

Les commandes (base)

Lancez votre premier conteneur en local

Dans cette deuxième partie :

- Nous allons prendre en main Docker.

Nous allons commencer par découvrir l'interface en ligne de commande :

- ✓ qui nous permet de discuter avec le daemon Docker installé précédemment.

Objectif :

Être capable de lancer et gérer vos conteneurs.

Rappel, le Docker Hub :

Le Docker Hub est la registry officielle de Docker.



Une **registry** est un logiciel qui permet de partager des images à d'autres personnes. C'est un composant majeur dans l'écosystème Docker, car il permet :

- à des **développeurs** de **distribuer des images** prêtes à l'emploi et de les **versionner** avec un système de tags ;
- à des **outils** d'intégration en continu de jouer une suite de **tests**, sans avoir besoin d'autre chose que de Docker ;
- à des **systèmes automatisés** de **déployer ces applications** sur vos environnements de développement et de production.

Démarrez votre premier conteneur Docker

Structure d'une commande :

```
docker <catégorie> <commande> <options>
```

Exemple :

```
docker image ls  
// Liste les images (ou 'docker images')
```

Ou encore :

```
docker container -a ls  
// Liste les containers, même inactif (ou 'docker ps -a')
```

Pour démarrer votre premier conteneur, vous devez utiliser la commande : « **run** »

```
➤ docker run hello-world
```

Quand vous utilisez cette commande :

- ✓ le daemon Docker va chercher si l'image hello-world est disponible en local.
- ✓ Dans le cas contraire, il va la récupérer sur la registry Docker officielle.

Résultat :

```
Hello from Docker!
```

```
This message shows that your installation appears to be working correctly.
```

```
To generate this message, Docker took the following steps:
```

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

```
To try something more ambitious, you can run an Ubuntu container with:
```

```
$ docker run -it ubuntu bash
```

```
Share images, automate workflows, and more with a free Docker ID:
```

```
https://hub.docker.com/
```

```
For more examples and ideas, visit:
```

```
https://docs.docker.com/get-started/
```



Dans notre cas, le conteneur a démarré, puis affiché du contenu, et il a fini par s'arrêter. Si vous souhaitez **que votre conteneur reste allumé jusqu'à l'arrêt du service qu'il contient**, vous devez ajouter l'argument `--detach (-d)`. Celui-ci permet de ne pas rester attaché au conteneur, et donc de pouvoir lancer plusieurs conteneurs. Nous allons voir dans la section suivante comment utiliser l'argument `-d`.

Démarrez un serveur Nginx avec un conteneur Docker

Vous savez maintenant :

- ✓ lancer un conteneur : 'docker run',
- ✓ et vous avez compris les actions effectuées par le daemon Docker lors de l'utilisation de la commande docker run.

Maintenant, nous allons aller plus loin avec celui-ci :

- Nous allons lancer un conteneur qui démarre un serveur Nginx :

```
> docker run -p 8080:80 nginx
```

Résultat :

```
2023/12/03 16:06:10 [notice] 1#1: start worker process 39
2023/12/03 16:06:10 [notice] 1#1: start worker process 40
2023/12/03 16:06:10 [notice] 1#1: start worker process 41
2023/12/03 16:06:10 [notice] 1#1: start worker process 42
2023/12/03 16:06:10 [notice] 1#1: start worker process 43
2023/12/03 16:06:10 [notice] 1#1: start worker process 44
2023/12/03 16:06:10 [notice] 1#1: start worker process 45
2023/12/03 16:06:10 [notice] 1#1: start worker process 46
2023/12/03 16:06:10 [notice] 1#1: start worker process 47
2023/12/03 16:06:10 [notice] 1#1: start worker process 48
172.17.0.1 - - [03/Dec/2023:16:06:14 +0000] "GET / HTTP/1.1" 200 615 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36" "-"
```

Problème :

- ✓ L'image est lancée (serveur Nginx)
- ✓ Mais le terminal est verrouillé.

Listons les container lancés :

```
> docker ps -a
```

ou

```
docker container ls
```

Résultat :

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
597aee1f9e84	nginx	"/docker-entrypoint..."	4 minutes ago	Exited (0) 9 seconds ago		inspiring_ride

Nous pouvons le stopper via son ID :

```
docker stop 597aee1f9e84
```

Listons à nouveau les container lancés :

```
> docker ps -a
```

ou

```
docker container ls
```

Résultat :

```
PS D:\AFCI\Cours\8 - conteneurisation> docker container ls
CONTAINER ID   IMAGE          COMMAND         CREATED        STATUS        PORTS          NAMES
PS D:\AFCI\Cours\8 - conteneurisation> |
```

