Commandes de base

Exécuter un conteneur : **docker run**

Commande de base :

```
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

Le nom de l'image docker à utiliser doit être spécifié avec le paramètre `image`. Quelques exemples :

- Distributions Linux : `debian`, `ubuntu`, `alpine`
- Logiciels et CMS : `wordpress`, `prestashop`
- Langages de programmation : `php`, `python`
- Bases de données : `mysql`, `postgresql`

> Il existe actuellement 11 millions d'image (août 2025) sur le Docker Hub (que nous approfondirons plus tard).

La commande **docker run** propose de nombreux paramètres dont voici les principaux :

Paramètre	Description	Exemple

`--name`	Attribuer un nom au conteneur. Si il n'est pas défini, Docker générera un nom aléatoire à partir d'un dictionnaire de mots.	`docker run ubuntu --name=test1`
		`docker run ubuntu --name=test2`
		`docker run ubuntu --name=test3`

`-ti`	Les paramètres **-t** et **-i** permettent d'ouvrir un terminal interactif dans le conteneur : **-t** alloue un pseudo-terminal, et **-i** garde l'entrée ouverte.	`docker run -ti ubuntu` `# je suis redirigé dans la console du conteneur`

Paramètre	Description	Exemple
`-d`, `--detached`	Lance le conteneur en arrière-plan, en mode **détaché**.	`docker run -d ubuntu` `# le conteneur se lance en arrière-plan`

`--rm`	Supprimer le conteneur lorsqu'il est arrêté.	`docker run --rm -ti ubuntu`
		`# le conteneur se lance dans la console`
		`# Il sera supprimé après la sorite (exit)`

-e, --env	Passer une variable d'environnement à l'application conteneurisée	`docker run -e VAR1=value1` `# la variable VAR1 sera définie dans le conteneur`

```
debian@debian:~$ docker run -ti --name=test1 ubuntu
root@81aae328d11d:/# echo "Je suis dans mon conteneur"
Je suis dans mon conteneur
root@81aae328d11d:/# exit
exit
debian@debian:~$ docker run -d --name=test2 ubuntu
85d0402c2f0d48a7ec7dbc73cf8dc65fb66d3079e5ceb8c46a77580985569859
debian@debian:~$ echo "Mon conteneur test2 a été lancé en arrière-plan"
Mon conteneur test2 a été lancé en arrière-plan
debian@debian:~$
```

> Nous approfondirons plus tard d'autres paramètres de la commande **docker run**.

La commande `docker run` se termine par la commande à lancer dans le conteneur ainsi que ses paramètres. L'ajout de cette commande est optionnel, il n'est pas toujours nécessaire de l'indiquer.

Certaines images ne lancent pas automatiquement d'invite de commande `bash`.
Par exemple, l'image officielle python démarre sur une invite de commande python :

```
debian@debian:~$ docker run -ti python
Python 3.13.5 (main, Jul 22 2025, 04:26:21) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> str("Je suis en python ici car je n'ai pas précisé de commande.")
"Je suis en python ici car je n'ai pas précisé de commande."
>>> exit
debian@debian:~$ docker run -ti python bash
root@aafde6d872d3:/# echo "Maintenant je suis en bash dans mon conteneur python"
Maintenant je suis en bash dans mon conteneur python
root@aafde6d872d3:/# python --version
Python 3.13.5
```

Lister les conteneurs : **docker ps**

La commande `docker ps` liste les conteneurs démarrés sur la machine. Le paramètre `--all`, `-a` permet d'afficher les conteneurs stoppés.

```
docker ps
docker ps -a
```

```
debian@debian:~$ docker run -tid --name=test_1 ubuntu
8343b7be708201965140558 38b7130c9db59a2f04bb13ad062d7e1182d375a35
debian@debian:~$ docker run -ti --name=test_2 ubuntu
root@b5909c1ce8eb:/# exit
exit
debian@debian:~$ docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
8343b7be7082   ubuntu    "/bin/bash"   30 seconds ago   Up 30 seconds             test_1
debian@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS                      PORTS     NAMES
b5909c1ce8eb   ubuntu    "/bin/bash"   12 seconds ago   Exited (0) 4 seconds ago              test_2
8343b7be7082   ubuntu    "/bin/bash"   32 seconds ago   Up 32 seconds                         test_1
debian@debian:~$
```

- `test_1` est lancé en mode détaché (`-d`), il tourne en arrière-plan.
- `test_2` est lancé en mode attaché, il s'arrête lorsque je le quitte avec `exit`.

> Astuce : il est possible de se détacher d'un conteneur sans l'arrêter avec **CTRL+P** puis **CTRL+Q** !

