DOCKER

INTRODUCTION

Commandes Docker de base

1. Docker version et info

docker --version : Affiche la version de Docker installée.

docker info : Affiche les informations détaillées sur l'installation Docker, y compris le nombre de conteneurs et d'images.

2. Gestion des conteneurs

docker run : Lance un nouveau conteneur.

docker run -d -p 80:80 nginx

Ici, -d lance le conteneur en arrière-plan (détaché) et -p mappe le port 80 du conteneur au port 80 de l'hôte.

docker ps : Affiche la liste des conteneurs en cours d'exécution.

docker ps

docker ps -a : Affiche la liste de tous les conteneurs, y compris ceux arrêtés.

docker ps -a

docker stop : Arrête un conteneur en cours d'exécution.

docker stop <container\_id>

docker start : Démarre un conteneur arrêté.

docker start <container\_id>

docker rm : Supprime un conteneur.

docker rm <container\_id>

docker logs : Affiche les logs d'un conteneur.

docker logs <container\_id>

3. Gestion des images

docker pull : Télécharge une image depuis Docker Hub.

docker pull ubuntu

docker images : Affiche la liste des images disponibles localement.

docker images

docker rmi : Supprime une image.

docker rmi <image\_id>

4. Dockerfile et construction d'images

docker build : Construit une image à partir d'un Dockerfile.

docker build -t myapp:latest .

Ici, -t permet de nommer l'image (ici myapp:latest), et . spécifie le contexte de construction (le répertoire actuel).

5. Réseau

docker network ls : Liste les réseaux Docker.

docker network ls

docker network create : Crée un nouveau réseau.

docker network create mynetwork

docker network connect : Connecte un conteneur à un réseau.

docker network connect mynetwork <container\_id>

docker network disconnect : Déconnecte un conteneur d'un réseau.

docker network disconnect mynetwork <container\_id>

Exemple pratique

Pour illustrer comment utiliser ces commandes, voici un exemple simple où nous allons créer et exécuter une application web simple avec Docker.

1. Créer un Dockerfile

Créez un fichier nommé Dockerfile avec le contenu suivant :

Dockerfile

FROM nginx:latest

COPY . /usr/share/nginx/html

2. Construire l'image

docker build -t mynginxapp:latest .

3. Lancer un conteneur

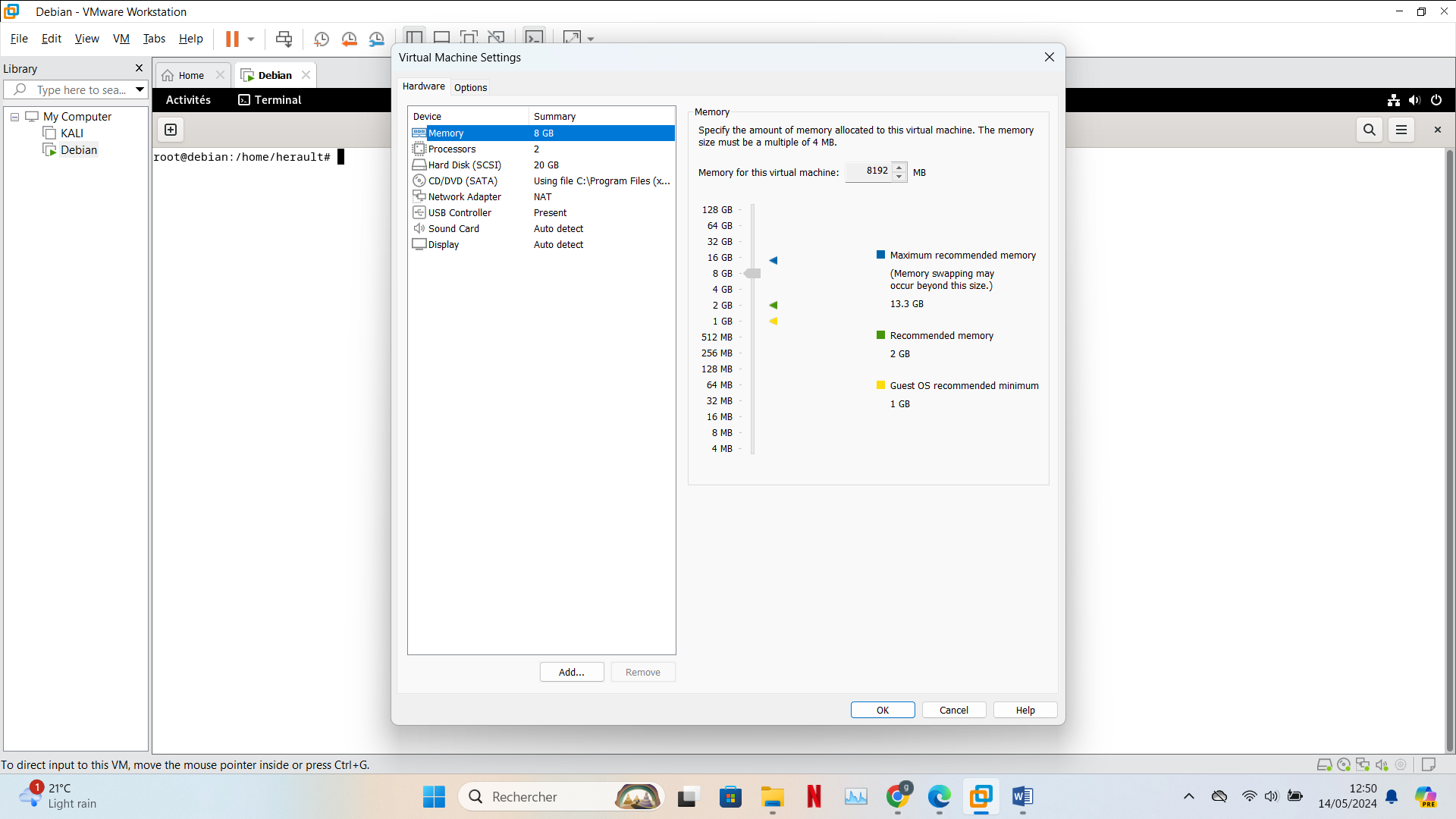
docker run -d -p 8080:80 mynginxapp:latest

4. Vérifier l'exécution

Ouvrez votre navigateur et allez à http://localhost:8080 pour voir votre application en cours d'exécution.

JOB1 :

* INSTALLATION DE VM



- INSTALLATION DE DOCKER EN CLI

**Mettre à jour la liste des paquets :**

sudo apt update

**Installer les prérequis pour utiliser un dépôt sur HTTPS :**

sudo apt install apt-transport-https ca-certificates curl gnupg2 software-properties-common lsb-release

**Ajouter la clé GPG officielle de Docker :**

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

**Configurer le dépôt de Docker**

**Ajoutez le dépôt de Docker aux sources APT de Debian pour accéder aux paquets de la CLI sans installer le daemon Docker :**

echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

**Mettre à jour la liste des paquets après l'ajout du nouveau dépôt :**

sudo apt update

**Installer Docker CLI :**

sudo apt install docker-ce-cli

**Vérifier l'installation de la CLI**

**Vous pouvez vérifier que la CLI de Docker a été correctement installée en exécutant :**

docker --version

JOB2 : TEST DE L’INSTALLATION DE DOCKER AVEC << helloword>>

**Vérifier le statut du service Docker :**

sudo systemctl status docker

**Démarrer le service Docker**

**Si Docker est installé mais le service n'est pas actif, essayez de le démarrer avec :**

sudo systemctl start docker

**Activer le démarrage automatique de Docker**

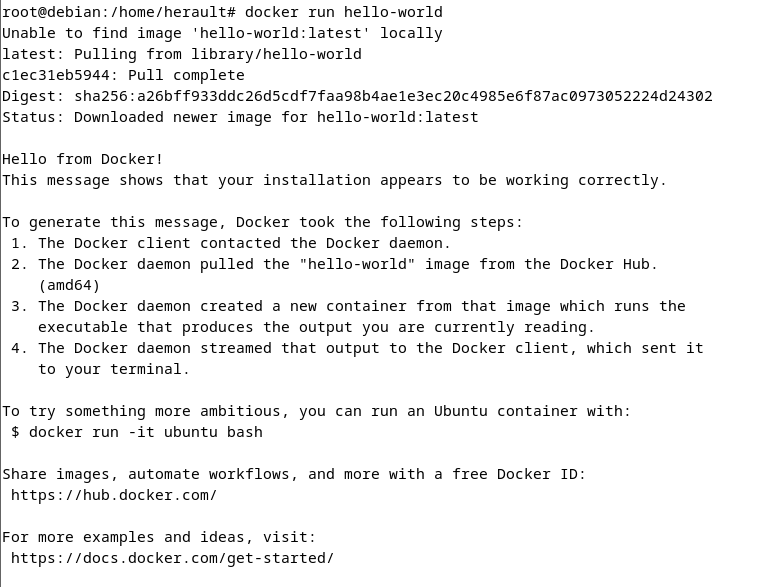
**Pour que Docker démarre automatiquement à chaque démarrage du système, utilisez :**

sudo systemctl enable docker

**Tester Docker**

**Enfin, testez Docker pour vous assurer qu'il fonctionne correctement :**

docker run hello-world



JOB3 : Utilisation de « Dockerfile » pour recréer le conteneur « helloworld » depuis une

Image Debian minimum.

