partir de l'image est ensuite lancé sur la machine hôte (run), rendant ainsi l'application contenue dans le conteneur disponible aux utilisateurs.

Toutes les commandes Docker commencent par le mot docker, suivi du nom de la commande, puis des paramètres et enfin du nom du conteneur.

Pour obtenir des informations sur le système Docker installé

`docker info`



Dans ce qui suit, le mot clé <conteneur> représente tout aussi bien l'ID du conteneur que son nom.

B. GESTION DES IMAGES

Chercher une image sur le Hub officiel

```
docker search <mot_clé>
```

Télécharger une image depuis le Hub officiel

```
docker pull <nom_image>
```

Le nom de l'image peut comporter un tag comme debian:bookworm. Si le tag est omis, il prend automatiquement la valeur « latest ».

Le nom de l'image, comme reseaucerta/ubuntu:ssh, est préfixé par un nom qui correspond au compte sur le Docker Hub si cette dernière n'est pas créée par l'équipe de Docker.

Envoyer une image sur le Hub public

```
docker push <nom_image>
```



Il est nécessaire d'avoir un compte sur le Docker Hub et que le préfixe de l'image corresponde au login créé.

Lister les images disponibles

```
docker images
```

Créer une nouvelle image à partir d'un conteneur

```
docker commit <conteneur> <nom_image>
```

Sauvegarder une image dans un fichier « tar »

```
docker save <nom_image> > <nom_fichier.tar>
```

Restaurer une image depuis un fichier sauvegardé

```
docker load -i <nom_fichier.tar>
```

Afficher la séquence de commandes qui ont été utilisées pour construire l'image

```
docker history <nom_image>
```

Supprimer une image

```
docker rmi <nom_image>
```

```
docker image rm <nom_image>
```

Docker crée des images automatiquement en lisant les instructions d'un Dockerfile qui est un fichier texte contenant toutes les instructions, dans l'ordre, nécessaires pour créer une image donnée.

Construire une image

```
docker build -t <nom_image>:<tag> <dossier_dockerfile>
```

Voir le récapitulatif des principales instructions dans un Dockerfile dans le fichier de documentation correspondant.

C. GESTION DES CONTENEURS

Lancer un conteneur

```
docker run [OPTIONS] <nom image> [COMMANDE] [ARG]
```

Quelques paramètres de la commande run utilisés dans les activités :

- t : fourni un terminal au docker.
- i : permet d'écrire dans le conteneur (couplé à -t).
- d : exécute le conteneur en arrière plan.
- p : permet de mapper un port sur le conteneur vers un port sur l'hôte
- name : donne un nom au conteneur
- rm : supprime un conteneur dès qu'il a rempli sa fonction (utile, par exemple, si le conteneur a pour seule vocation de lancer une commande)

Créer un conteneur sans le démarrer

```
docker create [OPTIONS] <nom image> [COMMANDE] [ARG]
```

Lister les conteneurs démarrés

```
docker ps
```

- a pour afficher tous les conteneurs (même ceux qui ne sont pas actifs)
- q pour n'afficher que les « id »

Arrêter un conteneur

```
docker stop <conteneur> ou docker kill <conteneur>
```

Réactiver un conteneur et y accéder en interactif :

```
docker start <conteneur>
docker attach <conteneur>
```

Supprimer un conteneur

```
docker rm <conteneur>
```

Le conteneur doit avoir été stoppé sinon il faut utiliser l'option « -f » : `docker rm -f <conteneur>` :

Supprimer rapidement tous les conteneurs actifs

```
docker rm -f $(docker ps -aq)
```

Mettre en pause un conteneur

```
docker pause <conteneur>
```

Sortir de la pause un conteneur

```
docker unpause <conteneur>
```

Afficher les processus en cours d'un conteneur

```
docker top <conteneur>
```

Afficher les statistiques d'un ou des conteneurs (CPU, mémoire, etc)

```
docker stats [<conteneur>]
```

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
4cfc7c35645d	servWeb	0.04%	11.21MiB / 512MiB	2.19%	23.6kB / 21.8kB	606kB / 0B	8
d8d73eb9780e	servSQL	0.19%	89.83MiB / 512MiB	17.55%	9.64kB / 14.1kB	54.2MB / 142MB	30