Nous pouvons maintenant relancer notre container

Ajoutons une option : '-d' ou '--detach'

```
PS D:\AFCI\Cours\8 - conteneurisation> docker run -d -p 8080:80 nginx  
c6ad1f2e56c723c641217f3ded3f6b056cfe9c74361a0728ddac52cae5d4dfc7  
PS D:\AFCI\Cours\8 - conteneurisation> |
```

Notre container tourne sans bloquer l'accès au terminal ...


```
PS D:\AFCI\Cours\8 - conteneurisation> docker run -d -p 8080:80 nginx  
c6ad1f2e56c723c641217f3ded3f6b056cfe9c74361a0728ddac52cae5d4dfc7  
PS D:\AFCI\Cours\8 - conteneurisation> |
```

Dans cette commande, nous avons utilisé deux options :

- ✓ -d pour détacher le conteneur du processus principal de la console.

Il vous permet de continuer à utiliser la console pendant que votre conteneur tourne sur un autre processus ;

- ✓ -p pour définir l'utilisation de ports.

Dans notre cas, nous lui avons demandé de transférer le trafic du port 8080 vers le port 80 du conteneur.

Rendons nous ici : <http://127.0.0.1:8080>

➤ C'est la page par défaut de Nginx.

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Travailler dans un conteneur

Vous pourriez aussi avoir besoin de "rentrer" dans votre conteneur Docker pour pouvoir y effectuer des actions.

➤ Pour cela, vous devez utiliser la commande :

```
docker exec -ti ID_RETournÉ_LORS_DU_DOCKER_RUN bash
```

Dans cette commande, l'argument -ti permet d'avoir un shell bash pleinement opérationnel.

```
PS D:\AFCl\Cours\8 - conteneurisation> docker exec -ti c6ad1f2e56c723c641217f3ded3f6b056cfe9c74361a0728ddac52cae5d4dfc7 bash
root@c6ad1f2e56c7:/# |
```

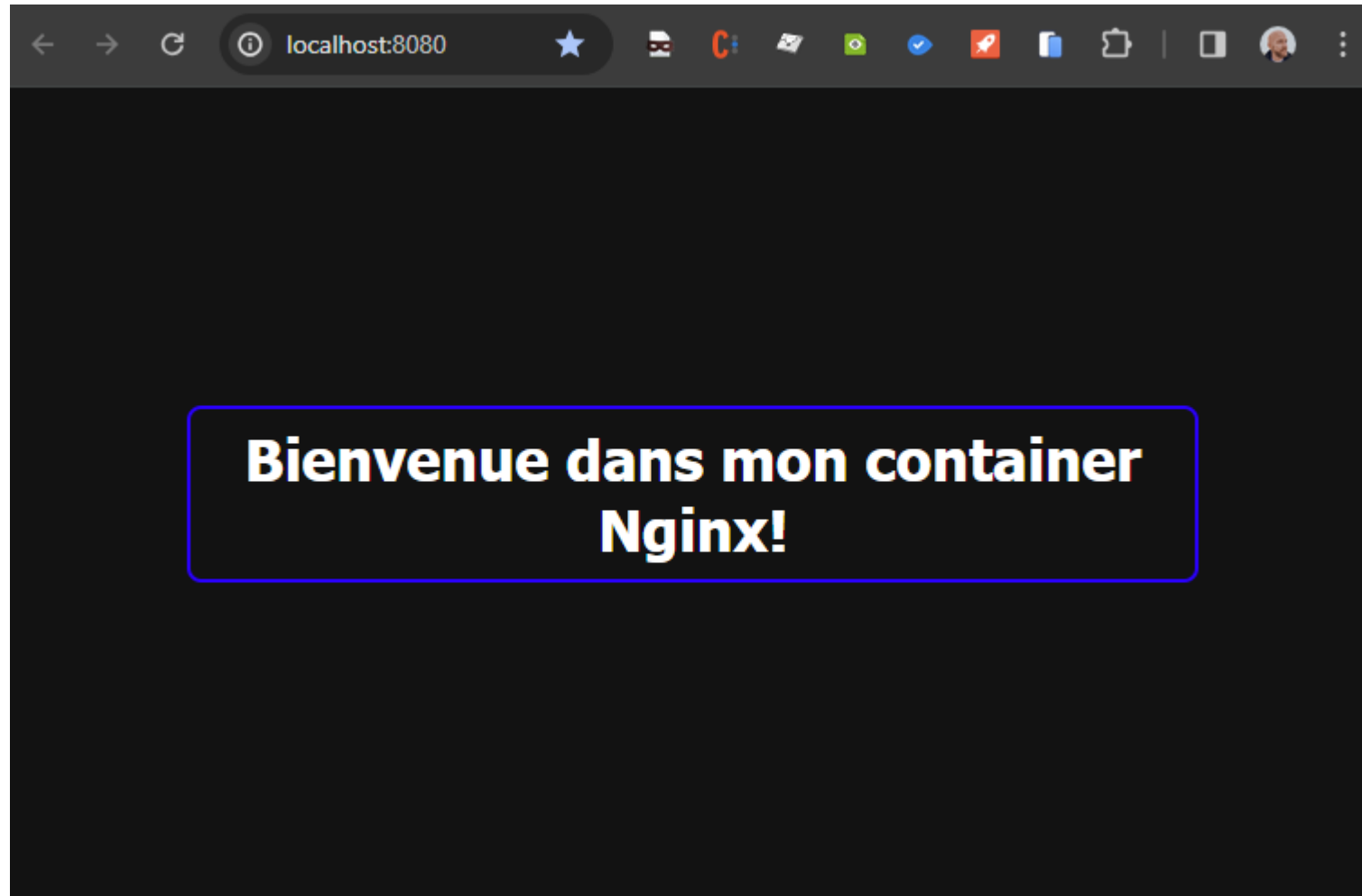
Une fois que vous êtes dans votre conteneur :