```
debian@debian:~$ docker run -ti --name=test_3 ubuntu
root@53272edb658e:/# debian@debian:~$
debian@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS                      PORTS     NAMES
53272edb658e   ubuntu    "/bin/bash"   10 seconds ago   Up 10 seconds                         test_3
b5909c1ce8eb   ubuntu    "/bin/bash"   4 minutes ago    Exited (0) 4 minutes ago              test_2
8343b7be7082   ubuntu    "/bin/bash"   4 minutes ago    Up 4 minutes                          test_1
debian@debian:~$
```

Après avoir pressé **CTRL+P** puis **CTRL+Q**, je suis détaché du conteneur `test_3` qui tourne encore en arrière-plan.

Arrêt d'un conteneur : **docker stop** et **docker start**

```
docker start MON_CONTENEUR  # démarre le conteneur
docker stop MON_CONTENEUR # arrête le conteneur
docker rm MON_CONTENEUR # supprime le conteneur
```

```
debian@debian:~$ docker run -tid --name=toto alpine
353bc1b72eb8bc941e689c5f880054cc3290500ff84b8050f0a3dd15e9f1533e
debian@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND     CREATED         STATUS         PORTS     NAMES
353bc1b72eb8   alpine    "/bin/sh"   4 seconds ago   Up 3 seconds             toto
debian@debian:~$ docker stop toto
toto
debian@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND     CREATED          STATUS                       PORTS     NAMES
353bc1b72eb8   alpine    "/bin/sh"   23 seconds ago   Exited (137) 4 seconds ago             toto
debian@debian:~$ docker start toto
toto
debian@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND     CREATED          STATUS         PORTS     NAMES
353bc1b72eb8   alpine    "/bin/sh"   37 seconds ago   Up 7 seconds             toto
debian@debian:~$ docker stop toto
toto
debian@debian:~$ docker rm toto
toto
debian@debian:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
debian@debian:~$
```

Visualiser les logs : **docker logs**

```
docker logs MON_CONTENEUR
docker logs -f MON_CONTENEUR # -f pour --follow, permet de visualiser en direct le log.
```

```
debian@debian:~$ docker run -tid --name=test alpine date
d3ae72c6496375046f6101aeb036170f111d41fec491396c10c3a0a0466493f4
debian@debian:~$ docker logs test
Sat Aug  2 21:11:37 UTC 2025
debian@debian:~$
```

La commande date est lancée au démarrage du conteneur détaché, je ne vois pas le résultat.
J'y accède plus tard avec la commande `docker logs`.

S'attacher à un conteneur : **docker attach**

```
docker attach MON_CONTENEUR
```

```
debian@debian:~$ docker run -tid --name=test_debian debian
a436317b1b38823d1709539da831b8a9b45e5300eed1f42bce54fb9ff035ba92
debian@debian:~$ docker attach test_debian
root@a436317b1b38:/# echo "Je suis dans mon conteneur"
```

Le conteneur a été lancé en arrière-plan (mode détaché) avec `-d`. Je peux y accéder plus tard.

Exécuter une commande : **docker exec**

```
docker exec [OPTIONS] CONTAINER COMMAND [ARG...]
```

La syntaxe de cette commande ressemble à `docker run`. La différence est qu'ici, la commande est lancée dans un conteneur déjà en cours d'exécution.

La commande `docker exec` est utile pour lancer une invite de commande (`bash`) dans un conteneur qui n'en propose pas forcément. Exemple avec python :

```
debian@debian:~$ docker run -tid --name=test_python python
be97292531739fa1634c0d3c24f28b4afff0a4a983aa51ac5f0faba79c1b9b91
debian@debian:~$ docker attach test_python
>>> str('Je suis en Python après un docker attach')
'Je suis en Python après un docker attach'
>>> read escape sequence
debian@debian:~$ docker exec -ti test_python bash
root@be9729253173:/# echo "Et maintenant je suis en bash, dans le conteneur"
Et maintenant je suis en bash, dans le conteneur
root@be9729253173:/# exit
exit
debian@debian:~$ docker logs test_python
Python 3.13.5 (main, Jul 22 2025, 04:26:21) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> str('Je suis en Python après un docker attach')
'Je suis en Python après un docker attach'
debian@debian:~$
```

Je lance un conteneur python sans préciser de commande. La commande par défaut de l'image docker est utilisée.
Lorsque je m'attache au conteneur avec `docker attach`, je suis dans une invite de commande python.
Je me détache du conteneur et je lance une invite de commande `bash` avec `docker exec`.

> La commande**docker exec** exécute un nouveau processus dans le conteneur. La sortie de ce processus n'est pas visible dans les logs du conteneur !

Revision #40
Created 2 August 2025 20:09:14 by Thibaud FRICHET
Updated 7 September 2025 19:12:01 by Thibaud FRICHET