Exécuter une commande dans un conteneur existant

```
docker exec [OPTIONS] <conteneur> COMMAND [ARG...]
```

« Docker exec » est utilisé pour lancer des commandes dans un conteneur qui tourne en mode détaché. Par exemple, exécuter un bash en attachant le container nommé test :

```
docker exec -it <nom conteneur> bash
```

Afficher les logs d'un conteneur

```
docker logs -t -f <conteneur>
```

L'option -t (facultative) permet d'afficher l'horodatage (timestamp) de la sortie, ce qui peut s'avérer utile pour un éventuel débogage.

L'option -f (facultative) permet de suivre les logs « à chaud ».

Voir en temps réel les évènements sur un conteneur

```
docker events <conteneur>
```

Connaître la configuration et les éléments d'un conteneur

```
docker inspect <conteneur>
```

Créer une image à partir d'un conteneur

```
docker commit <conteneur> <nom future image>
```

Sauvegarder une image en local

```
docker save <image> > <nom_fichier.tar>
```

Restaurer depuis un conteneur en local

```
docker load -i <nom_fichier.tar>
```

Voir les différences apportées par rapport à une image d'origine :

```
docker diff <conteneur>
```

Renommer un conteneur

```
docker rename <ancien nom conteneur> <nouveau nom conteneur>
```

D. GESTION DES VOLUMES

Créer un volume

```
docker volume create <nom_du_volume>
```

Lister les volumes

```
docker volume ls
```

Inspecter un volume

```
docker volume inspect public-html
```

Supprimer un volume

```
docker volume rm public-html
```

E. GESTION DU RÉSEAU

Créer un réseau docker

```
docker network create --driver <type_de_driver> <nom_du_réseau>
```

Lister les réseaux docker

```
docker network ls
```

Supprimer un réseau docker

```
docker network rm <nom_du_réseau>
```

Récolter des informations sur un réseau docker

```
docker network inspect <nom_du_réseau>
```

-v ou --verbose : mode verbose pour un meilleur diagnostic

Supprimer tous les réseaux docker non utilisés

```
docker network prune
```

-f ou --force : forcer la suppression

Connecter un conteneur à un réseau docker

```
docker network connect <nom_du_réseau> <nom_du_conteneur>
```

Déconnecter un conteneur d'un réseau docker

```
docker network disconnect <nom_du_réseau> <nom_du_conteneur>
```

-f ou --force : forcer la déconnexion

Démarrer un conteneur et le connecter à un réseau docker

```
docker run --network <nom_du_réseau> <nom_image>
```

F. NETTOYAGE DE DOCKER

Docker a la fâcheuse habitude de laisser traîner plein de choses sur votre système. Fonctionnant par couches (*layers*), après des commandes build passées, beaucoup de ces *layers* ne sont plus utiles.

Supprimer tous les conteneurs, réseaux, images inutilisés et, éventuellement, les volumes.

```
docker system prune [OPTIONS]
```

Option	Description
Nom, raccourci	
--all,-a	Supprimer toutes les images inutilisées
--filter	Fournir des valeurs de filtre (par exemple label=<key>=<value>)
--force,-f	Ne pas demander de confirmation
--volumes	Supprimer les volumes

Supprimer les images inutilisées

```
docker image prune [OPTIONS]
```

Option	Description
Nom, raccourci	
--all,-a	Supprimer toutes les images inutilisées
--filter	Fournir des valeurs de filtre (par exemple until=<timestamp>)
--force,-f	Ne pas demander de confirmation

Supprimer tous les conteneurs arrêtés

```
docker container prune [OPTIONS]
```

Option	Description
Nom, raccourci	
--filter	Fournir des valeurs de filtre (par exemple until=<timestamp>)
--force,-f	Ne pas demander de confirmation

Supprimer tous les volumes locaux inutilisés

```
docker volume prune [OPTIONS]
```

Option	Description
Nom, raccourci	
--all,-a	Supprimer tous les volumes inutilisés
--filter	Fournir des valeurs de filtre (par exemple label=<label>)
--force,-f	Ne pas demander de confirmation