- vous pouvez vous rendre, via la commande `cd /usr/share/nginx/html`, dans le répertoire où se trouve le fichier `index.html` , pour modifier son contenu et voir le résultat en direct à l'adresse <http://127.0.0.1:8080>

```
root@c6ad1f2e56c7:/# cd usr/share/nginx/html/  
root@c6ad1f2e56c7:/usr/share/nginx/html# ls  
50x.html  index.html  
root@c6ad1f2e56c7:/usr/share/nginx/html# |
```

Exercise

Modifier votre fichier index.html pour obtenir ce résultat :



Indices :

- ✓ Vous êtes dans un conteneur qui tourne sous débien

Trouver les commandes linux pour :

1. Installer un éditeur de texte (ex: nano)
2. Réussir à :

- Ouvrir le fichier
- Le modifier
- L'enregistrer
- Le quitter

Correction

Test ouverture fichier :

```
root@c6ad1f2e56c7:/usr/share/nginx/html# nano index.html  
bash: nano: command not found
```

Nano n'est pas installé ...

Après une petite recherche Google :

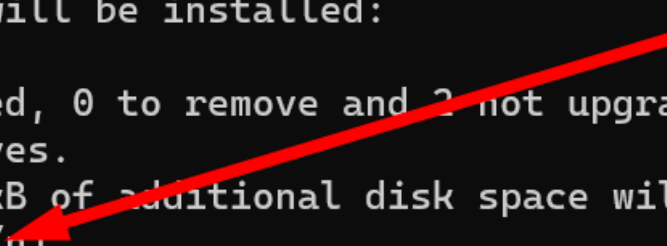
- ✓ Il faut mettre à jour les dépôts,
- ✓ Installer le paquet 'nano'

Mettre à jour les dépôts :

```
root@c6ad1f2e56c7:/usr/share/nginx/html# apt-get update
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [52.1 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8780 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [6668 B]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [106 kB]
Fetched 9144 kB in 19s (476 kB/s)
Reading package lists... Done
root@c6ad1f2e56c7:/usr/share/nginx/html# apt-get install nano
Reading package lists... Done
Building dependency tree... Done
```

Installer le paquet 'nano' :

```
root@c6ad1f2e56c7:/usr/share/nginx/html# apt-get install nano
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libgpm2 libncursesw6
Suggested packages:
  gpm hunspell
The following NEW packages will be installed:
  libgpm2 libncursesw6 nano
0 upgraded, 3 newly installed, 0 to remove and 2 not upgraded.
Need to get 837 kB of archives.
After this operation, 3339 kB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 http://deb.debian.org/debian bookworm/main amd64 libncursesw6 amd64 6.4-4 [134 kB]
Get:2 http://deb.debian.org/debian bookworm/main amd64 libgpm2 amd64 1.20.7-1 [5694 B]
Get:3 http://deb.debian.org/debian bookworm/main amd64 nano amd64 2.9.4-1 [284 kB]
```



Par défaut : entrer

Editer le fichier 'index.html' :

```
root@c6ad1f2e56c7:/usr/share/nginx/html# nano index.html
```

Faire les modifications :

```
body { display: flex; min-height: 100vh; align-items:center; justify-content: center; width: 100%;  
font-family: Tahoma, Verdana, Arial, sans-serif; }  
h1 { border: 2px solid blue;text-align: center; padding: 8px; border-radius: 8px }  
</style>  
</head>  
<body>  
<h1>Bienvenue dans mon container Nginx!</h1>  
</body>
```

Le fichier complet :

```
GNU nano 7.2                                index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { display: flex; min-height: 100vh; align-items:center; justify-content: center; width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
h1 { border: 2px solid blue;text-align: center; padding: 8px; border-radius: 8px }
</style>
</head>
<body>
<h1>Bienvenue dans mon container Nginx!</h1>
</body>
</html>
```

