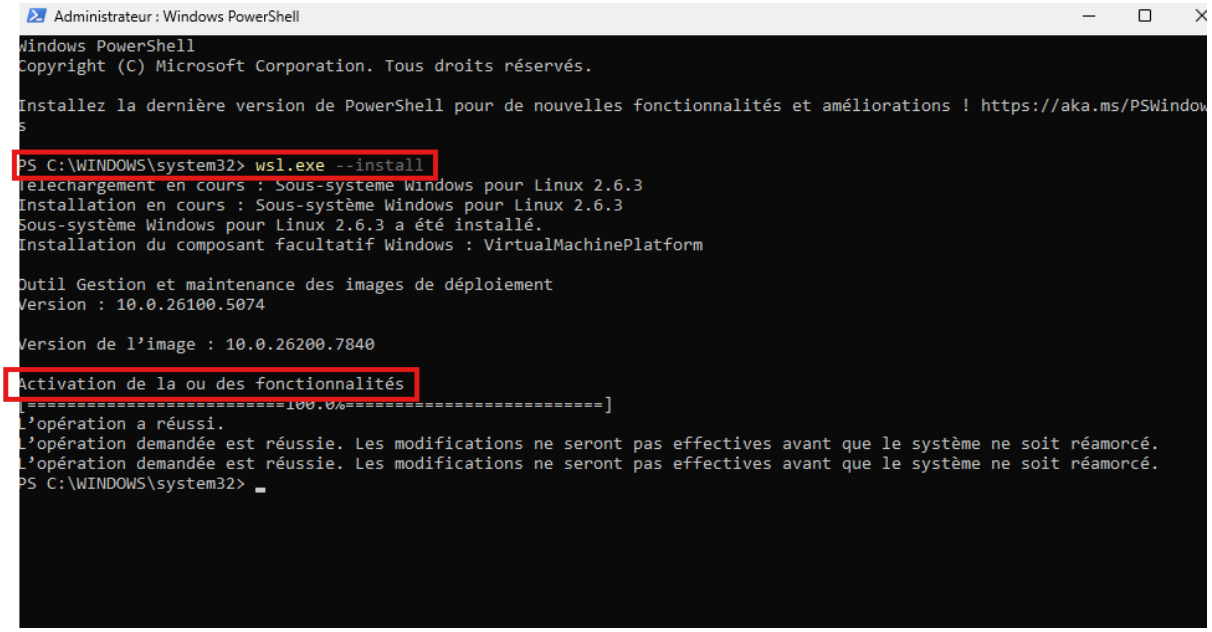
Chapitre 4 – Docker sous Windows (Docker Desktop et WSL 2)

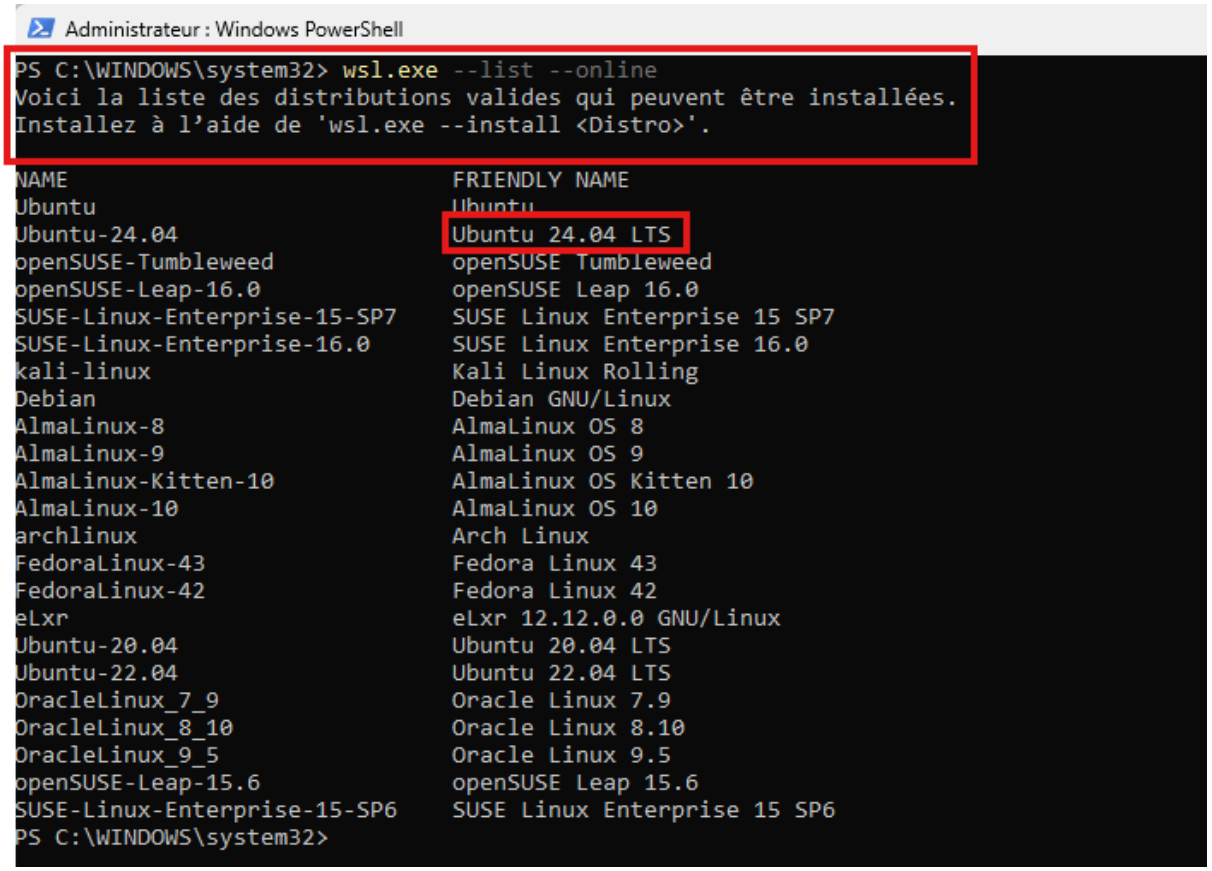
Table des Matières :

1. Installation / Activation de WSL.

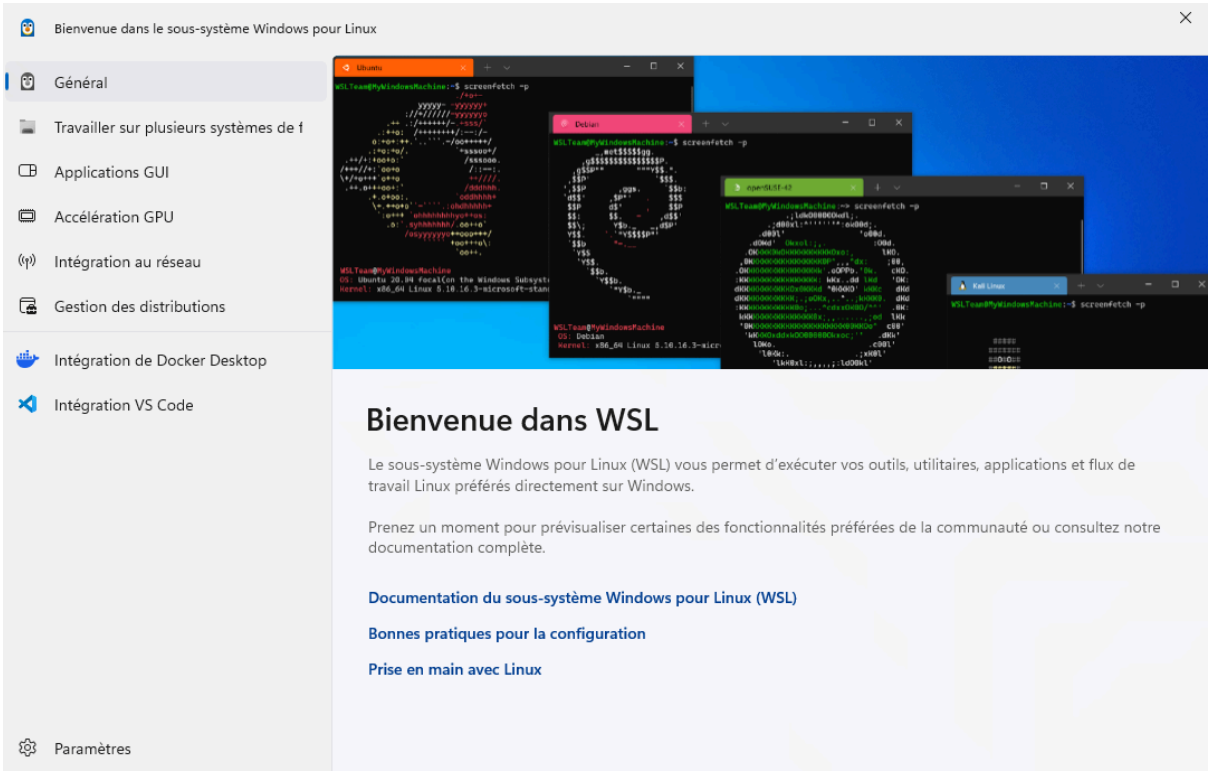
▪ Nous procédons à l'installation de WSL

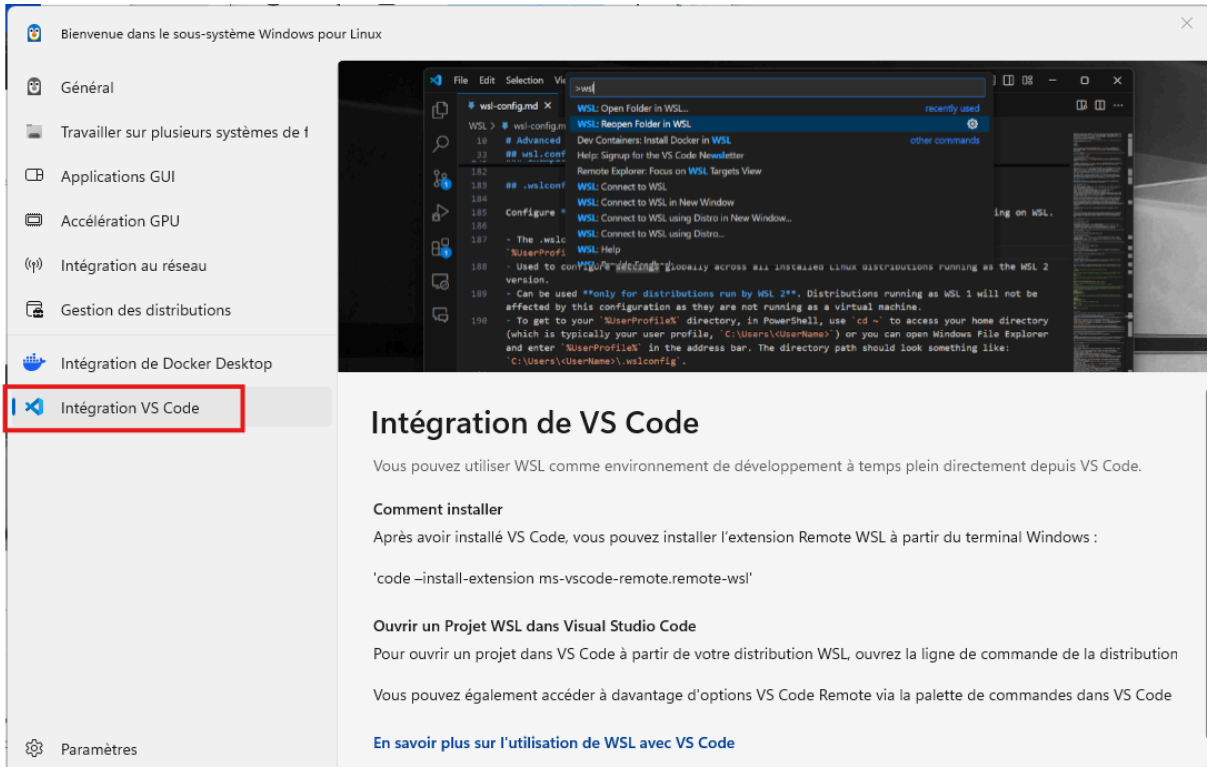


▪ Cette commande permet de consulter le catalogue officiel, ainsi de vérifier les noms exacts.

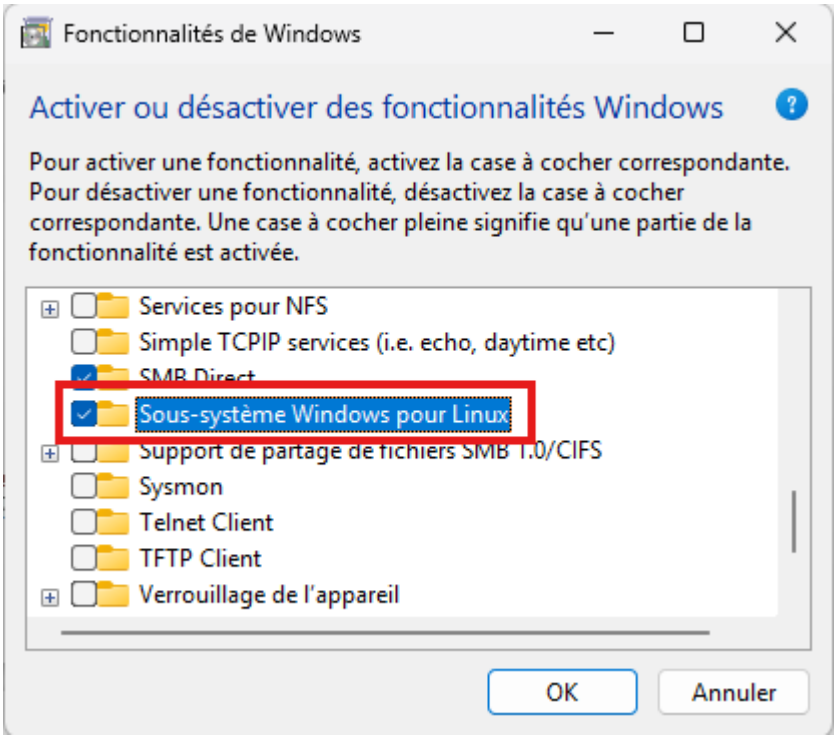


- Nous procédons par une vérification :

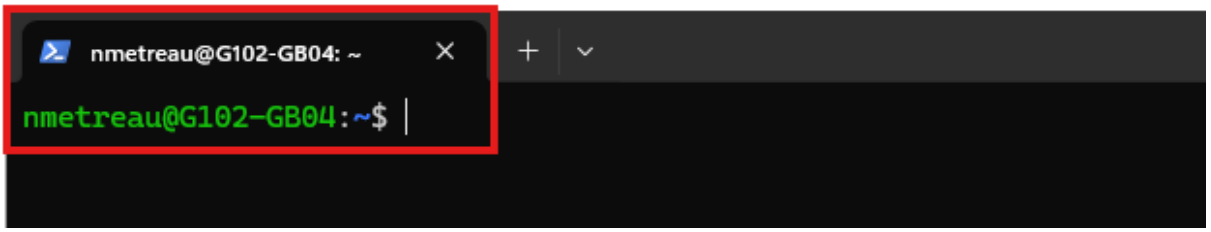


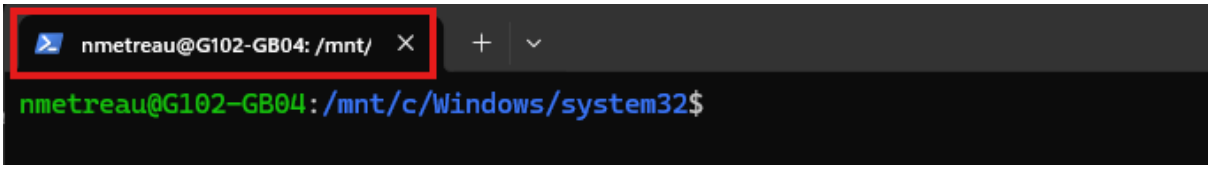


▪ Nous activons “Sous-système Windows pour Linux” afin de créer une machine virtuelle sur windows



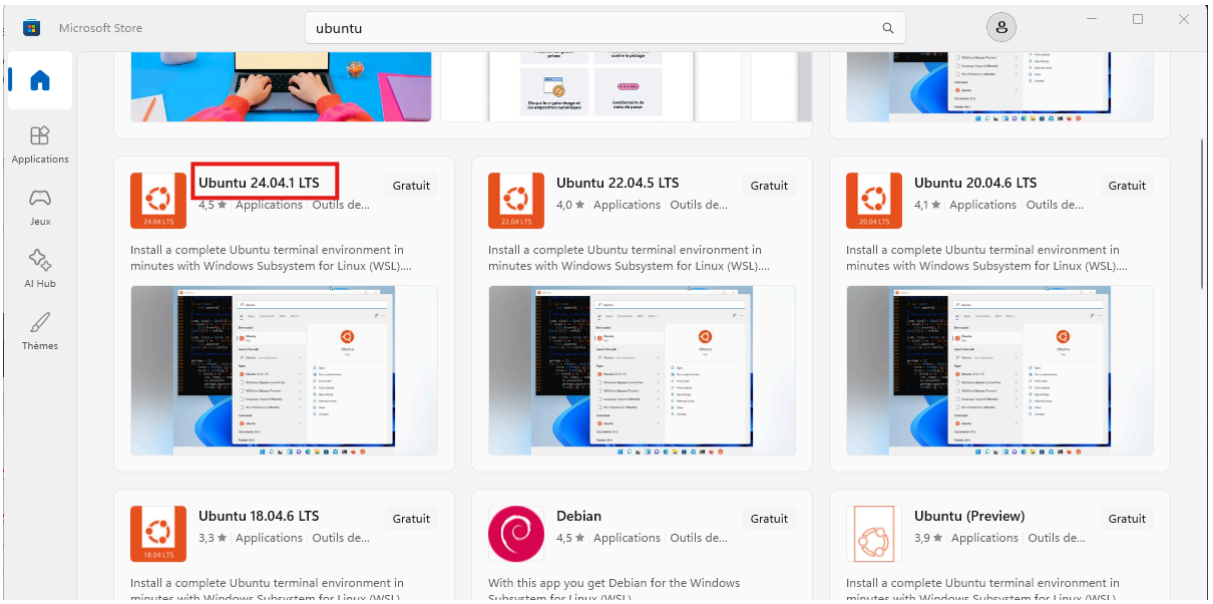
▪ Nous essayons de nous connecter sur le compte créé sur ubuntu au préalable

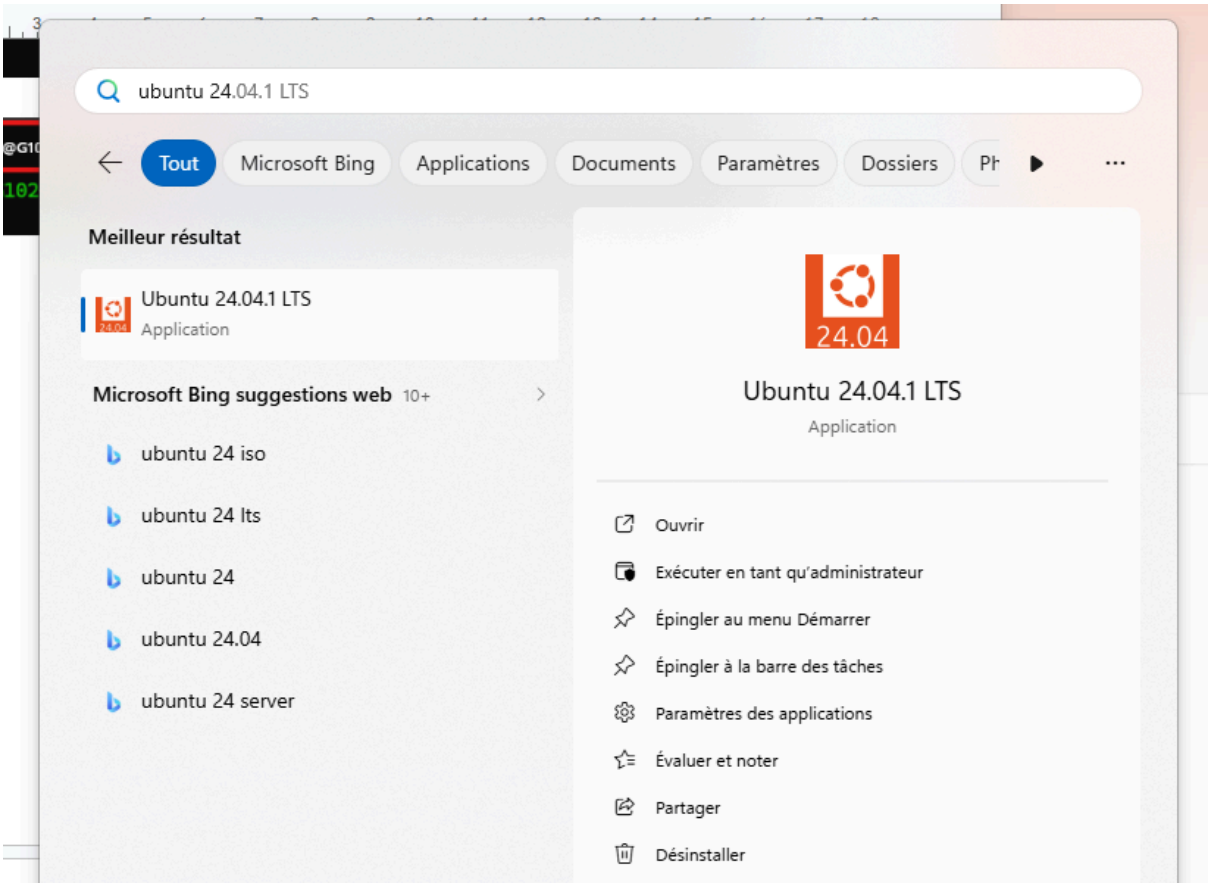


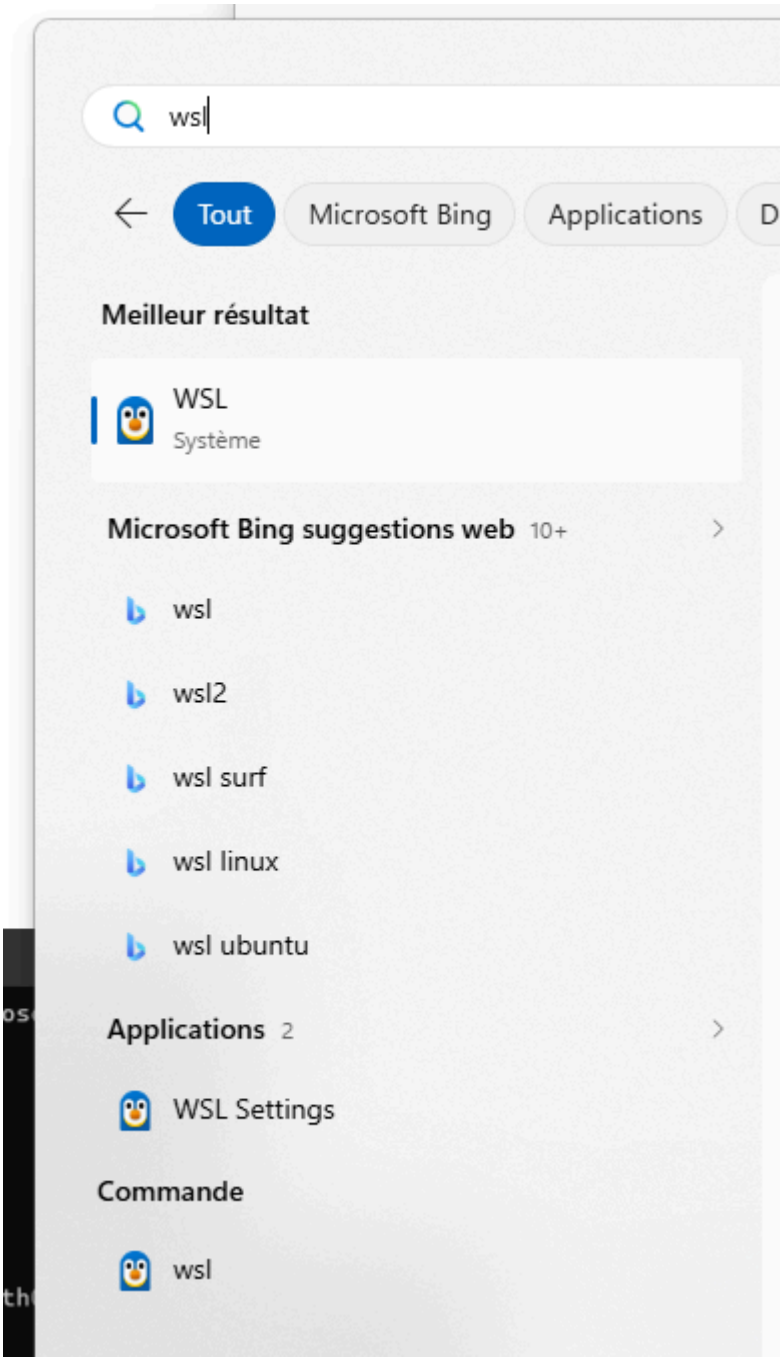


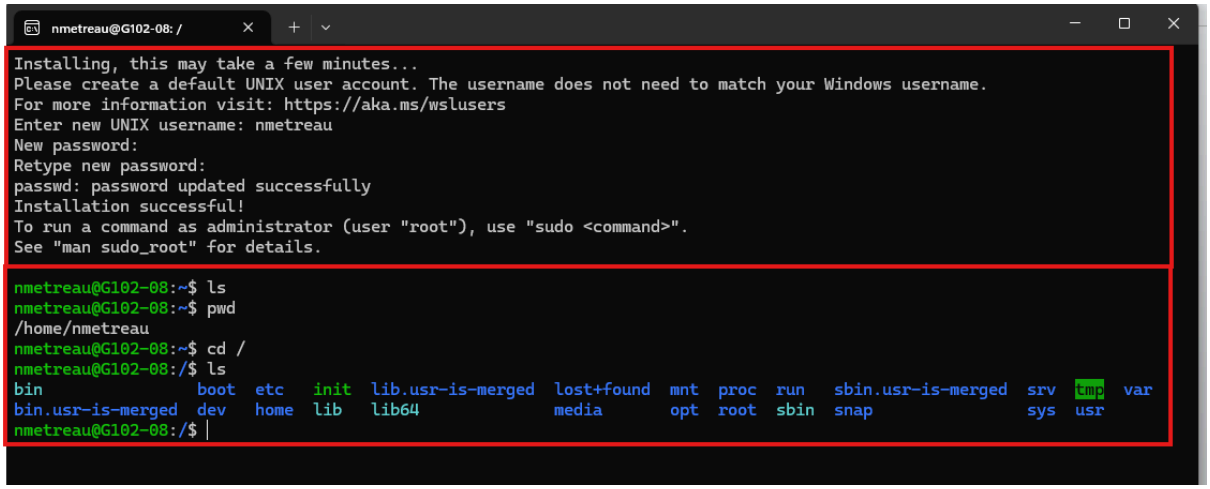
2. Installation Ubuntu.

▪ Nous installons ubuntu 24.04.1 LTS depuis le microsoft store :

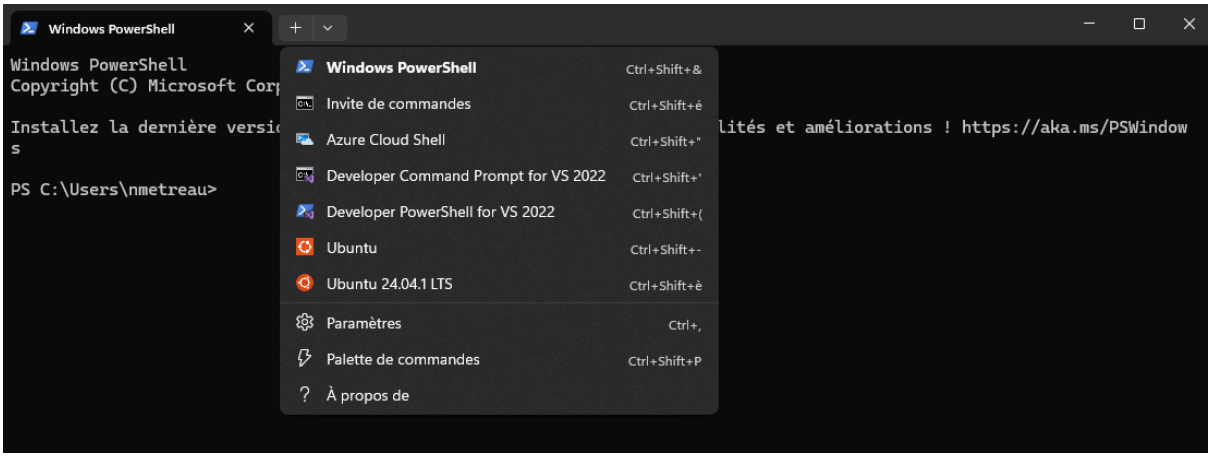


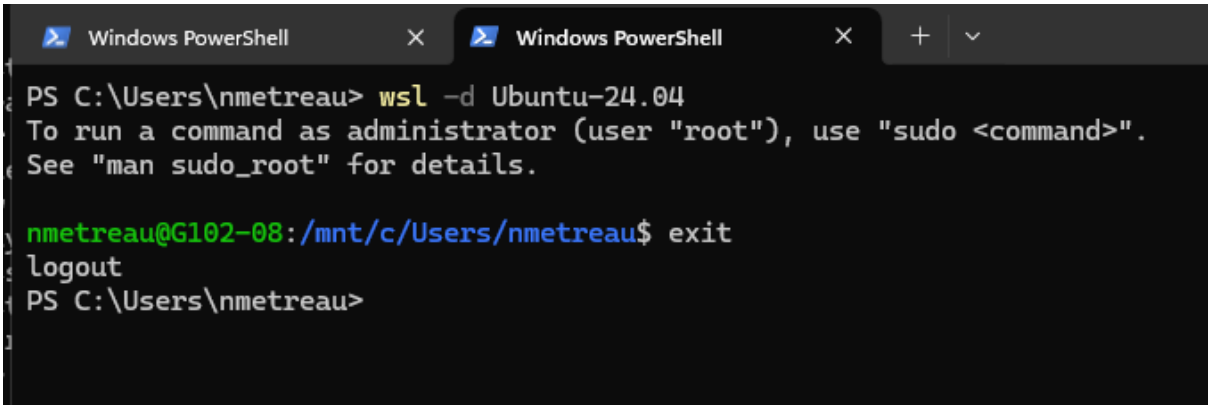



wsl
Tout
Microsoft Bing
Applications
Meilleur résultat
WSL
Système
Microsoft Bing suggestions web 10+
wsl
wsl2
wsl surf
wsl linux
wsl ubuntu
Applications 2
WSL Settings
Commande
wsl



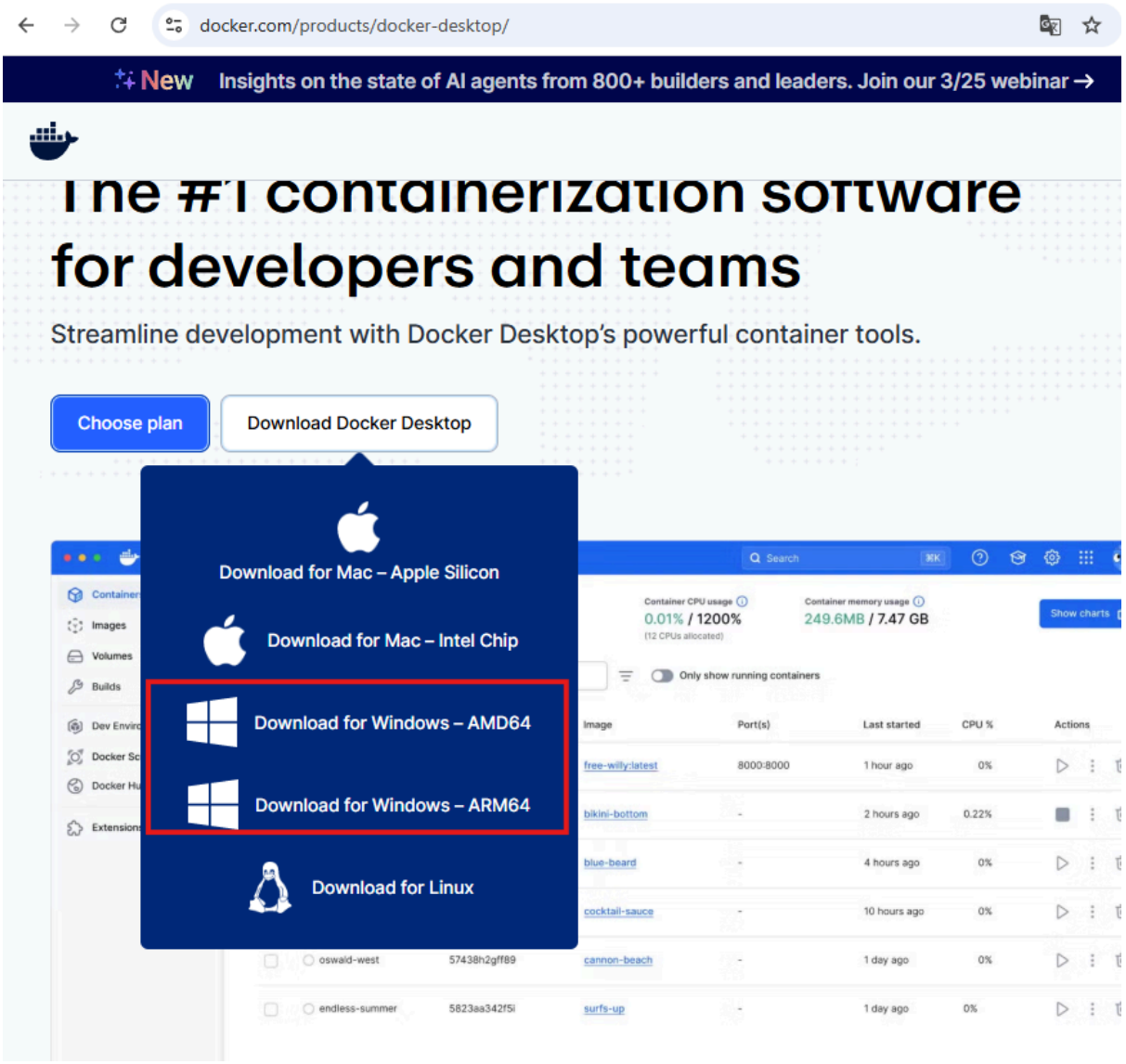
▪ Nous vérifions depuis le terminal windows :

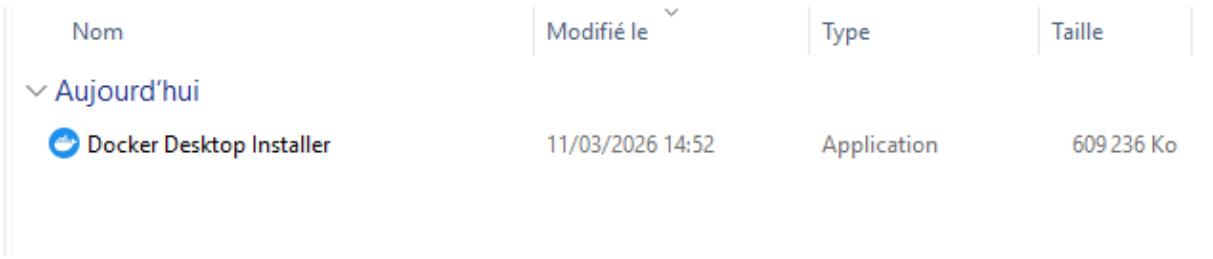




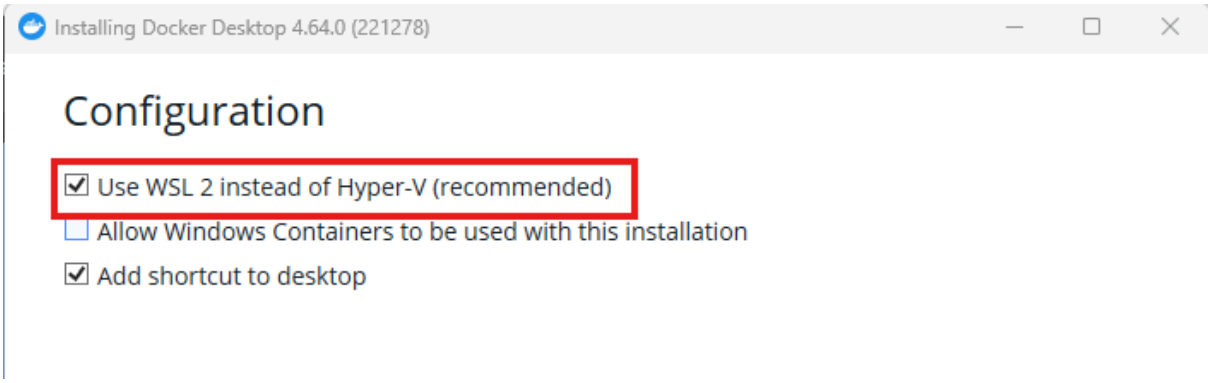
3. Installation Docker.

