

Commandes pour Docker

La commande `docker --help` permet de voir les nombreuses commandes disponibles.

Obtenir des informations sur le système Docker installé

```
docker info
```

Chercher une image sur le Hub officiel

```
docker search <mot_clé>
```

Télécharger une image depuis le Hub officiel

```
docker pull <nom image>
```

`nom_image` peut comporter un tag comme `debian:jessie`

Lister les images disponibles

```
docker images
```

Supprimer une image

```
docker rmi <nom_image>
```

Supprimer toutes les images

```
docker rmi $(docker images -q)
```

Supprimer toutes les images "dangling"

Efface les couches (layers) qui ne mènent à aucune image définitive

```
docker rmi $(docker images -q -f dangling=true)
```

Renommer une image

```
docker tag IMAGE[:TAG] [REGISTRYHOST/][USERNAME/]NAME[:TAG]
```

Historique d'une image

```
docker history <image>
```

Créer un conteneur

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

Quelques paramètres de la commande `run` :

`-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

`-v` : permet de gérer des volumes de données 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)

`-e` : permet de définir des variables

`--restart always` (ou `on-failure`) : permet de redémarrer un conteneur automatiquement

Lister les conteneurs démarrés

```
docker ps
```

-a pour afficher tous les conteneurs

-q pour n'afficher que les « id »

Arrêter un conteneur

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

Arrêter tous les conteneurs

```
docker stop $(docker ps -q) ou docker kill $(docker ps -q)
```

Réactiver/démarrer un conteneur

```
docker start <conteneur>
```

Accéder à un conteneur actif en interactif

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

Supprimer un conteneur

```
docker rm <conteneur>
```

Le conteneur doit avoir été stoppé sinon **il faut utiliser l'option « -f »** :

```
docker rm -f <conteneur>
```

Pour supprimer également le ou les volumes associés au conteneur

```
docker rm -fv <conteneur>
```

Supprimer tous les conteneurs (y compris ceux qui sont démarrés)

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

Supprimer tous les conteneurs avec les volumes associés

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

- **docker system prune**

WARNING! This will remove:

- all stopped containers
- all networks not used by at least one container
- all dangling images
- all dangling build cache

- **docker system prune -a**

WARNING! This will remove:

- all stopped containers
- all networks not used by at least one container
- all images without at least one container associated to them
- all build cache

- **docker image prune -a**

WARNING! This will remove all images without at least one container associated to them. Are you sure you want to continue? [y/N]

Supprimer les volumes liés à aucun container

```
docker volume rm $(docker volume ls -qf dangling=true)
```

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 <conteneur>
```

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>
```

Commandes pour les Dockerfile

FROM [:] [AS]

Importe l'image à partir des repos officiels Docker, ou des repos privés si ceux-ci ont été paramétrés. On peut faire apparaître plusieurs frm dans le même Dockerfile.

LABEL

Un label est une manière d'associer des métadatas avec une clé et une valeur

Les instructions pour lancer des commandes :

RUN

La commande RUN permet de lancer des commandes **pour la construction de l'image**.

Format :

RUN commande (la commande est lancée dans un shell, par défaut "/bin/sh -c" sur Linux ou "cmd /S /C" sur Windows)

OU

RUN ["executable", "param1", "param2"]

CMD

La commande CMD permet de lancer des instructions **au lancement du conteneur**. Contrairement à la commande ENTRYPOINT, la commande CMD peut être surchargé lorsqu'on lance un "docker run".

Si on indique plusieurs CMD dans le Dockerfile, seul le dernier sera pris en compte.

Format :

CMD commande (la commande est lancée dans un shell) -

-- OU --

CMD ["executable", "param1", "param2"] (exec form, this is the preferred form)

-- OU --

CMD ["param1", "param2"] (as default parameters to ENTRYPOINT)

ENTRYPOINT

C'est la commande qui sera lancé par défaut au démarrage du conteneur.

Format :

ENTRYPOINT command param1 param2

Les instructions pour manipuler des fichiers ou des répertoires :

COPY src dest

ADD src dest

src peut être une URL et ADD décompresse automatique les fichiers

WORKDIR

Définit le répertoire de travail. Si le répertoire n'existe pas, il sera créé.

Commandes pour Docker Compose

docker-compose config vérifie la configuration du fichier docker-compose.yml.

docker-compose up crée les conteneurs et démarre les services décrits dans le fichier docker-compose.yml et ne rend pas la main.

docker-compose up -d démarre les services décrits dans le fichier docker-compose.yml et rend la main une fois que les services sont démarrés.

docker-compose build [SERVICE...] construit et/ou reconstruit les images des services avant de les lancer. Elle n'est utile que si l'on souhaite qu'un container utilise une configuration définie dans un Dockerfile (si un Dockerfile est modifié, un docker-compose up ne le prendra pas en compte, il faut effectuer un rebuild du service correspondant (ou de tous les services)).

Par défaut, le cache sera utilisé jusqu'à la première ligne modifiée.

Construire ou reconstruire les images avec les dernières versions des images de base :

docker-compose build --pull [SERVICE...]

Construire ou reconstruire les images sans utiliser le cache :

docker-compose build --no-cache [SERVICE...]

docker-compose logs retourne l'ensemble des logs des services depuis le dernier démarrage et rend la main.

docker-compose logs <nom_d'un_service> retourne les logs du service correspondant.

docker-compose ps affiche des informations sur les conteneurs créés.

docker-compose logs -f affiche les logs des services et continue à les « écouter » sans rendre la main.

docker-compose logs -f <nom_d'un_service> retourne les logs du service correspondant et continue à les « écouter » sans rendre la main.

docker-compose stop arrête l'ensemble des conteneurs.

docker-compose stop <nom_d'un_service> arrête le conteneur correspondant.

docker-compose start démarre l'ensemble des conteneurs.

docker-compose start <nom_d'un_service> démarre le conteneur correspondant.

docker-compose restart redémarre l'ensemble des services.

docker-compose restart <nom_d'un_service> redémarre le service.

docker-compose down stoppe et supprime l'ensemble des conteneurs.

docker-compose down <nom_d'un_service> stoppe le service et supprime le conteneur correspondant.

docker-compose rm supprime l'ensemble des conteneurs (le ou les conteneurs doivent avoir été arrêtés au préalable).

docker-compose rm <nom_d'un_service> supprime le conteneur correspondant (le conteneur doit avoir été arrêté au préalable).

docker-compose exec <nom_d'un_service> bash fournit une console bash au sein du conteneur correspondant.

docker-compose run exécute une commande dans un conteneur.