Une fois modifié, exécuter :

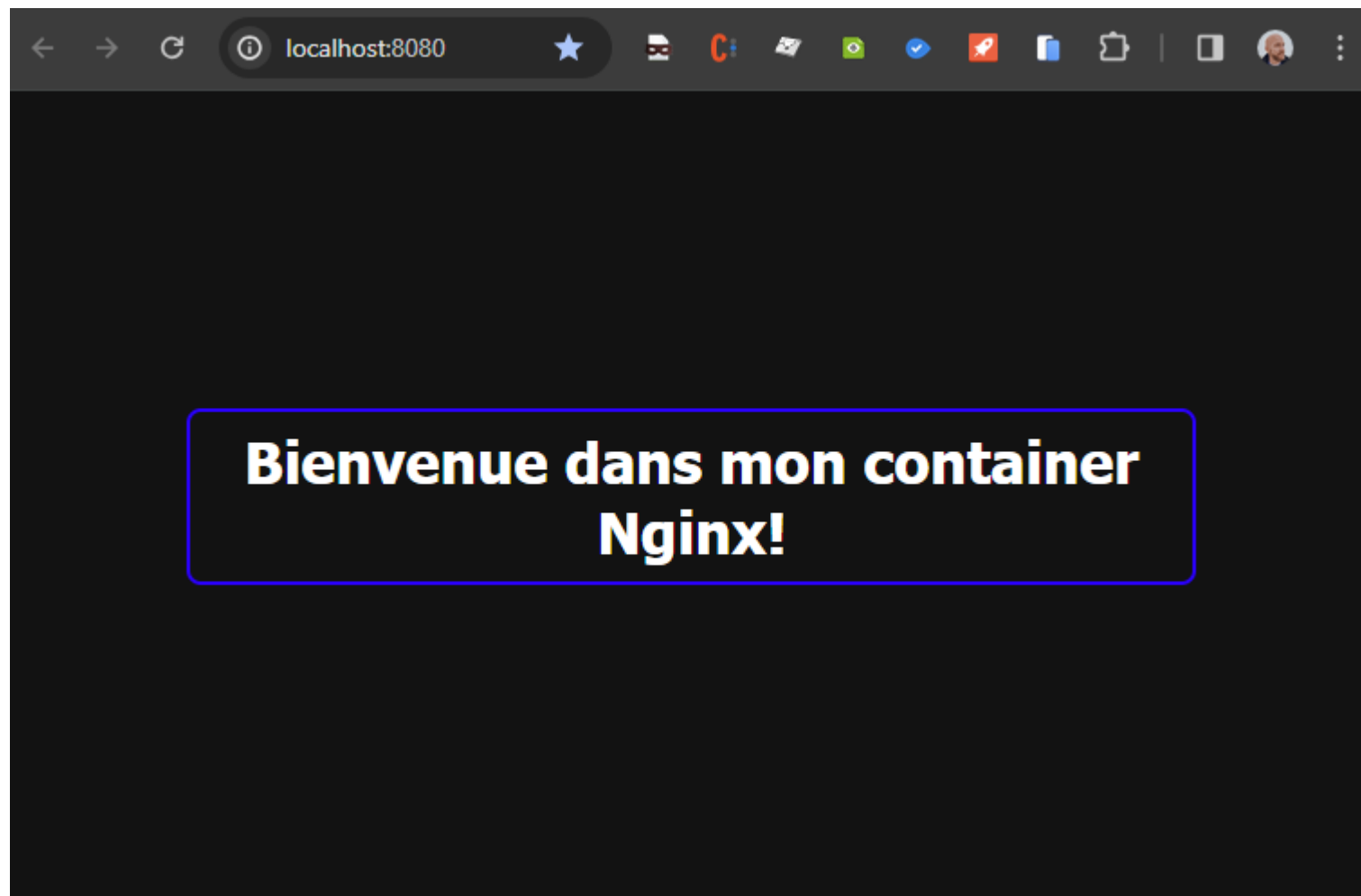
✓ Ctrl + s

Pour enregistrer les modifications

✓ Ctrl + x

Pour quitter

Résultat :



Arrêtez votre conteneur Docker

Vous avez créé un conteneur avec l'option `-detach`,

➤ vous aurez donc sûrement besoin de l'arrêter !