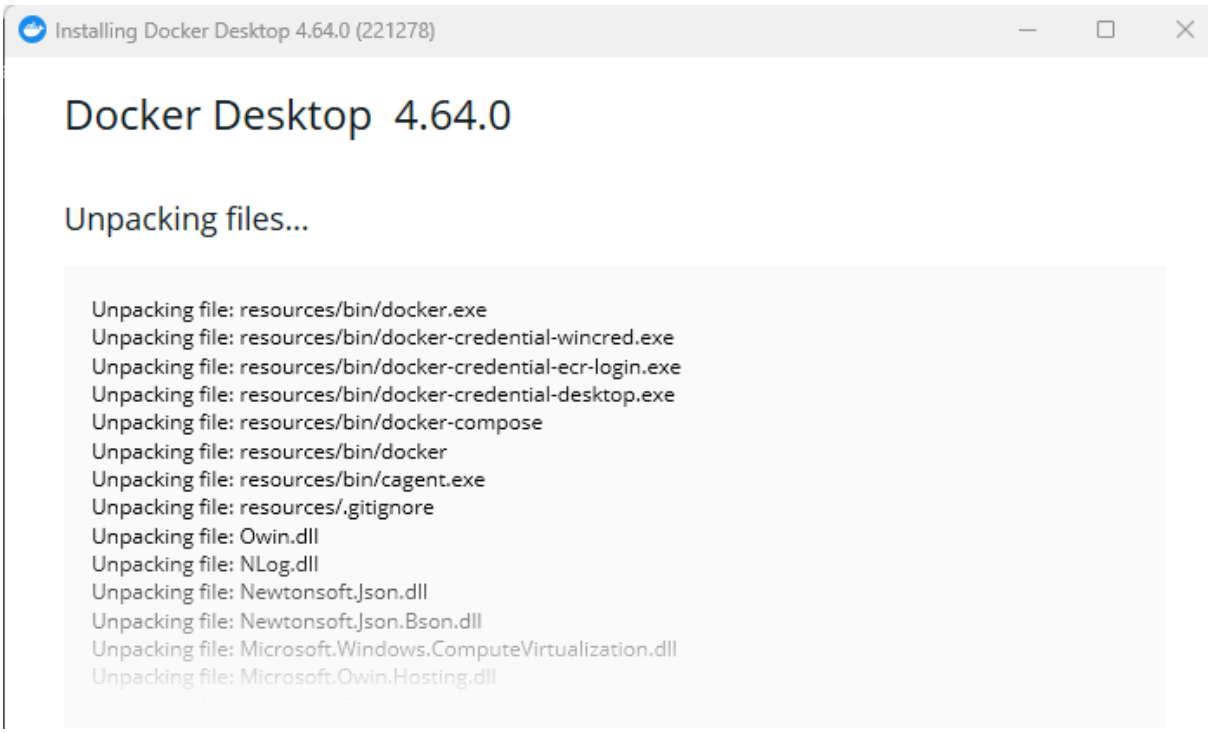
▪ Nous sélectionnons AMD64

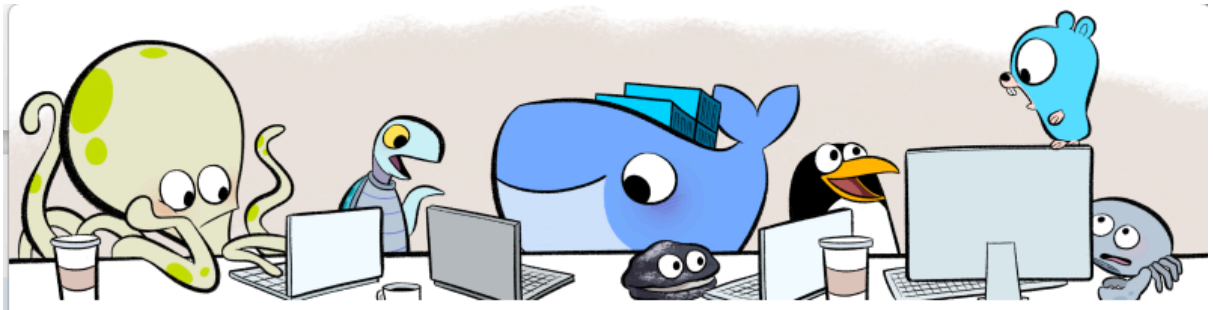




▪ Nous cochons Use WSL2… :







Docker Subscription Service Agreement

By selecting **accept**, you agree to the Subscription Service Agreement, the Docker Data Processing Agreement, the Data Privacy Policy and the Docker AI Supplemental Terms.

Commercial use of Docker Desktop at a company of more than 250 employees OR more than $10 million in annual revenue requires a paid subscription (Pro, Team, or Business). See subscription details

View Full Terms | Accept | Close

docker

Create your account

Work | Personal

Work email: metreau.nicolas@gmail.com

Username: aeaeaezz

Password: ••••••••••••

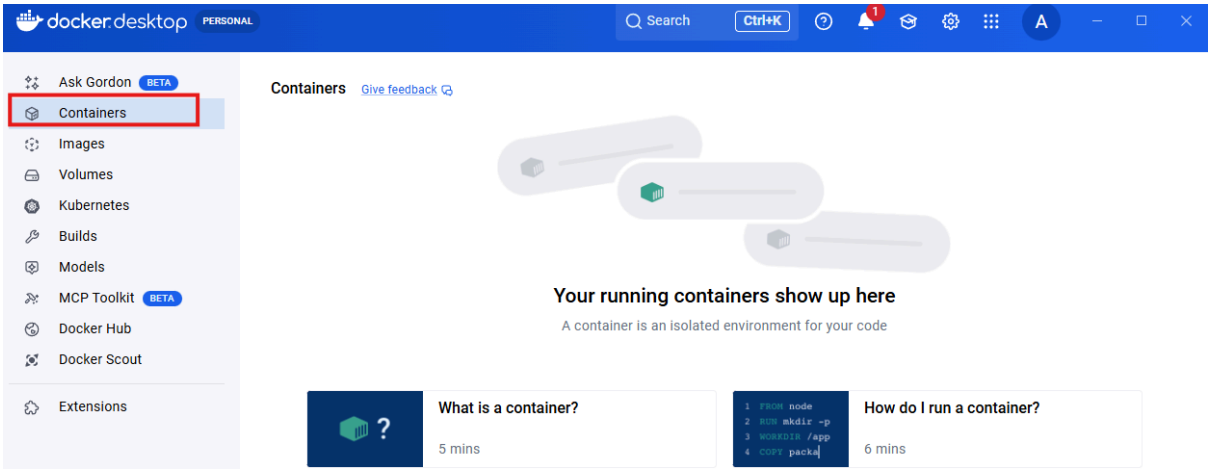
Send me occasional product updates and announcements.

Sign up

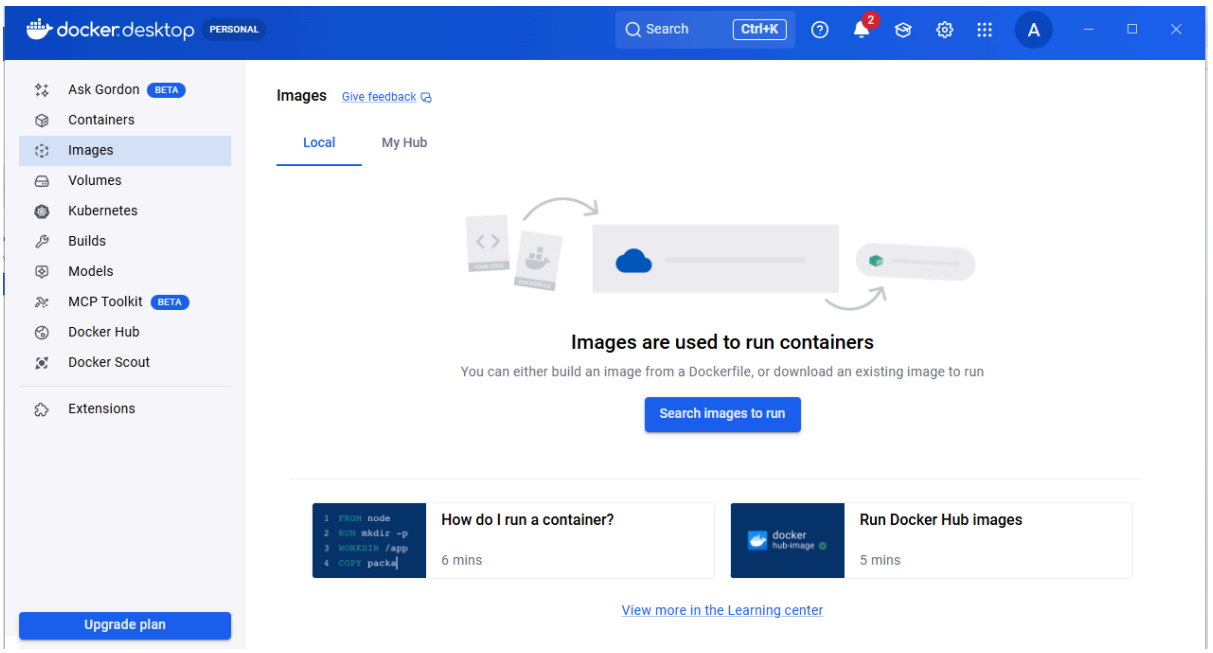
OR

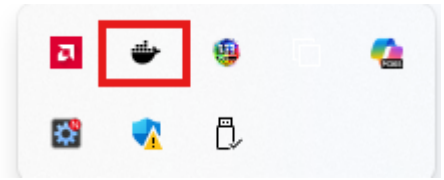
Continue with Google | Continue with GitHub

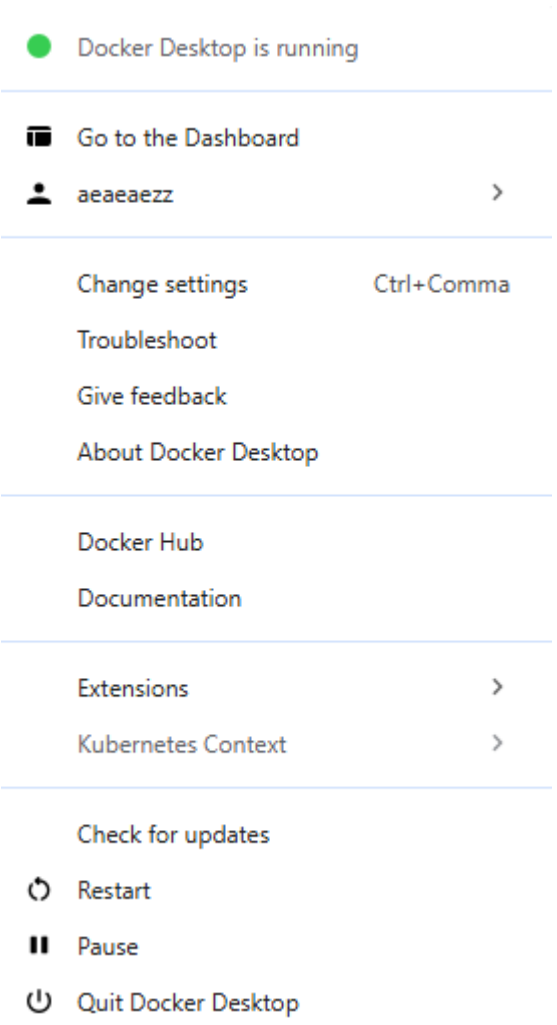
Already have an account? Sign in

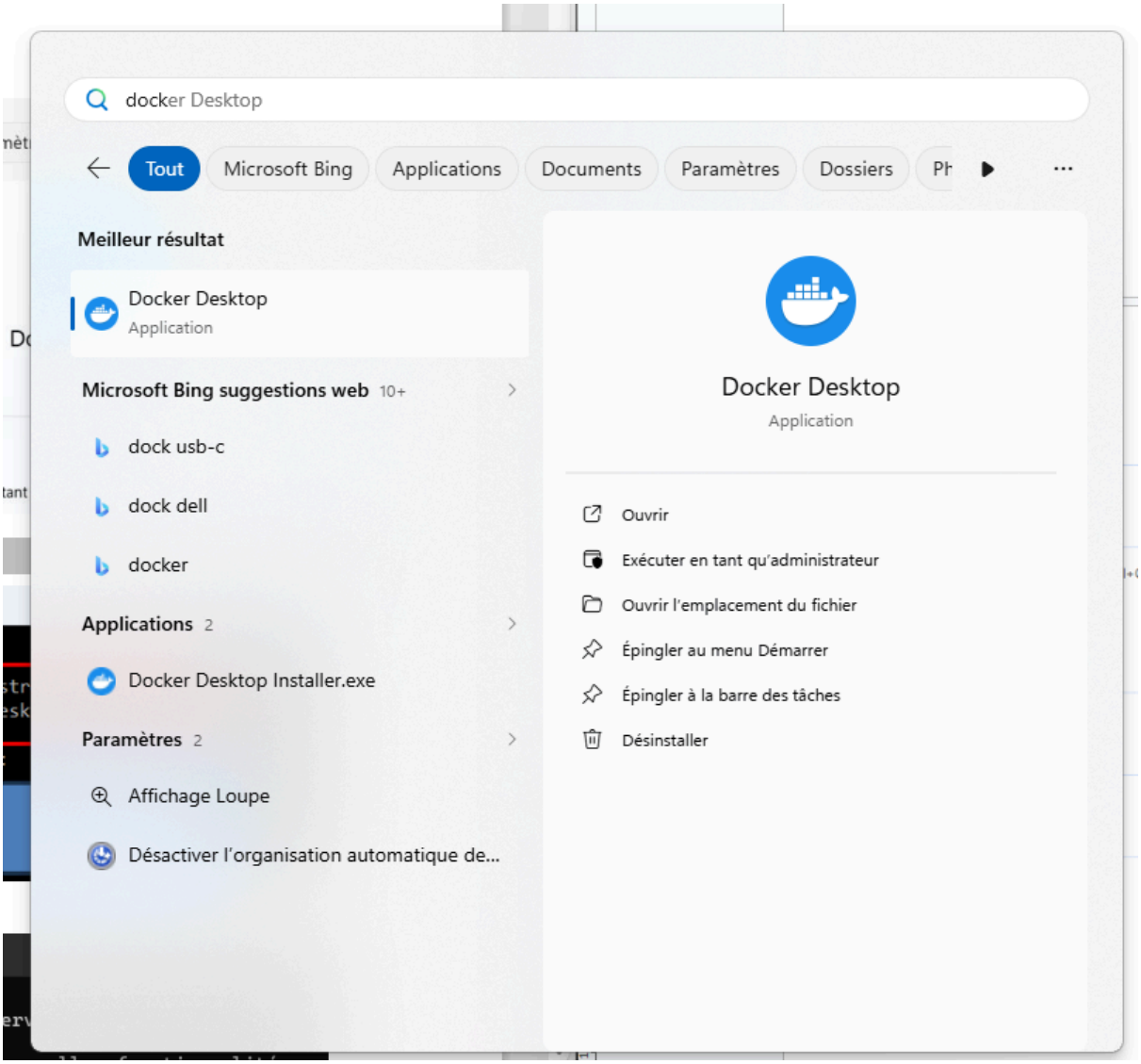


- Nous vérifions que Docker soit bien lancé :



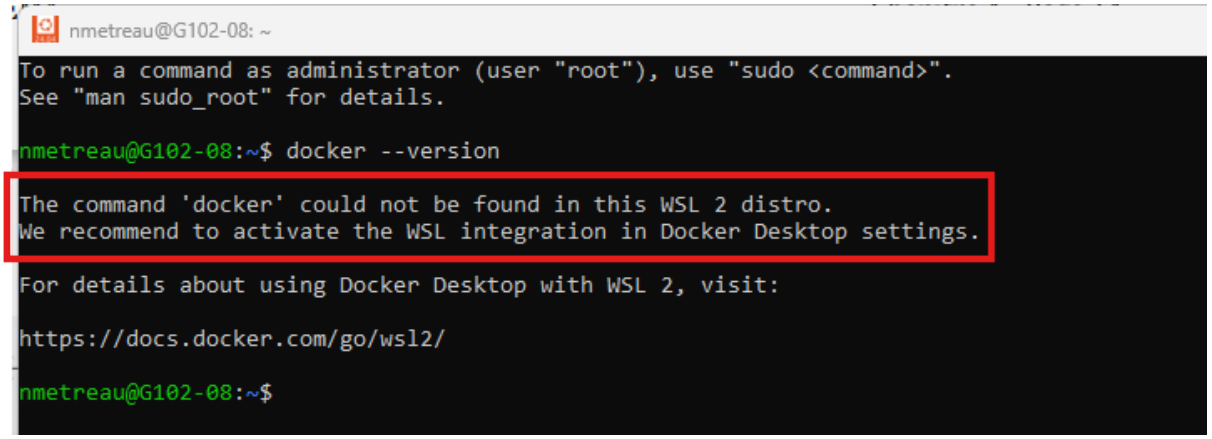



Docker Desktop is running
Go to the Dashboard
aeaeaezz
Change settings
Ctrl+Comma
Troubleshoot
Give feedback
About Docker Desktop
Docker Hub
Documentation
Extensions
Kubernetes Context
Check for updates
Restart
Pause
Quit Docker Desktop

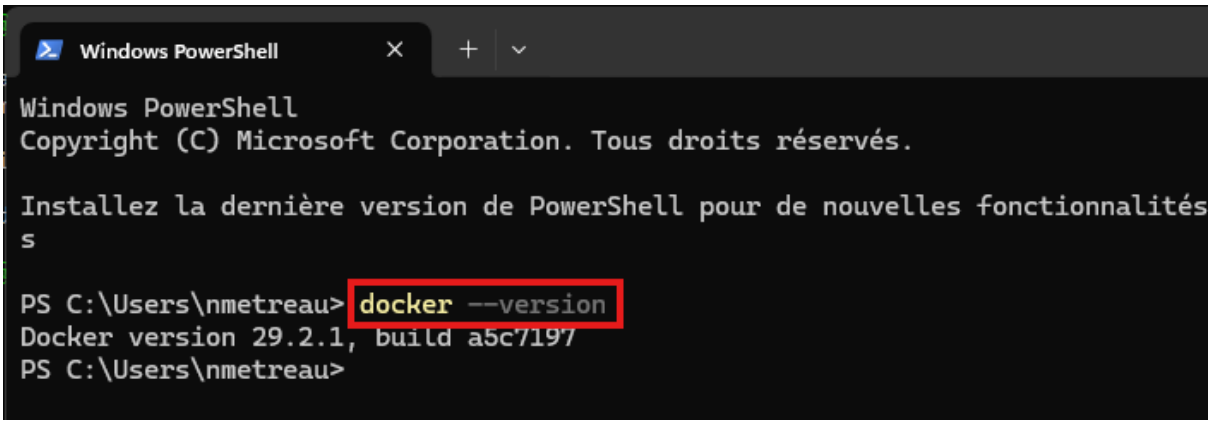

docker Desktop
Tout
Microsoft Bing
Applications
Documents
Paramètres
Dossiers
Meilleur résultat
Docker Desktop
Application
Microsoft Bing suggestions web 10+
dock usb-c
dock dell
docker
Applications 2
Docker Desktop Installer.exe
Paramètres 2
Affichage Loupe
Désactiver l'organisation automatique de...
Docker Desktop
Application
Ouvrir
Exécuter en tant qu'administrateur
Ouvrir l'emplacement du fichier
Épingler au menu Démarrer
Épingler à la barre des tâches
Désinstaller

4. Premier test.

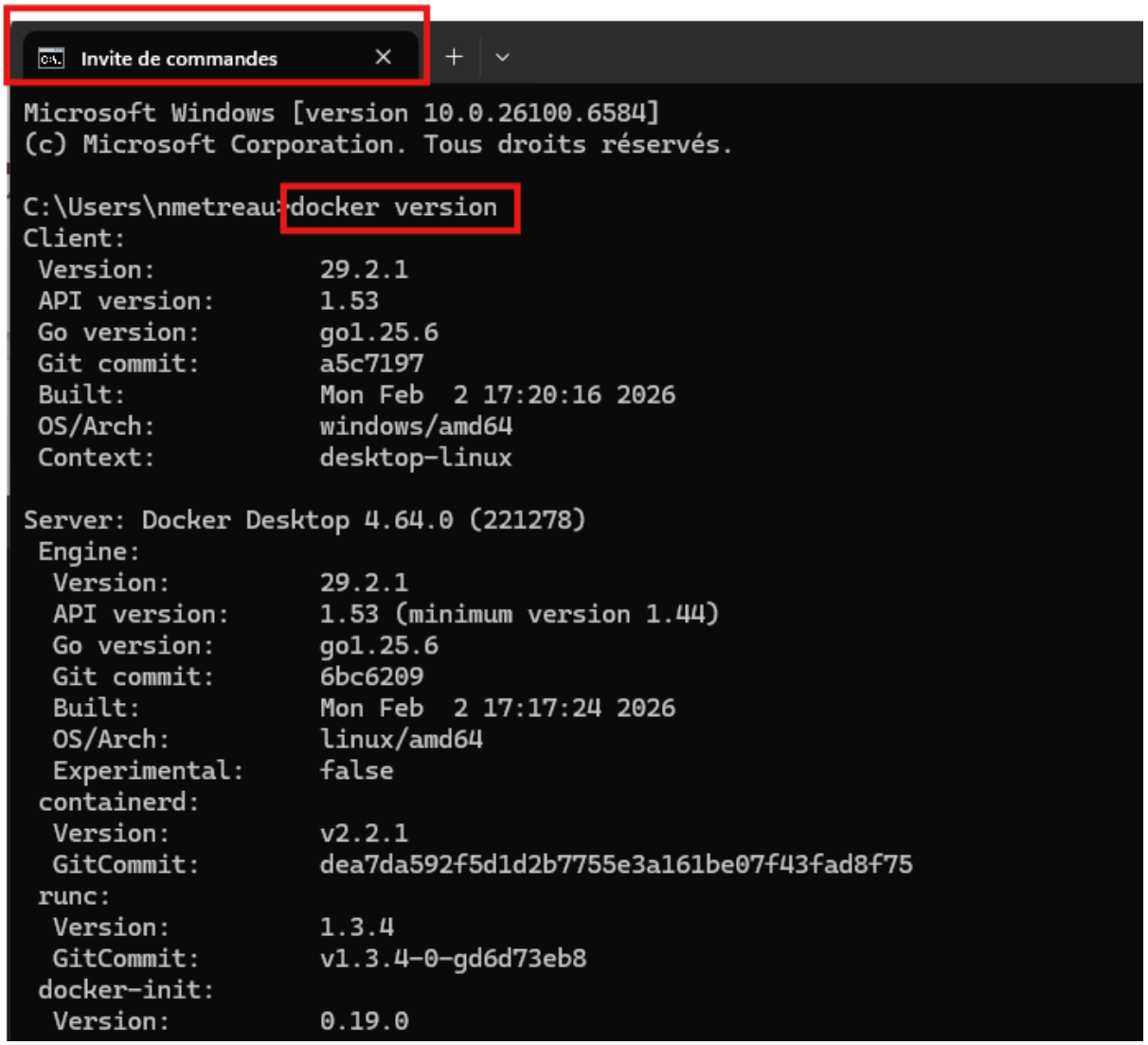
▪ Nous essayons de vérifier la version de docker sur l'application ubuntu, sauf que l'on doit activer l'intégration dans les paramètres dans docker :



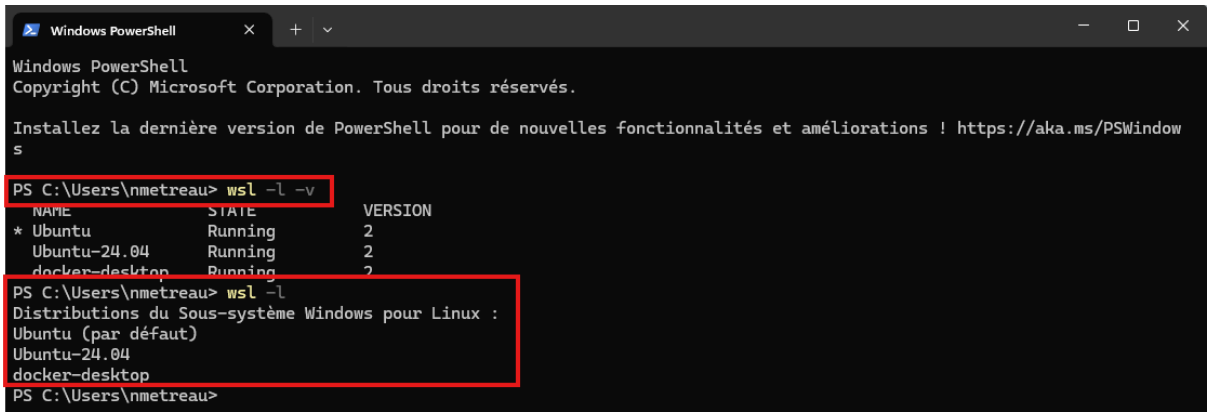
▪ Depuis le terminal Windows, nous vérifions la version de docker :



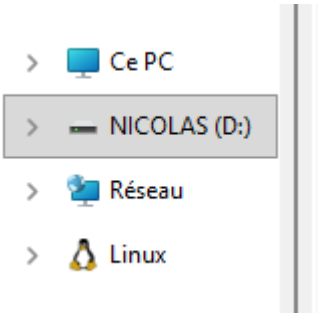
▪Nous vérifions, depuis le terminal, la version de docker :



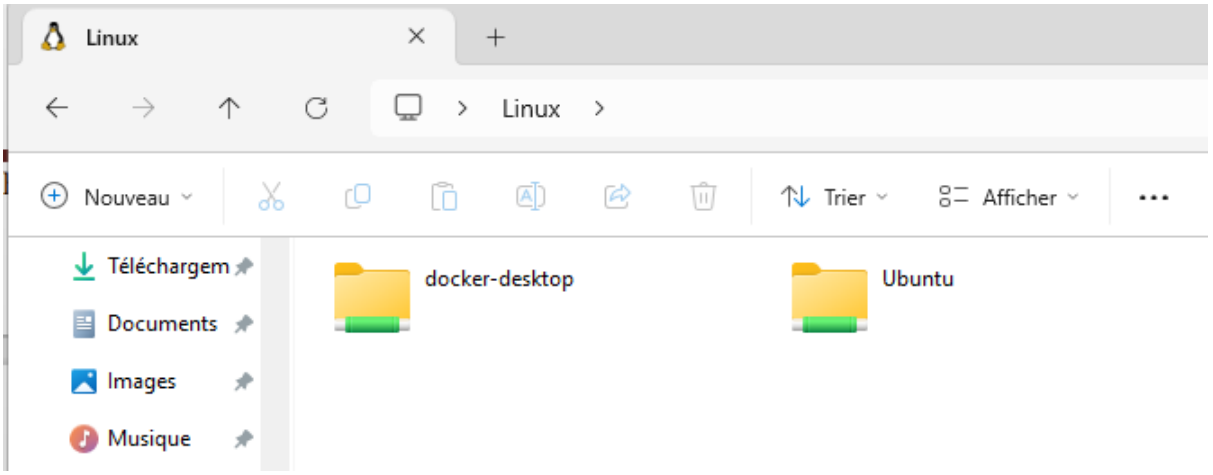
▪



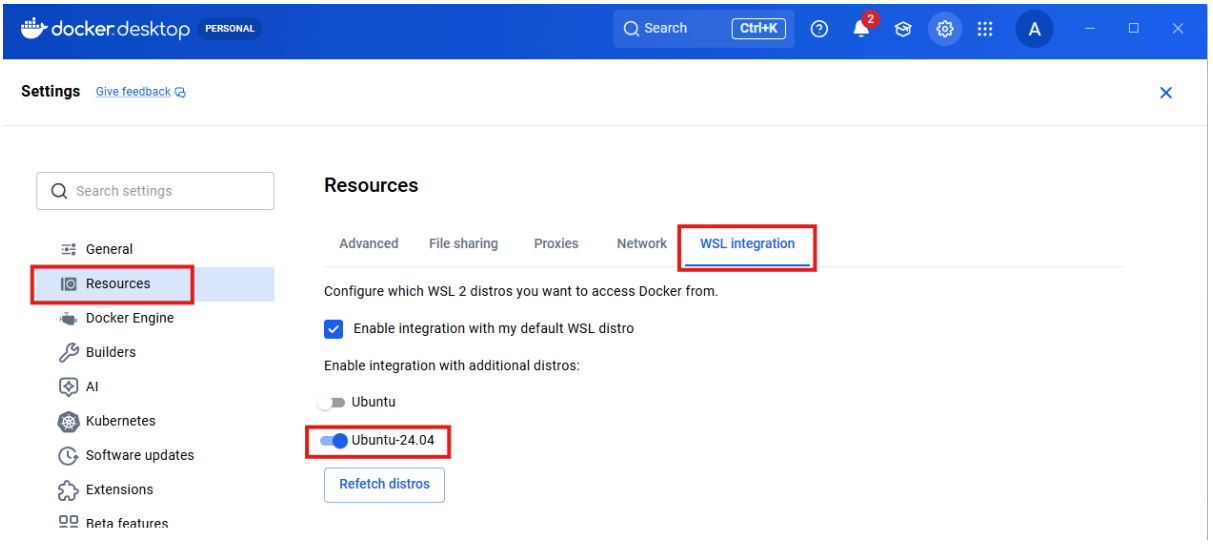
▪ Nous vérifions la disponibilité de l'environnement virtuel Linux



▪ Nous vérifions la disponibilité des répertoires :

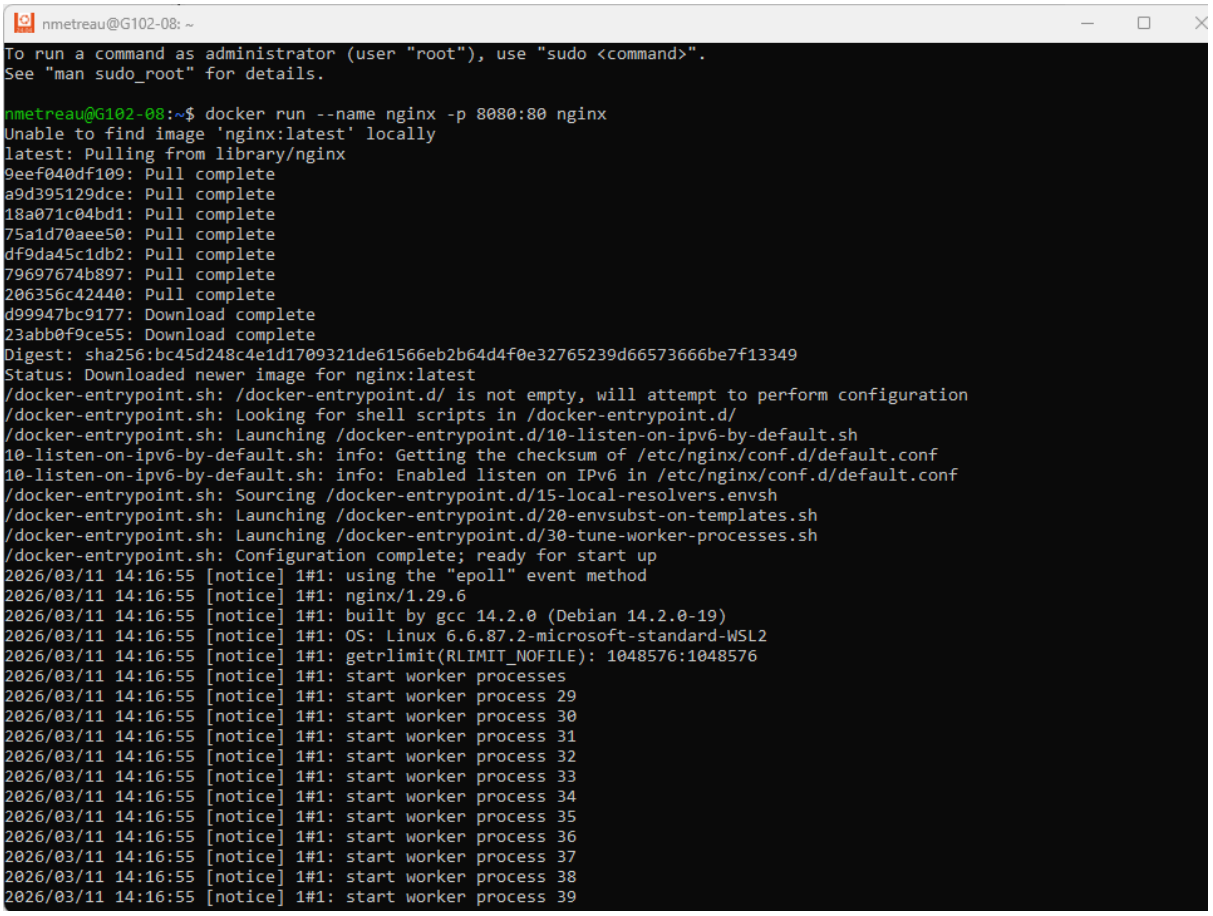


▪ Dans les intégrations de WSL, nous cochons la case Ubuntu 24.04 :

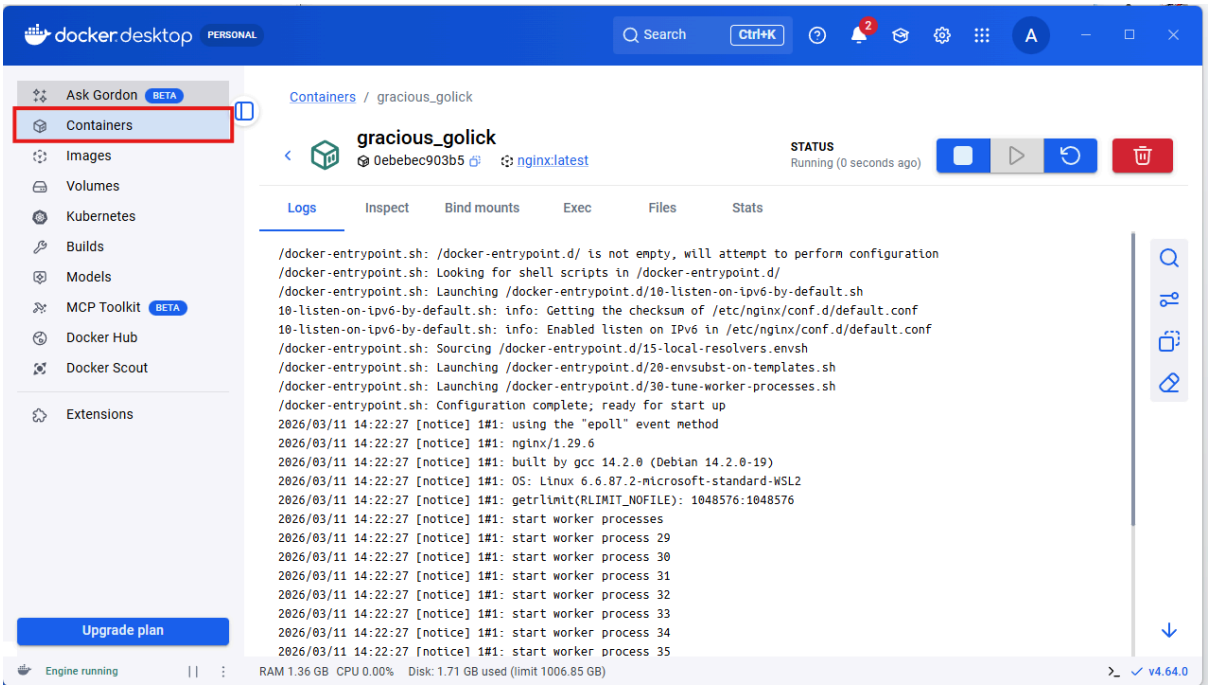


5. Deuxième test.

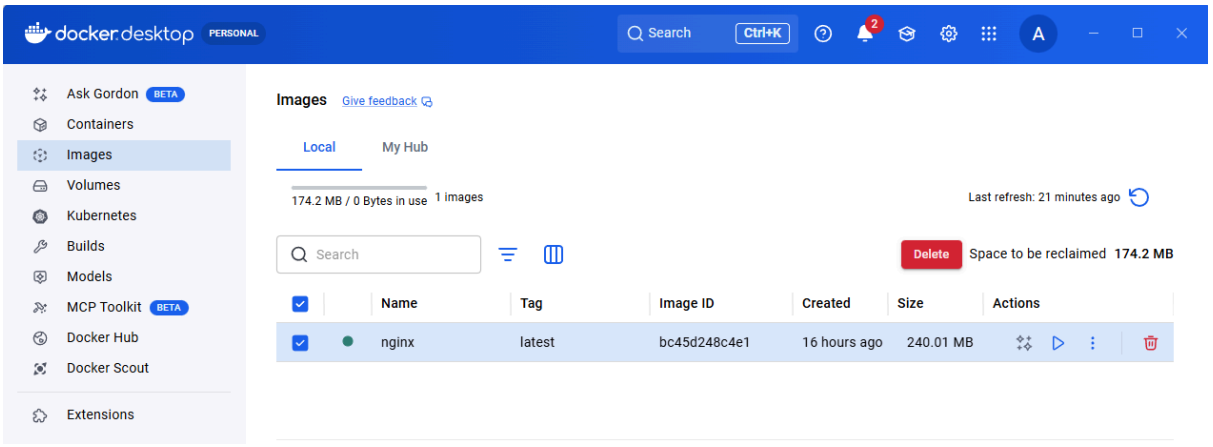
▪ Création d'un conteneur ayant pour image nginx.