Créez un nouveau répertoire pour votre projet Docker :

**mkdir herault-hello-world**

**cd herault-hello-world**

Créez un fichier Dockerfile :

Utilisez votre éditeur de texte préféré pour créer un fichier Dockerfile sans extension dans ce répertoire.

**nano Dockerfile**

Ajoutez le contenu suivant dans le Dockerfile :

**# Utiliser une image de base Debian minimale**

**FROM debian:stable-slim**

**# Mettre à jour les paquets et installer curl (facultatif, pour plus de fonctionnalité)**

**RUN apt-get update && apt-get install -y curl && rm -rf /var/lib/apt/lists/\***

**# Ajouter un script ou une commande qui sera exécuté lorsque le conteneur démarre**

**CMD ["echo", "Hello from Docker! This message shows that your installation appears to be working correctly."]**

Sauvegardez et fermez le fichier.

Ici, FROM debian:stable-slim indique que nous utilisons l'image Debian la plus réduite comme base.

RUN exécute des commandes pour mettre à jour les paquets et installer curl.

CMD définit la commande par défaut qui sera exécutée lors du démarrage du conteneur.

Construire l'image Docker

Dans le répertoire contenant votre Dockerfile, exécutez la commande suivante pour construire l'image Docker.

Remplacez mon-image-hello-world par le nom que vous souhaitez donner à votre image :

**docker build -t herault-hello .**

root@debian:/home/herault/herault-hello-world# **docker run herault-hello**

**Hello from Docker-HERAULT-BEA-GERALD! This message shows that your installation appears to be working correctly.**

root@debian:/home/herault/herault-hello-world#

JOB4 :

Pour créer une image Docker avec un serveur SSH configuré pour écouter sur un port non standard et avec un accès pour l'utilisateur root (mot de passe root123)

Je vais détailler comment préparer le Dockerfile, construire l'image, et lancer un conteneur avec cette image, y compris la redirection des ports.

1. Préparation du Dockerfile

Créez un nouveau répertoire pour votre projet Docker et entrez dedans :

**mkdir docker-ssh**

**cd docker-ssh**

Créez le Dockerfile avec le contenu suivant :

**nano dockerfile**

coller le code

**# Utiliser l'image de base Debian**

**FROM debian:stable-slim**

**# Installer SSH et net-tools (pour ifconfig, etc.)**

**RUN apt-get update && apt-get install -y openssh-server net-tools**

**# Configurer le mot de passe de l'utilisateur root**

**RUN echo 'root:root123' | chpasswd**

**# Configurer SSH pour autoriser l'accès root avec un mot de passe**

**RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd\_config**

**# Pas de PAM**

**RUN sed -i 's/UsePAM yes/UsePAM no/' /etc/ssh/sshd\_config**

**# Exposer un port non standard pour SSH, par exemple 2222**

**EXPOSE 2222**

**# Dossier pour les données de runtime SSH**

**RUN mkdir /var/run/sshd**

**# Commande pour démarrer le serveur SSH sur le port 2222**

**CMD ["/usr/sbin/sshd", "-D", "-p", "2222"]**

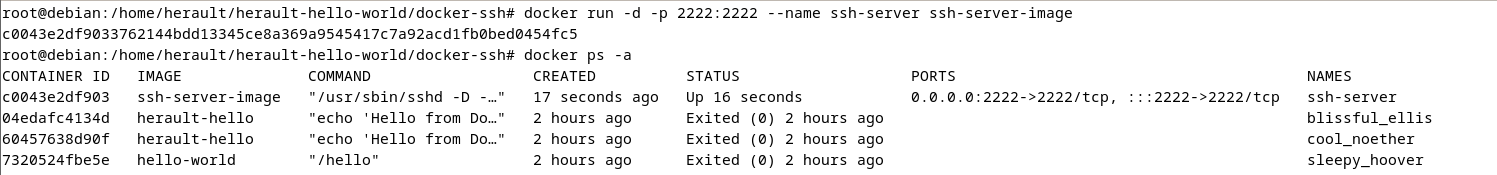
2. Construction de l'image Docker

Construisez l'image Docker depuis le répertoire contenant votre Dockerfile :

**docker build -t ssh-server-image .**

3. Lancer le conteneur Docker

Lancez un conteneur à partir de l'image que vous venez de construire, en redirigeant le port 2222 du conteneur vers un port de votre choix sur l'hôte (par exemple, 2222 également) :

**docker run -d -p 2222:2222 --name ssh-server ssh-server-image**

4. Connexion SSH au conteneur

Connectez-vous au serveur SSH du conteneur depuis votre machine hôte. Vous aurez besoin de l'adresse IP de votre machine hôte et du port que vous avez redirigé (2222 dans cet exemple) :

**ssh root@localhost -p 2222**

**job 5 :**

**Ouvrez votre fichier .bashrc avec un éditeur de texte, par exemple Nano :**

nano ~/.bashrc

**Ajoutez des alias pour les commandes Docker fréquemment utilisées. Voici quelques exemples que vous pourriez trouver utiles :**

# Pour lister tous les conteneurs actifs

alias dcps='docker ps'

# Pour lister tous les conteneurs, actifs ou non

alias dcpsa='docker ps -a'

# Pour lister toutes les images Docker

alias dcimages='docker images'

# Pour arrêter un conteneur Docker

alias dcstop='docker stop'

# Pour arrêter tous les conteneurs Docker

alias dcstopall='docker stop $(docker ps -a -q)'

# Pour supprimer un conteneur Docker

alias dcrm='docker rm'

# Pour supprimer tous les conteneurs Docker

alias dcrmall='docker rm $(docker ps -a -q)'

# Pour supprimer une image Docker

alias dcrmi='docker rmi'

# Pour supprimer toutes les images Docker

alias dcrmiall='docker rmi $(docker images -q)'

# Pour démarrer un conteneur Docker

alias dcstart='docker start'

# Pour redémarrer un conteneur Docker

alias dcrestart='docker restart'

# Pour voir les logs d'un conteneur Docker

alias dclogs='docker logs'

**Sauvegardez et fermez le fichier. Si vous utilisez Nano, vous pouvez le faire en appuyant sur Ctrl+X, puis en tapant Y pour confirmer la sauvegarde, et enfin Enter pour sortir.**

**Activez les alias en rechargeant le fichier .bashrc :**

source ~/.bashrc

**job 6** :

Utilisation de volumes entre deux conteneurs

Pour partager des volumes entre deux conteneurs, vous pouvez procéder comme suit :

Créer un volume Docker :

Pour commencer, vous devez créer un volume Docker que vous allez partager entre les conteneurs. Vous pouvez le faire avec la commande suivante :

**docker volume create mon\_volume**

Attacher le volume à des conteneurs :

Lorsque vous démarrez des conteneurs, vous pouvez monter le volume créé sur les conteneurs qui ont besoin d'accéder aux mêmes données. Par exemple, pour monter ce volume sur deux conteneurs différents :

**docker run -d -v mon\_volume:/data --name conteneur1 alpine**

**docker run -d -v mon\_volume:/data --name conteneur2 alpine**

Dans cet exemple, mon\_volume est monté sur le répertoire /data à l'intérieur de chaque conteneur. Tous les fichiers écrits dans /data dans conteneur1 seront accessibles sous /data dans conteneur2, et vice-versa.