Pour cela, faites appel à la commande :

✓ `docker stop ID_RETOURNÉ_LORS_DU_DOCKER_RUN`

Dans l'ordre

✓ Trouver l'ID :

```
PS D:\AFCI\Cours\8 - conteneurisation> docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
c6ad1f2e56c7	nginx	"/docker-entrypoint..."	About an hour ago	Up About

✓ Stopper le conteneur :

```
PS D:\AFCI\Cours\8 - conteneurisation> docker stop c6ad1f2e56c7
```

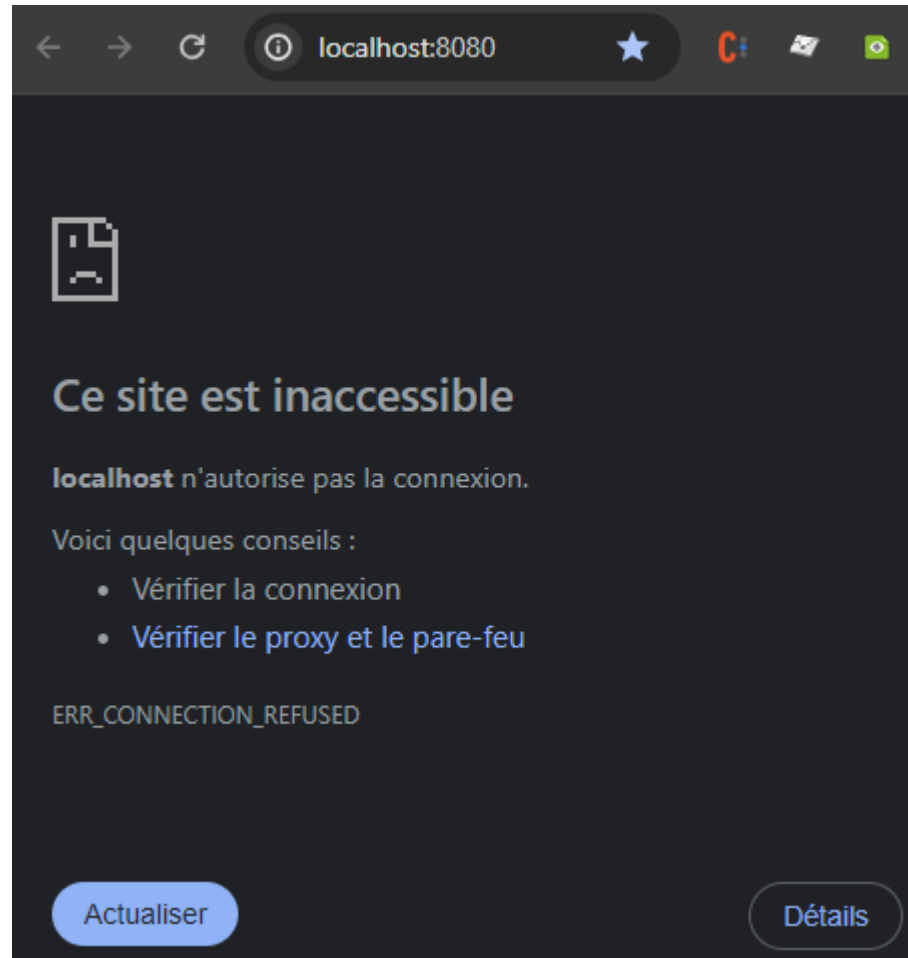
c6ad1f2e56c7

✓ Vérifier :

```
PS D:\AFCI\Cours\8 - conteneurisation> docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
c6ad1f2e56c7	nginx	"/docker-entrypoint..."	About an hour ago	Exited (0) 3 seconds ago

Notre serveur nginx est arrêté :



Supprimer votre conteneur Docker

Maintenant que votre conteneur Docker a été arrêté :

➤ vous pouvez le supprimer avec la commande :

```
docker rm ID_RETOURNÉ_LORS_DU_DOCKER_RUN
```

✓ Celle-ci va détruire le conteneur et son contenu ;

cependant, vous pouvez toujours recréer votre conteneur avec la commande docker run vue plus haut.

Dans l'ordre

✓ Trouver l'ID :

```
PS D:\AFCI\Cours\8 - conteneurisation> docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
c6ad1f2e56c7	nginx	"/docker-entrypoint..."	About an hour ago	Up About

✓ supprimer le conteneur :

```
PS D:\AFCI\Cours\8 - conteneurisation> docker rm c6ad1f2e56c7
```

```
c6ad1f2e56c7
```


```
PS D:\AFCI\Cours\8 - conteneurisation> |
```

✓ Vérifier :

```
PS D:\AFCI\Cours\8 - conteneurisation> docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
--------------	-------	---------	---------	--------	-------	-------

PS D:\AFCI\Cours\8 - conteneurisation> |

 **disparu**

Récupérez une image du Docker Hub

Vous pouvez aussi avoir besoin de récupérer des images sur le Docker Hub sans pour autant lancer de conteneur.