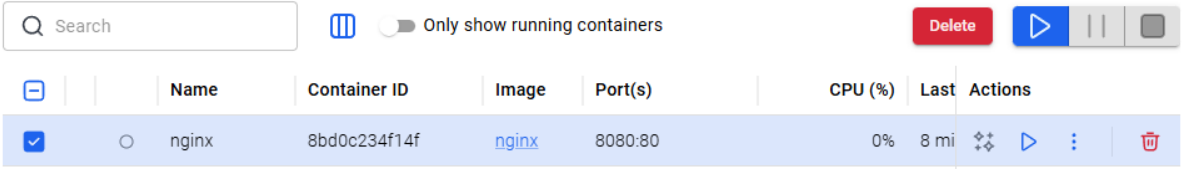
▪ L'instance a été créée de cette image qui est le conteneur



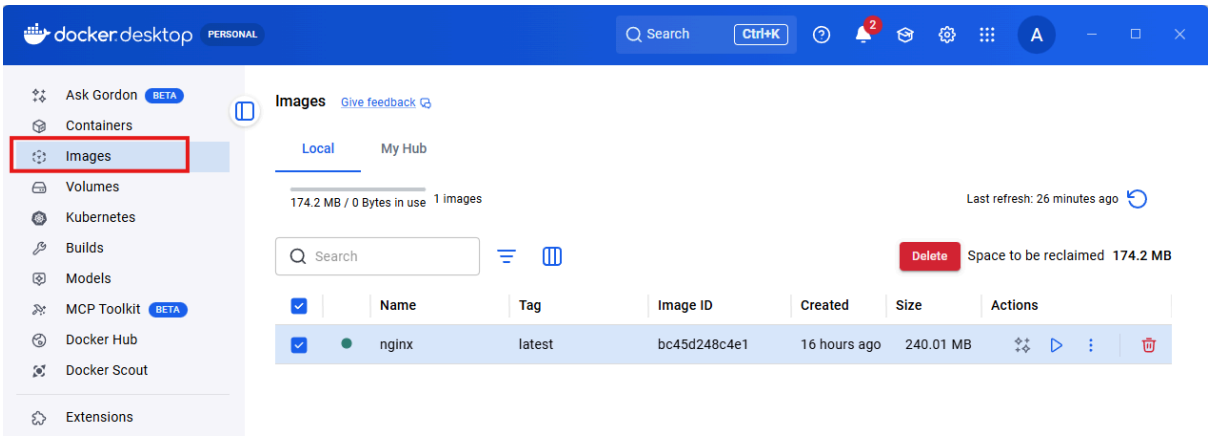
- Nous arrêtons le conteneur
  - On doit alors actionner la touche stop



- Pour delete l'image, nous devons d'abord laisser apparaître l'icône de flèche "play" :

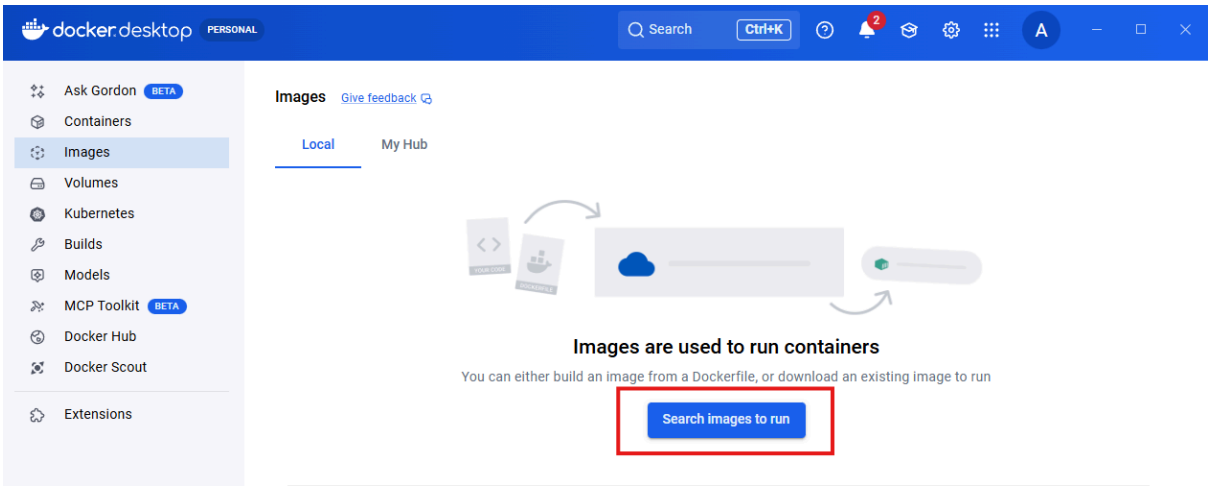


- Nous pouvons delete l'image en cliquant sur l'icône de la poubelle :

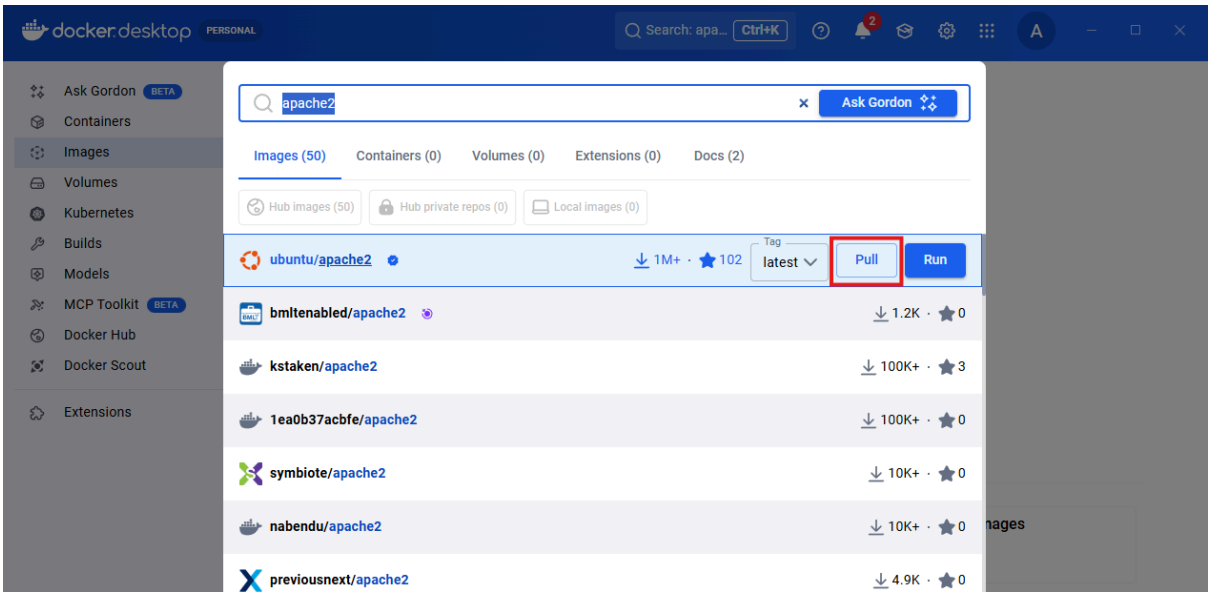


6. Troisième test.

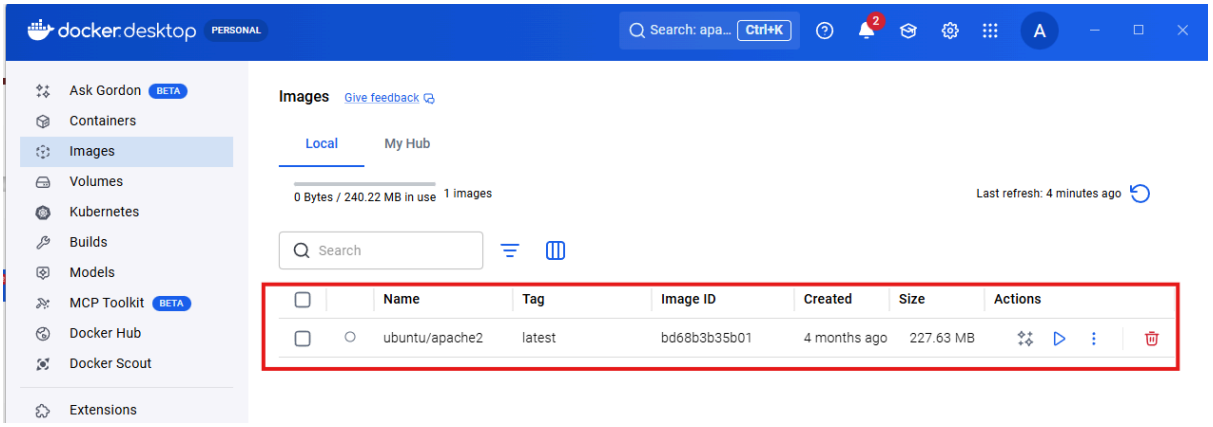
▪ Nous cliquons sur search images to run :



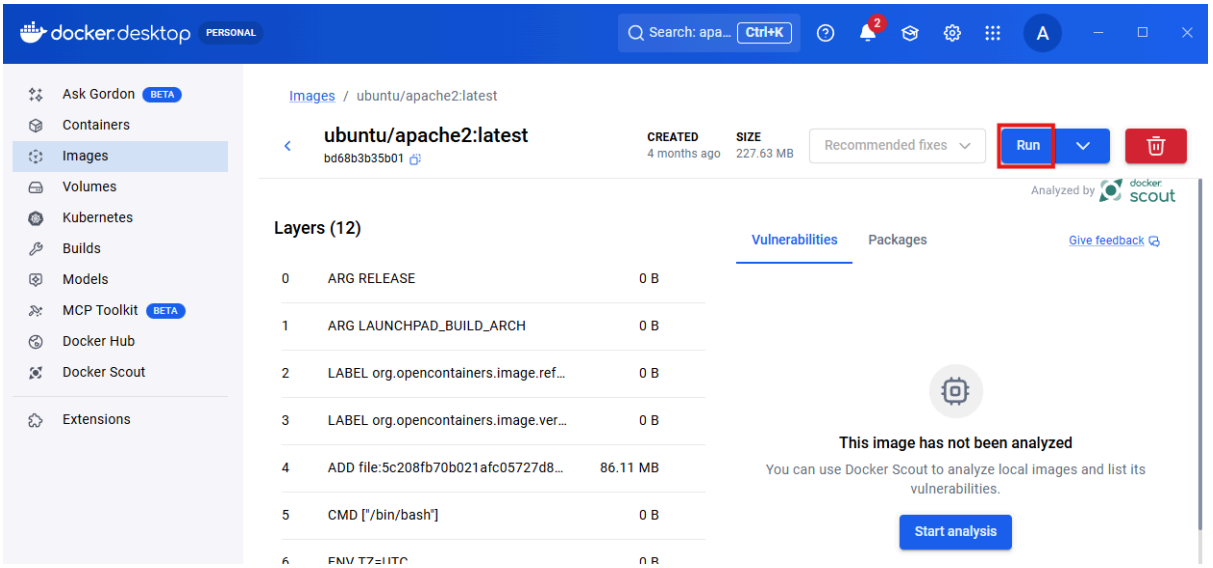
▪ Nous cliquons sur pull afin de prendre l'image de ubuntu/apache2 :



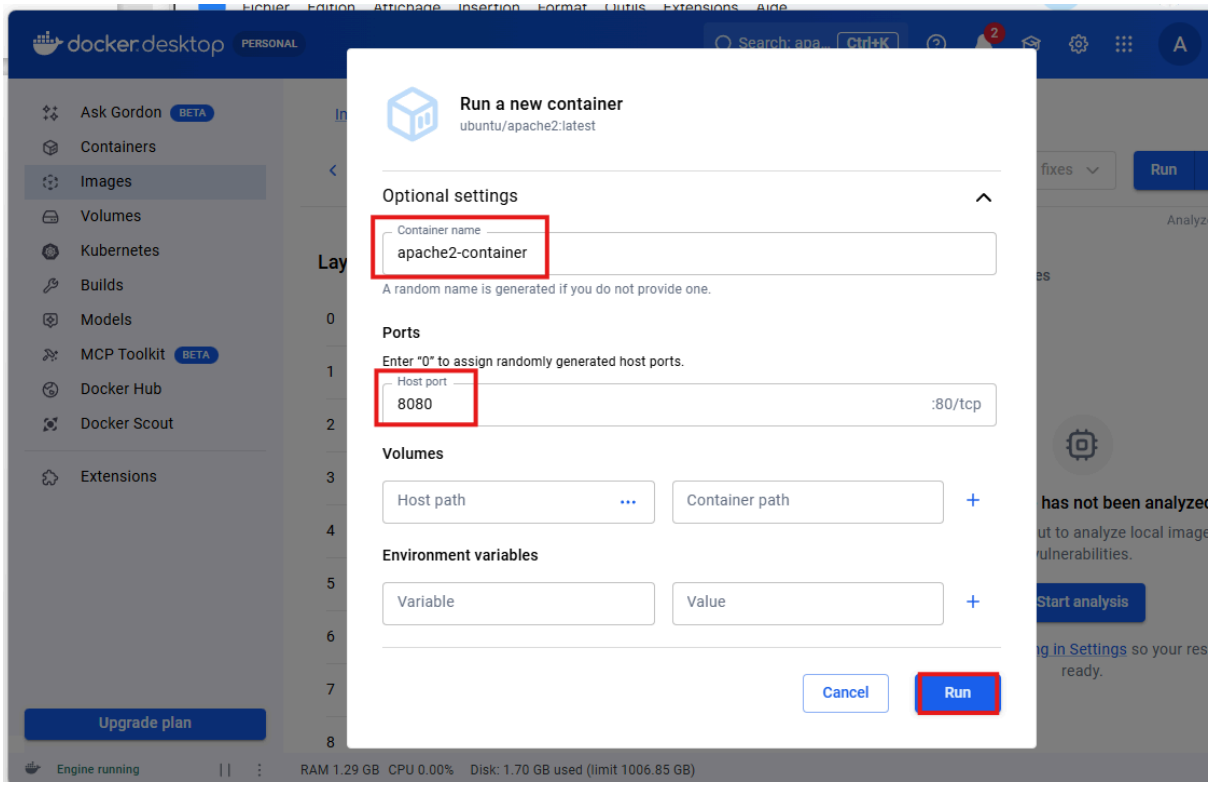
▪ Nous constatons sa présence :



- Nous lançons l'image

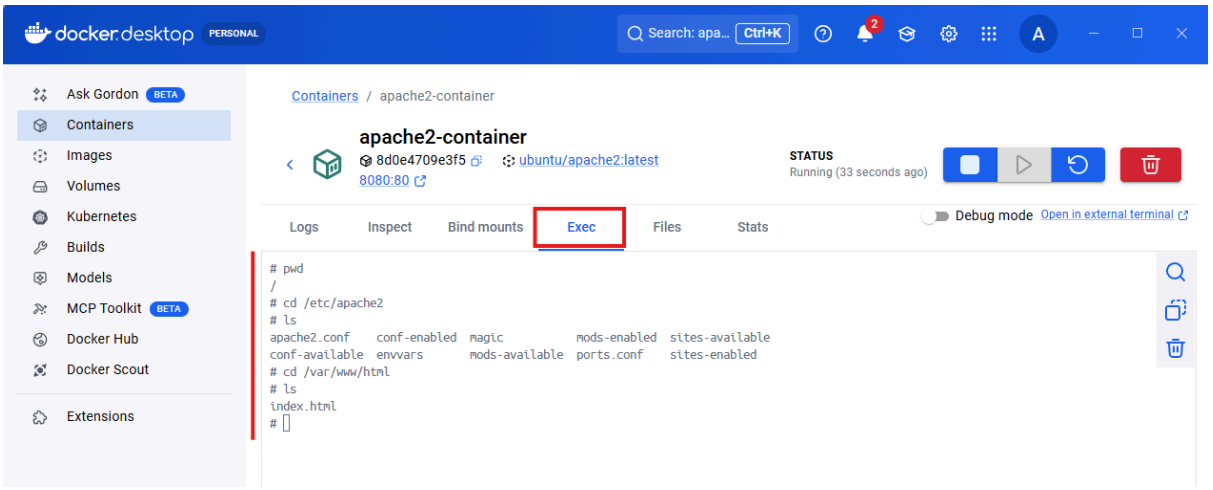


- Nous constatons son nom ainsi que son port d'écoute :

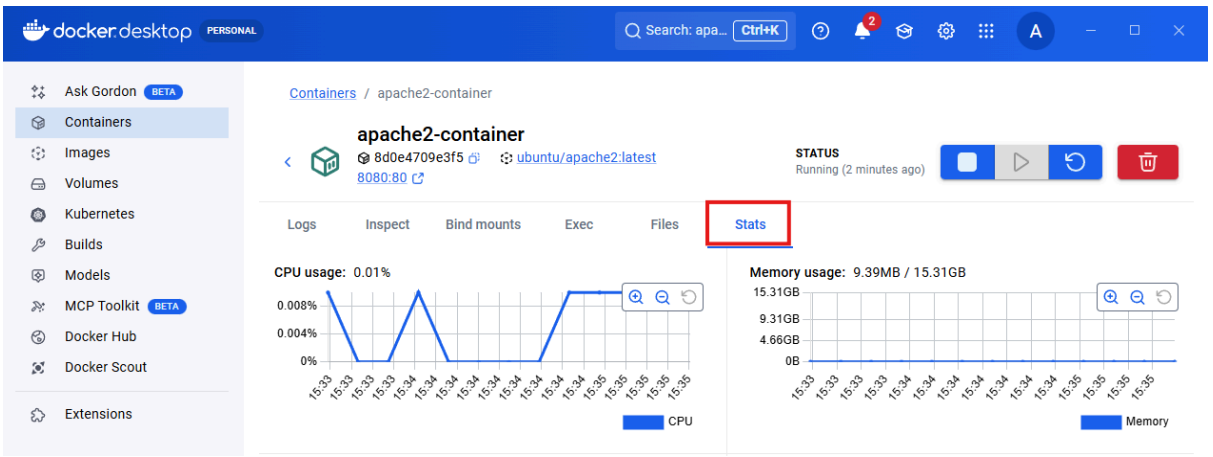


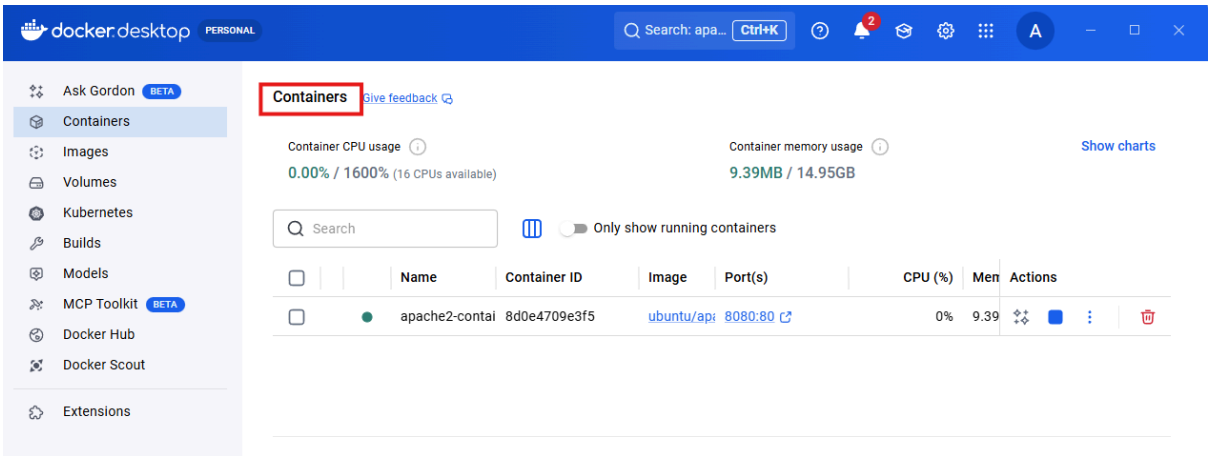
- Nous vérifions les logs dans l'onglet conteneur :

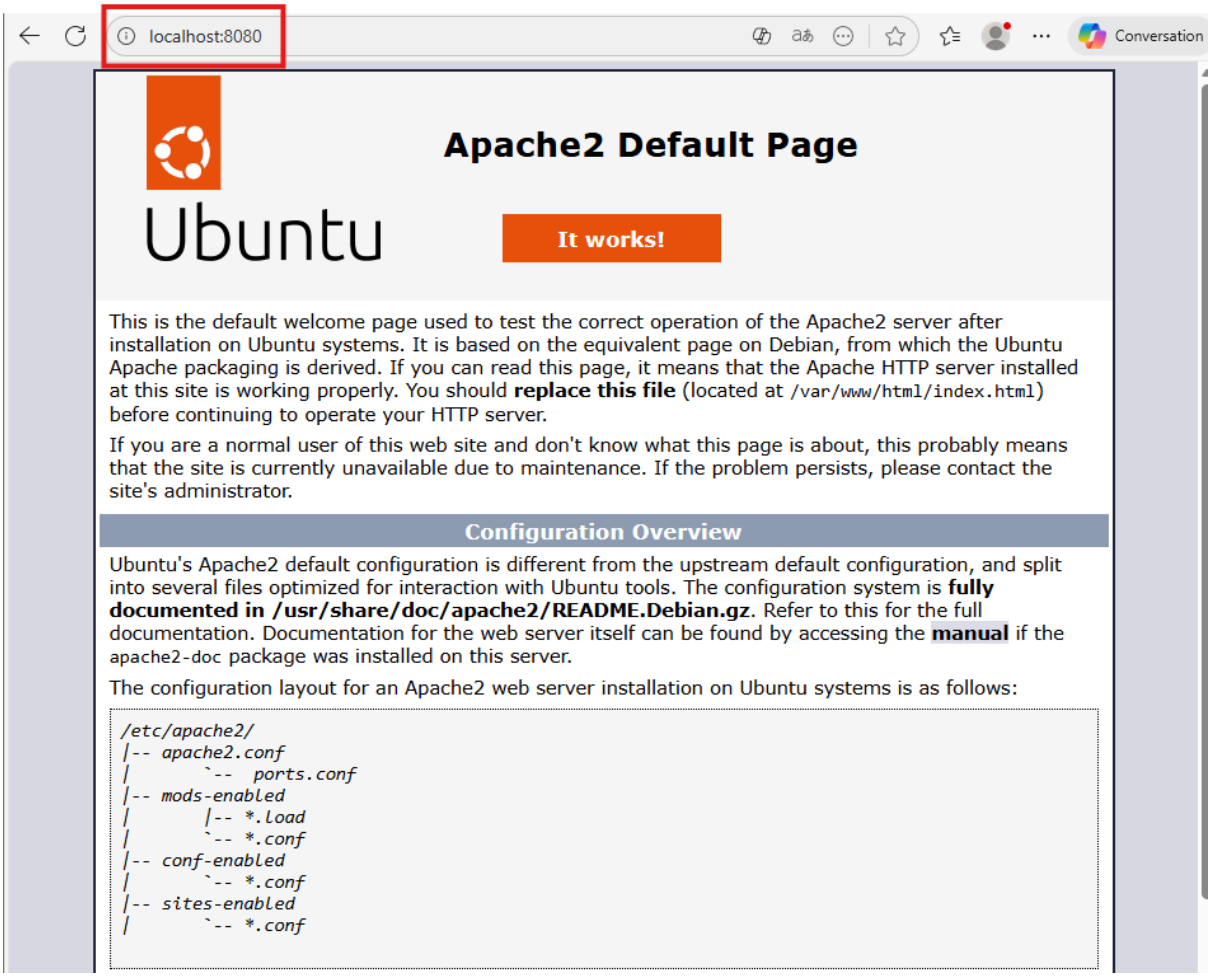
-



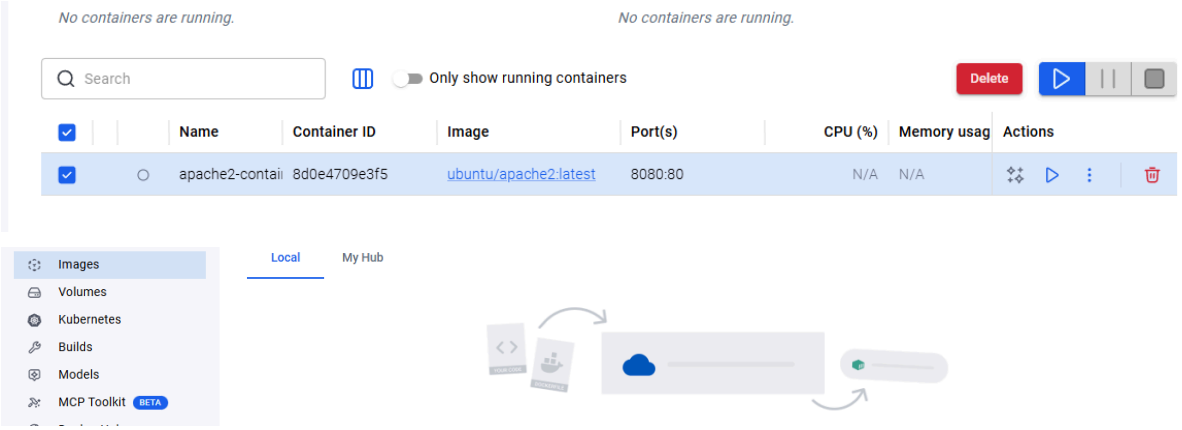
-



- 
docker desktop
PERSONAL
Search: apa... Ctrl+K
Ask Gordon BETA
Containers
Images
Volumes
Kubernetes
Builds
Models
MCP Toolkit BETA
Docker Hub
Docker Scout
Extensions
Containers
Give feedback
Container CPU usage
0.00% / 1600% (16 CPUs available)
Container memory usage
9.39MB / 14.95GB
Show charts
Search
Only show running containers
Name
Container ID
Image
Port(s)
CPU (%)
Mem
Actions
apache2-contai
8d0e4709e3f5
ubuntu/apa
8080:80
0%
9.39

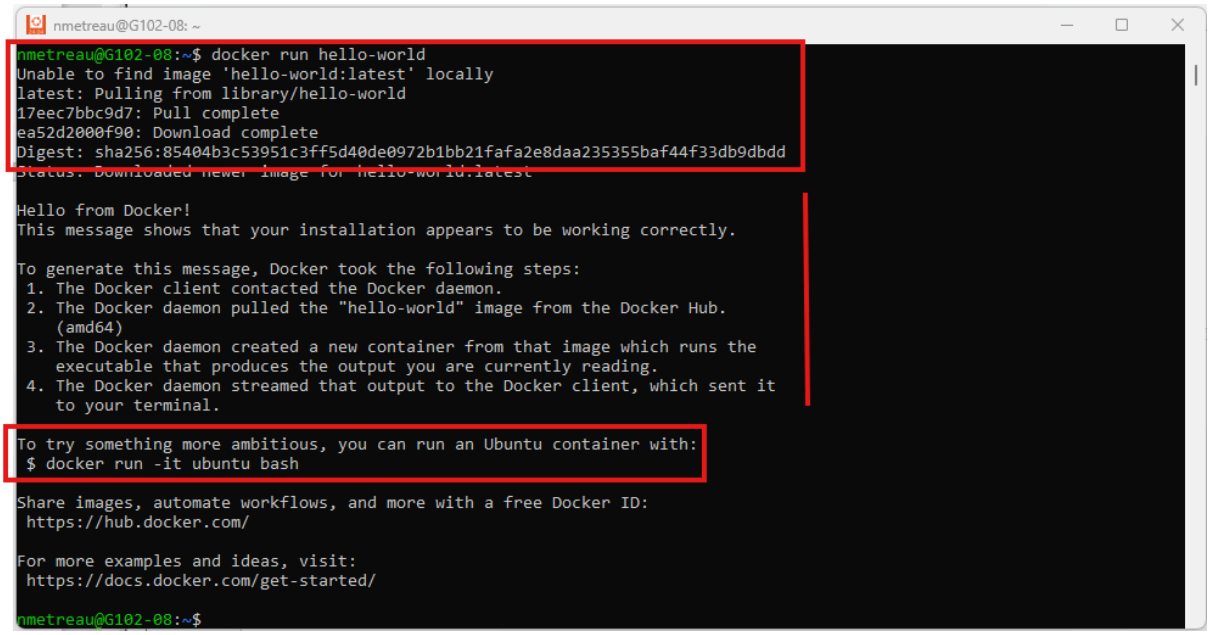
- 
localhost:8080
Conversation
Apache2 Default Page
Ubuntu
It works!
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should replace this file (located at /var/www/html/index.html) before continuing to operate your HTTP server.
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.
Configuration Overview
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is fully documented in /usr/share/doc/apache2/README.Debian.gz. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the manual if the apache2-doc package was installed on this server.
The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:
/etc/apache2/
|-- apache2.conf
| `-- ports.conf
|-- mods-enabled
| |-- *.load
| `-- *.conf
|-- conf-enabled
| `-- *.conf
|-- sites-enabled
| `-- *.conf

- Nous arrêtons et supprimons le container puis l'image.