Gestion des volumes

Lister les volumes

Pour voir tous les volumes sur votre système :

**docker volume ls**

Inspecter un volume

Pour obtenir plus d'informations sur un volume spécifique :

**docker volume inspect mon\_volume**

Supprimer un volume

Pour supprimer un volume (assurez-vous qu'aucun conteneur actif ne l'utilise) :

**docker volume rm mon\_volume**

Nettoyer les volumes inutilisés

Pour supprimer tous les volumes qui ne sont attachés à aucun conteneur :

**docker volume prune**

**JOB 7 :**

**Nous allons commencer par installer docker-compose**

**sudo curl -L "https://github.com/docker/compose/releases/download/v2.11.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose**

**-sudo install docker-compose**

**Étape 1 : Préparez votre fichier HTML localement**

**Créez un fichier index.html localement dans un dossier webdata :**

**html**

**Copier le code**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>My Web Page</title>**

**</head>**

**<body>**

**<h1>Nom: HERAULT BEA</h1>**

**<h1>Prénom: GERALD </h1>**

**</body>**

**</html>**

**Étape 2 : Créez un fichier de configuration Nginx localement**

**Créez un fichier nginx.conf dans le même répertoire que votre docker-compose.yml et ajoutez-y le contenu suivant :**

**nginx**

**Copier le code**

**server {**

**listen 80;**

**server\_name 192.168.107.129;**

**location / {**

**root /usr/share/nginx/html;**

**index index.html index.htm;**

**}**

**}**

**Étape 3 : Modifiez votre fichier docker-compose.yml**

**Assurez-vous que votre fichier docker-compose.yml est configuré pour monter les volumes correctement :**

**yaml**

**Copier le code**

**version: '3.8'**

**services:**

**nginx:**

**image: nginx:latest**

**ports:**

**- "8080:80"**

**volumes:**

**- ./webdata:/usr/share/nginx/html**

**- ./nginx.conf:/etc/nginx/conf.d/default.conf**

**networks:**

**- webnet**

**ftp:**

**image: stilliard/pure-ftpd:hardened**

**ports:**

**- "21:21"**

**environment:**

**PUBLICHOST: "localhost"**

**FTP\_USER\_NAME: "bea"**

**FTP\_USER\_PASS: "root"**

**FTP\_USER\_HOME: "/home/herault/"**

**volumes:**

**- webdata:/home/herault/**

**networks:**

**- webnet**

**volumes:**

**webdata:**

**networks:**

**webnet:**

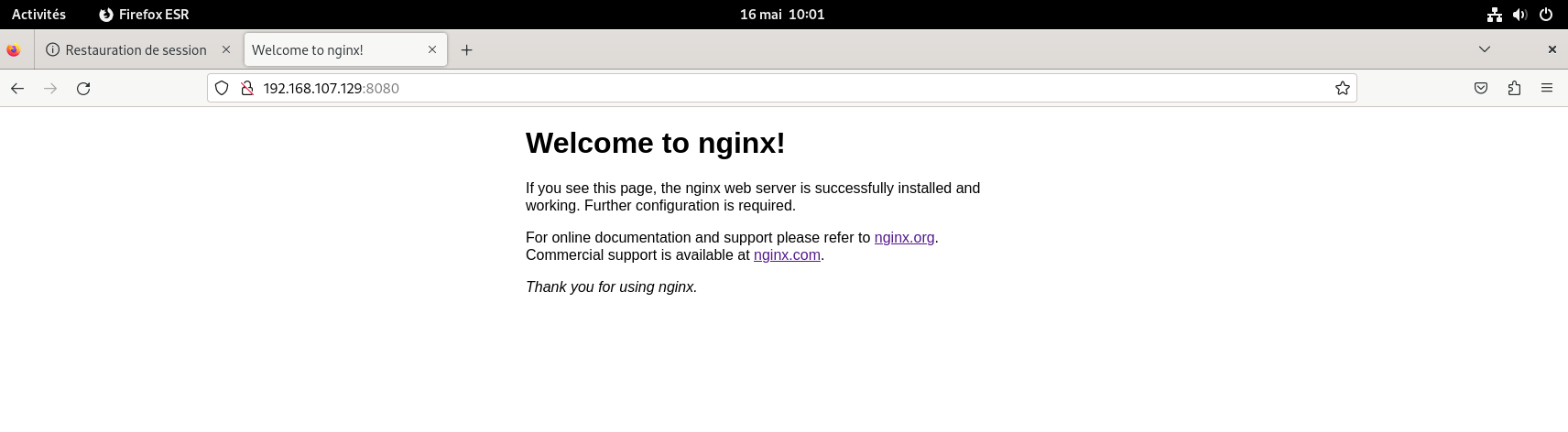
**Étape 4 : Redémarrez les conteneurs Docker Compose**

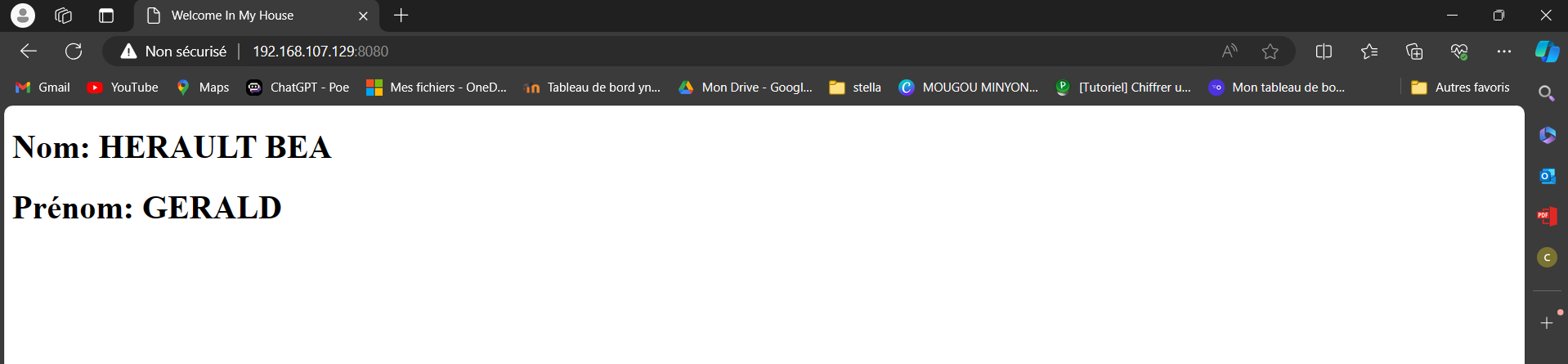
**Arrêtez les conteneurs en cours d'exécution :**

**docker-compose down**

**Démarrez les conteneurs :**

**docker-compose up -d**





JOB 8 :

* Dans /home/herault/herault-hello-world/job8 que nous avons cree notre DockerFile ; notre index.html et notre nginx.conf

**DockerFifle :**

# Utilisez l'image de base nginx

FROM nginx

# Copiez le fichier de configuration personnalisé dans le conteneur

COPY index.html /usr/share/nginx/html

# Exposez le port 80 pour les connexions entrantes

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

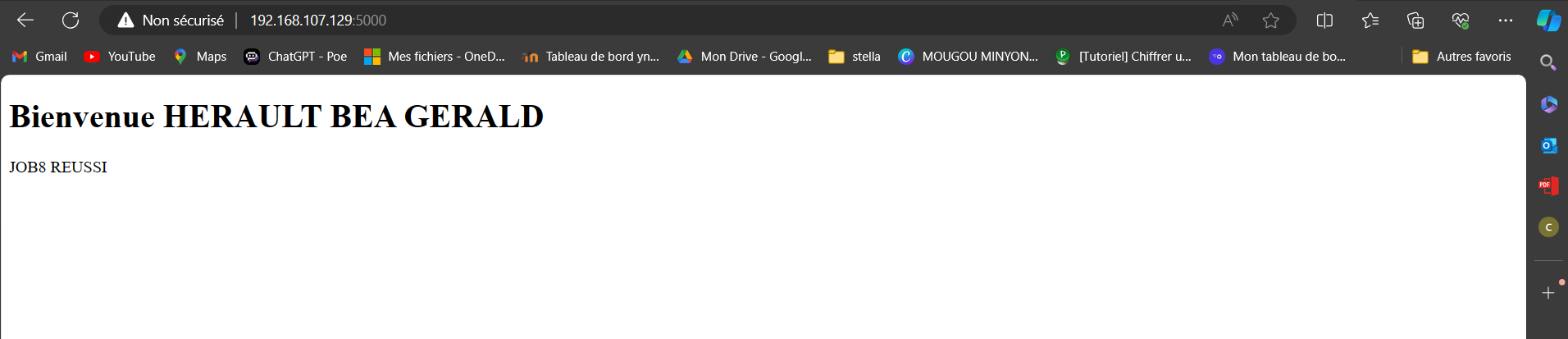
**Apres ca nous avons cree une image :**

sudo docker build -t job8-nginx .