✓ Pour cela, vous avez besoin de lancer la commande suivante :

```
> docker pull NOM_DE_L_IMAGE|
```

Commençons par lister les images en local :

```
PS D:\AFCI\Cours\8 - conteneurisation> docker images -a
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	a6bd71f48f68	12 days ago	187MB
hello-world	latest	9c7a54a9a43c	7 months ago	13.3kB

```
PS D:\AFCI\Cours\8 - conteneurisation>
```

2 images existent déjà en local

Pour supprimer une image :

```
hello-world    latest    9c7a54a9a43c    7 months ago    13.3kB
PS D:\AFCI\Cours\8 - conteneurisation> docker image rm 9c7a54a9a43c
Untagged: hello-world:latest
Untagged: hello-world@sha256:c79d06dfdfd3d3eb04cafd0dc2bacab0992ebc243e083cabe208bac4dd7759e0
Deleted: sha256:9c7a54a9a43cca047013b82af109fe963fde787f63f9e016fdc3384500c2823d
Deleted: sha256:01bb4fce3eb1b56b05adf99504dafd31907a5aadac736e36b27595c8b92f07f1
PS D:\AFCI\Cours\8 - conteneurisation> |
```

Vérifions :

```
PS D:\AFCI\Cours\8 - conteneurisation> docker images -a
REPOSITORY    TAG        IMAGE ID        CREATED        SIZE
nginx         latest    a6bd71f48f68    12 days ago    187MB
PS D:\AFCI\Cours\8 - conteneurisation> |
```

L'image 'hello-world' n'existe plus en local ...

Récupérer une image en local :

```
PS D:\AFCI\Cours\8 - conteneurisation> docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:c79d06dfdfd3d3eb04cafd0dc2bacab0992ebc243e083cabe208bac4dd7759e0
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview hello-world
PS D:\AFCI\Cours\8 - conteneurisation>
```

Vérifions :

```
View a summary of image vulnerabilities and recommendations →
PS D:\AFCI\Cours\8 - conteneurisation> docker images -a
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest             9c7a54a9a43c       7 months ago       13.3kB
PS D:\AFCI\Cours\8 - conteneurisation>
```

L'image 'hello-world' existe en local ...

Inspector

Il est possible d'inspecter les objets docker :

- ✓ Conteneur,
- ✓ Image;
- ✓ Volumes;
- ✓ Network;
- ✓ Etc ...

L'inspection donne des détails plus approfondis de l'objet.

Exemple :

```
C:\Users\conta>docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS          NAMES
851d4ed93c57   hello-world "/hello"                3 minutes ago   Exited (0) 3 minutes ago          keen_zhukovsky 1

C:\Users\conta>docker inspect keen_zhukovsky 2
[
  {
    "Id": "851d4ed93c578609d4c6e150cddf7743505fdf91e933fedd20c0fa8da0811efd",
    "Created": "2023-12-05T17:40:32.277762435Z",
    "Path": "/hello",
    "Args": [],
    "State": {
      "Status": "exited",
      "Running": false,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 0,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2023-12-05T17:40:32.602149941Z",
      "FinishedAt": "2023-12-05T17:40:32.601120616Z"
    },
    "Image": "sha256:9c7a54a9a43cca047013b82af109fe963fde787f63f9e016fdc3384500c2823d",
    "ResolvConfPath": "/var/lib/docker/containers/851d4ed93c578609d4c6e150cddf7743505fdf91e933fedd20c0fa8da0811efd/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/851d4ed93c578609d4c6e150cddf7743505fdf91e933fedd20c0fa8da0811efd/hostname",
    "HostsPath": "/var/lib/docker/containers/851d4ed93c578609d4c6e150cddf7743505fdf91e933fedd20c0fa8da0811efd/hosts",
    "LogPath": "/var/lib/docker/containers/851d4ed93c578609d4c6e150cddf7743505fdf91e933fedd20c0fa8da0811efd/851d4ed93c578609d4c6e150cddf7743505fdf91e933fedd20c0fa8da0811efd-json.log",
    "Name": "/keen_zhukovsky",
    "RestartCount": 0,
    "Driver": "overlay2",
    "Platform": "linux",
    "MountLabel": "",
    "ProcessLabel": "",
    "AppArmorProfile": "",
    "ExecIDs": null,
    "HostConfig": {
      "Binds": null,
      "ContainerIDFile": "",
      "LogConfig": {
        "Type": "json-file",
        "Config": {}
      },
      "NetworkMode": "default",
      "PortBindings": {},
      "RestartPolicy": {
        "Name": "no",
        "MaximumRetryCount": 0
      },
      "AutoRemove": false,
    }
  }
]
```

Comment nettoyer mon système

Après avoir fait de nombreux tests sur votre ordinateur, vous pouvez avoir besoin de faire un peu de ménage.

Pour cela, vous pouvez :

- ✓ supprimer l'ensemble des ressources manuellement dans Docker.
- ✓ Ou vous pouvez laisser faire Docker pour qu'il fasse lui-même le ménage.

Voici la commande que vous pouvez utiliser pour faire le ménage :

➤ `docker system prune`

Test:

```
PS D:\AFCI\Cours\8 - conteneurisation> 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

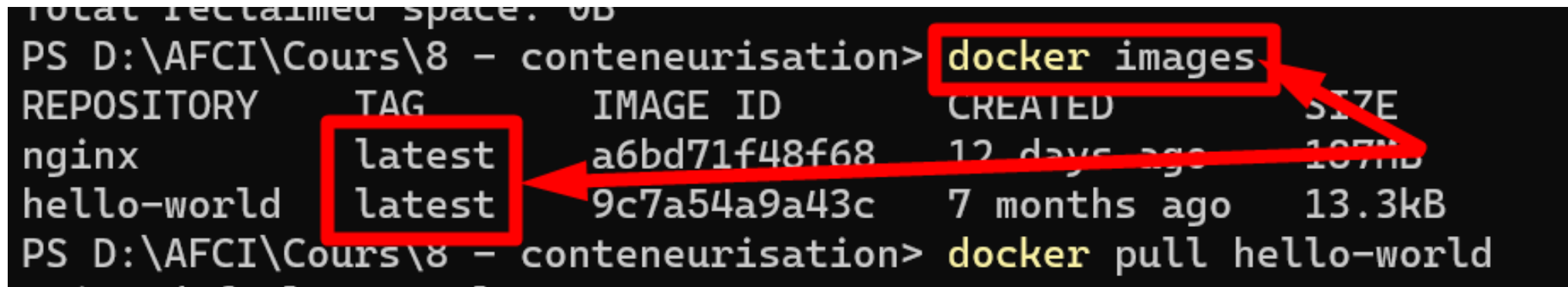
Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
PS D:\AFCI\Cours\8 - conteneurisation> docker ps -a
```

Cette commande supprime les données suivantes :

- ✓ l'ensemble des conteneurs Docker qui ne sont pas en status running ;
- ✓ l'ensemble des réseaux créés par Docker qui ne sont pas utilisés par au moins un conteneur ;
- ✓ l'ensemble des images Docker non taggées ;
- ✓ l'ensemble des caches utilisés pour la création d'images Docker.

Cas des images :

```
Total reclaimed space: 0B
PS D:\AFCI\Cours\8 - conteneurisation> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx                latest              a6bd71f48f68       12 days ago        107MB
hello-world          latest              9c7a54a9a43c       7 months ago       13.3kB
PS D:\AFCI\Cours\8 - conteneurisation> docker pull hello-world
```



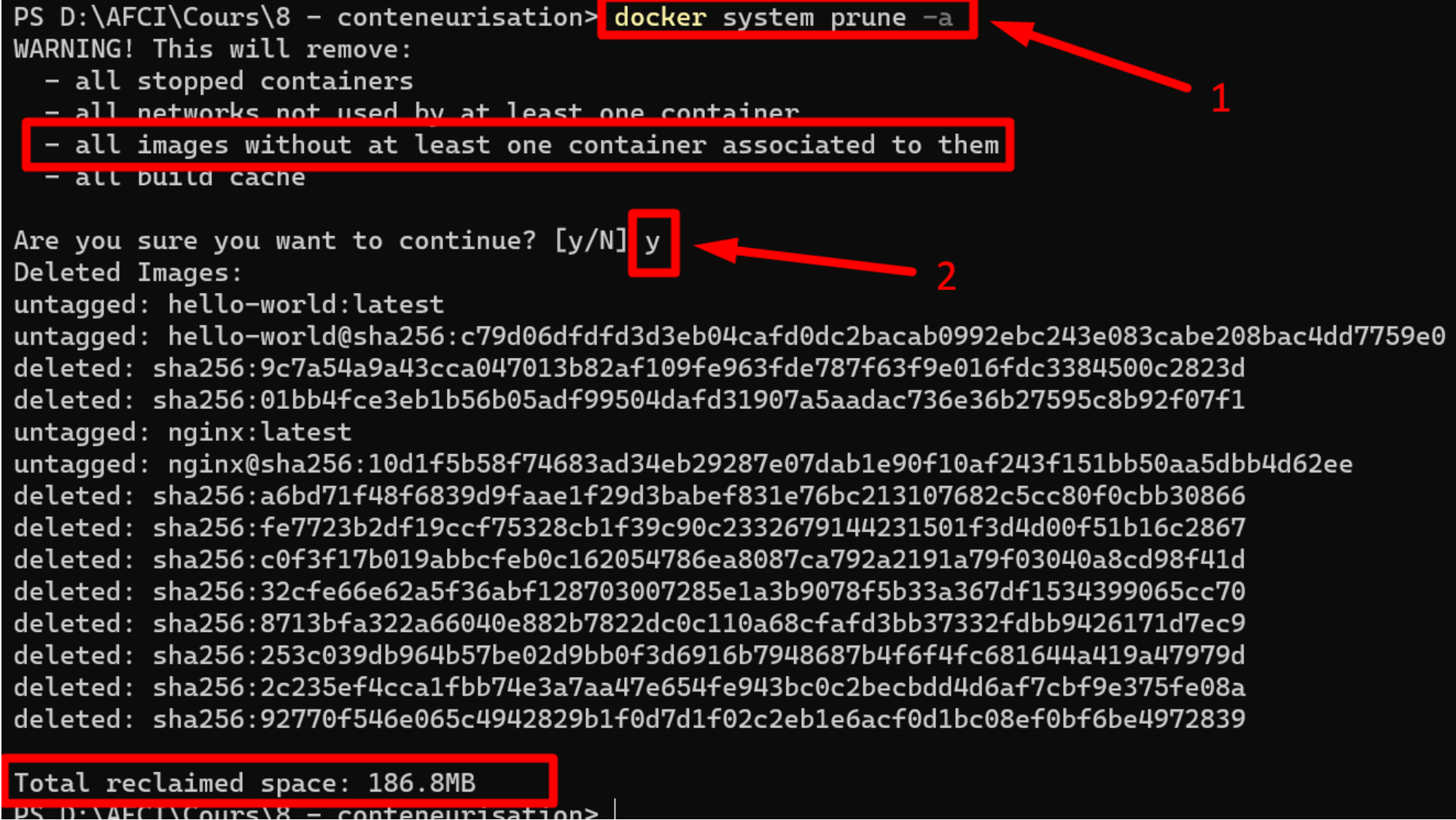
Les images sont toujours présentes car « taggées »

Pour forcer l'effacement complet :