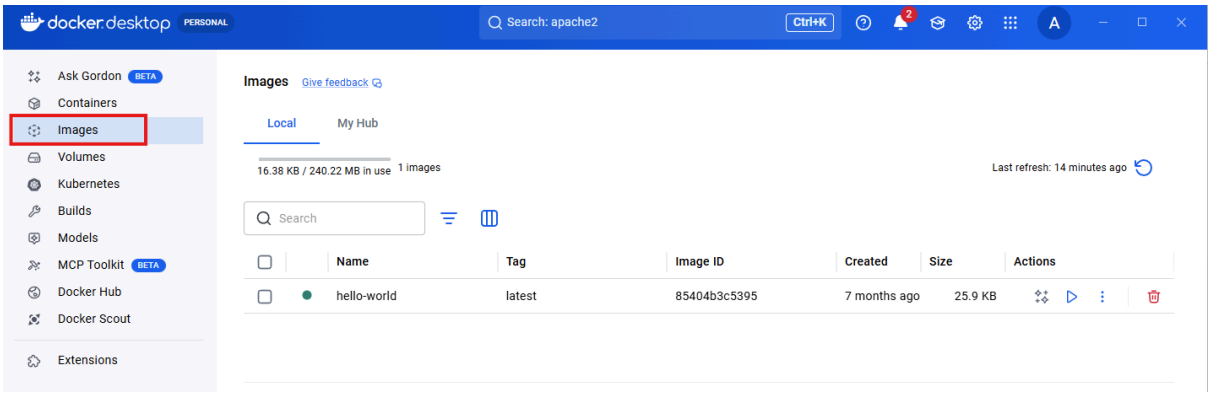


7. Quatrième test.

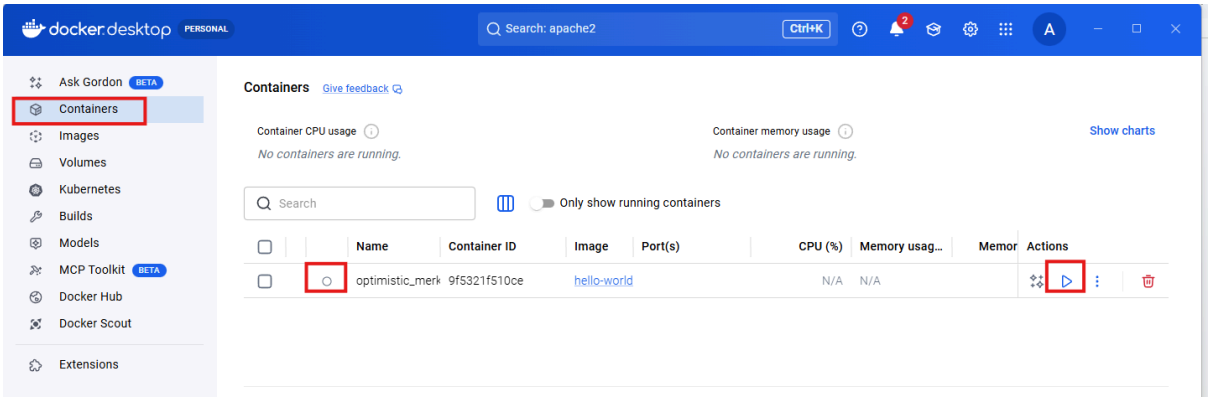
▪ Nous effectuons un docker run hello-world afin de chercher l'image hello-world en local :

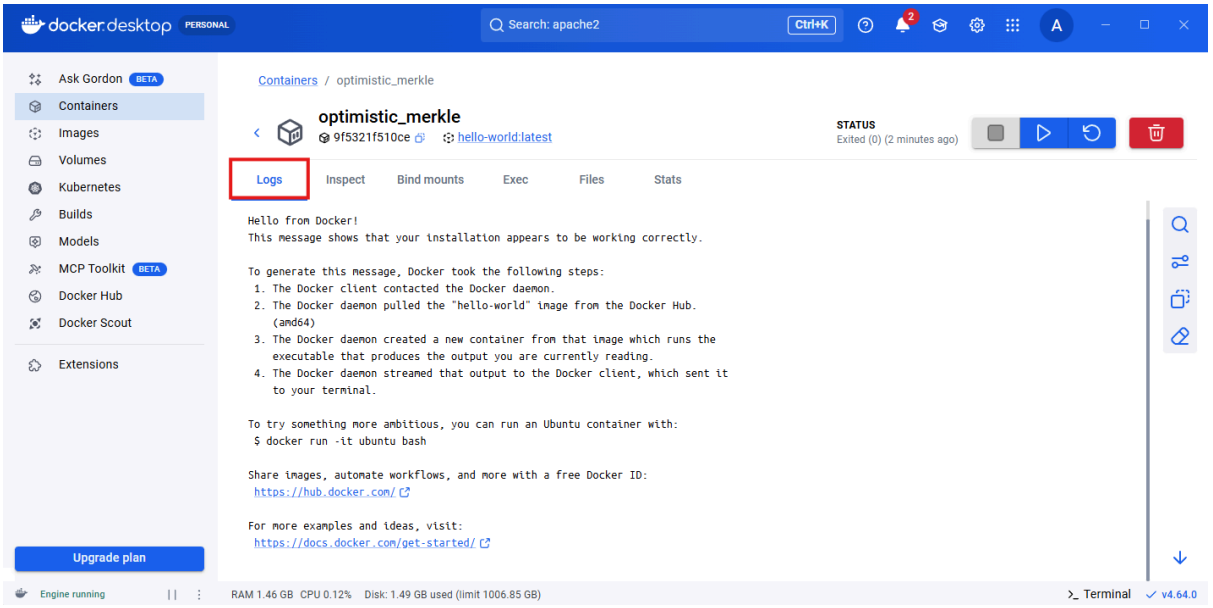


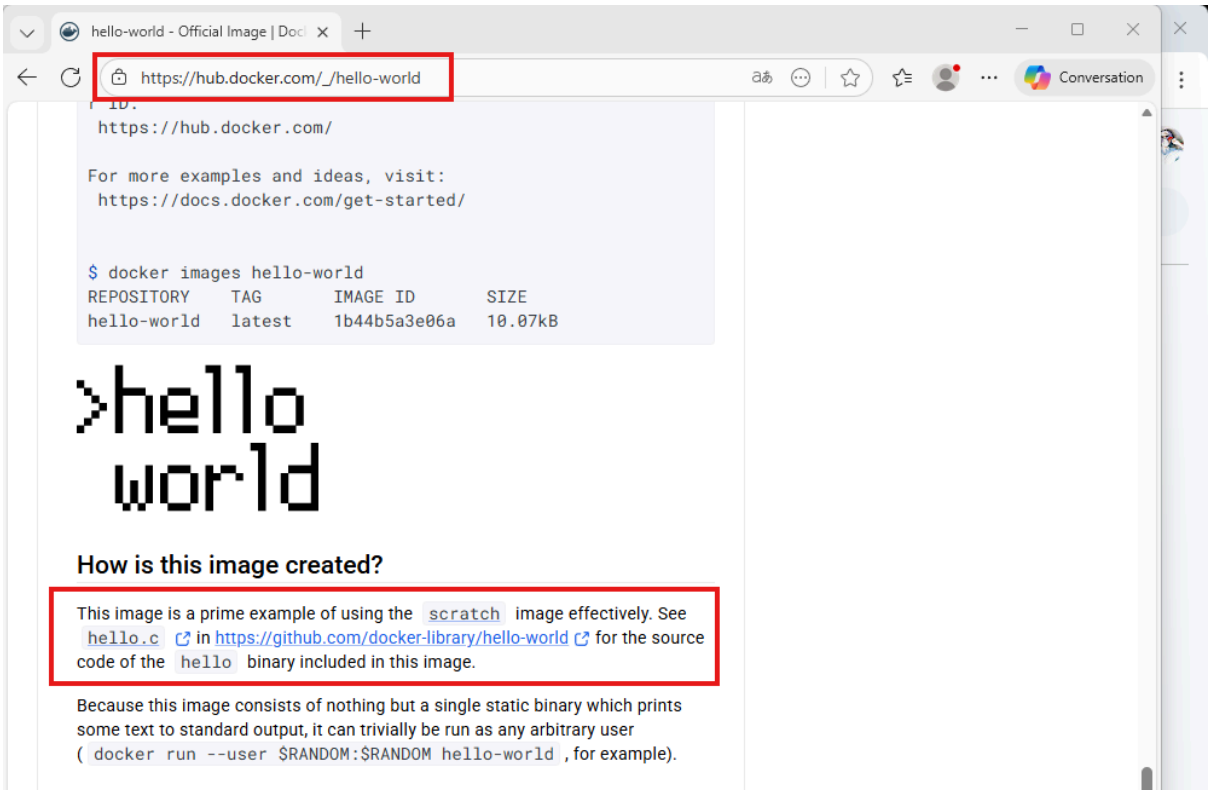
▪ Nous constatons la création de l'image



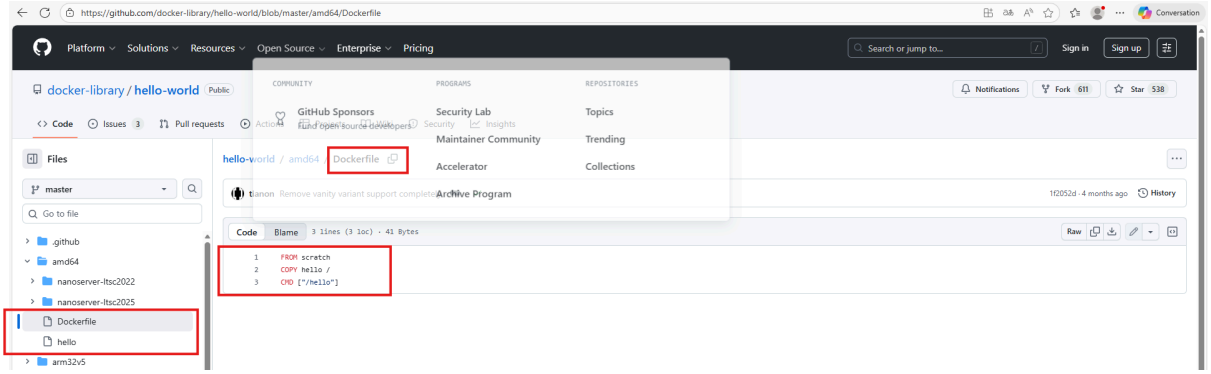
Le conteneur est arrêté après l'affichage du message :

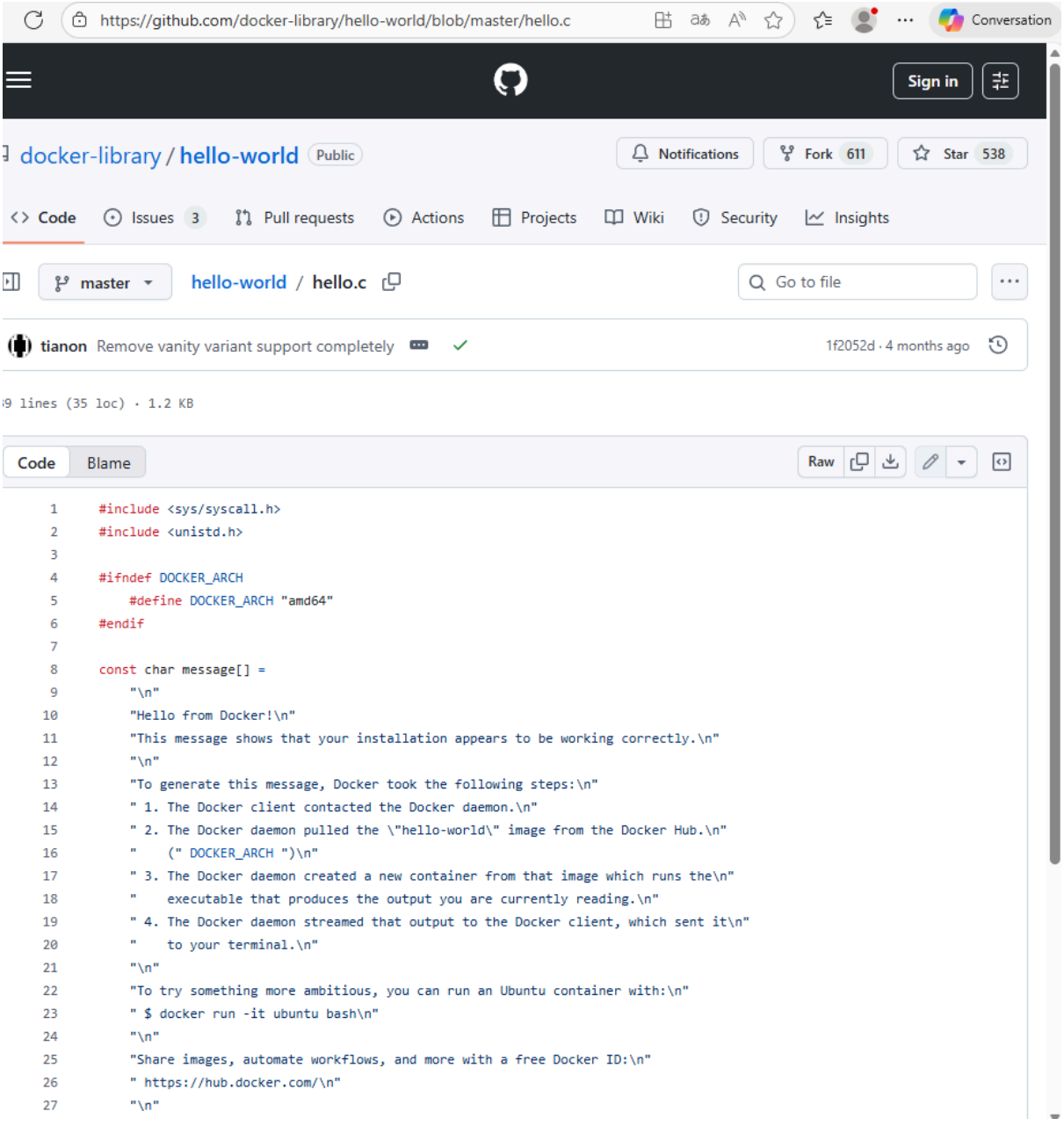



docker desktop
PERSONAL
Search: apache2
Ctrl+K
Ask Gordon
BETA
Containers
Images
Volumes
Kubernetes
Builds
Models
MCP Toolkit
BETA
Docker Hub
Docker Scout
Extensions
Upgrade plan
Containers / optimistic_merkle
optimistic_merkle
9f5321f510ce
hello-world:latest
STATUS
Exited (0) (2 minutes ago)
Logs
Inspect
Bind mounts
Exec
Files
Stats
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/
Engine running
RAM 1.46 GB CPU 0.12% Disk: 1.49 GB used (limit 1006.85 GB)
Terminal
v4.64.0


hello-world - Official Image | Doc
https://hub.docker.com/_/hello-world
Conversation
https://hub.docker.com/
For more examples and ideas, visit:
https://docs.docker.com/get-started/
$ docker images hello-world
REPOSITORY TAG IMAGE ID SIZE
hello-world latest 1b44b5a3e06a 10.07kB
>hello world
How is this image created?
This image is a prime example of using the scratch image effectively. See hello.c in https://github.com/docker-library/hello-world for the source code of the hello binary included in this image.
Because this image consists of nothing but a single static binary which prints some text to standard output, it can trivially be run as any arbitrary user (docker run --user $RANDOM:$RANDOM hello-world , for example).

Les fichiers qui servent à créer l'image hello-word :

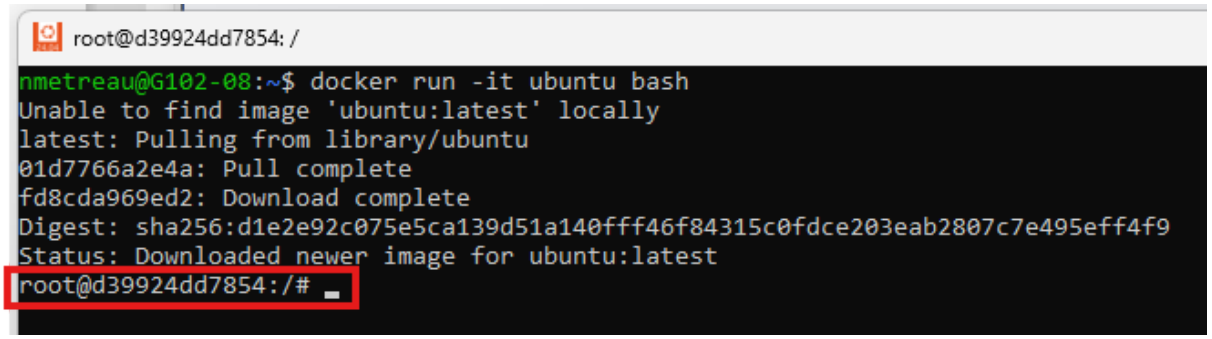


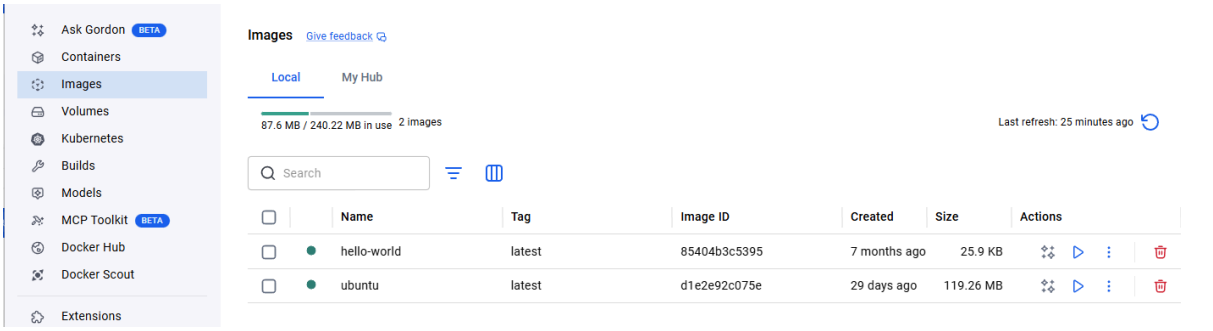


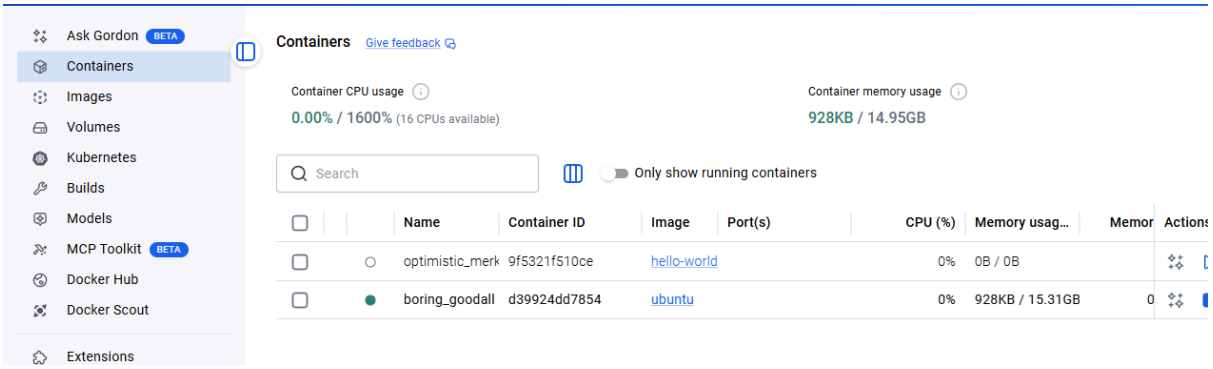
8. Cinquième test : mode interactif d'utilisation d'un conteneur.

▪ Commande docker images : liste toutes les images sur le cache local de la machine (identifiant = code hexadécimal)

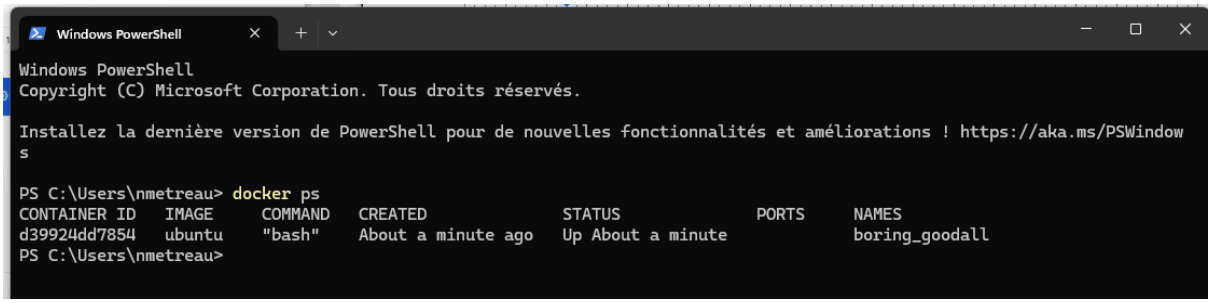
▪ Lancement d'un conteneur en mode interactif qui lancera un shell bash qui interprètera les commandes :



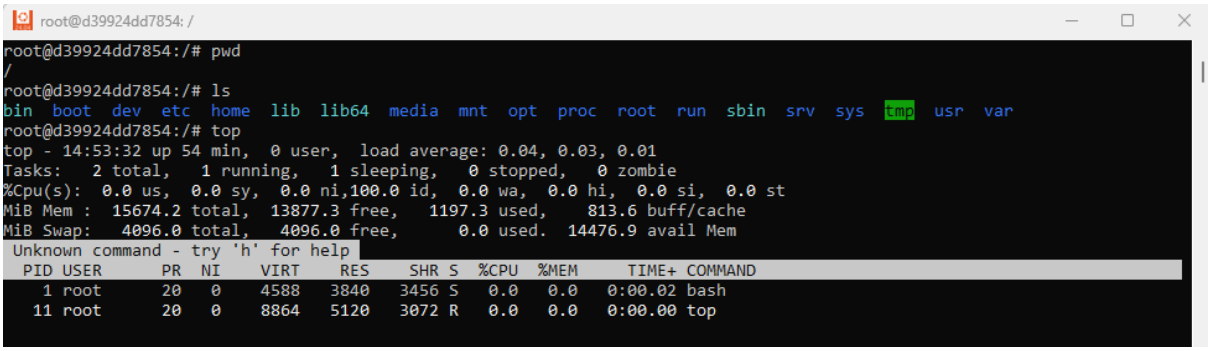


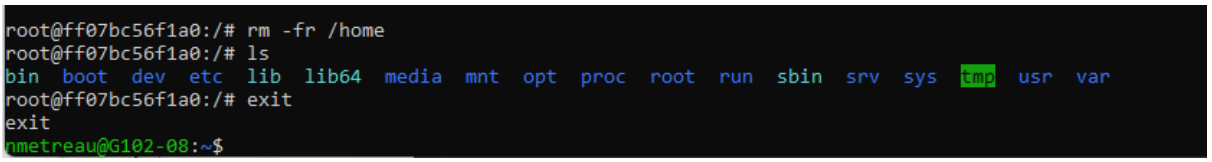


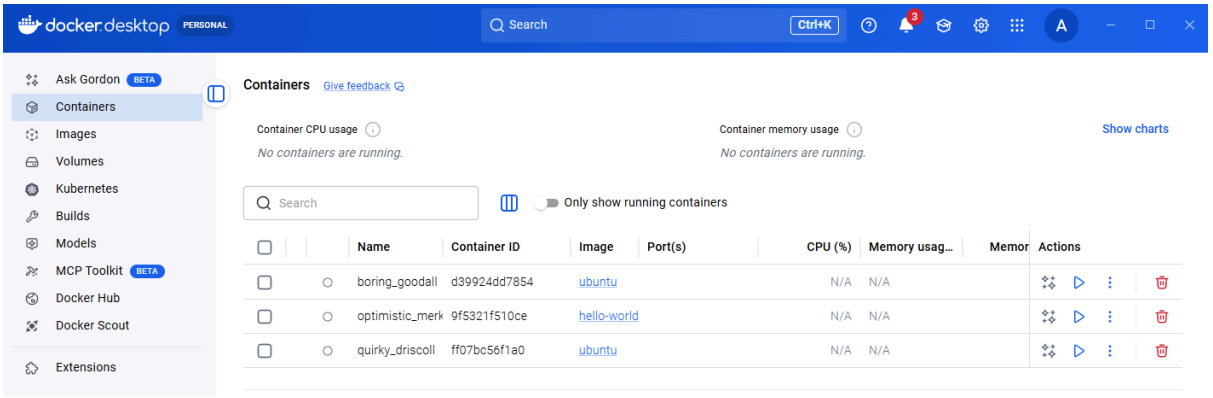
▪ La commande docker ps permet de lister les conteneurs en cours d'exécution :

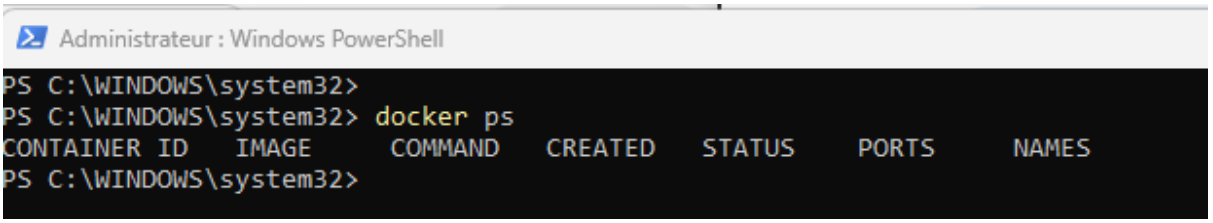


▪ Nous tapons les commandes pwd ainsi que ls et top









▪ Nous affichons les conteneurs

```
PS C:\WINDOWS\system32> docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED         STATUS                      PORTS     NAMES
ff07bc56f1a0   ubuntu        "bash"     6 minutes ago   Exited (0) 4 minutes ago              quirky_driscoll
d39924dd7854   ubuntu        "bash"     12 days ago     Exited (255) 7 minutes ago            boring_goodall
9f5321f510ce   hello-world   "/hello"   12 days ago     Exited (0) 12 days ago                optimistic_merkle
PS C:\WINDOWS\system32>
```

▪ Nous effectuons “start” du conteneur commençant par “ff07”

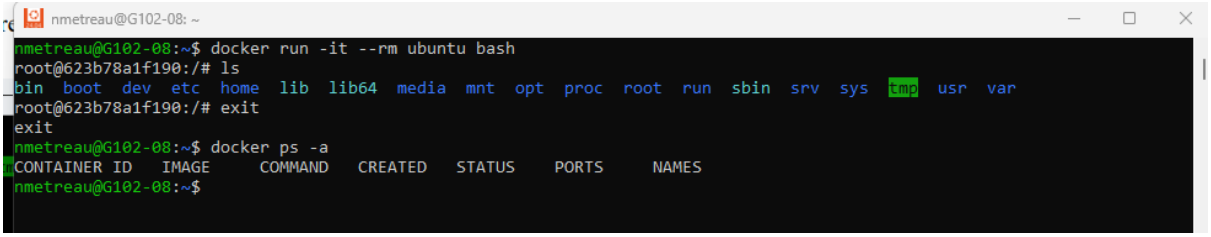
```
nmetreau@G102-08: ~
nmetreau@G102-08:~$ docker start ff07
ff07
nmetreau@G102-08:~$ docker stop ff07
ff07
nmetreau@G102-08:~$ dcoker start -ai ff07
dcoker: command not found
nmetreau@G102-08:~$ docker start -ai ff07
root@ff07bc56f1a0:/# ls
bin  boot  dev  etc  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@ff07bc56f1a0:/# exit
exit
nmetreau@G102-08:~$
```

```
nmetreau@G102-08: ~
nmetreau@G102-08:~$ docker start ff07
ff07
nmetreau@G102-08:~$ docker exec -it ff07 bash
root@ff07bc56f1a0:/# exit
exit
nmetreau@G102-08:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS         PORTS     NAMES
ff07bc56f1a0   ubuntu    "bash"    13 minutes ago   Up 2 minutes             quirky_driscoll
nmetreau@G102-08:~$ docker stop ff07
ff07
nmetreau@G102-08:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
nmetreau@G102-08:~$ docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                       PORTS     NAMES
ff07bc56f1a0   ubuntu        "bash"     14 minutes ago   Exited (137) 15 seconds ago           quirky_driscoll
d39924dd7854   ubuntu        "bash"     12 days ago      Exited (137) 5 minutes ago            boring_goodall
9f5321f510ce   hello-world   "/hello"   12 days ago      Exited (0) 6 minutes ago              optimistic_merkle
nmetreau@G102-08:~$
```

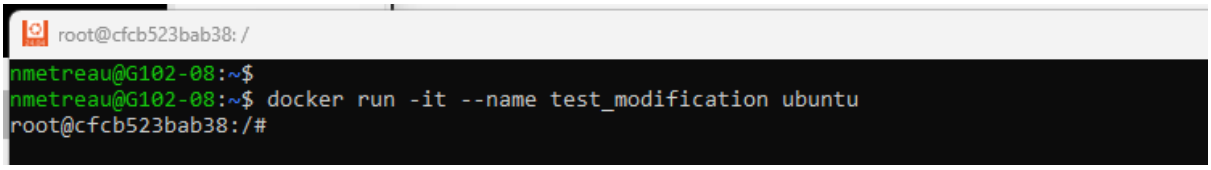
```
nmetreau@G102-08: ~
nmetreau@G102-08:~$ docker start ff07
ff07
nmetreau@G102-08:~$ docker exec -it ff07 bash
root@ff07bc56f1a0:/# exit
exit
nmetreau@G102-08:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS         PORTS     NAMES
ff07bc56f1a0   ubuntu    "bash"    13 minutes ago   Up 2 minutes             quirky_driscoll
nmetreau@G102-08:~$ docker stop ff07
ff07
nmetreau@G102-08:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
nmetreau@G102-08:~$ docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                       PORTS     NAMES
ff07bc56f1a0   ubuntu        "bash"     14 minutes ago   Exited (137) 15 seconds ago           quirky_driscoll
d39924dd7854   ubuntu        "bash"     12 days ago      Exited (137) 5 minutes ago            boring_goodall
9f5321f510ce   hello-world   "/hello"   12 days ago      Exited (0) 6 minutes ago              optimistic_merkle
nmetreau@G102-08:~$ docker rm boring_goodall
boring_goodall
nmetreau@G102-08:~$ docker rm quirky_driscoll
quirky_driscoll
nmetreau@G102-08:~$ docker rm quirky_driscoll rm 9f
9f
Error response from daemon: No such container: quirky_driscoll
Error response from daemon: No such container: rm
nmetreau@G102-08:~$ docker rm 9f
Error response from daemon: No such container: 9f
nmetreau@G102-08:~$
nmetreau@G102-08:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
nmetreau@G102-08:~$
```

9. Création d'une image à la main avec docker commit.

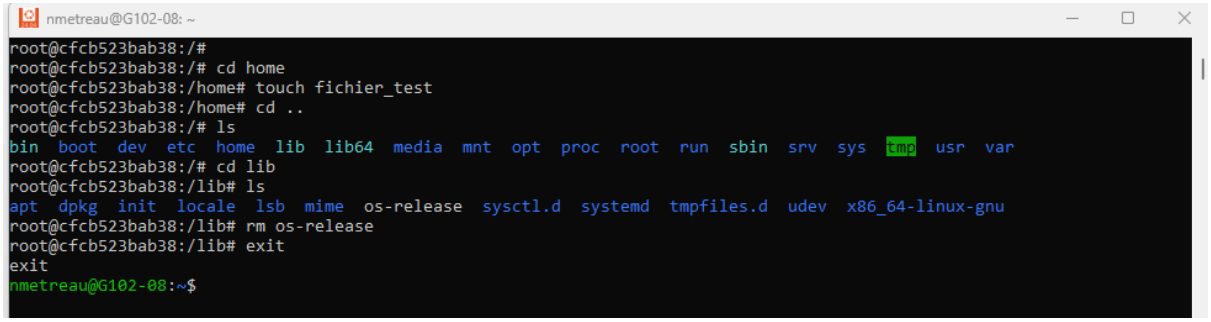
▪ Nous lançons un conteneur ayant pour image ubuntu qui sera détruit quand nous le quittons



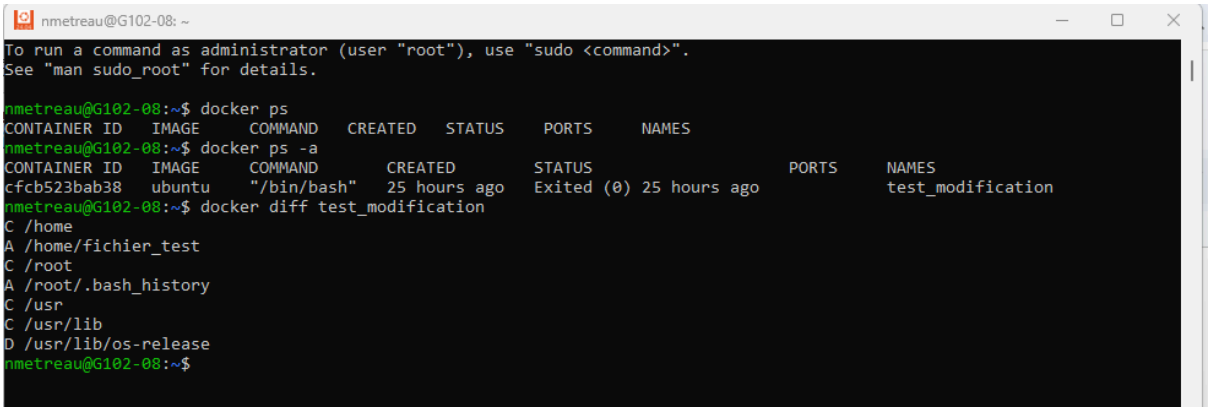
▪ Création d'un nouveau conteneur ayant pour nom "test_modification".



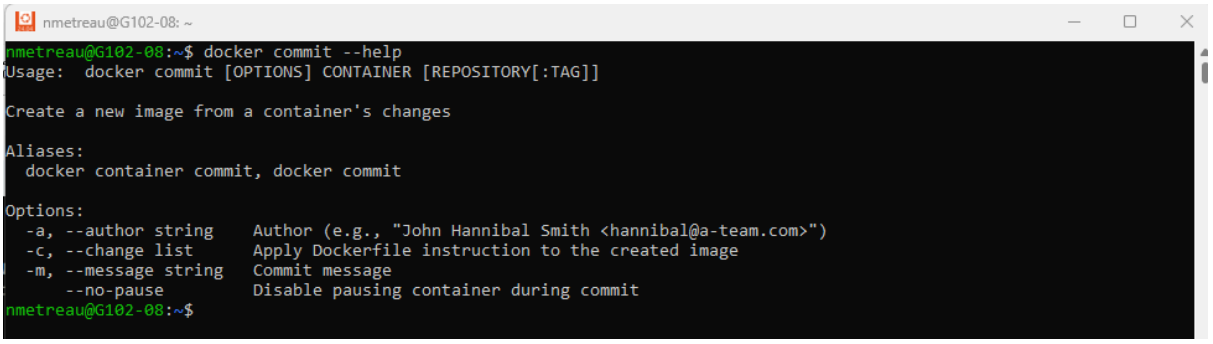
▪ Nous créons un fichier vide de test puis nous supprimons le fichier os et quittons ce conteneur.

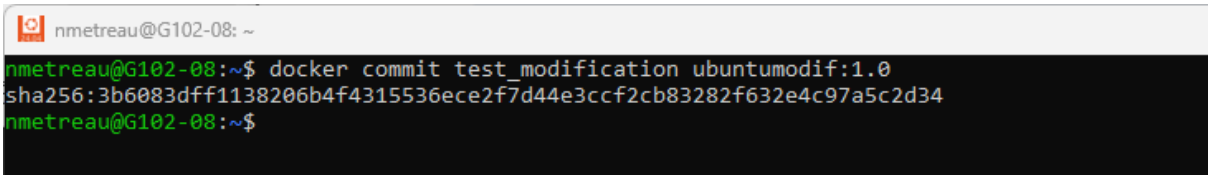


▪ Nous effectuons docker diff test_modification afin de lister tous les changements effectués dans le système de fichiers du conteneur par rapport à son image d'origine .

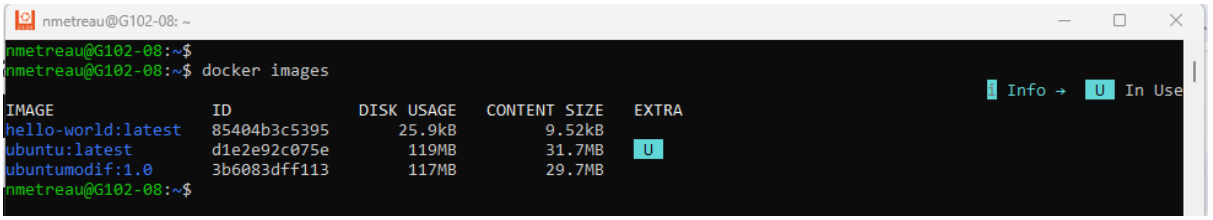


▪ Nous effectuons la commande docker commit afin de sauvegarder les modifications effectuées dans un conteneur

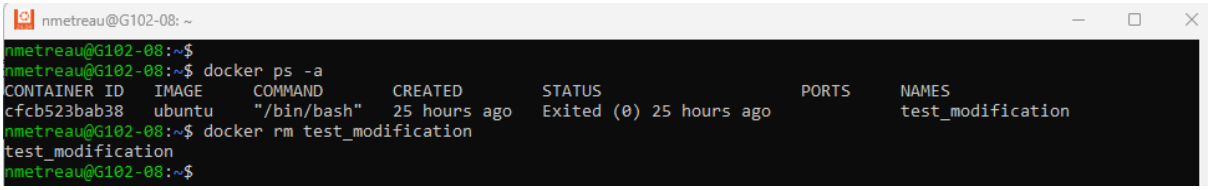




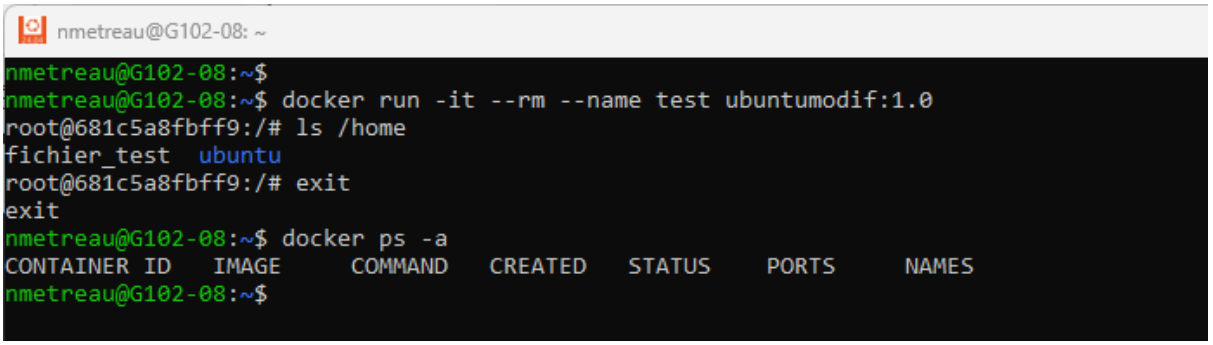
▪ Nous regardons que l'image ce soit bien créée



▪ Nous listons les conteneurs

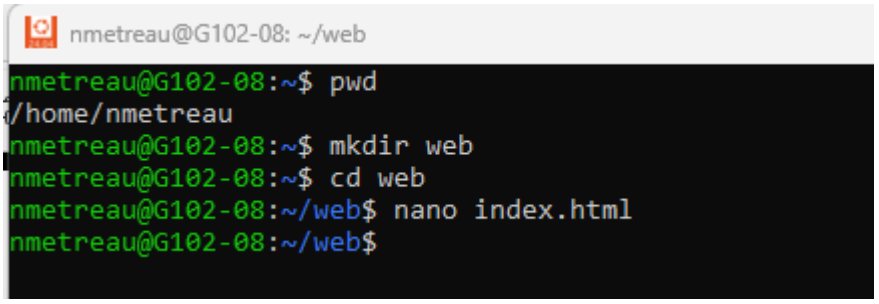


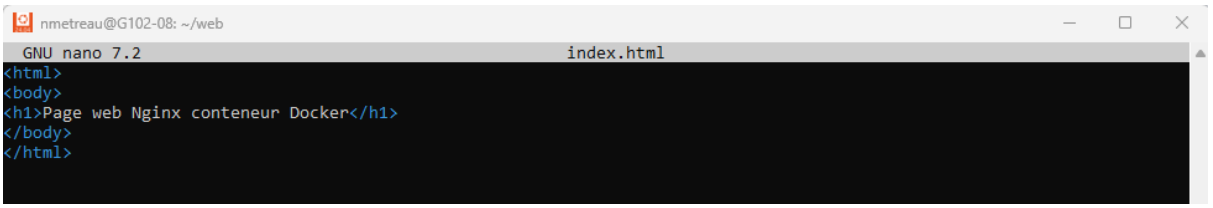
▪ Nous lançons un nouveau conteneur et nous vérifions s'il bien du fichier test



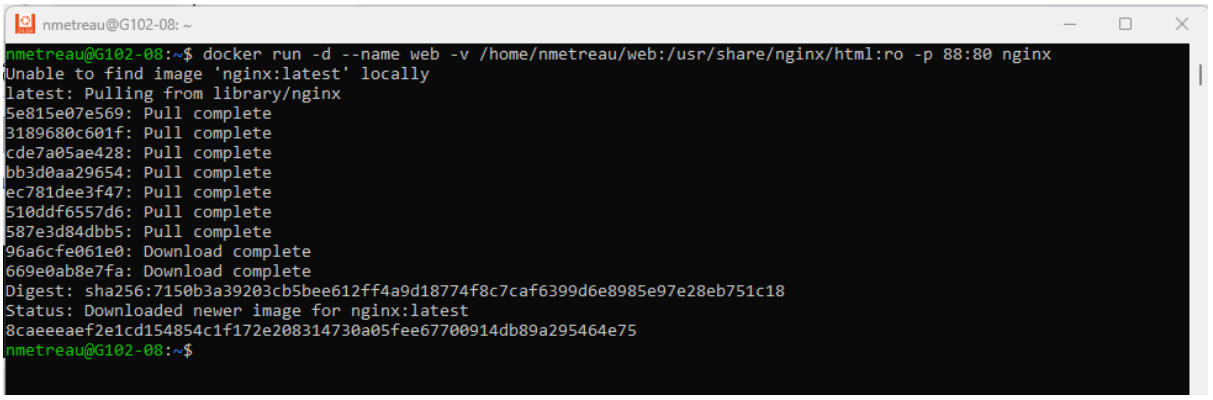
10. Première approche des volumes.