**puis nous allons executer notre contenaire :**

sudo docker run -d -p 5000:80 job8-nginx

Apres ca nous allons sur notre navigateur pour tapez : <http://192.168.107.129>: 5000



JOB 9 :

**sudo docker run -d -p 5000:5000 --name ui registry:latest**

**docker ps**

**sudo curl -X GET** [**http://192.168.107.129:5000/v2/\_catalog**](http://192.168.107.129:5000/v2/_catalog)

**sudo docker ps**

**sudo docker stop ui**

**sudo docker rm ui**

**sudo docker run -d -p 5000:5000 --restart=always --name registry -v $(pwd)/docker-registry:/var/lib/registry registry:latest**

**sudo docker tag nginx 192.168.107.129/nginx**

**sudo docker images**

**sudo nano /etc/docker/daemon.json**

**sudo service docker stop**

**sudo service docker start**

**sudo docker push 192.168.107.129/nginx**

**sudo docker tag nginx 192.168.107.129:5000/nginx**

**sudo docker push 192.168.107.129:5000/nginx**

**sudo docker rmi -f 192.168.107.129:5000/nginx**

**sudo docker pull 192.168.107.129:5000/nginx**

**exemple :**

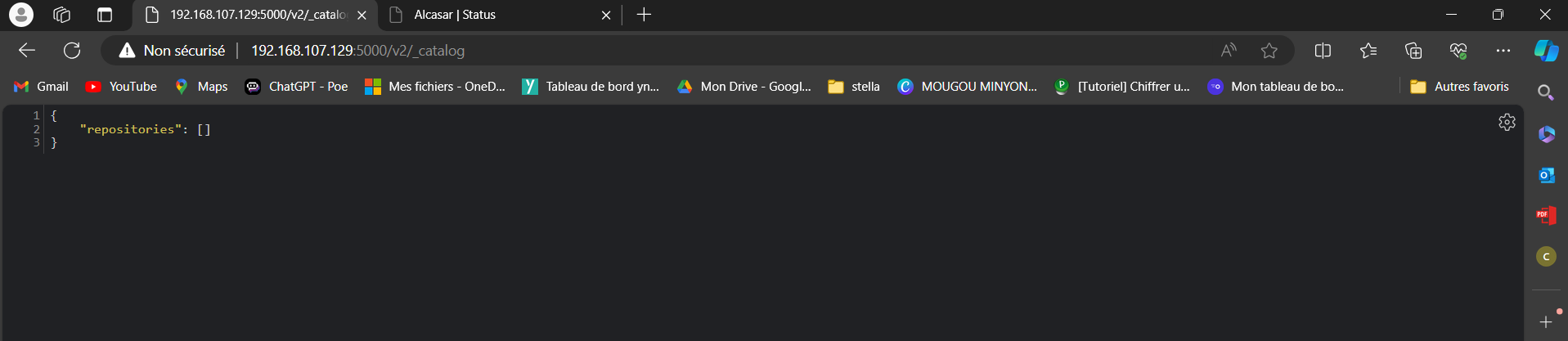
**docker images**

**-sudo docker tag herault-hello 192.168.107.129:5000/herault-hello #### permet de renomer une image apres faire le docker images pour voir si elle est bien enregistré**

**-sudo docker push 192.168.107.129:5000/herault-hello ### permet de l’enregistre sur le registre local qu’on a créé**

Puis faire ça sur notre navigateur http://[192.168.107.129:5000/v2/\_catalog](http://192.168.107.129:5000/v2/_catalog)

**root@debian:/home/herault/herault-hello-world/JOB9/docker-registry/docker/registry/v2/repositories#**



**JOB 10 :**

Touch purge-docker.sh

#!/bin/bash    
    
docker rmi -f \* ; docker rm -f \* ; docker volume rm -f \*     
    
apt-get autoremove **docker** --purge    
    
docker=$(find / -name "**docker**" 2>/dev/null)    
    
echo "$docker" | xargs rm -rf    
    
rm -rf /var/lib/docker    
    
    
for pkg in [[docker.io](http://docker.io)](<http://docker.io>) docker-doc docker-compose docker-compose-v2 podman-docker containerd runc -y ; do sudo apt-get remove $pkg -y; done

JOB 11

Installation de Portainer

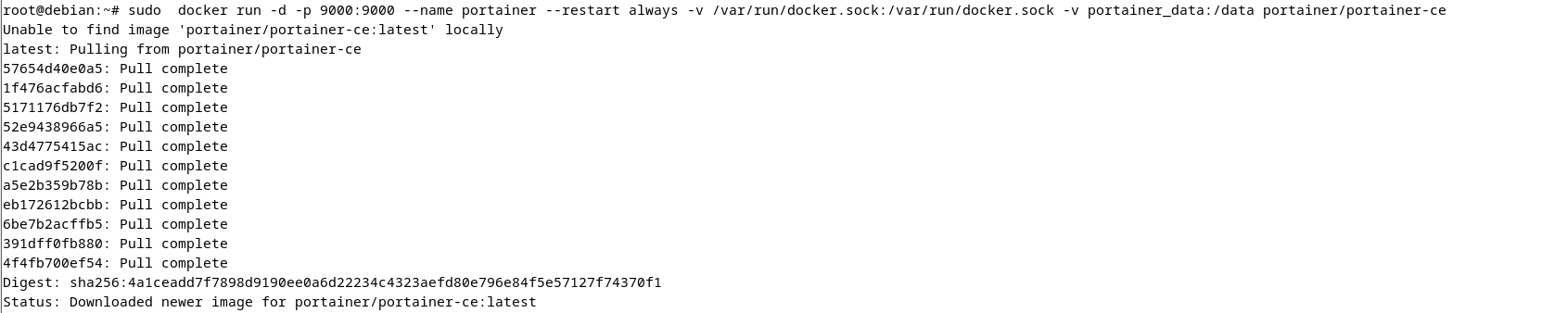
Il existe plusieurs façons d’installer Portainer, mais nous allons couvrir les deux méthodes les plus courantes : en utilisant une commande Docker run et en utilisant Docker Compose.

Commande Docker Run

Pour installer Portainer à l’aide d’une commande Docker run, exécutez la commande suivante :