```
PS D:\AFCI\Cours\8 - conteneurisation> 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

Are you sure you want to continue? [y/N] y
Deleted Images:
untagged: hello-world:latest
untagged: hello-world@sha256:c79d06dfdfd3d3eb04cafd0dc2bacab0992ebc243e083cabe208bac4dd7759e0
deleted: sha256:9c7a54a9a43cca047013b82af109fe963fde787f63f9e016fdc3384500c2823d
deleted: sha256:01bb4fce3eb1b56b05adf99504dafd31907a5aadac736e36b27595c8b92f07f1
untagged: nginx:latest
untagged: nginx@sha256:10d1f5b58f74683ad34eb29287e07dab1e90f10af243f151bb50aa5dbbb4d62ee
deleted: sha256:a6bd71f48f6839d9faae1f29d3babef831e76bc213107682c5cc80f0cbb30866
deleted: sha256:fe7723b2df19ccf75328cb1f39c90c2332679144231501f3d4d00f51b16c2867
deleted: sha256:c0f3f17b019abbcfeb0c162054786ea8087ca792a2191a79f03040a8cd98f41d
deleted: sha256:32cfe66e62a5f36abf128703007285e1a3b9078f5b33a367df1534399065cc70
deleted: sha256:8713bfa322a66040e882b7822dc0c110a68cfafd3bb37332fdbb9426171d7ec9
deleted: sha256:253c039db964b57be02d9bb0f3d6916b7948687b4f6f4fc681644a419a47979d
deleted: sha256:2c235ef4cca1fbb74e3a7aa47e654fe943bc0c2becbdd4d6af7cbf9e375fe08a
deleted: sha256:92770f546e065c4942829b1f0d7d1f02c2eb1e6acf0d1bc08ef0bf6be4972839

Total reclaimed space: 186.8MB
PS D:\AFCI\Cours\8 - conteneurisation>
```



Vérifions:

```
PS D:\AFCI\Cours\8 - conteneurisation> docker images
REPOSITORY    TAG          IMAGE ID      CREATED       SIZE
PS D:\AFCI\Cours\8 - conteneurisation> |
```

Nos images ont été supprimées en local.

En résumé

Nous savons maintenant

- télécharger, démarrer et arrêter des conteneurs ;

Nous sommes aussi capable de définir une registry Docker et vous savez pourquoi on l'utilise.

Pour rappel, voici les points importants du chapitre :

- ✓ démarrage d'un conteneur avec un docker run ;
- ✓ utilisation des arguments -d et -p lors du démarrage d'un conteneur ;
- ✓ récupération d'une image depuis une registry avec la commande docker pull ;
- ✓ nettoyage du système avec docker system prune

Questions ??

Prochain chapitre: La gestion des données.