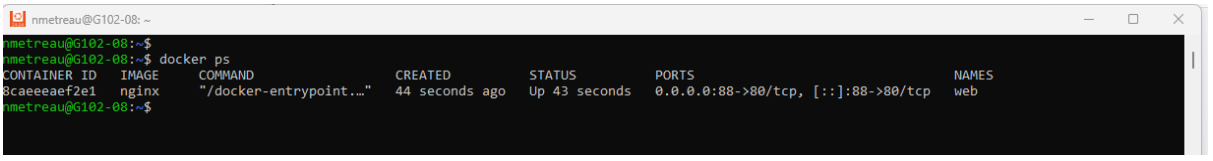
▪ Nous créons un répertoire web dans /home/”nmetreau” de la machine ubuntu ainsi qu’une page html :



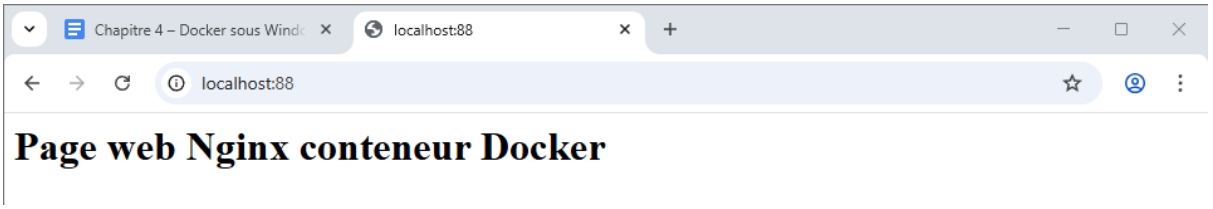


▪ Nous créons un conteneur qui va pointer sur le répertoire web du FS local accessible en lecture :

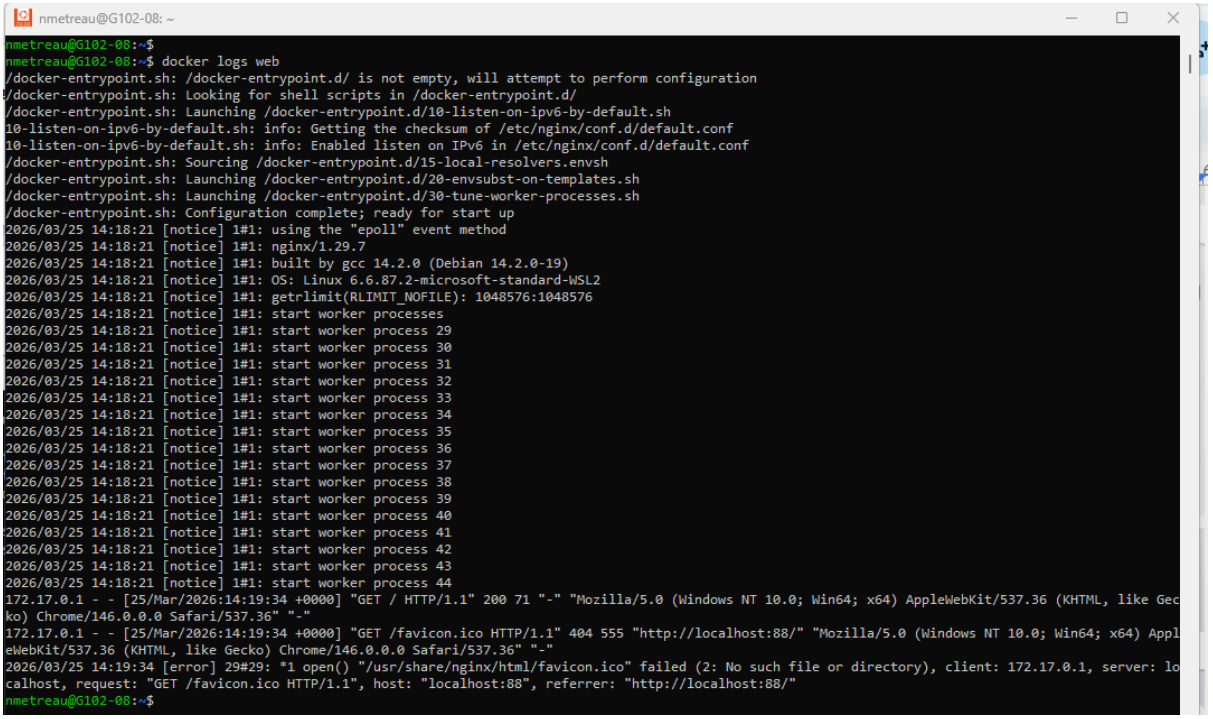




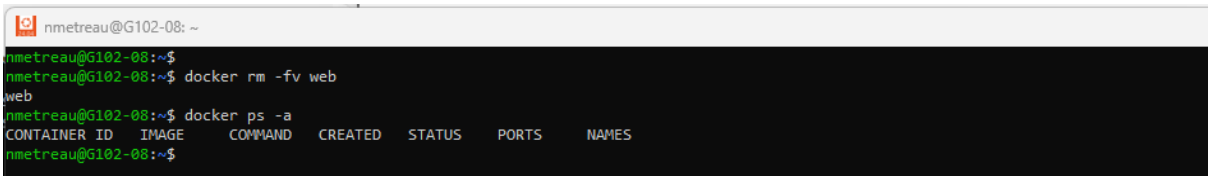
▪ Nous testons l’accès aux données :



▪ Affichage des logs du conteneur web :



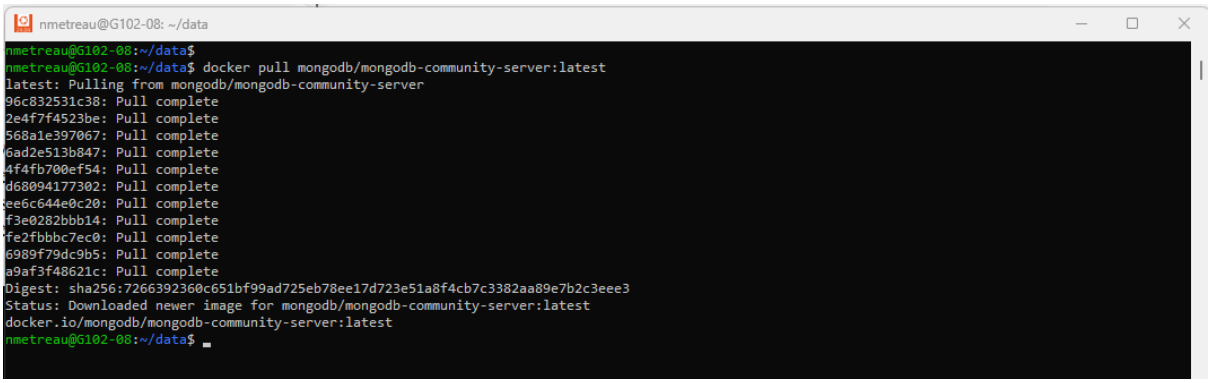
▪ Nous arrêtons et supprimons le conteneur web ainsi que le mappage de volume :



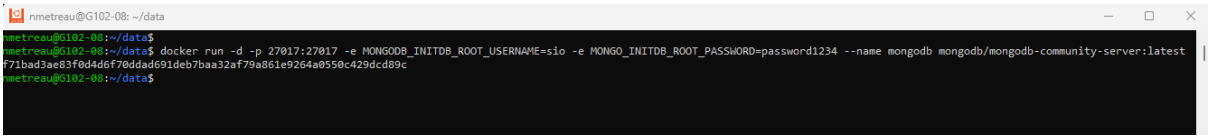
11. Gestion des volumes en écriture.

11.1. Premier type de volume : volume nommé

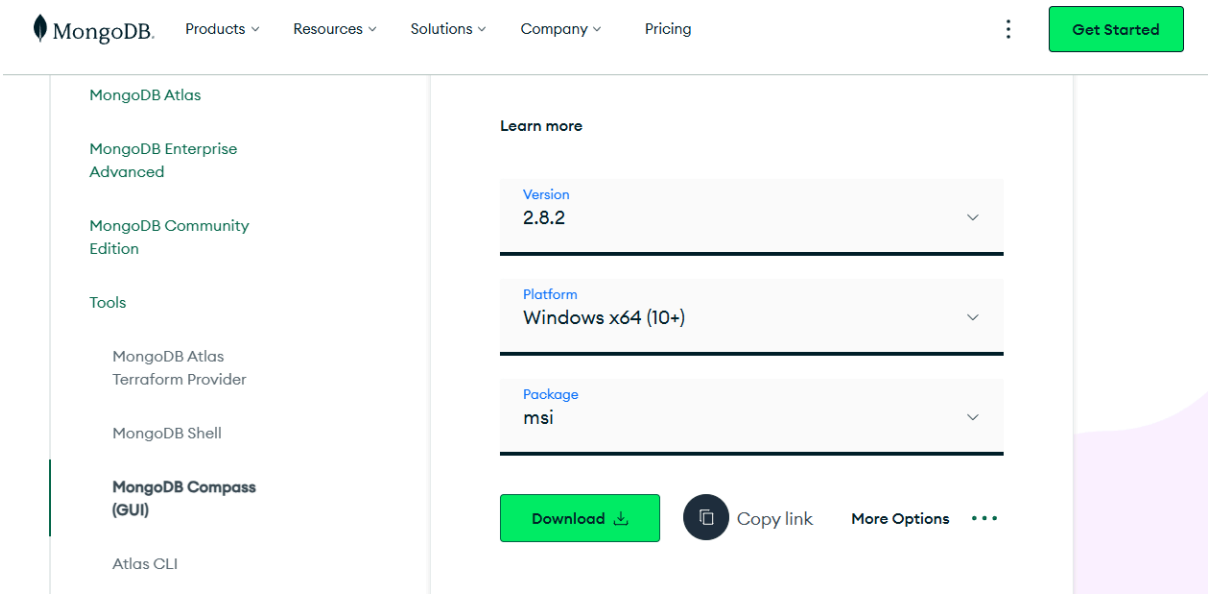
▪ Nous récupérons l'image mongodb-community-server avec la commande docker pull :



▪ Nous lançons un conteneur en mode serveur à partir de l'image mongodb-community-server :

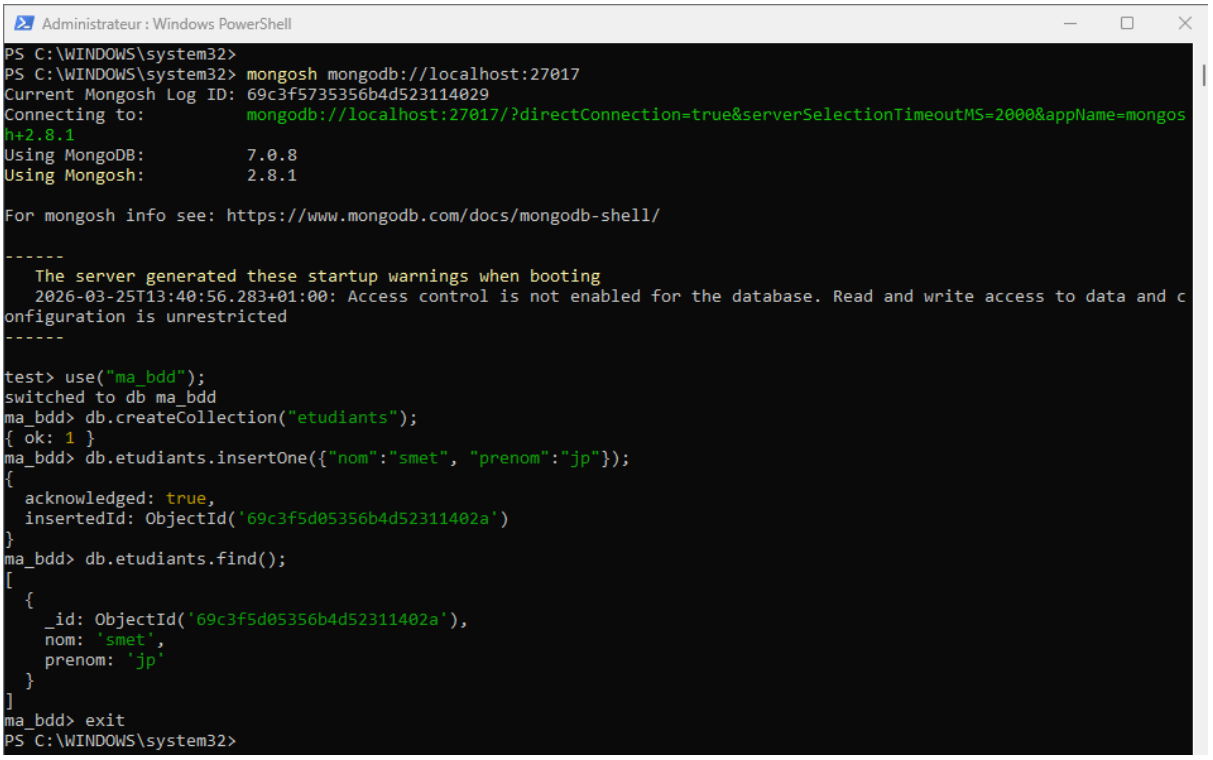


▪ Nous installons MongoDB Shell sur la machine physique pour pouvoir vous connecter au serveur MongoDB :

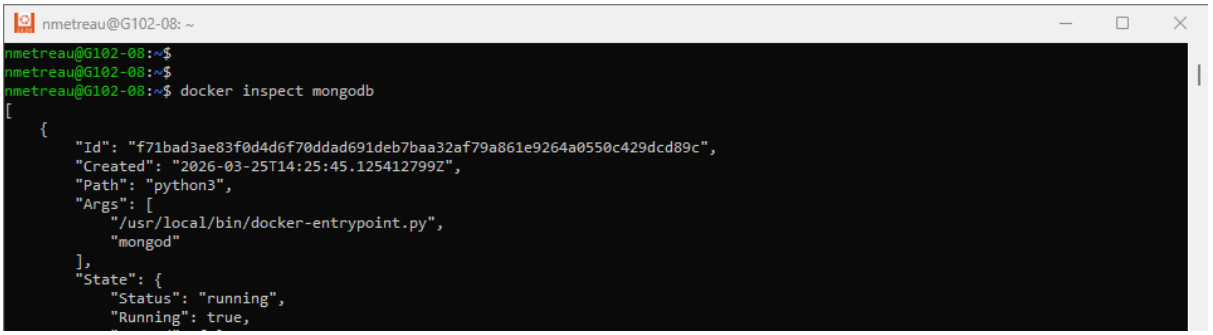


▪ Connexion au serveur MongoDB :

- Nous créons la base de données ma_bdd.
- Suite à la création de la bdd, nous créons une collection “etudiants” en ajoutant comme document “smet” pour le nom et “jp” pour son prénom.



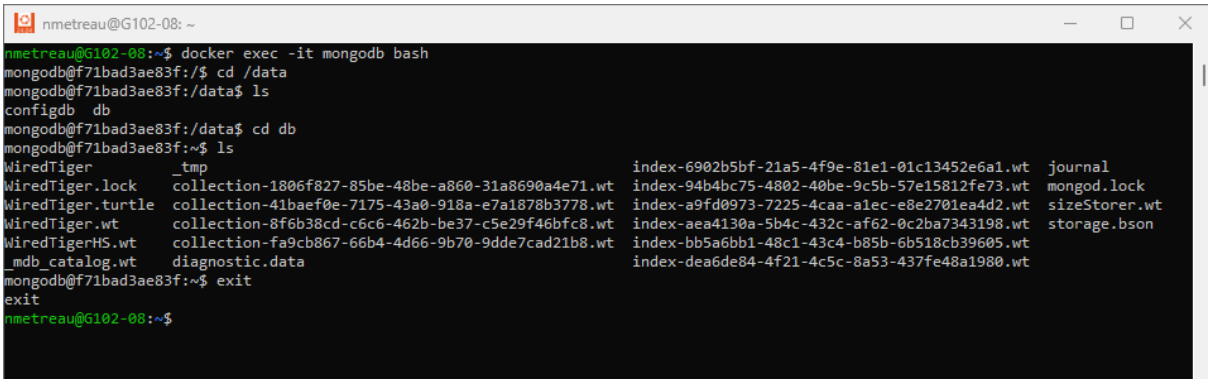
▪ Nous renvoyons toutes les informations techniques d'un conteneur (ou d'une image) au format JSON à l’aide de la commande docker inspect mongodb.



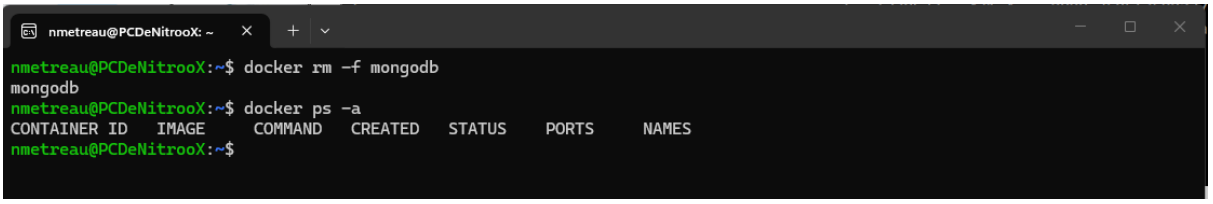
nmetreau@G102-08: ~


```
        },
        "Image": "sha256:7266392360c651bf99ad725eb78ee17d723e51a8f4cb7c3382aa89e7b2c3eee3",
        "ResolvConfPath": "/var/lib/docker/containers/f71bad3ae83f0d4d6f70ddad691deb7baa32af79a861e9264a0550c429dcd89c/res
        "HostnamePath": "/var/lib/docker/containers/f71bad3ae83f0d4d6f70ddad691deb7baa32af79a861e9264a0550c429dcd89c/hostn
        "HostsPath": "/var/lib/docker/containers/f71bad3ae83f0d4d6f70ddad691deb7baa32af79a861e9264a0550c429dcd89c/hosts",
        "LogPath": "/var/lib/docker/containers/f71bad3ae83f0d4d6f70ddad691deb7baa32af79a861e9264a0550c429dcd89c/f71bad3ae8
ad691deb7baa32af79a861e9264a0550c429dcd89c-json.log",
        "Name": "/mongodb",
        "RestartCount": 0,
        "Driver": "overlayfs",
        "Platform": "linux",
        "MountLabel": "",
        "ProcessLabel": "",
        "AppArmorProfile": "",
        "ExecIDs": null,
        "HostConfig": {
            "Binds": null,
            "ContainerIDFile": "",
            "LogConfig": {
                "Type": "json-file",
                "Config": {}
            },
            "NetworkMode": "bridge",
            "PortBindings": {
                "27017/tcp": [
                    {
                        "HostIp": "",
                        "HostPort": "27017"
                    }
                ]
            },
            "RestartPolicy": {
                "Name": "no",
                "MaximumRetryCount": 0
            },
            "AutoRemove": false,
            "VolumeDriver": "",
            "VolumesFrom": null,
            "ConsoleSize": [
                47,
                150
            ],
            "CapAdd": null,
            "CapDrop": null,
            "CgroupnsMode": "private",
            "Dns": null,
            "DnsOptions": [],
            "DnsSearch": [],
            "ExtraHosts": null,
            "GroupAdd": null,
            "IpcMode": "private",
            "Cgroup": "",
            "Links": null,
            "OomScoreAdj": 0,
            "PidMode": "",
            "Privileged": false,
            "PublishAllPorts": false,
            "ReadonlyRootfs": false,
            "SecurityOpt": null,
            "UTSMode": "",
            "UsernsMode": "",
            "ShmSize": 67108864,
            "Runtime": "runc",
            "Isolation": "",
            "CpuShares": 0,
            "Memory": 0,
```

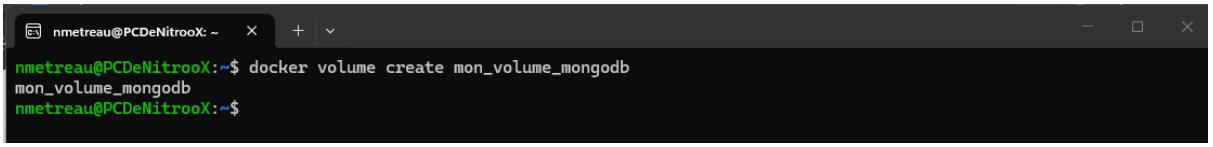
▪ Nous accédons au conteneur puis aux données et nous arrêtons le conteneur :

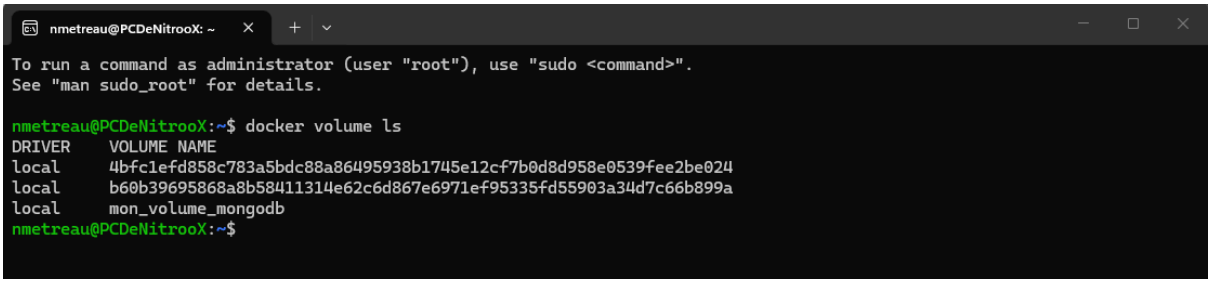


▪ Suppression du conteneur :

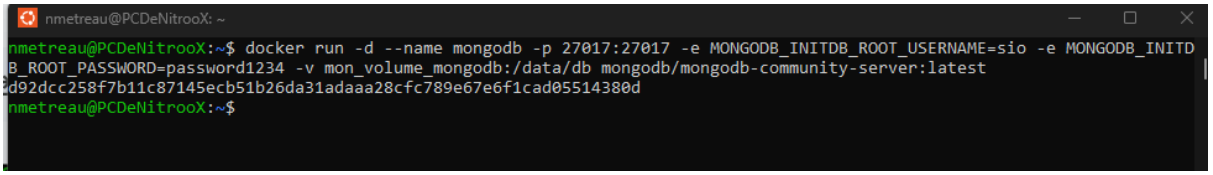


▪ Création d'un volume nommé à l'aide de la commande docker volume create :

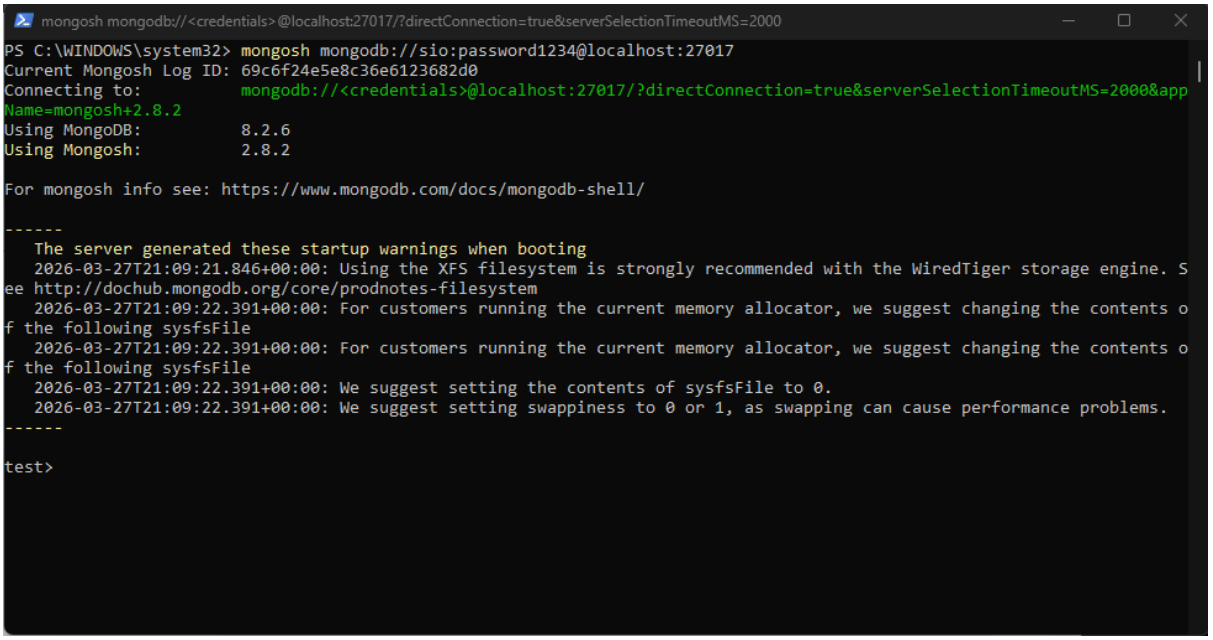


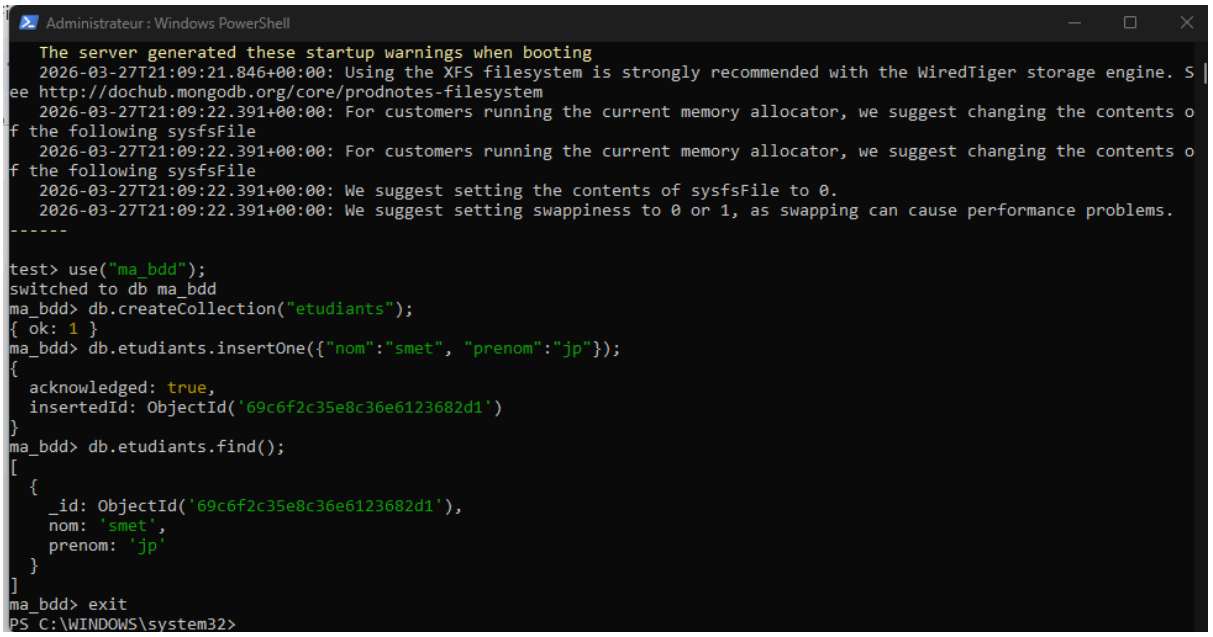


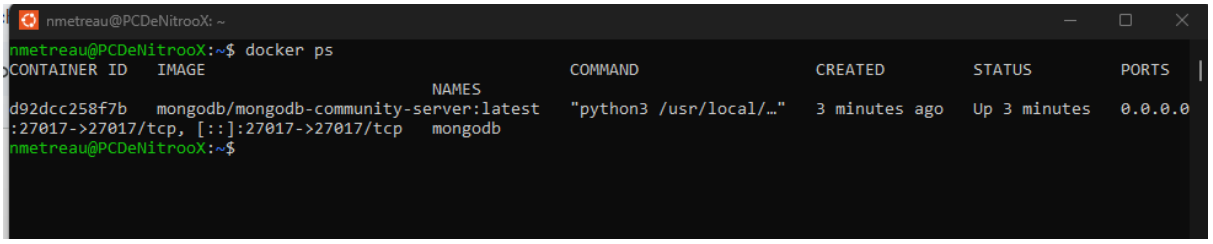
▪ Nous lançons un conteneur avec un mappage de volume (volume nommé:/volume du conteneur) :



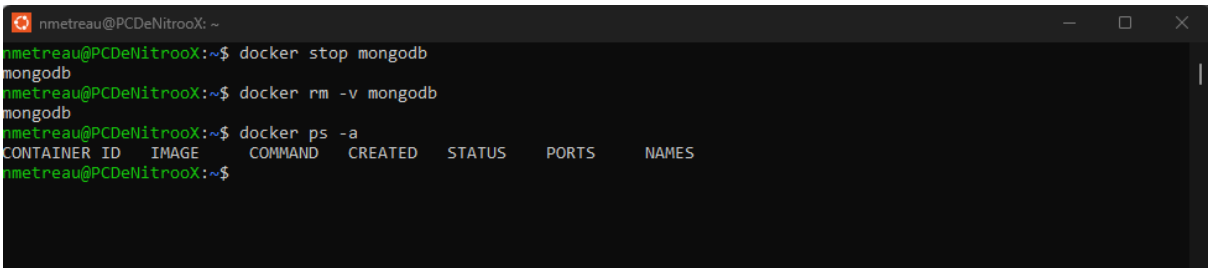
▪ Nous nous connectons au serveur MongoDB :



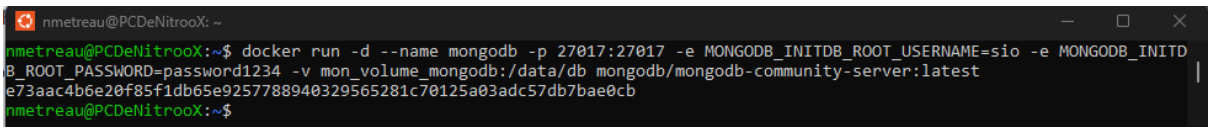




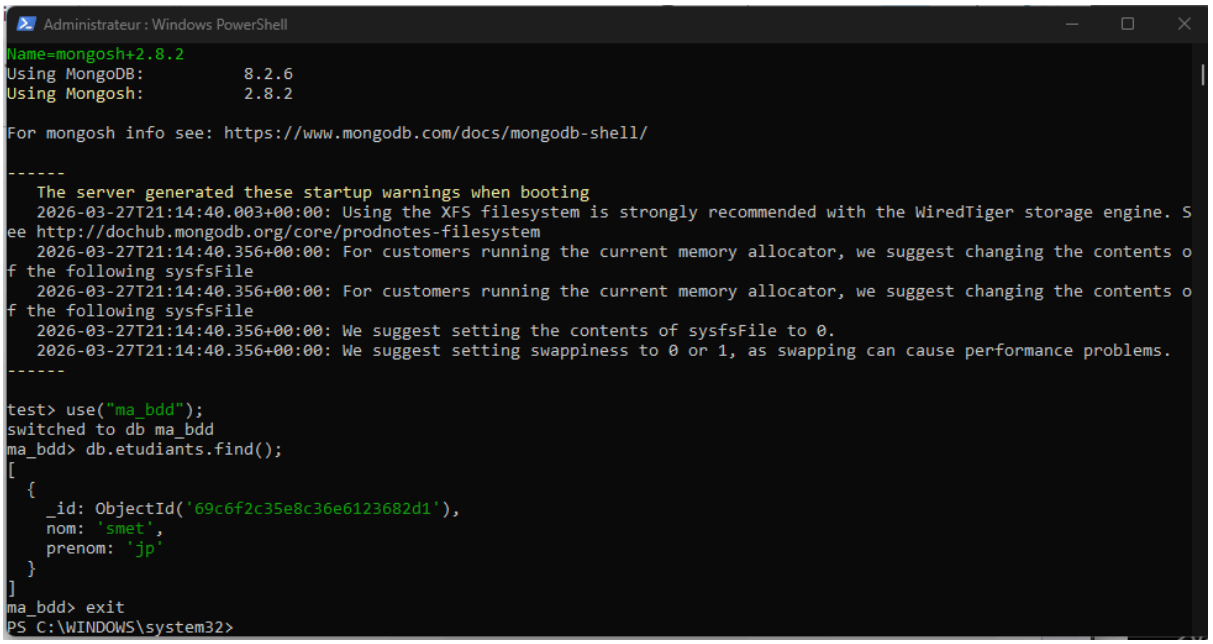
▪ Nous arrêtons le conteneur et nous le supprimons :



▪ Nous démarrons un conteneur mongo :