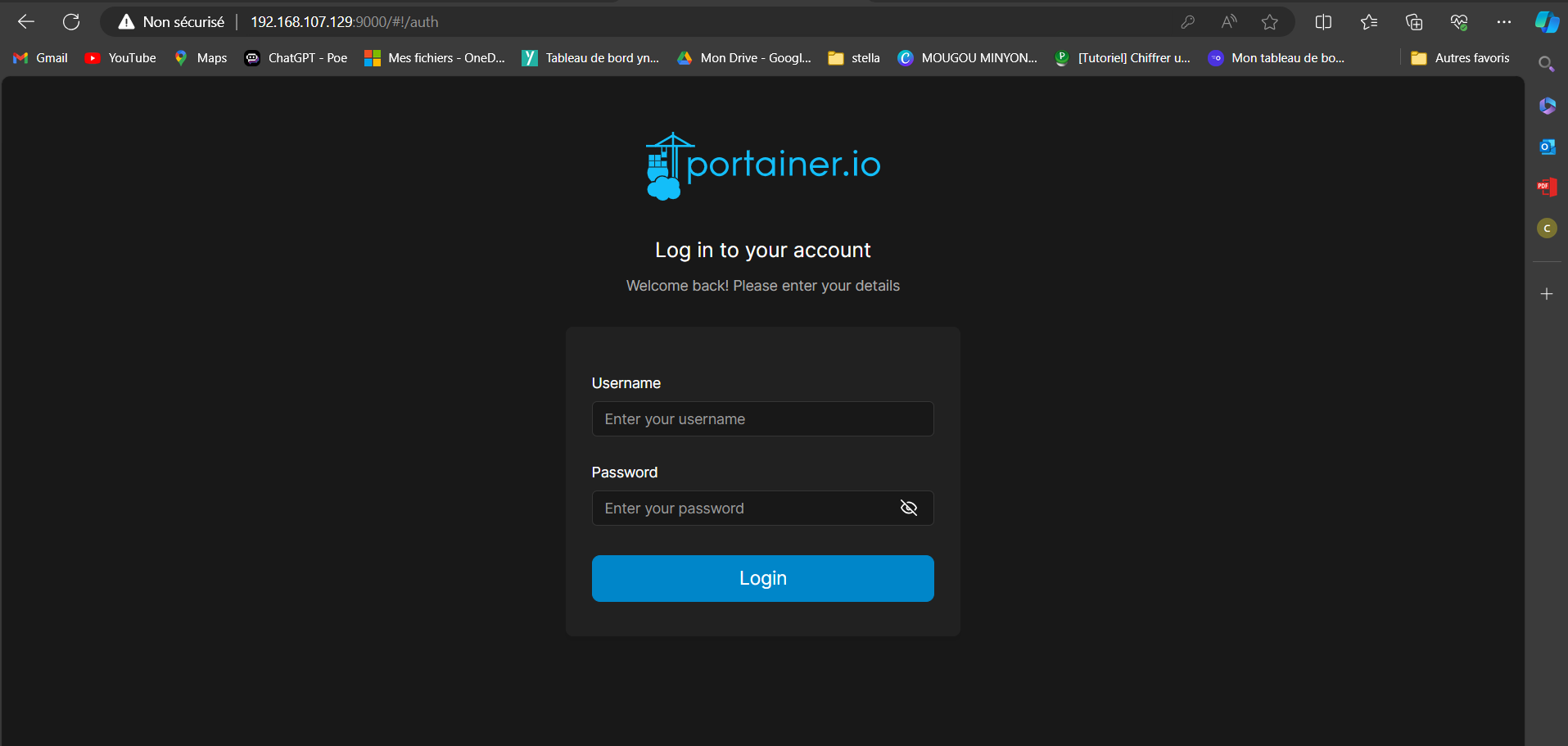
$ docker run -d -p 9000:9000 --name portainer --restart always -v /var/run/docker.sock:/var/run/docker.sock -v portainer\_data:/data portainer/portainer-ce

Voici ce que fait cette commande :

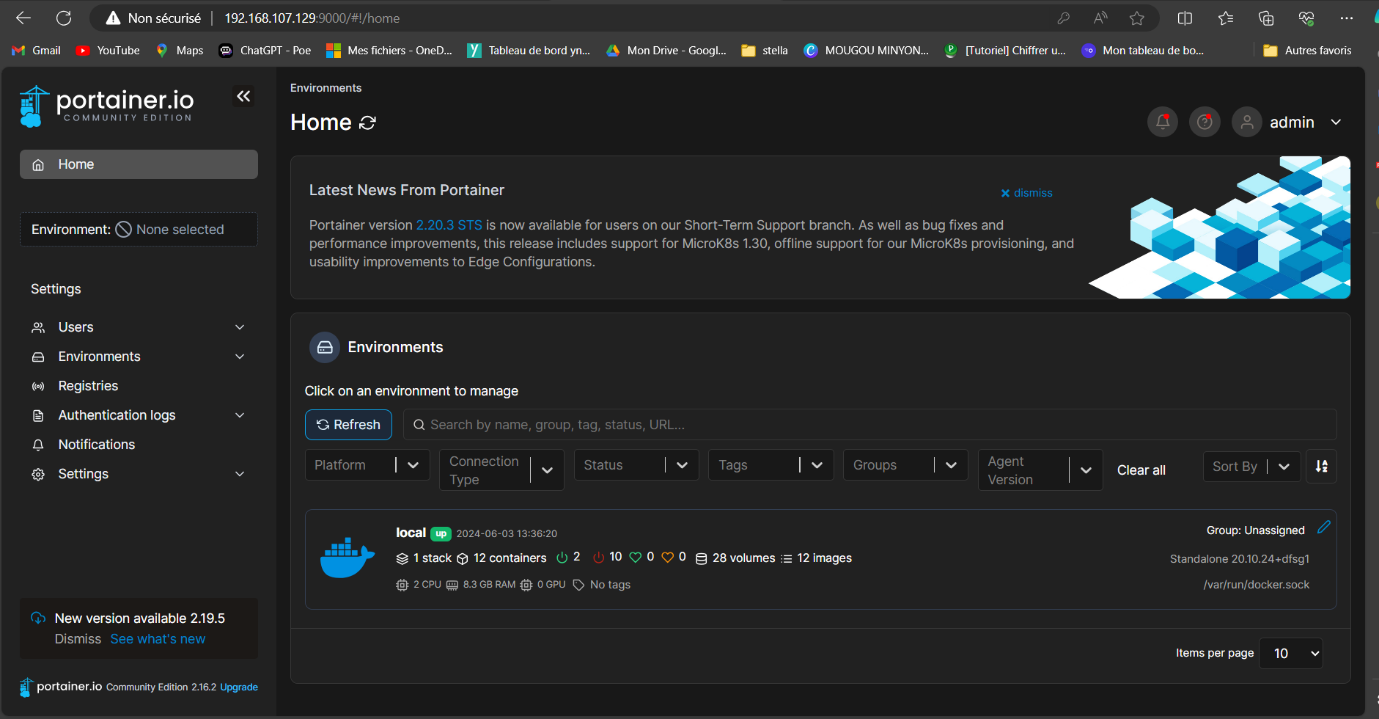
* -d : Exécute le conteneur en mode détaché.
* -p 9000:9000 : Mappe le port 9000 sur votre hôte au port 9000 dans le conteneur.
* --name portainer : Nomme le conteneur “portainer”.
* --restart always : Configure le conteneur pour redémarrer automatiquement.
* -v /var/run/docker.sock:/var/run/docker.sock : Donne à Portainer l’accès au socket Docker.
* -v portainer\_data:/data : Persiste les données de Portainer dans un volume nommé “portainer\_data”.
* portainer/portainer-ce : Spécifie l’image Portainer Community Edition.

Puis allons sur notre navigateur pour avoir accès a notre interface graphique

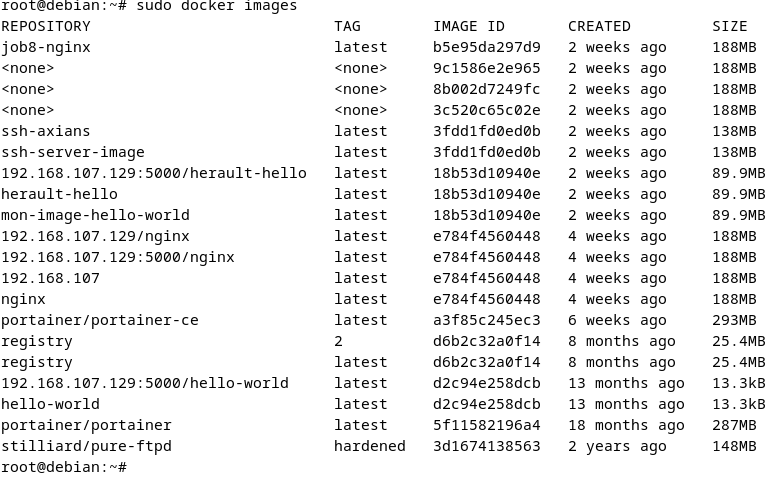
<http://192.168.107.129:9000>



Puis on met nos identifiants user : admin & password :

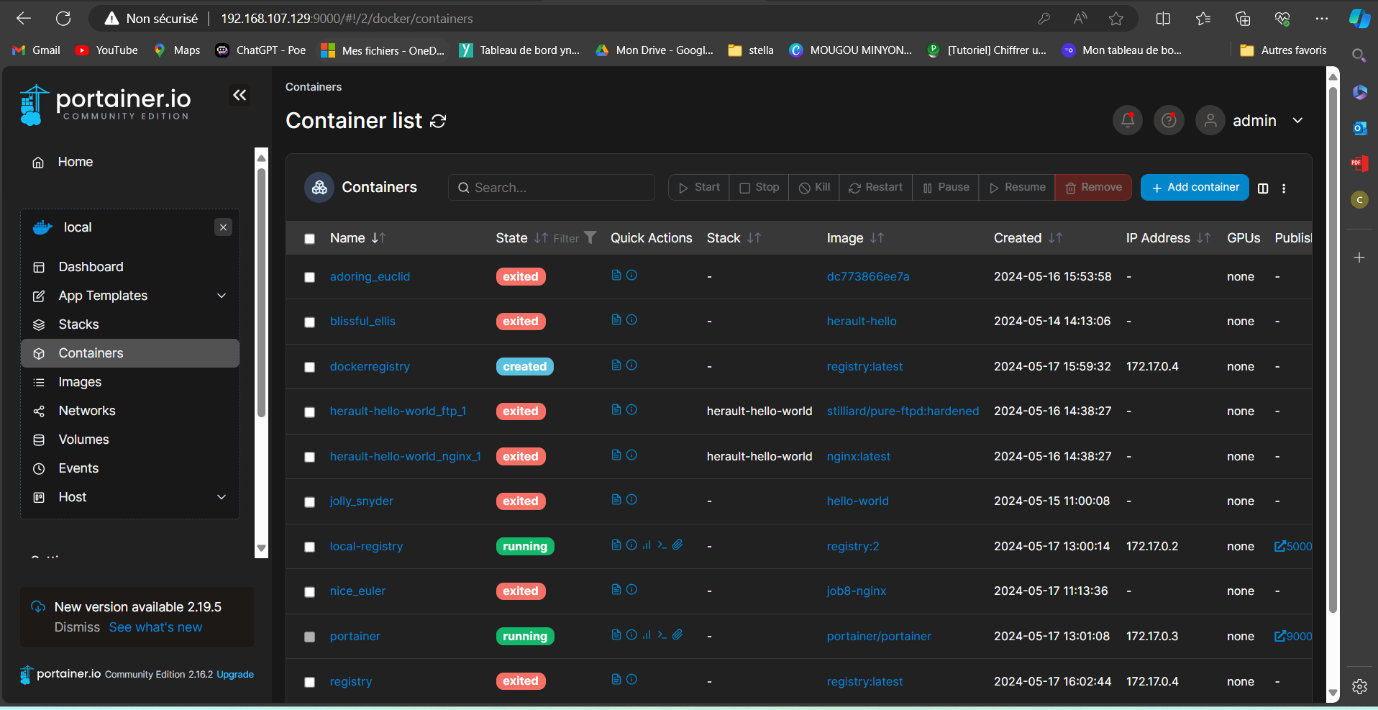


Voici mes différentes images sur docker

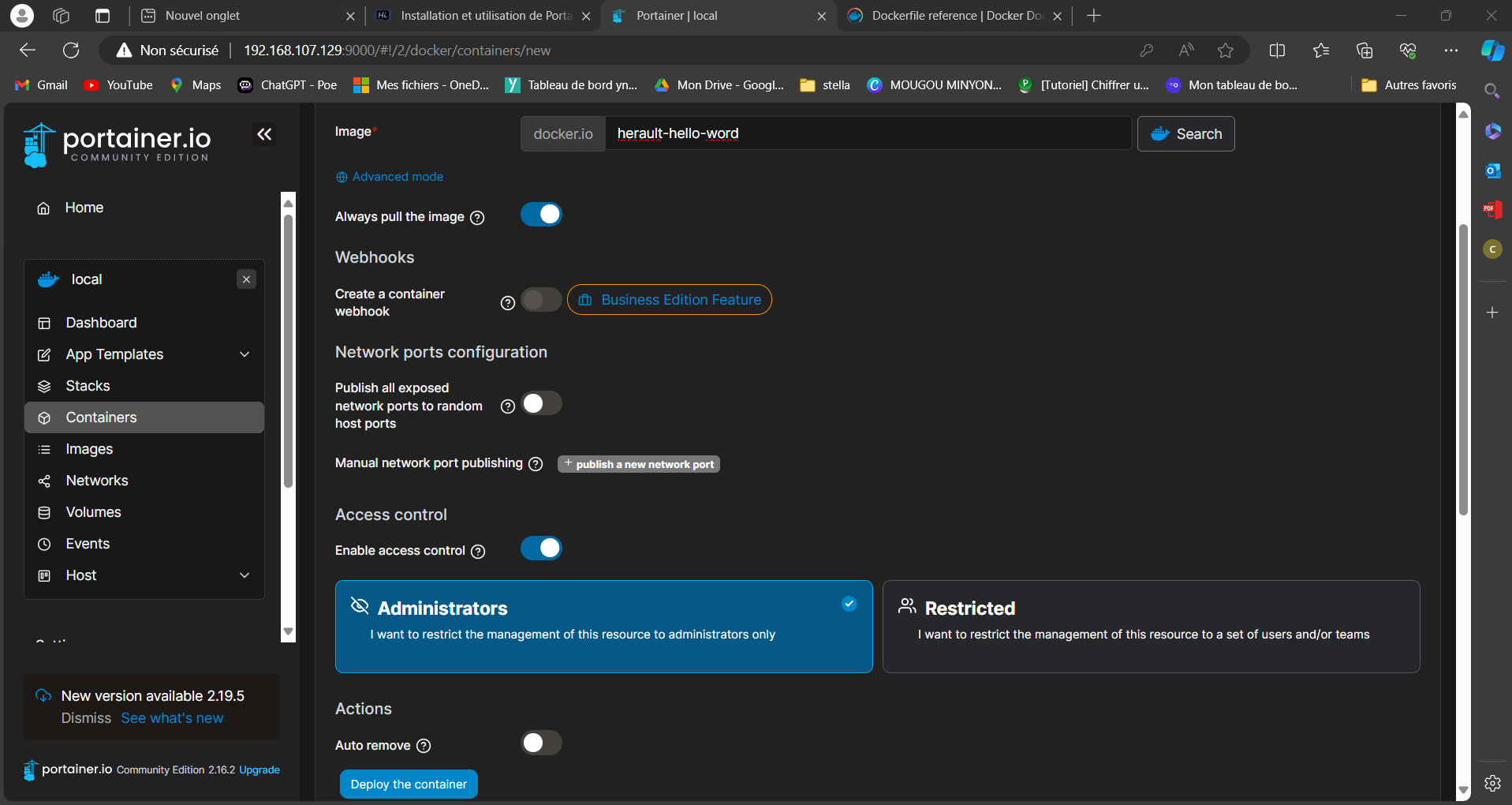


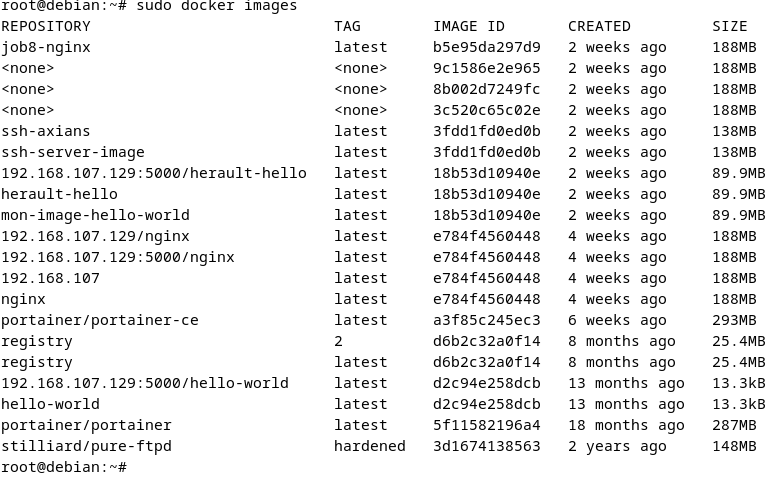
Nous allons donc télécharger le conteneur herault-hello sur le registry officiel de Docker:

On clique sur "Add container":



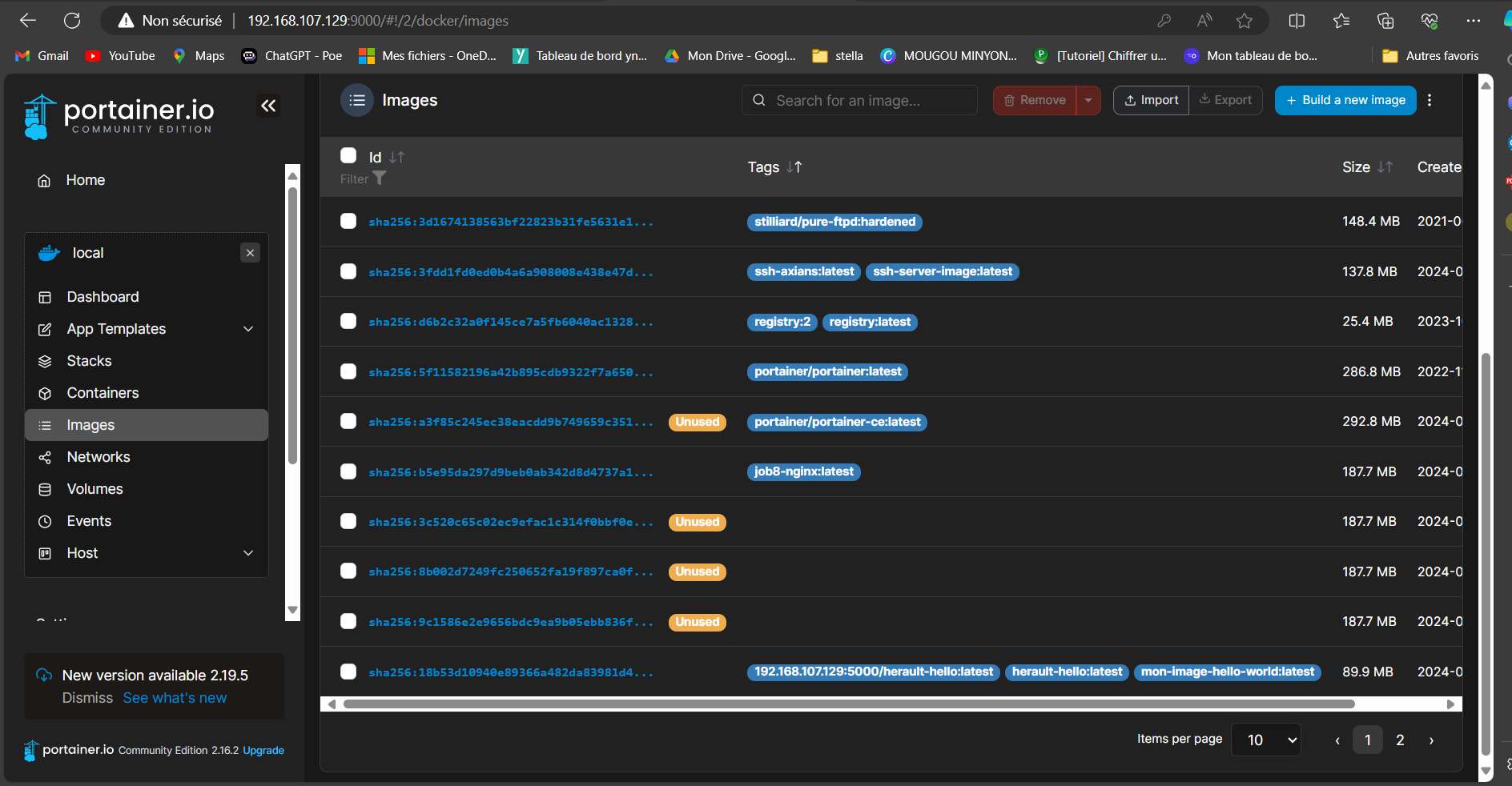
On entre ensuite le nom du conteneur a importer, puis on clique sur "Deploy the container":