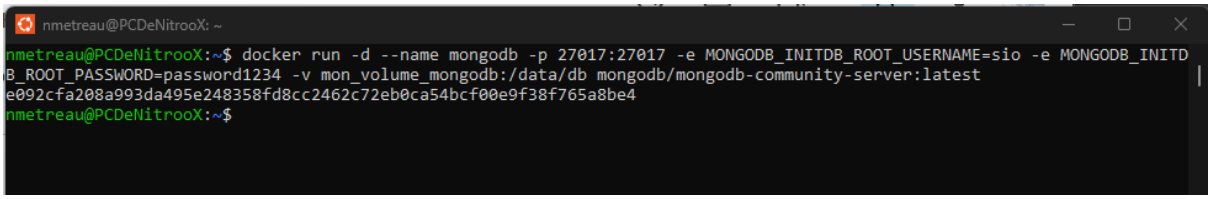


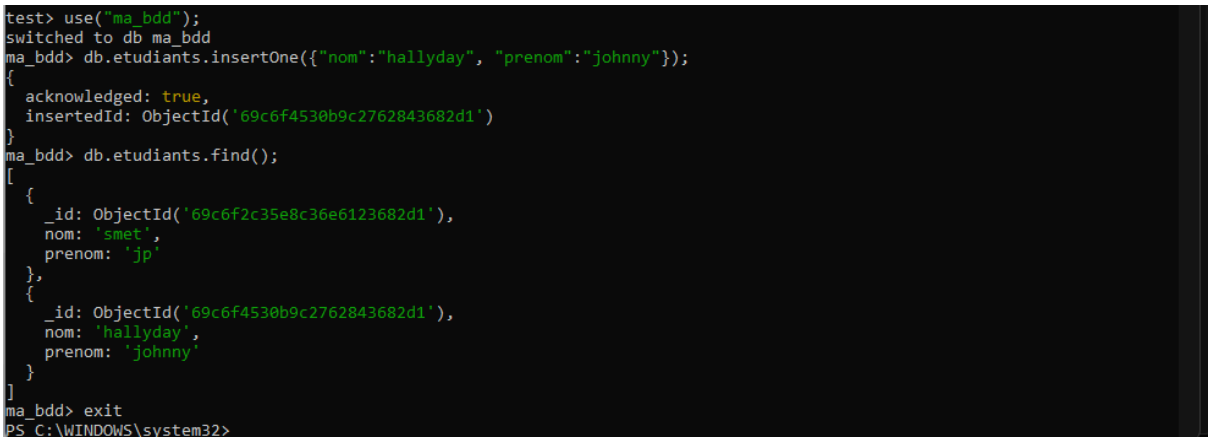
▪ Nous regardons si la collection “etudiants” est présente :



▪ Suppression du conteneur :

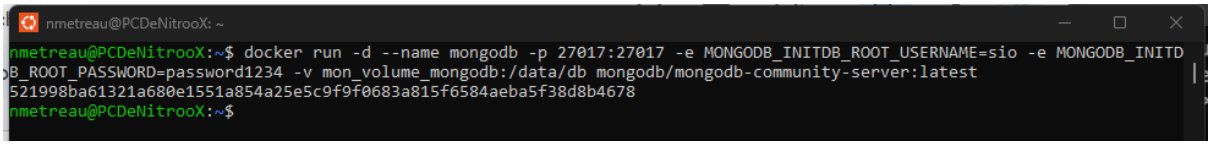
▪ Nous lançons un conteneur et nous ajoutons une deuxième ligne pour vérifier la persistance :





▪ Nous supprimons le conteneur :

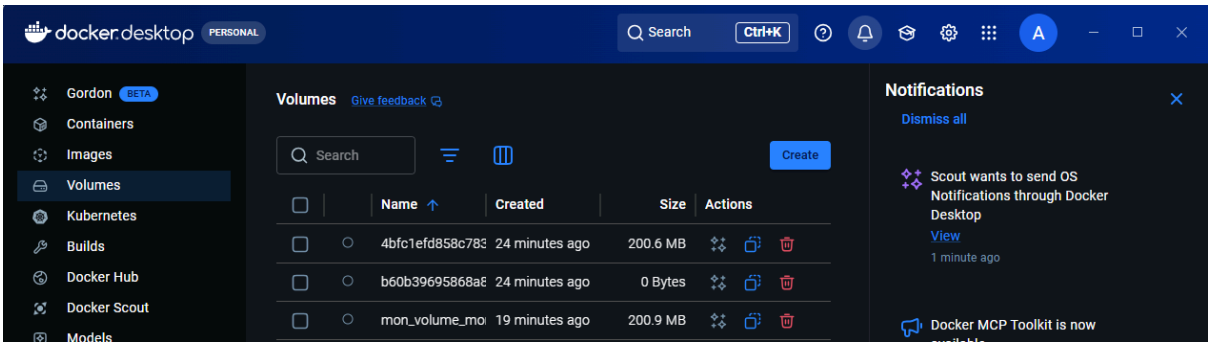
▪ Nous relançons un conteneur et nous ajoutons une deuxième ligne pour vérifier la persistance des données :

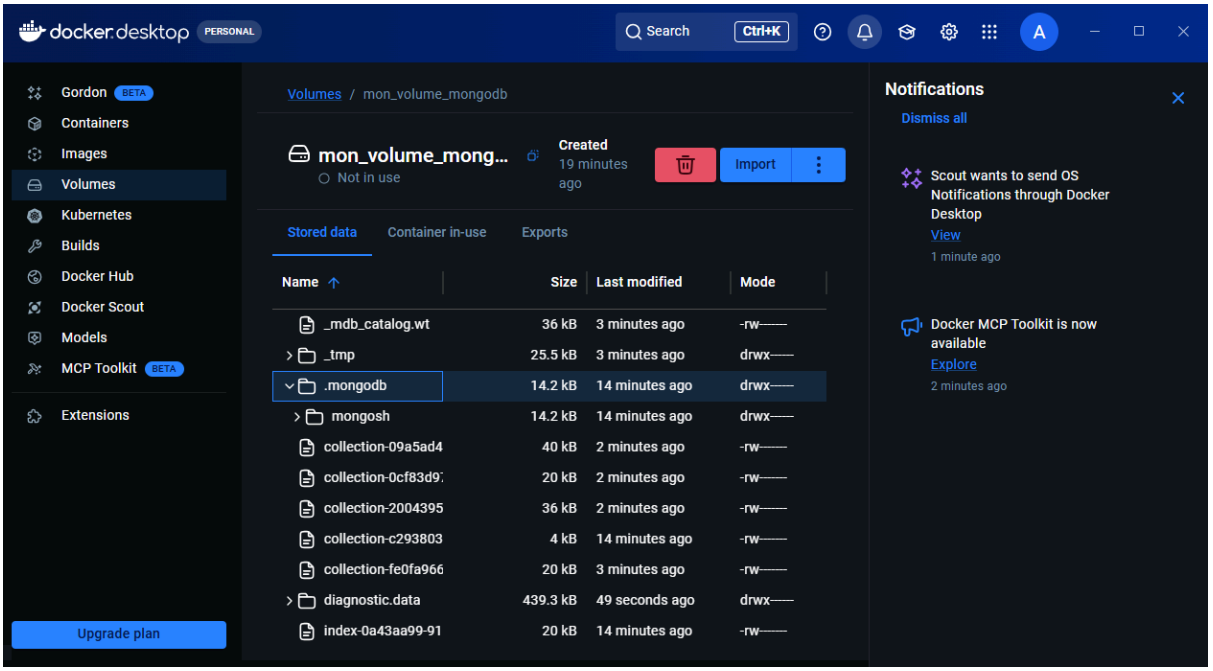


```
Administrateur : Windows PowerShell
PS C:\WINDOWS\system32> mongosh mongodb://sio:password1234@localhost:27017
Current Mongosh Log ID: 69c6f4f7c3f3461d373682d0
Connecting to:          mongodb://<credentials>@localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000&app
Name=mongosh+2.8.2
Using MongoDB:          8.2.6
Using Mongosh:          2.8.2

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2026-03-27T21:20:45.673+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. S
ee http://dochub.mongodb.org/core/prodnotes-filesystem
   2026-03-27T21:20:45.976+00:00: For customers running the current memory allocator, we suggest changing the contents o
f the following sysfsFile
   2026-03-27T21:20:45.976+00:00: For customers running the current memory allocator, we suggest changing the contents o
f the following sysfsFile
   2026-03-27T21:20:45.976+00:00: We suggest setting the contents of sysfsFile to 0.
   2026-03-27T21:20:45.976+00:00: We suggest setting swappiness to 0 or 1, as swapping can cause performance problems.
------

test> use("ma_bdd");
switched to db ma_bdd
ma_bdd> db.etudiants.find();
[
  {
    _id: ObjectId('69c6f2c35e8c36e6123682d1'),
    nom: 'smet',
    prenom: 'jp'
  },
  {
    _id: ObjectId('69c6f4530b9c2762843682d1'),
    nom: 'hallyday',
    prenom: 'johnny'
  }
]
ma_bdd> exit
PS C:\WINDOWS\system32>
```

- Nous supprimons le conteneur :

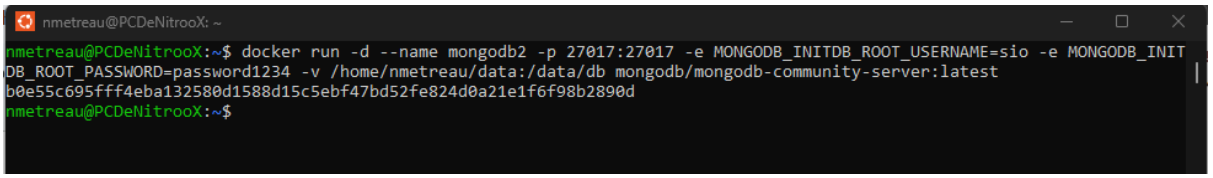
```
nmetreau@PCDeNitrooX: ~
nmetreau@PCDeNitrooX:~$ docker rm -fv mongodb
mongodb
nmetreau@PCDeNitrooX:~$
```



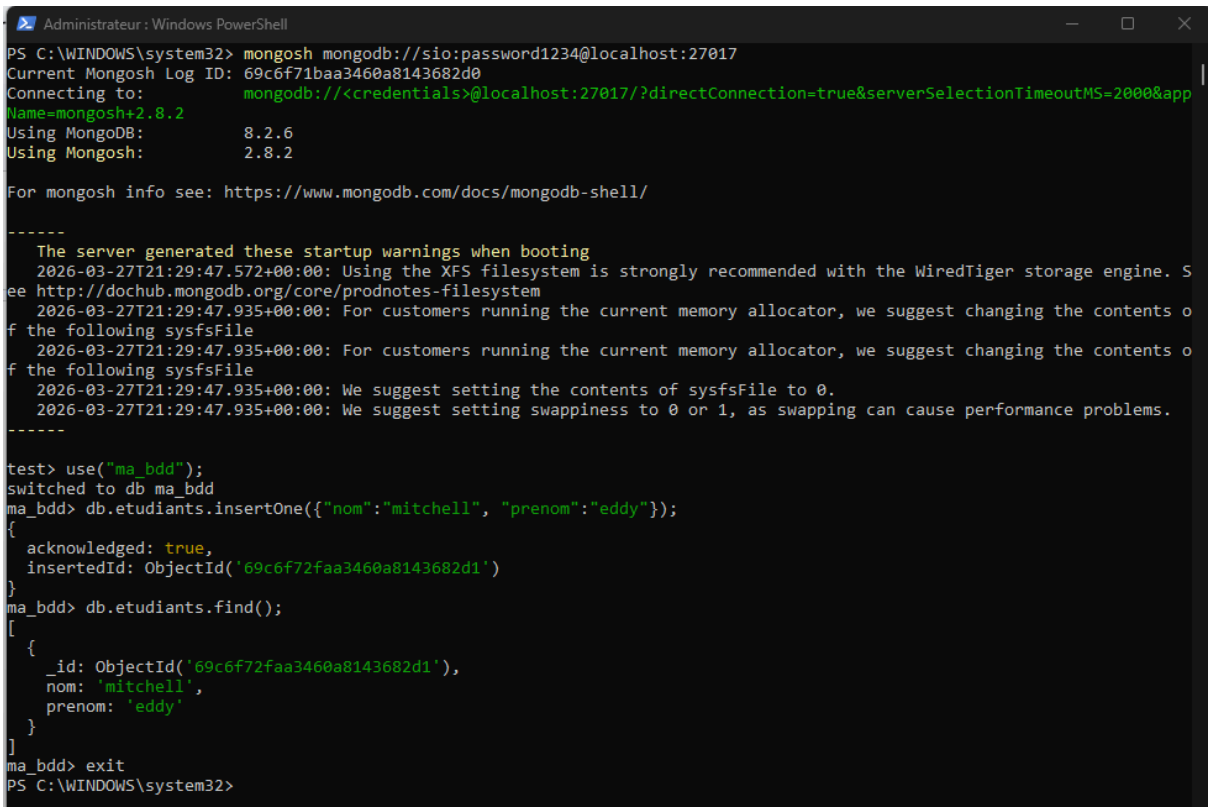

docker desktop
PERSONAL
Search
Ctrl+K
A
Gordon
BETA
Containers
Images
Volumes
Kubernetes
Builds
Docker Hub
Docker Scout
Models
MCP Toolkit
BETA
Extensions
Upgrade plan
Volumes / mon_volume_mongodb
mon_volume_mong...
Not in use
Created
19 minutes ago
Import
Stored data
Container in-use
Exports
Name
Size
Last modified
Mode
_mdb_catalog.wt 36 kB 3 minutes ago -rw-----
_tmp 25.5 kB 3 minutes ago drwx-----
.mongodb 14.2 kB 14 minutes ago drwx-----
mongosh 14.2 kB 14 minutes ago drwx-----
collection-09a5ad4 40 kB 2 minutes ago -rw-----
collection-0cf83d9 20 kB 2 minutes ago -rw-----
collection-2004395 36 kB 2 minutes ago -rw-----
collection-c293803 4 kB 14 minutes ago -rw-----
collection-fe0fa966 20 kB 3 minutes ago -rw-----
diagnostic.data 439.3 kB 49 seconds ago drwx-----
index-0a43aa99-91 20 kB 14 minutes ago -rw-----
Notifications
Dismiss all
Scout wants to send OS Notifications through Docker Desktop
View
1 minute ago
Docker MCP Toolkit is now available
Explore
2 minutes ago

11.2. Deuxième type de volume : montage lié (bind mounting)

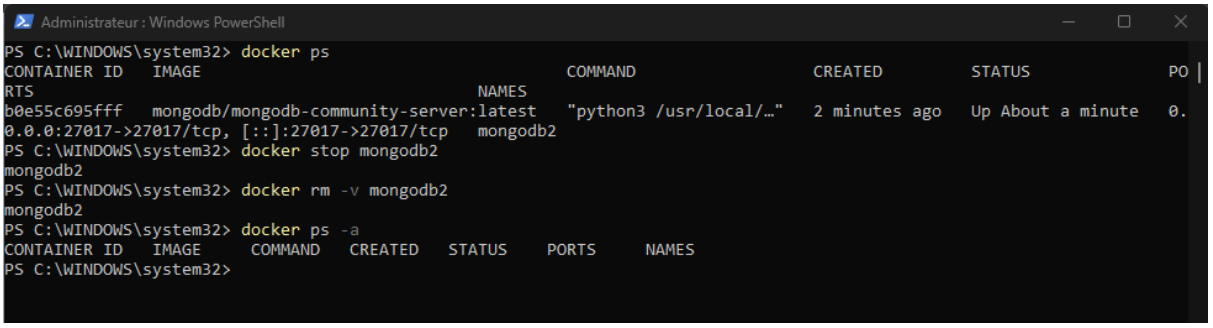
▪ Nous créons le répertoire data :

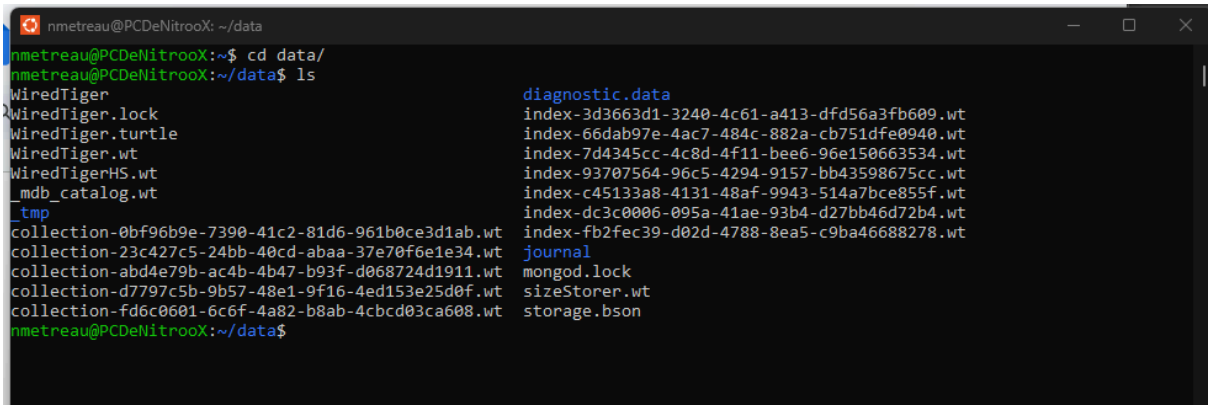


▪ Dans cette collection, nous ajoutons “eddy mitchemm”

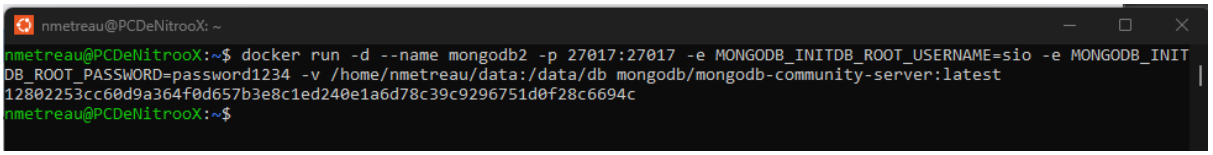


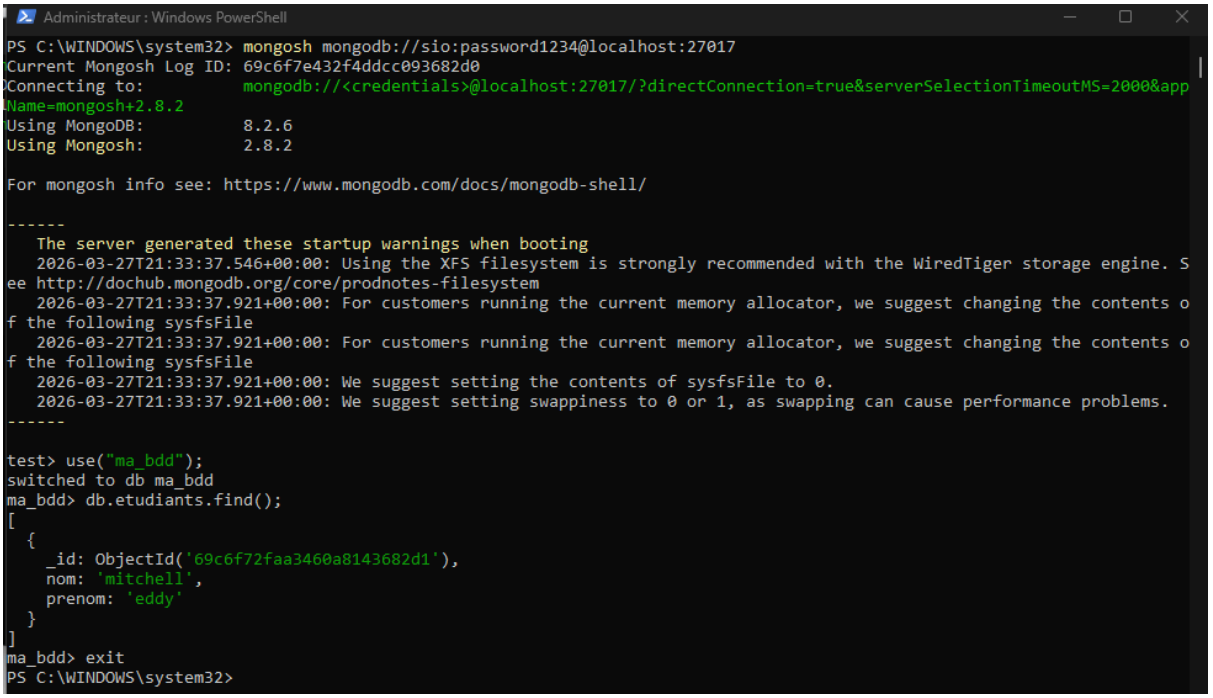
▪ Nous supprimons le conteneur :



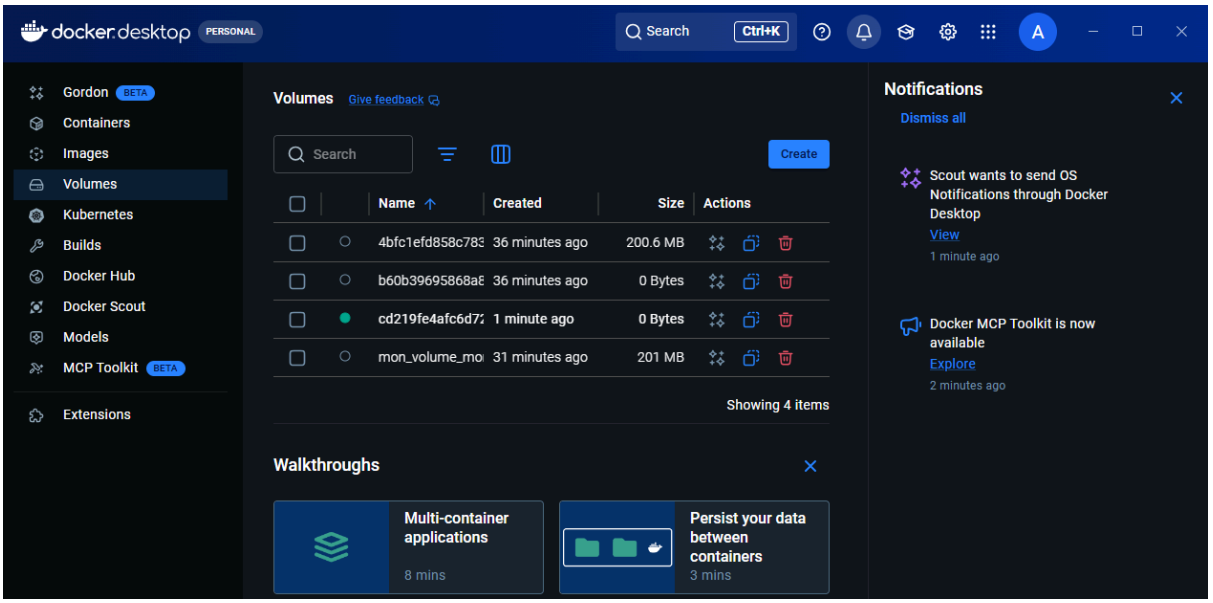


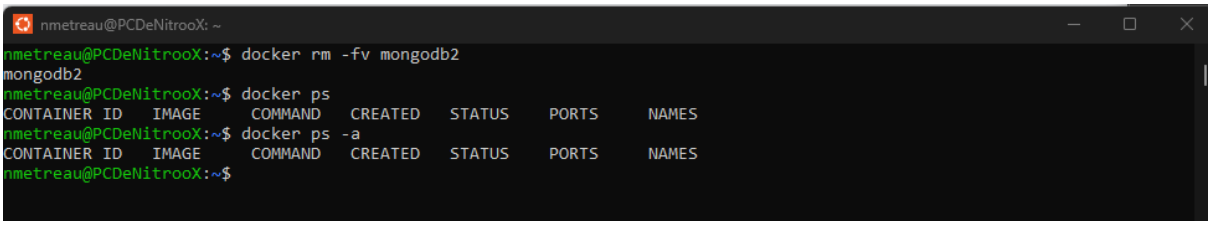
▪ Nous relançons une nouvelle instance de l'image :

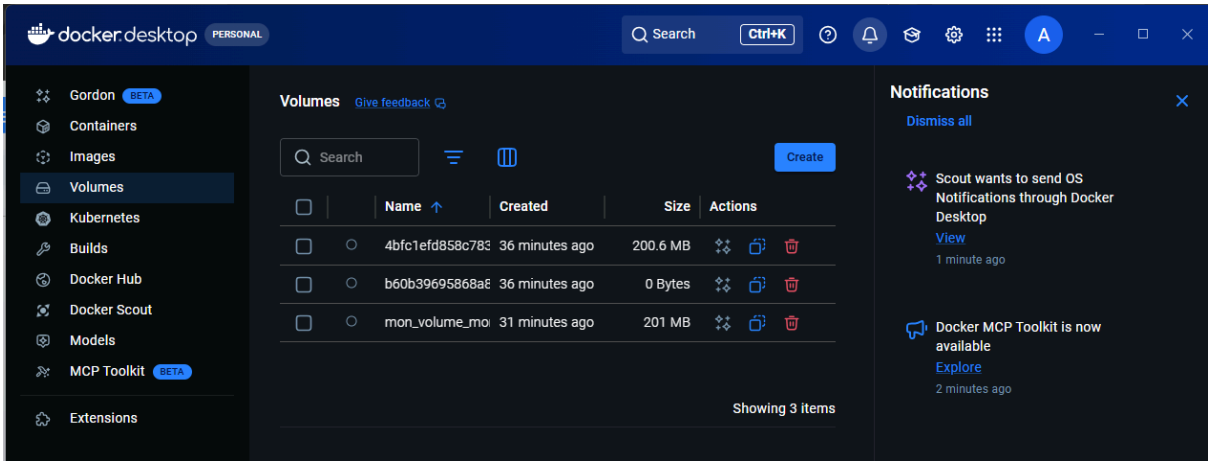




- Nous constatons la persistance

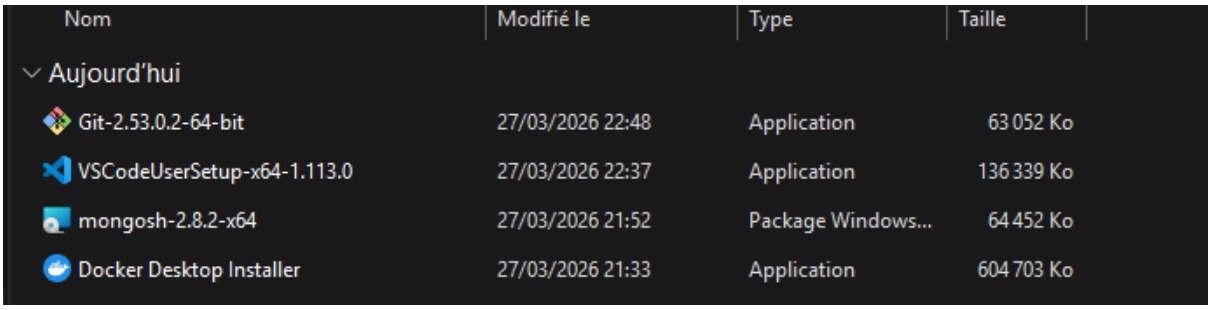






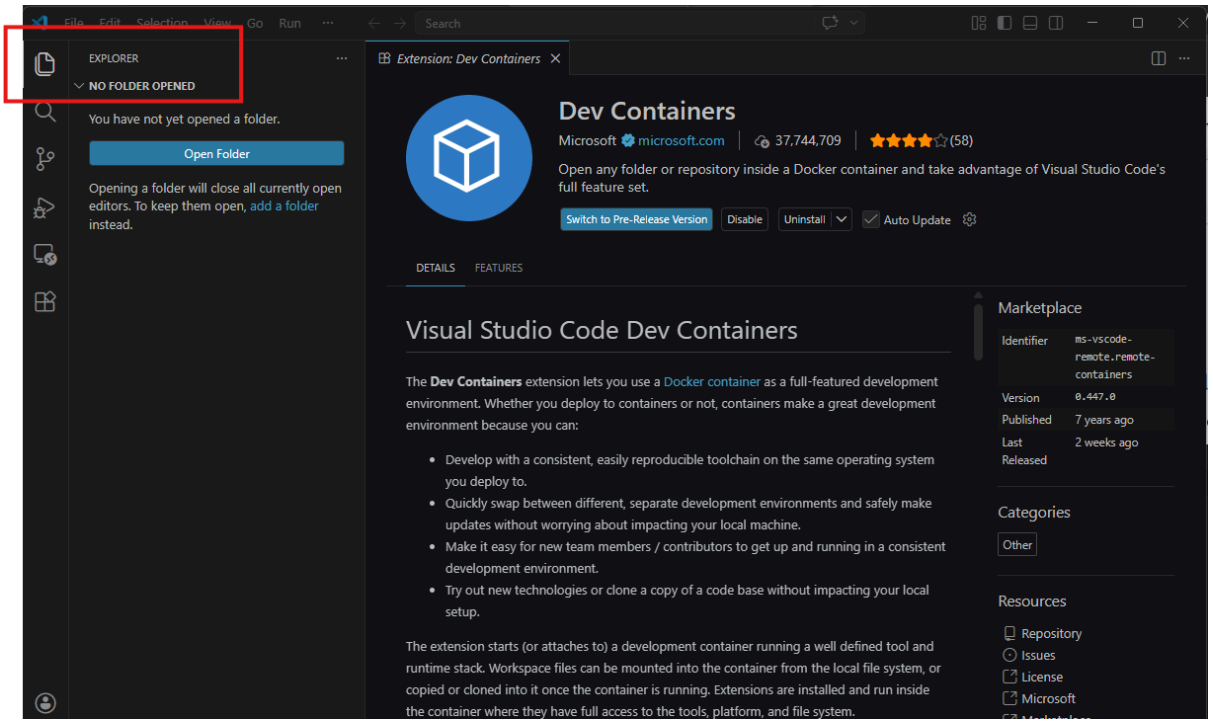
12. Utiliser un conteneur Docker et Visual Studio Code.

▪ Installation de Visual Studio Coded ainsi que Git.

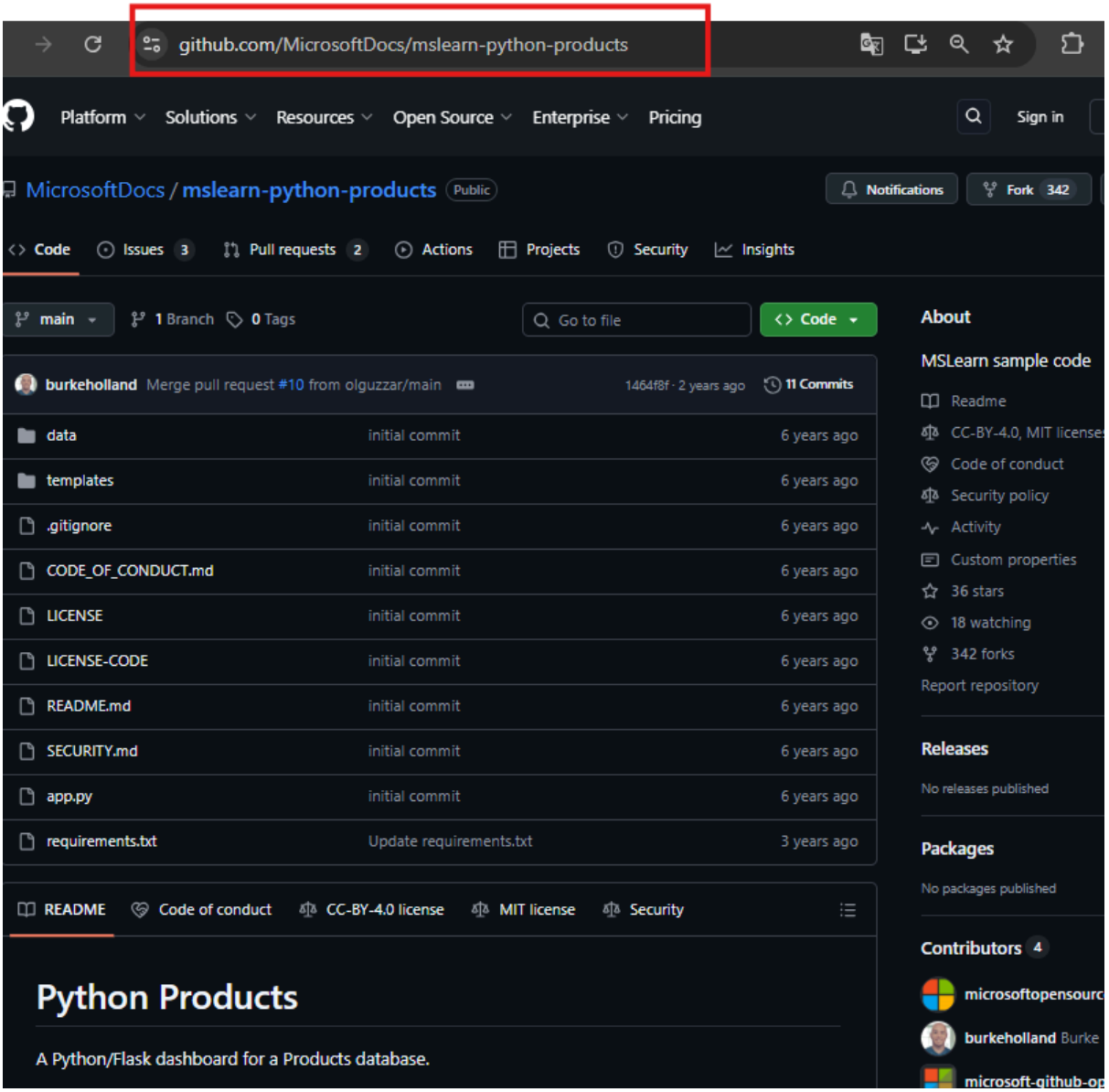


Nom	Modifié le	Type	Taille
Aujourd'hui			
Git-2.53.0.2-64-bit	27/03/2026 22:48	Application	63 052 Ko
VSCodeUserSetup-x64-1.113.0	27/03/2026 22:37	Application	136 339 Ko
mongosh-2.8.2-x64	27/03/2026 21:52	Package Windows...	64 452 Ko
Docker Desktop Installer	27/03/2026 21:33	Application	604 703 Ko

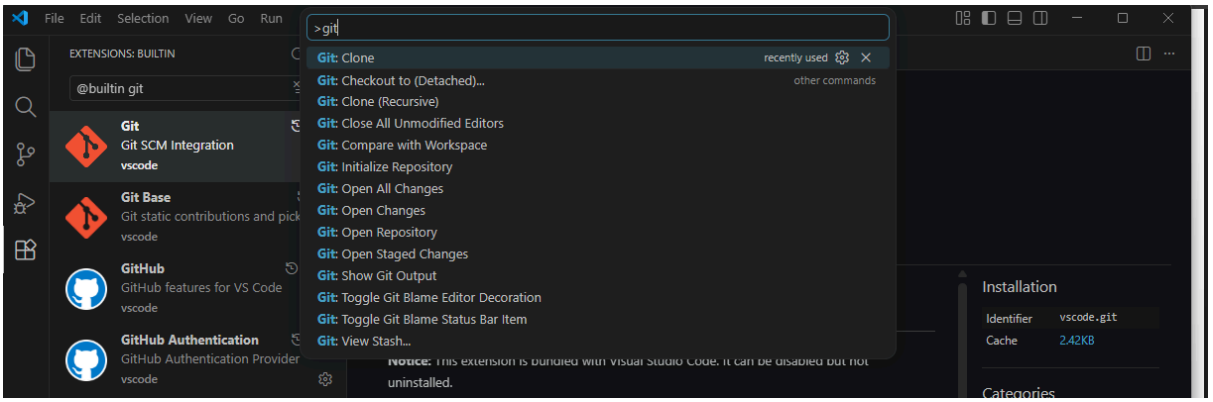
▪ Nous ouvrons une nouvelle instance dans VS Code :



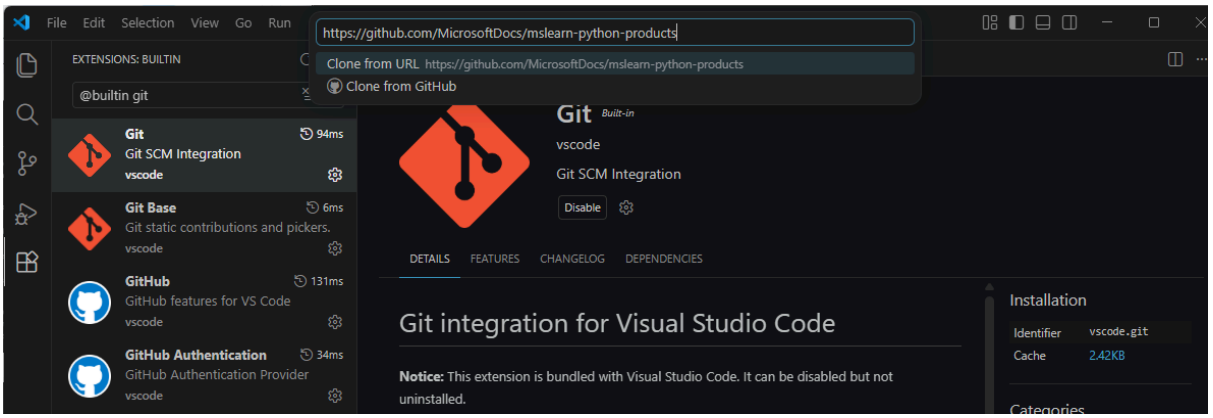
▪ Nous copions l'URL du projet mslearn-python-products qui nous servira d'exemple :



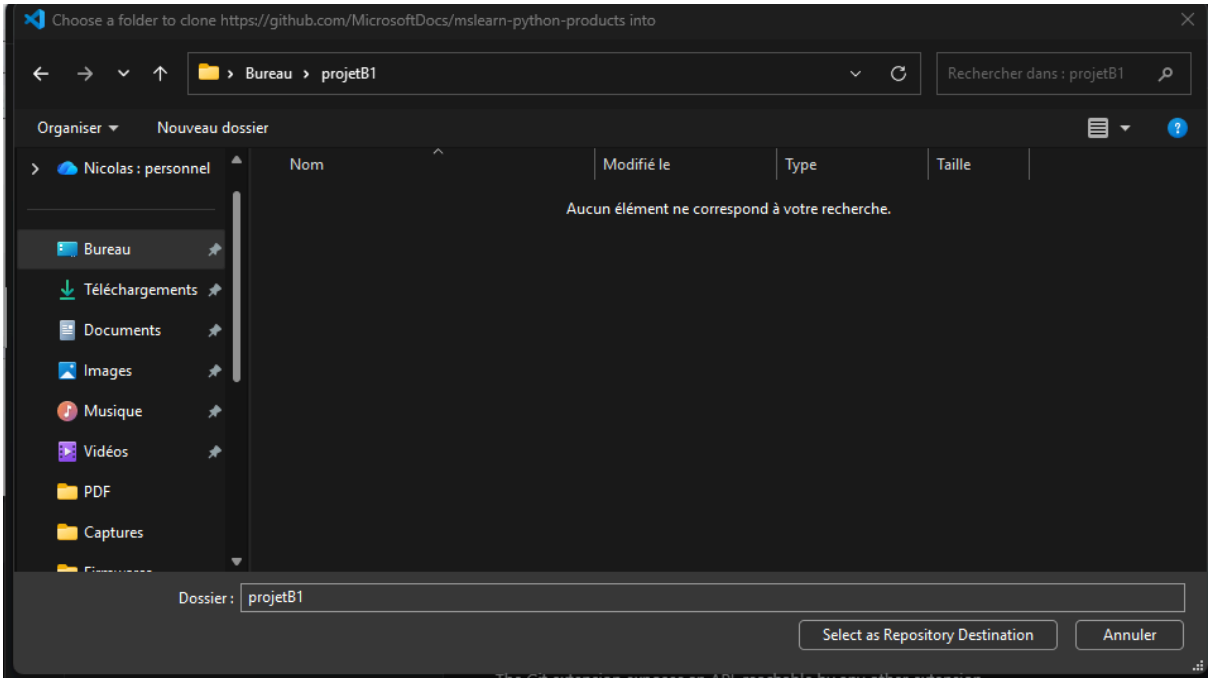
▪ Nous sélectionnons Clone Repository en appuyant sur F1 pour rechercher Git : Clone :

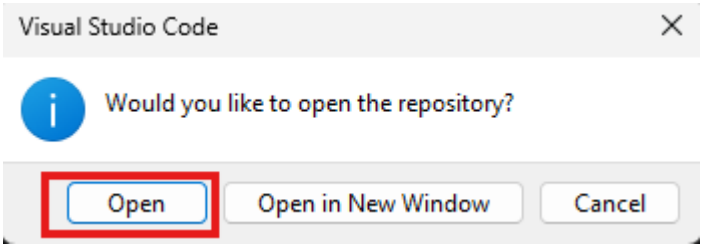


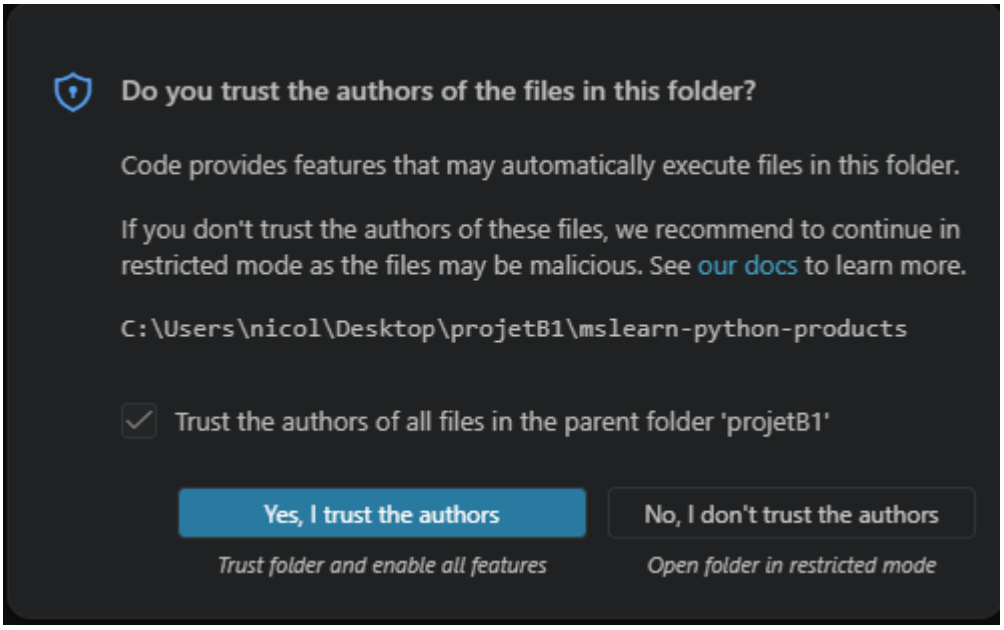
▪ Saisie de l'URL :



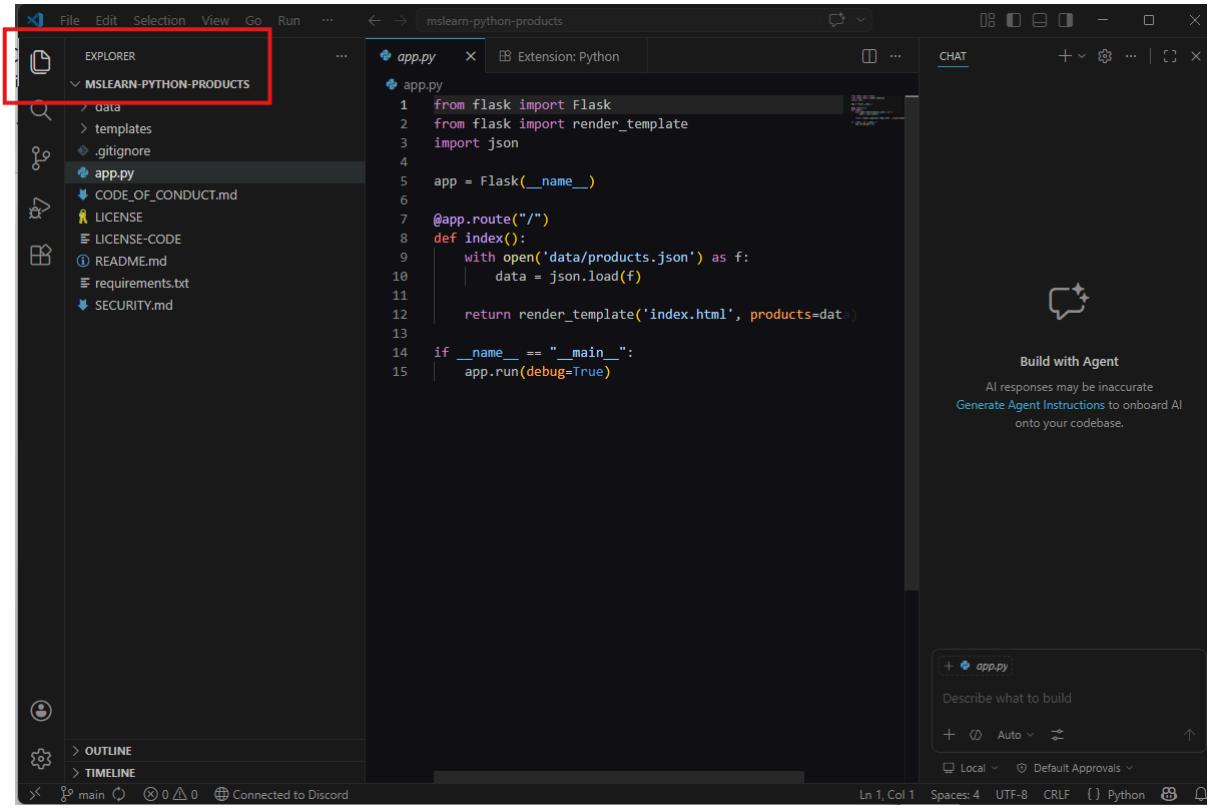
▪ Nous sélectionnons un emplacement libre pour le projet :

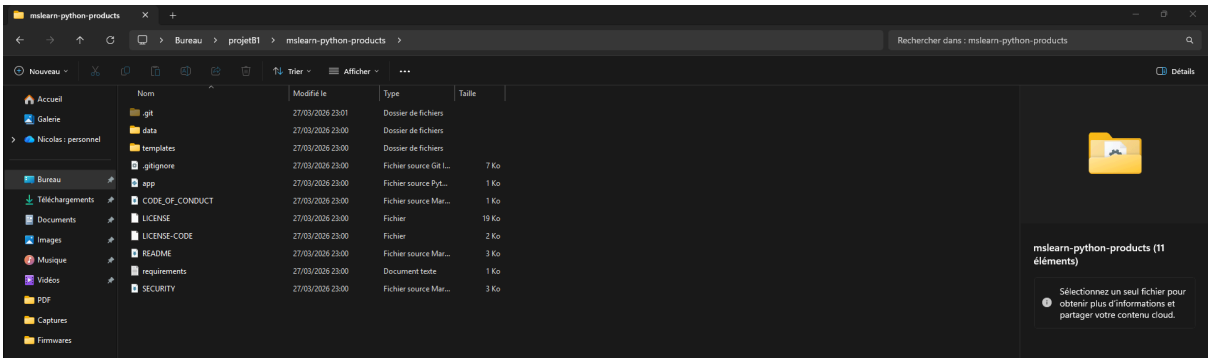




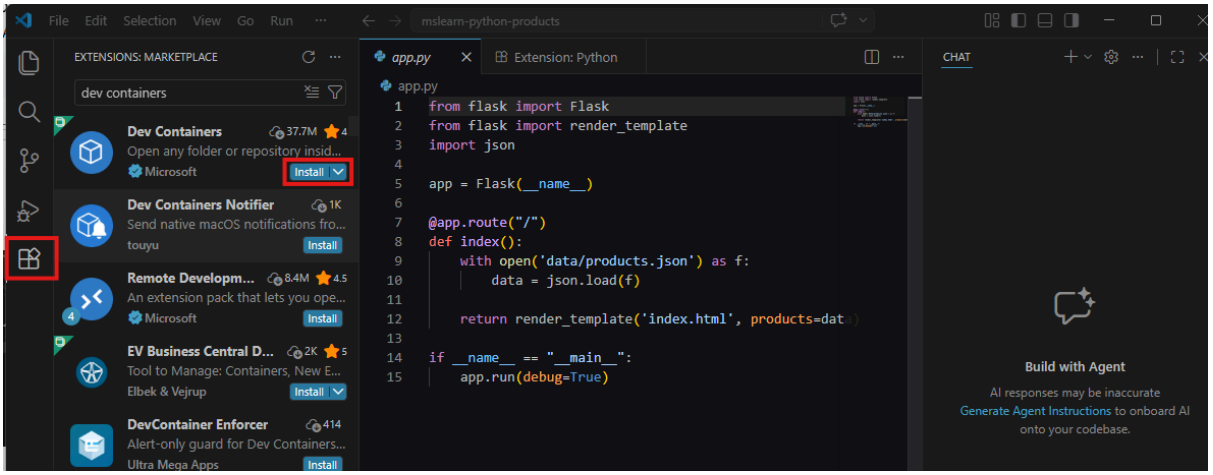


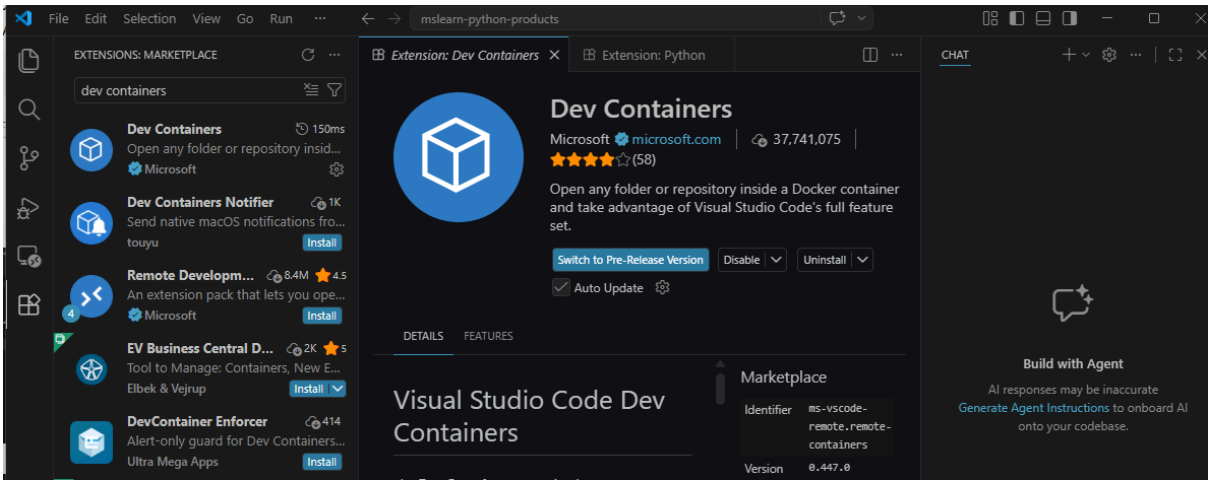
- Le projet est ouvert :



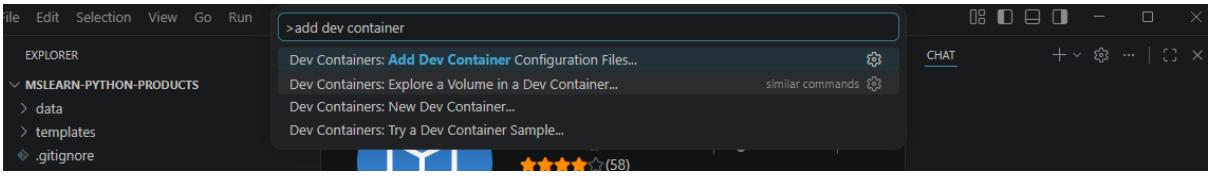


▪ Nous sélectionnons l'icône Extensions afin d'installer l'extension Dev Containers :

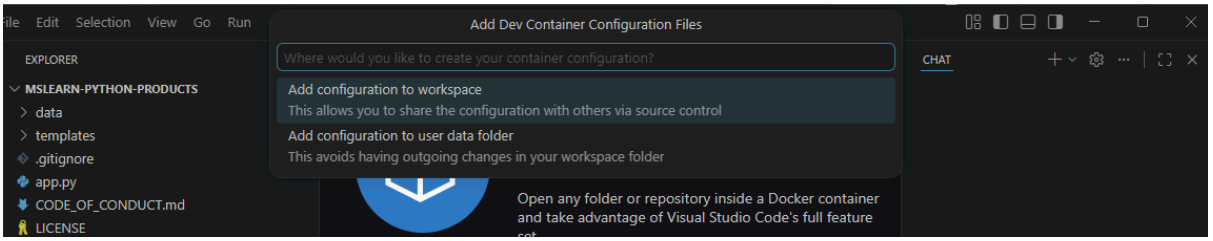


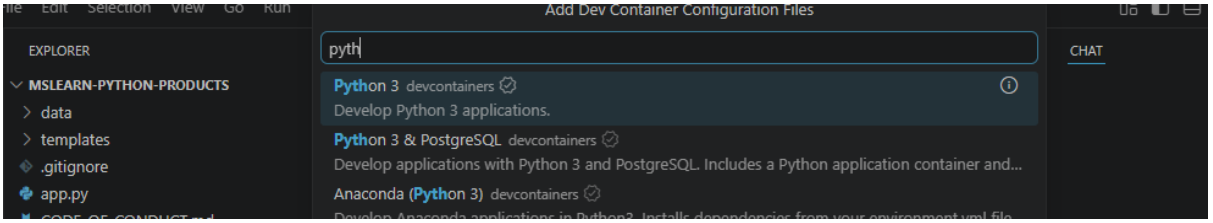


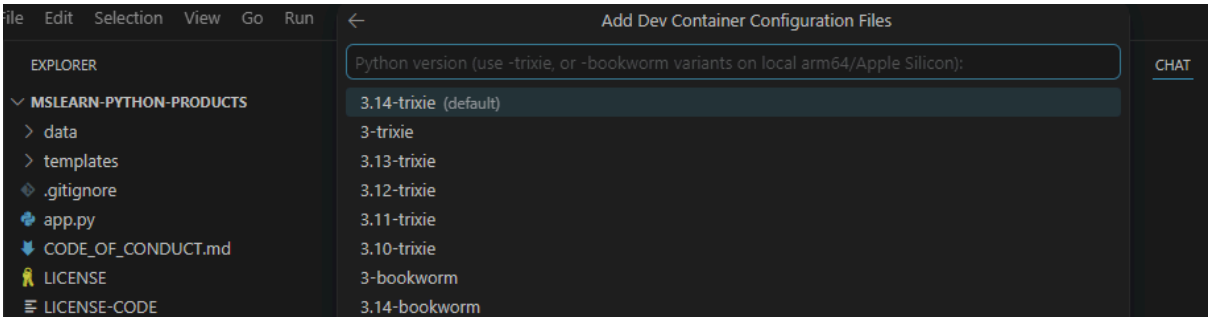
▪ On sélectionne Dev Containers: Add Dev Container Configuration Files.

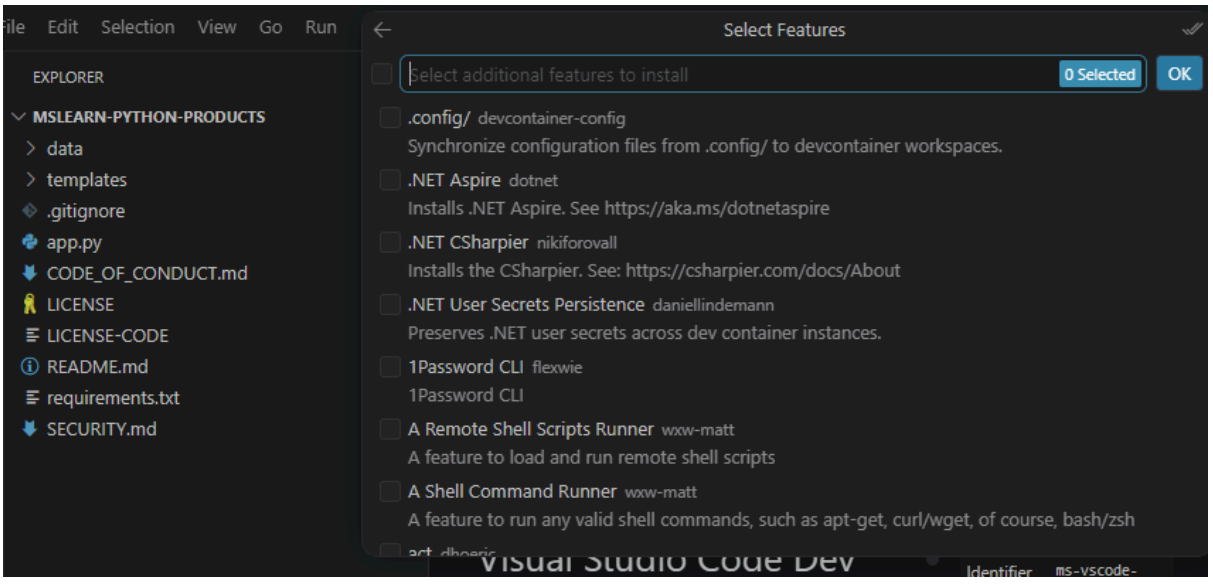


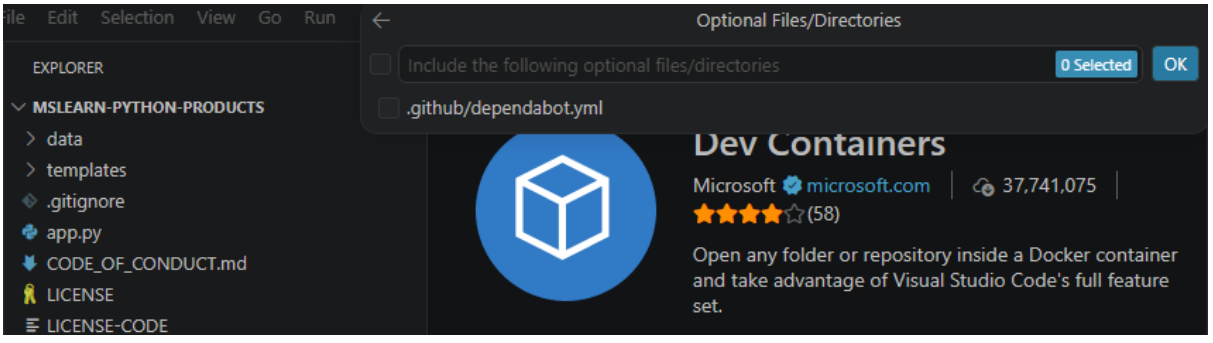
- On sélectionne ensuite Add configuration to workspace



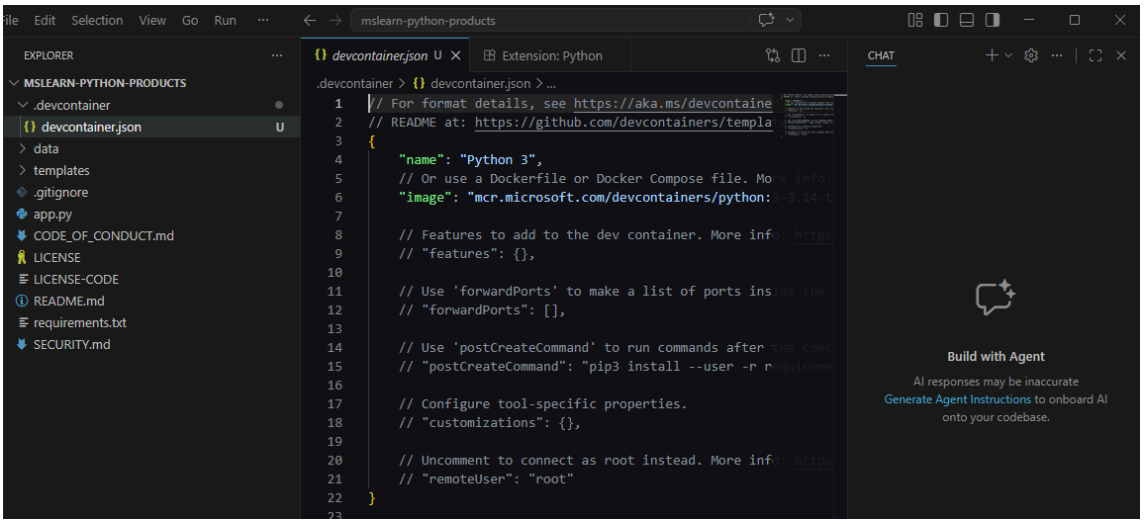




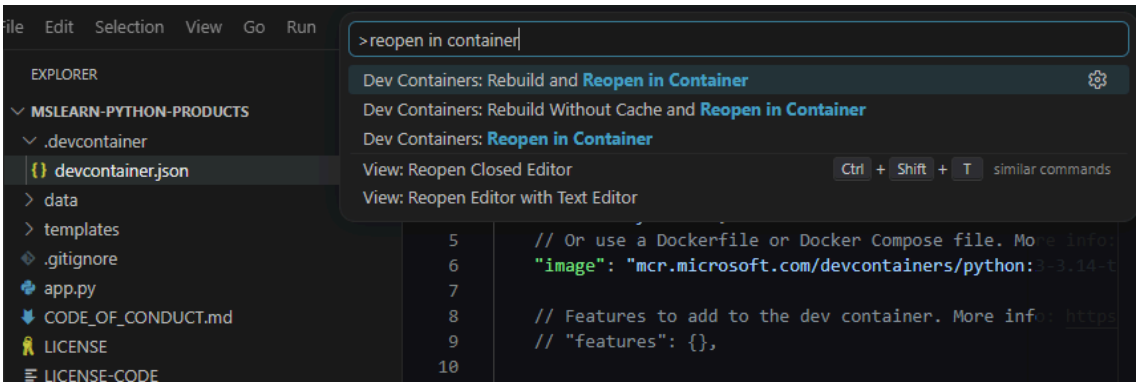




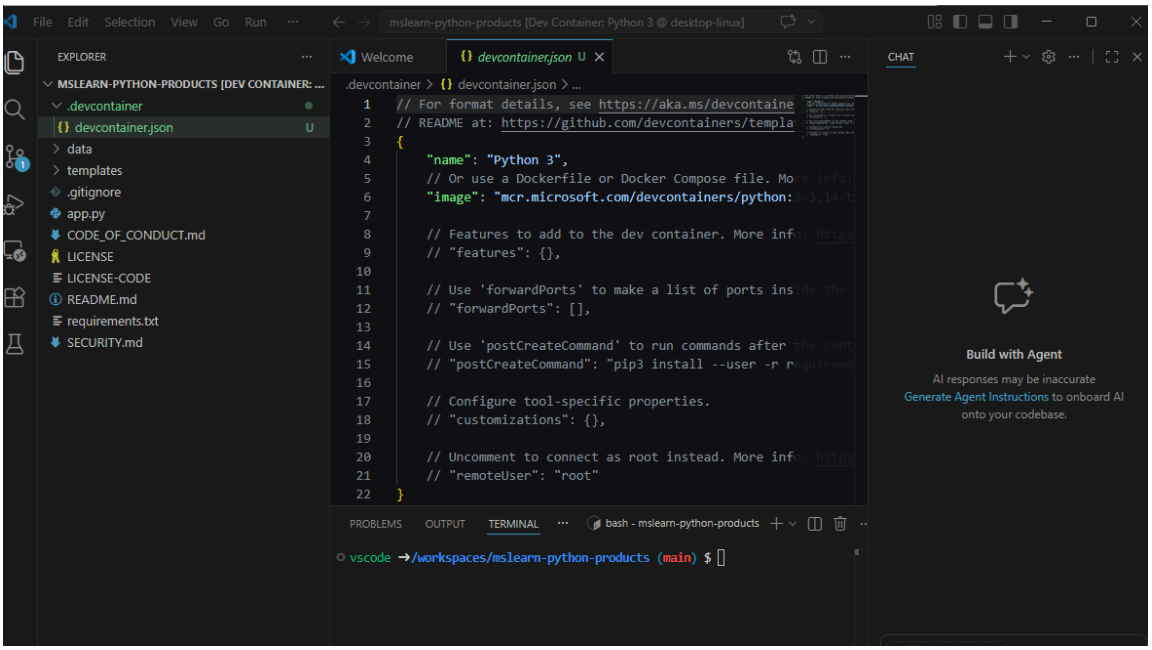
▪ Nous développons ce nouveau dossier contenant un fichier devcontainer.json.



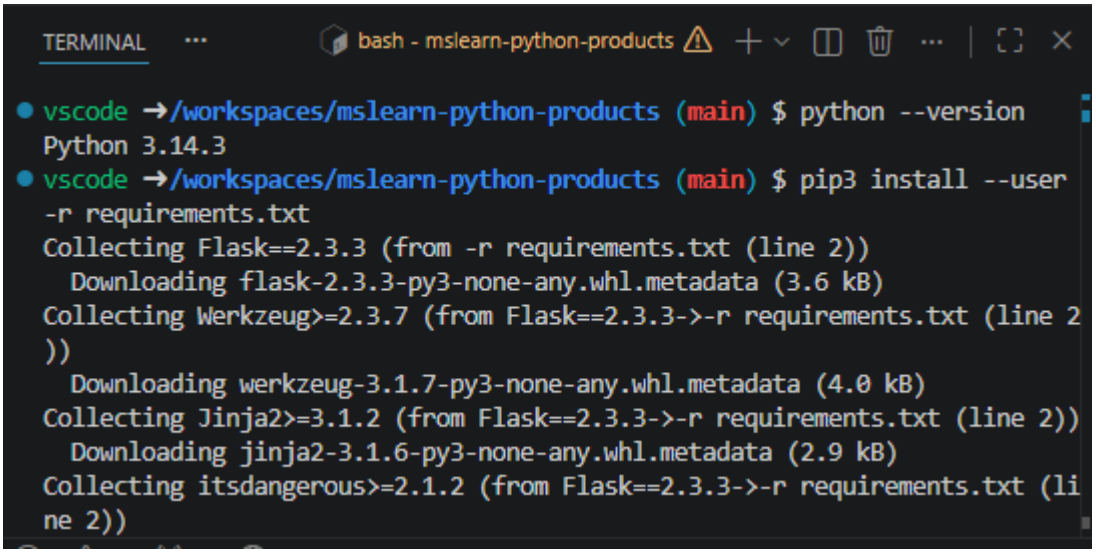
▪ Nous tapons reopen in container et sélectionnons Dev Containers: Reopen in Container dans la liste des options disponibles.



▪ Nous éxaminons l'indicateur distant en bas à gauche de VS Code : il affiche à présent « Dev Container: Python 3 »

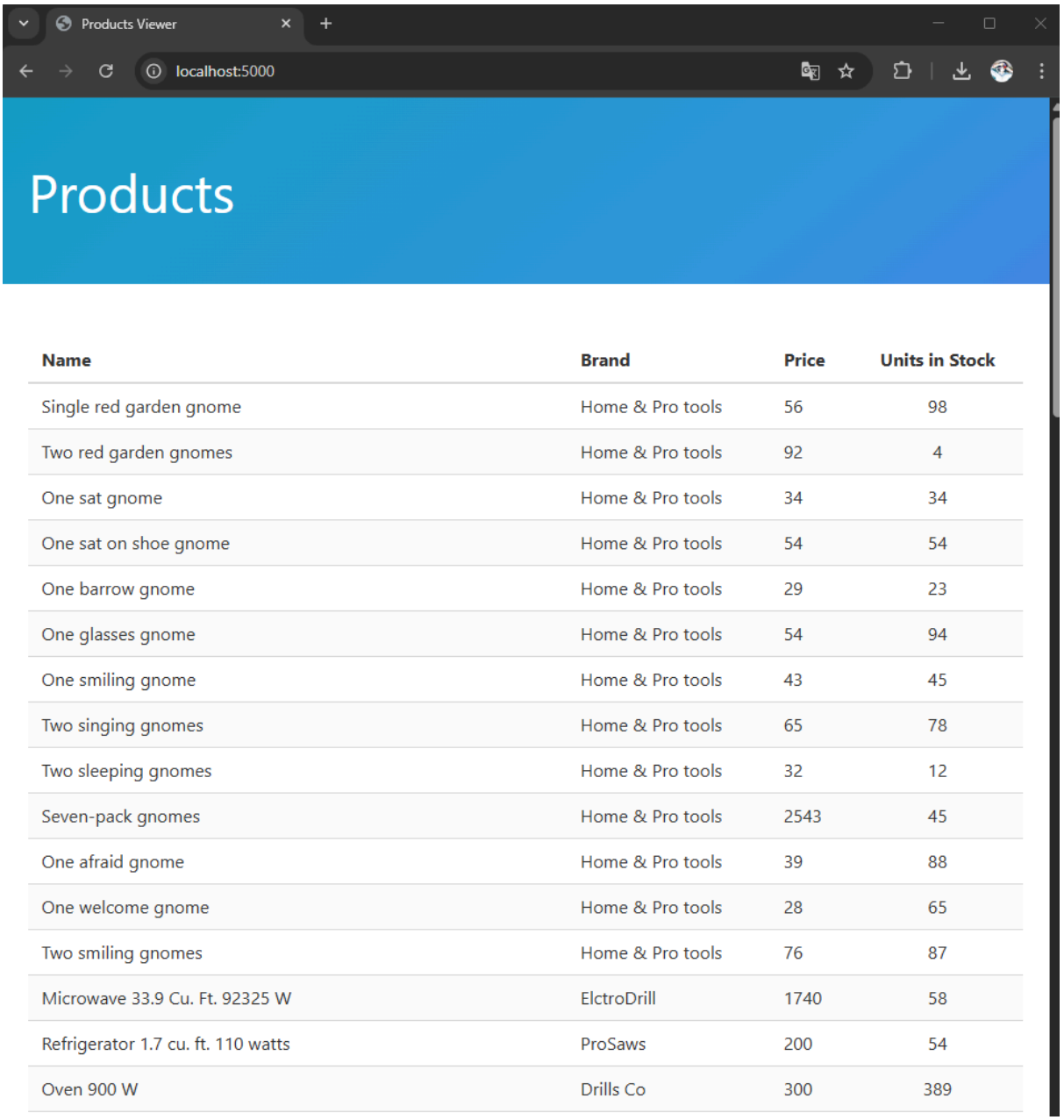


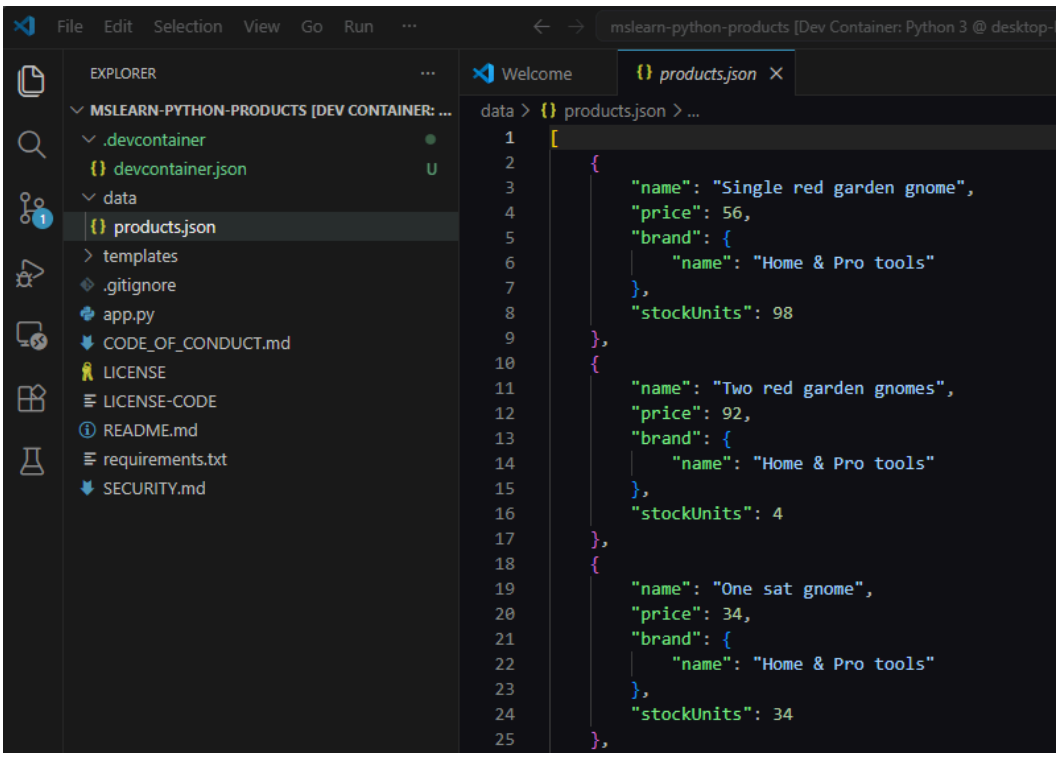
▪ Nous exécutons la commande python -version pour vérifier que Python est installé et pip3 install --user -r requirements.txt :

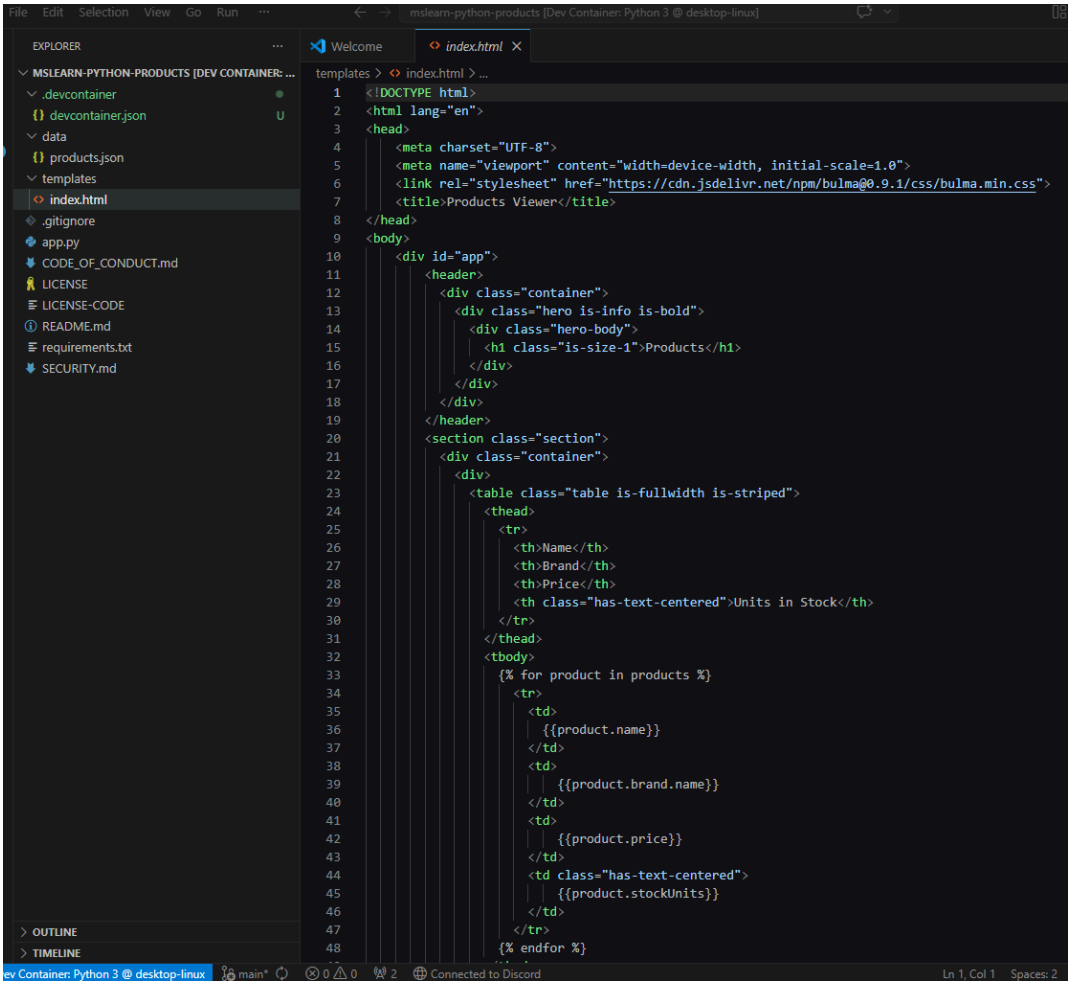


▪ Nous entrons la commande python app.py pour exécuter le projet :

▪ L'application web python s'est bien exécutée :




File Edit Selection View Go Run
mslearn-python-products [Dev Container: Python 3 @ desktop-
EXPLORER
MSLEARN-PYTHON-PRODUCTS [DEV CONTAINER: ...
.devcontainer
devcontainer.json U
data
products.json
templates
.gitignore
app.py
CODE_OF_CONDUCT.md
LICENSE
LICENSE-CODE
README.md
requirements.txt
SECURITY.md
Welcome
products.json
data > products.json > ...
[
{
"name": "Single red garden gnome",
"price": 56,
"brand": {
"name": "Home & Pro tools"
},
"stockUnits": 98
},
{
"name": "Two red garden gnomes",
"price": 92,
"brand": {
"name": "Home & Pro tools"
},
"stockUnits": 4
},
{
"name": "One sat gnome",
"price": 34,
"brand": {
"name": "Home & Pro tools"
},
"stockUnits": 34
},

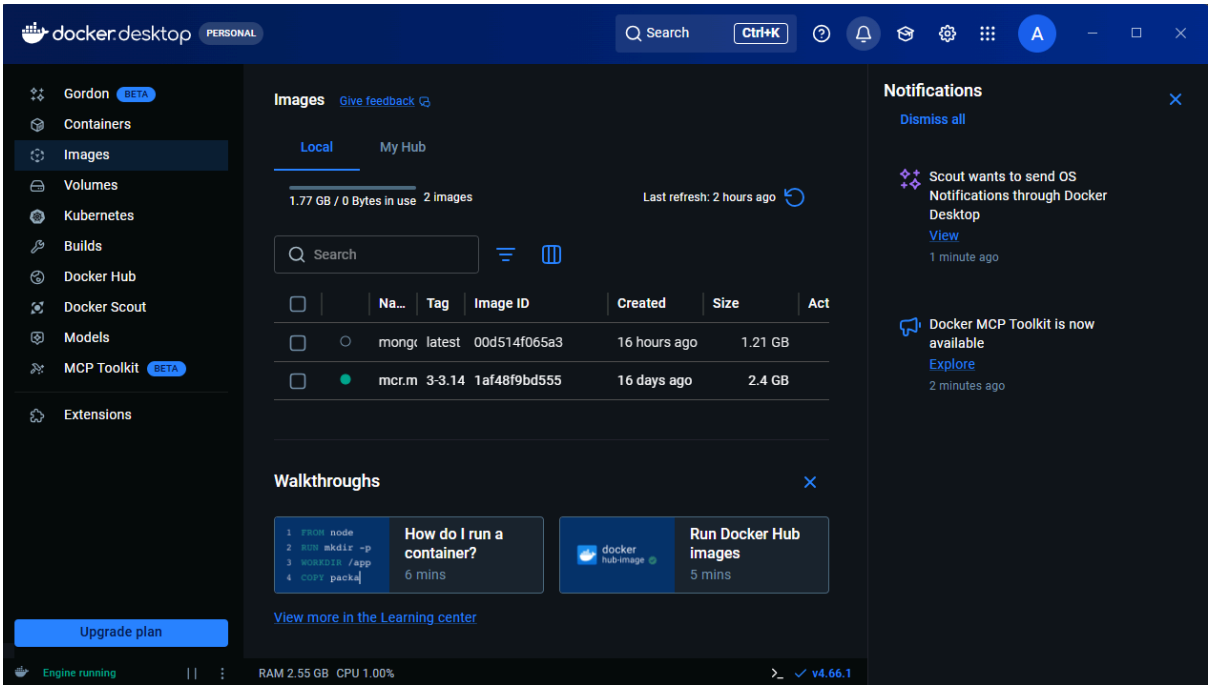

File Edit Selection View Go Run
mslearn-python-products [Dev Container: Python 3 @ desktop-linux]
EXPLORER
MSLEARN-PYTHON-PRODUCTS [DEV CONTAINER: ...
.devcontainer
devcontainer.json U
data
products.json
templates
index.html
.gitignore
app.py
CODE_OF_CONDUCT.md
LICENSE
LICENSE-CODE
README.md
requirements.txt
SECURITY.md
OUTLINE
TIMELINE
Welcome
index.html
templates > index.html > ...
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.9.1/css/bulma.min.css">
<title>Products Viewer</title>
</head>
<body>

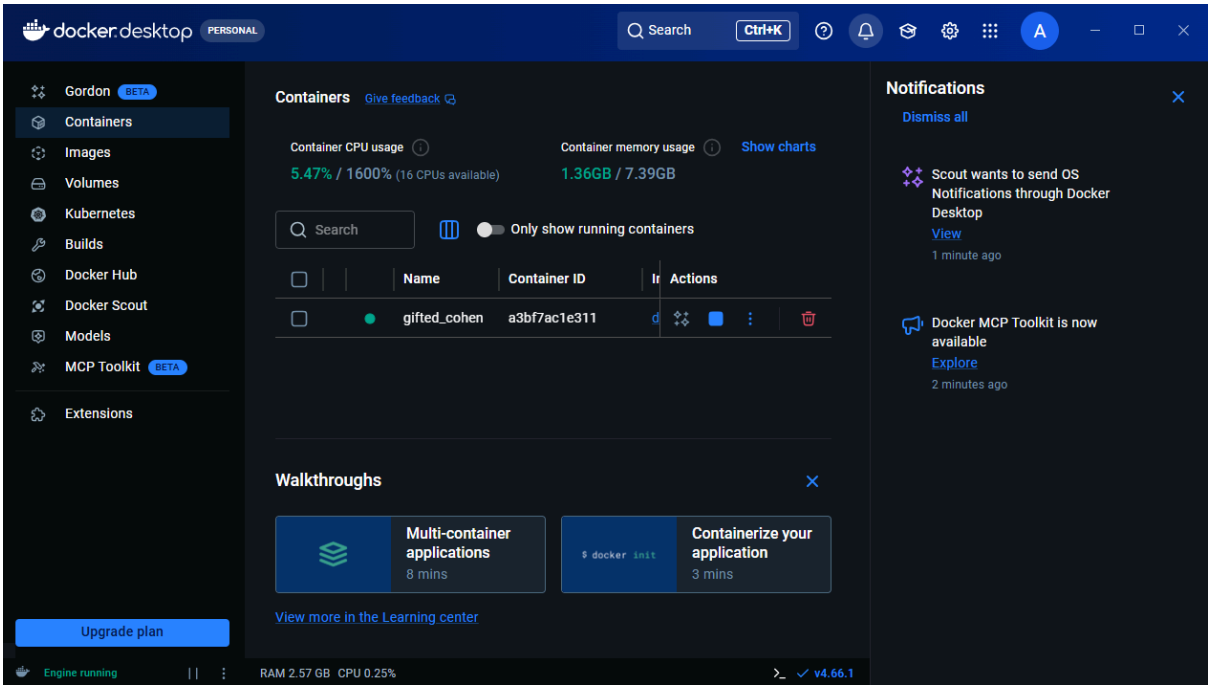
<header>

<h1 class="is-size-1">Products</h1>

</header>
<section class="section">

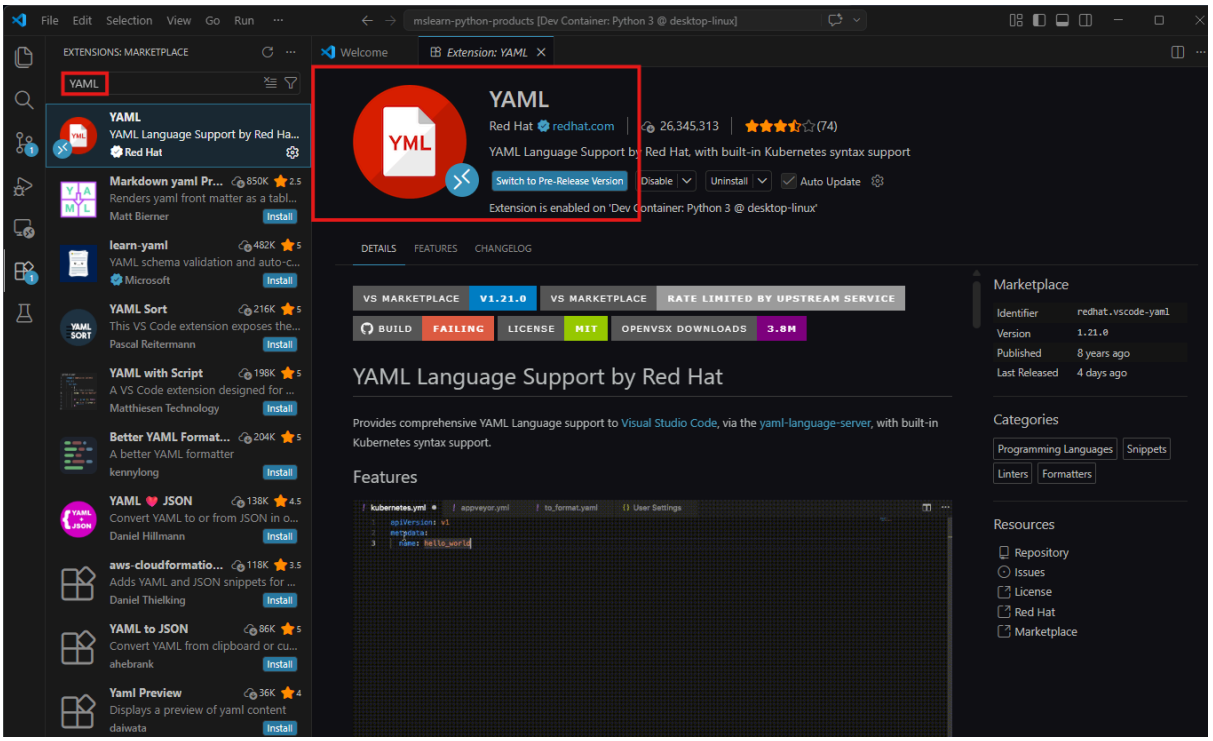
<table class="table is-fullwidth is-striped">
<thead>
<tr>
<th>Name</th>
<th>Brand</th>
<th>Price</th>
<th class="has-text-centered">Units in Stock</th>
</tr>
</thead>
<tbody>
{% for product in products %}
<tr>
<td>
{{product.name}}
</td>
<td>
{{product.brand.name}}
</td>
<td>
{{product.price}}
</td>
<td class="has-text-centered">
{{product.stockUnits}}
</td>
</tr>
{% endfor %}
ev Container: Python 3 @ desktop-linux
main*
Connected to Discord
Ln 1, Col 1
Spaces: 2

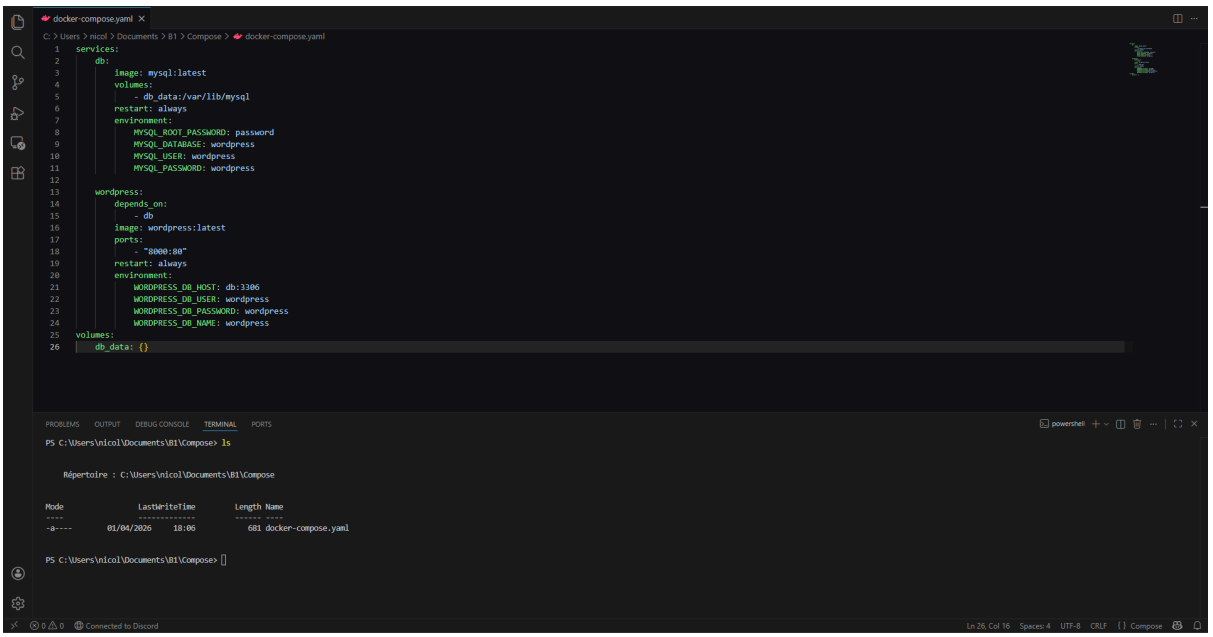

docker desktop
PERSONAL
Search
Ctrl+K
Gordon
BETA
Containers
Images
Volumes
Kubernetes
Builds
Docker Hub
Docker Scout
Models
MCP Toolkit
BETA
Extensions
Upgrade plan
Engine running
Images
Give feedback
Local
My Hub
1.77 GB / 0 Bytes in use
2 images
Last refresh: 2 hours ago
Search
Na...
Tag
Image ID
Created
Size
Act
mongo
latest
00d514f065a3
16 hours ago
1.21 GB
mcr.m
3-3.14
1af48f9bd555
16 days ago
2.4 GB
Walkthroughs
How do I run a container?
6 mins
Run Docker Hub images
5 mins
View more in the Learning center
RAM 2.55 GB CPU 1.00%
v4.66.1
Notifications
Dismiss all
Scout wants to send OS Notifications through Docker Desktop
View
1 minute ago
Docker MCP Toolkit is now available
Explore
2 minutes ago


docker desktop
PERSONAL
Search
Ctrl+K
Gordon
BETA
Containers
Images
Volumes
Kubernetes
Builds
Docker Hub
Docker Scout
Models
MCP Toolkit
BETA
Extensions
Upgrade plan
Engine running
Containers
Give feedback
Container CPU usage
5.47% / 1600% (16 CPUs available)
Container memory usage
1.36GB / 7.39GB
Show charts
Search
Only show running containers
Name
Container ID
Actions
gifted_cohen
a3bf7ac1e311
Walkthroughs
Multi-container applications
8 mins
Containerize your application
3 mins
View more in the Learning center
RAM 2.57 GB CPU 0.25%
v4.66.1
Notifications
Dismiss all
Scout wants to send OS Notifications through Docker Desktop
View
1 minute ago
Docker MCP Toolkit is now available
Explore
2 minutes ago

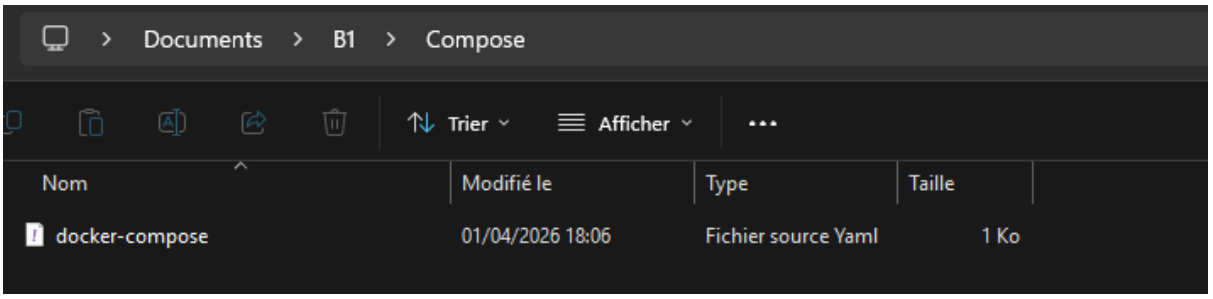
13. Monter une stack applicative avec Docker Compose.

- Installation de l'extension YAML dans VS Code :





▪ Création du fichier docker-compose.yml :

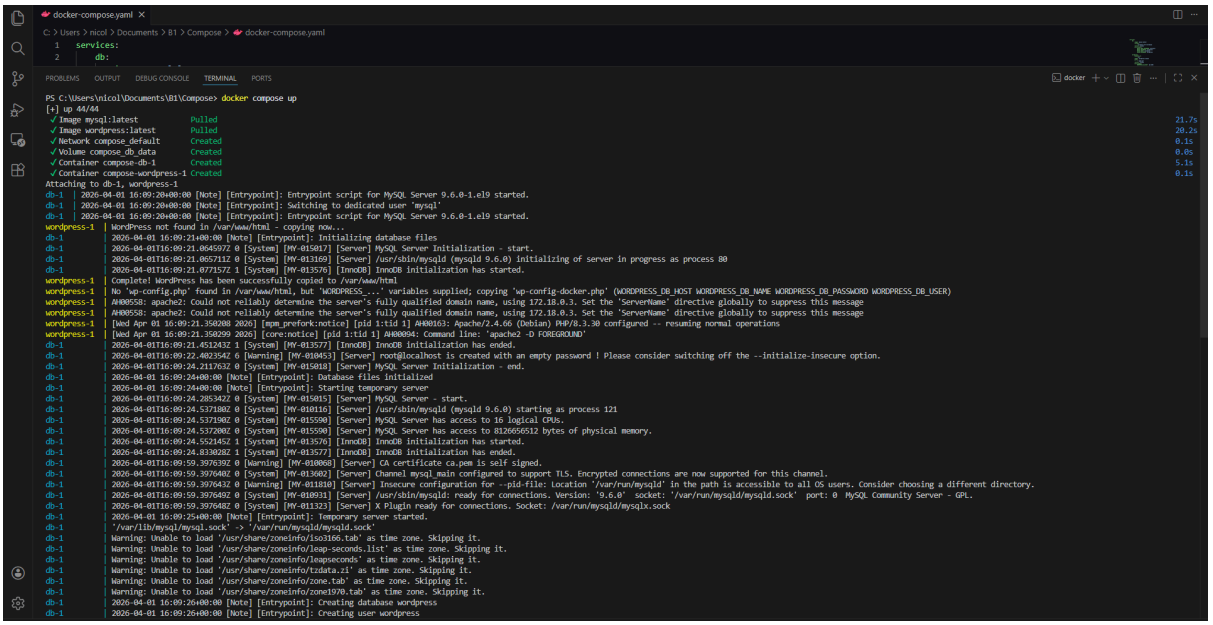


docker-compose.yaml

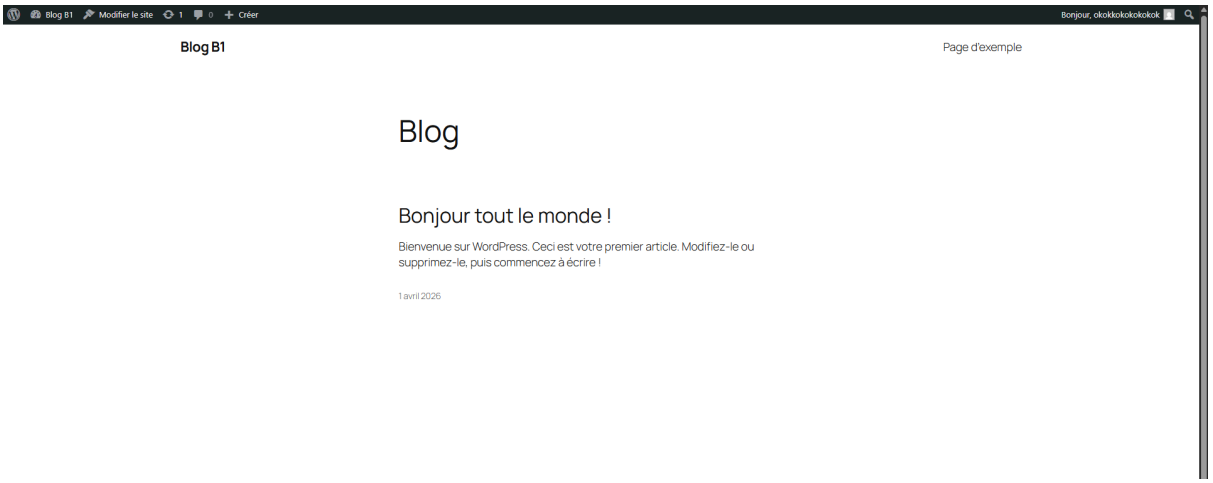
C: > Users > nicol > Documents > B1 > Compose > docker-compose.yaml

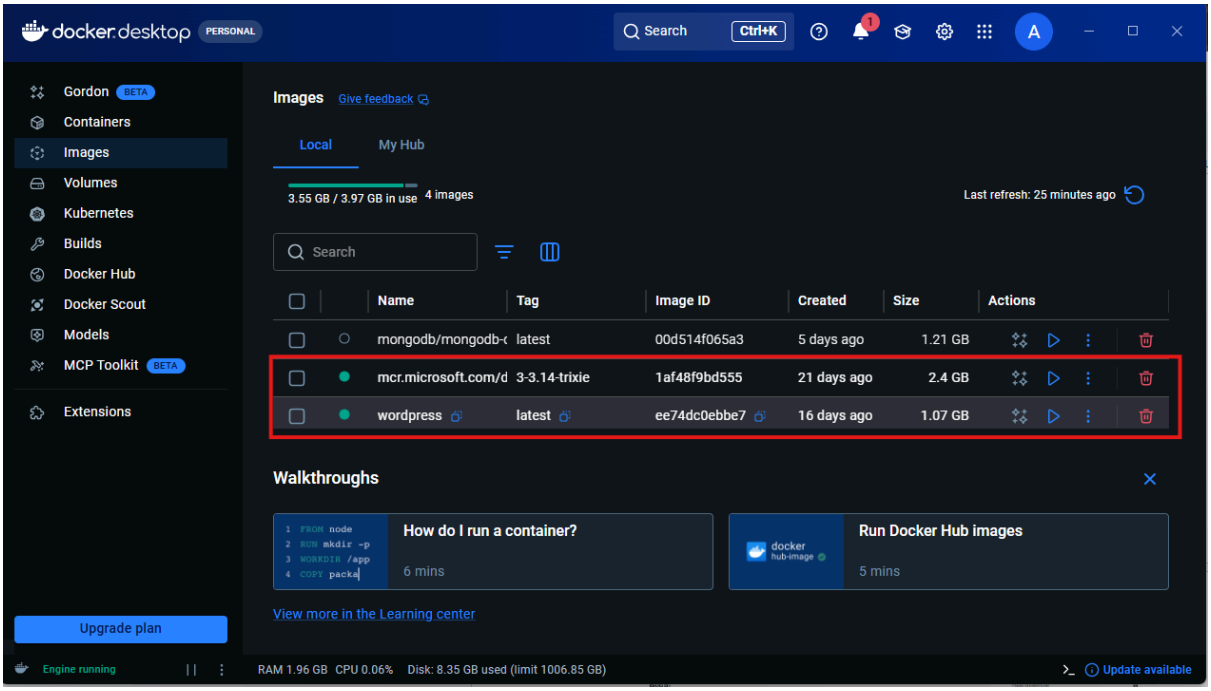
```yaml
services:
    db:
        image: mysql:latest
        volumes:
            - db_data:/var/lib/mysql
        restart: always
        environment:
            MYSQL_ROOT_PASSWORD: password
            MYSQL_DATABASE: wordpress
            MYSQL_USER: wordpress
            MYSQL_PASSWORD: wordpress

    wordpress:
        depends_on:
            - db
        image: wordpress:latest
        ports:
            - "8000:80"
        restart: always
        environment:
            WORDPRESS_DB_HOST: db:3306
            WORDPRESS_DB_USER: wordpress
            WORDPRESS_DB_PASSWORD: wordpress
            WORDPRESS_DB_NAME: wordpress
volumes:
    db_data: {}
```

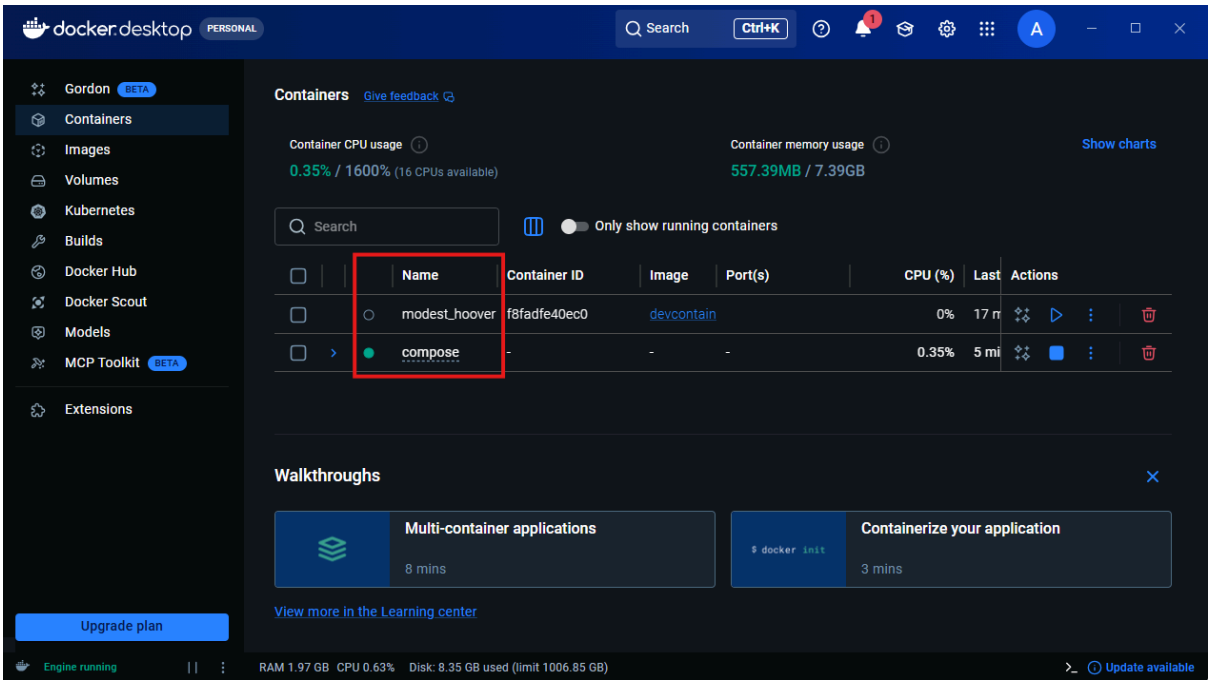
▪ Nous lançons la stack applicative avec la commande docker compose up :

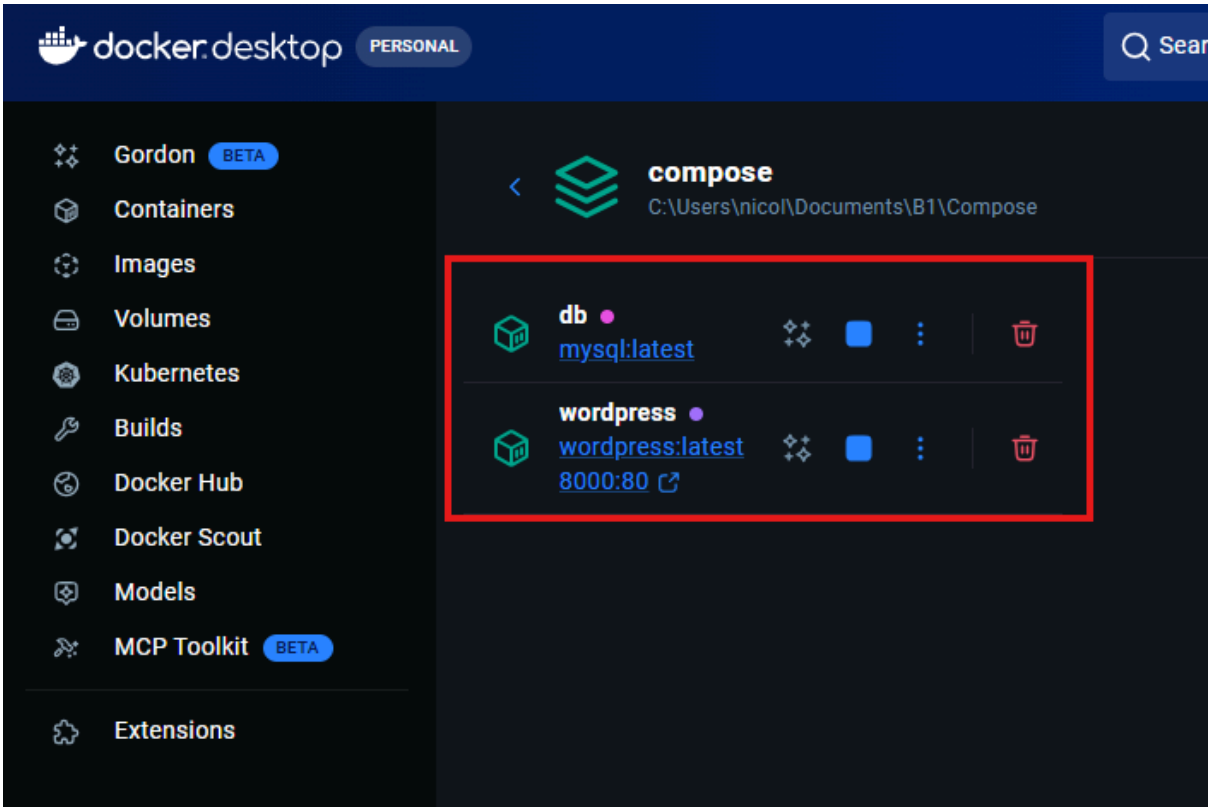


▪ Nous accédons au blog en saisissant l'url http://localhost:8000 :




docker desktop
PERSONAL
Search
Ctrl+K
Gordon
BETA
Containers
Images
Volumes
Kubernetes
Builds
Docker Hub
Docker Scout
Models
MCP Toolkit
BETA
Extensions
Upgrade plan
Images
Give feedback
Local
My Hub
3.55 GB / 3.97 GB in use
4 images
Last refresh: 25 minutes ago
Search
Name
Tag
Image ID
Created
Size
Actions
mongodb/mongodb-c
latest
00d514f065a3
5 days ago
1.21 GB
mcr.microsoft.com/d
3-3.14-trixie
1af48f9bd555
21 days ago
2.4 GB
wordpress
latest
ee74dc0ebbe7
16 days ago
1.07 GB
Walkthroughs
How do I run a container?
6 mins
Run Docker Hub images
5 mins
View more in the Learning center
Engine running
RAM 1.96 GB CPU 0.06%
Disk: 8.35 GB used (limit 1006.85 GB)
Update available


docker desktop
PERSONAL
Search
Ctrl+K
Gordon
BETA
Containers
Images
Volumes
Kubernetes
Builds
Docker Hub
Docker Scout
Models
MCP Toolkit
BETA
Extensions
Upgrade plan
Containers
Give feedback
Container CPU usage
0.35% / 1600% (16 CPUs available)
Container memory usage
557.39MB / 7.39GB
Show charts
Search
Only show running containers
Name
Container ID
Image
Port(s)
CPU (%)
Last
Actions
modest_hoover
f8fadfe40ec0
devcontain
0%
compose
0.35%
Walkthroughs
Multi-container applications
8 mins
Containerize your application
3 mins
View more in the Learning center
Engine running
RAM 1.97 GB CPU 0.63%
Disk: 8.35 GB used (limit 1006.85 GB)
Update available


docker desktop
PERSONAL
Gordon
BETA
Containers
Images
Volumes
Kubernetes
Builds
Docker Hub
Docker Scout
Models
MCP Toolkit
BETA
Extensions
compose
C:\Users\nicol\Documents\B1\Compose
db
mysql:latest
wordpress
wordpress:latest
8000:80