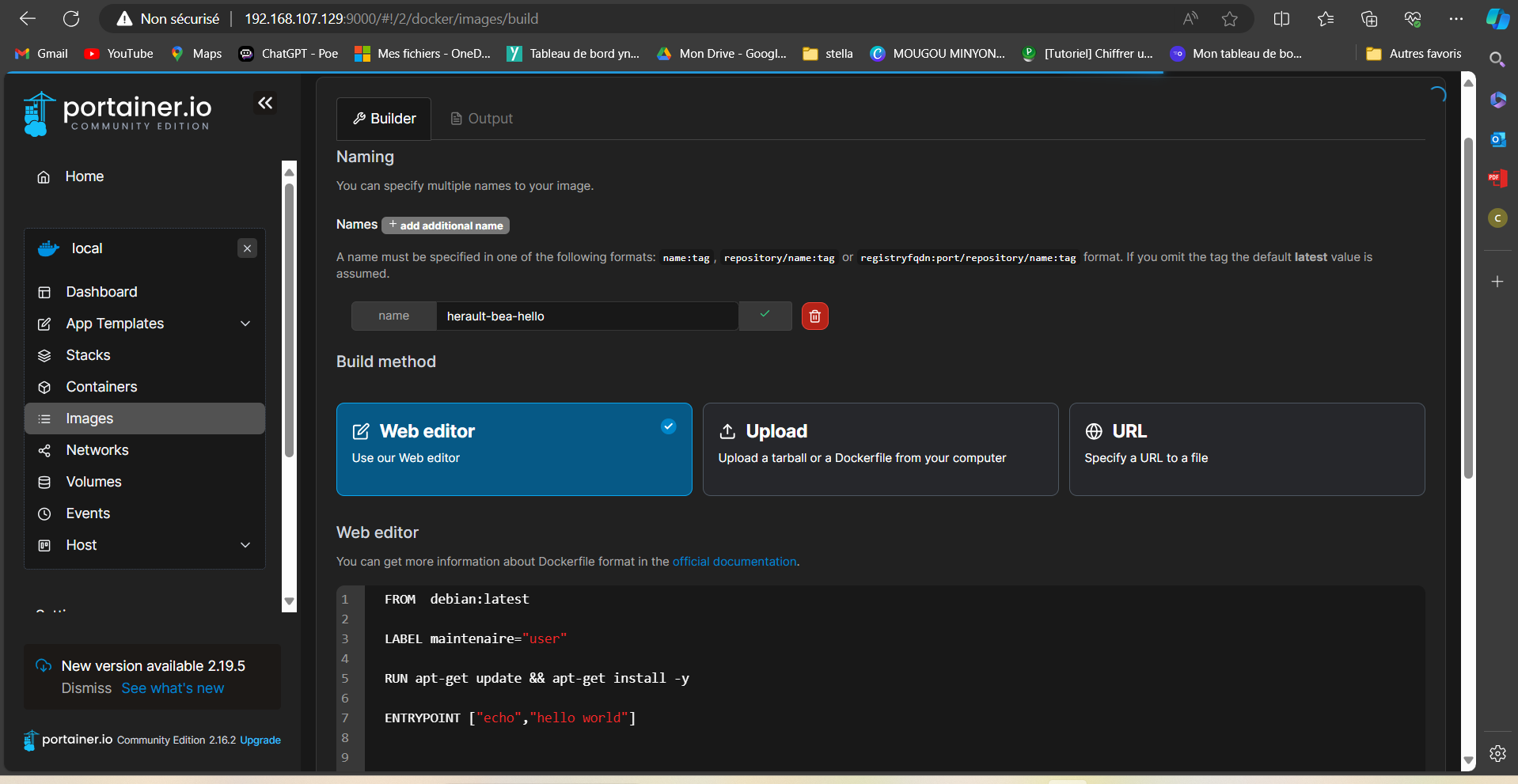


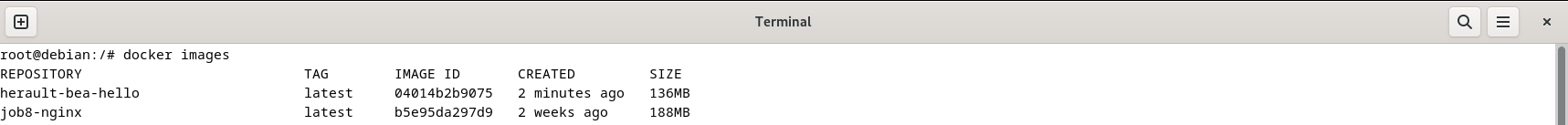


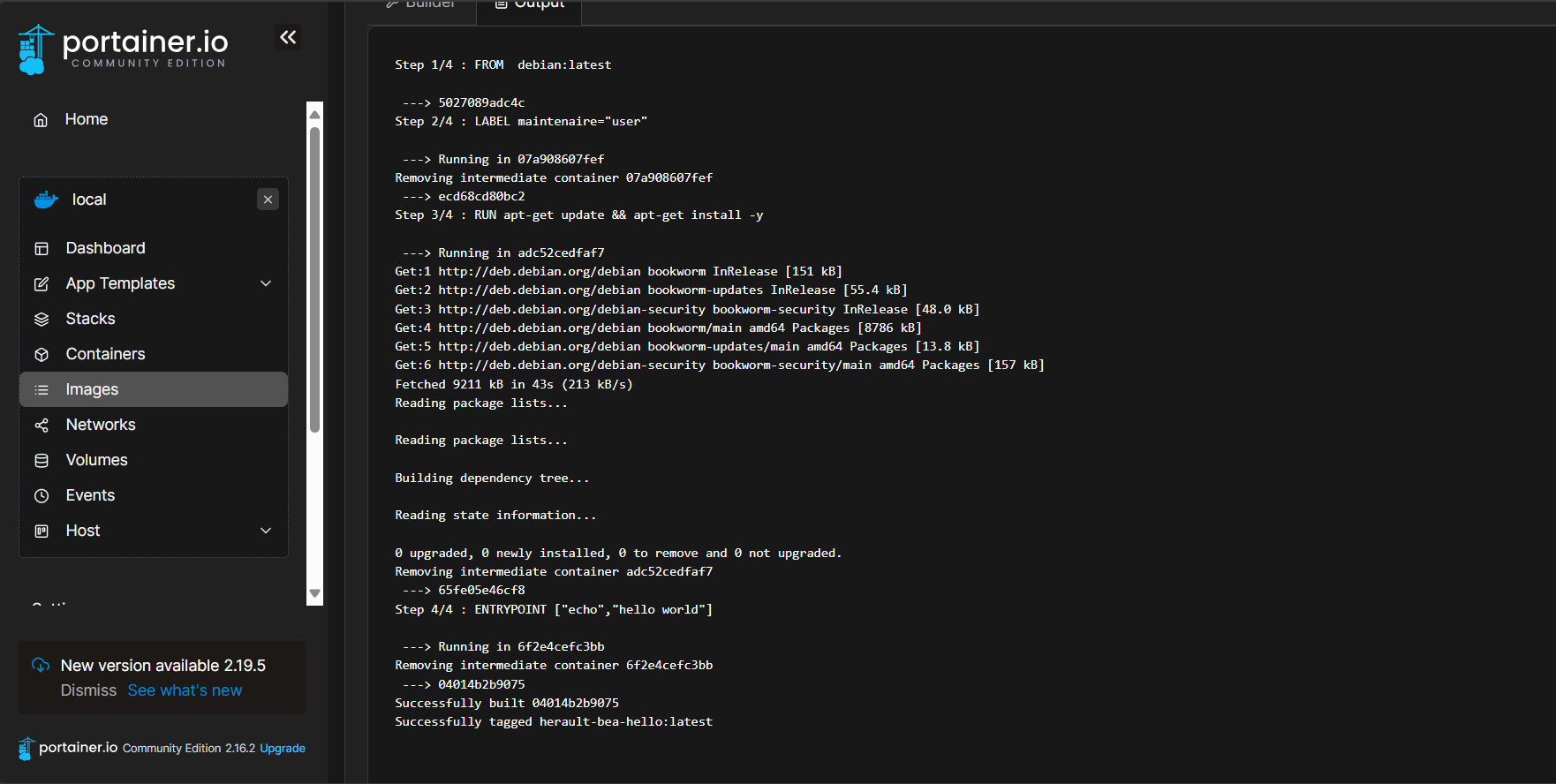
Nous devons maintenant déployer nous même un conteneur qui nous affichera "hello world". Pour se faire:

Nous cliquons sur "Images", puis sur "Build a new image":



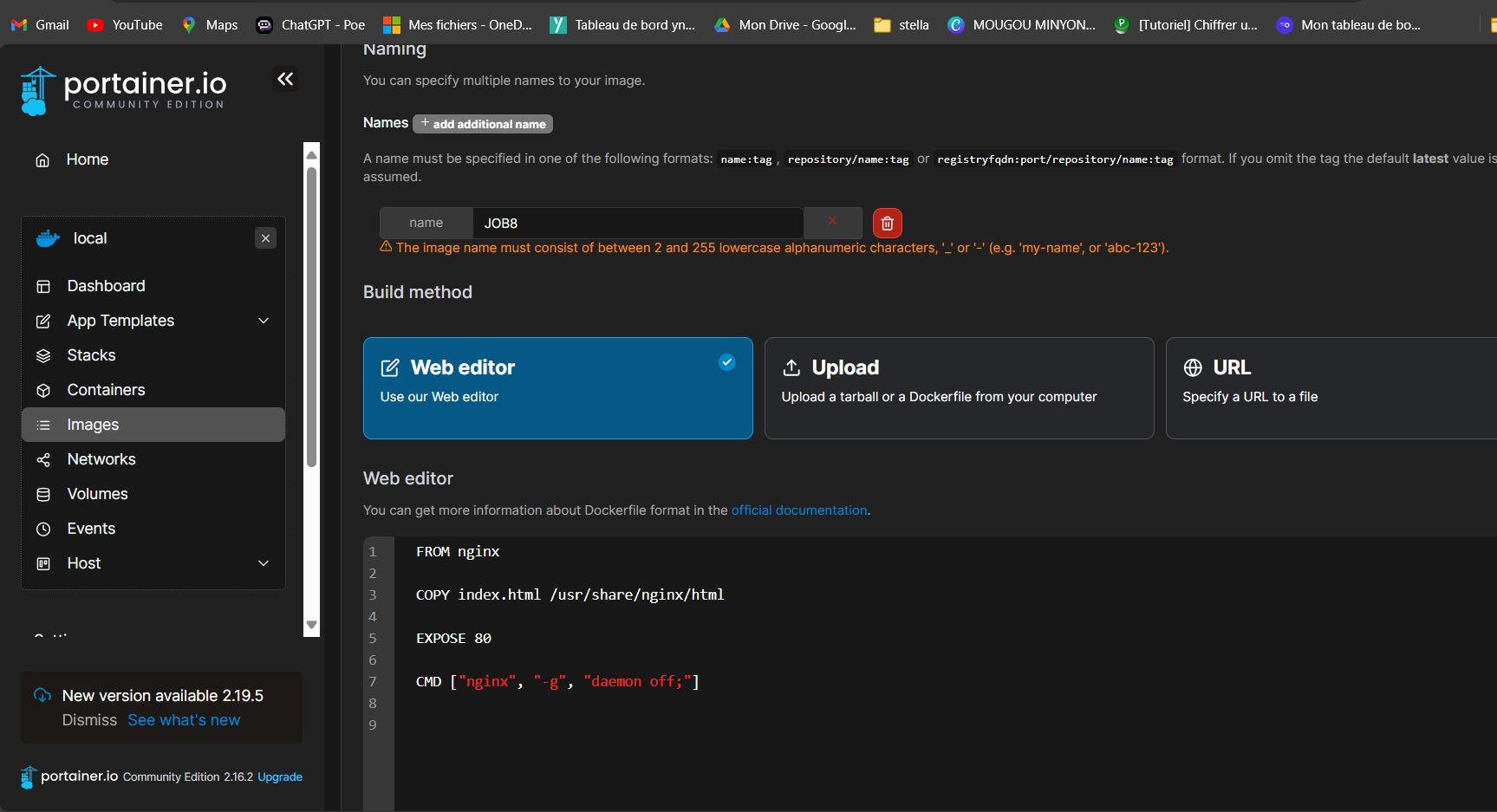
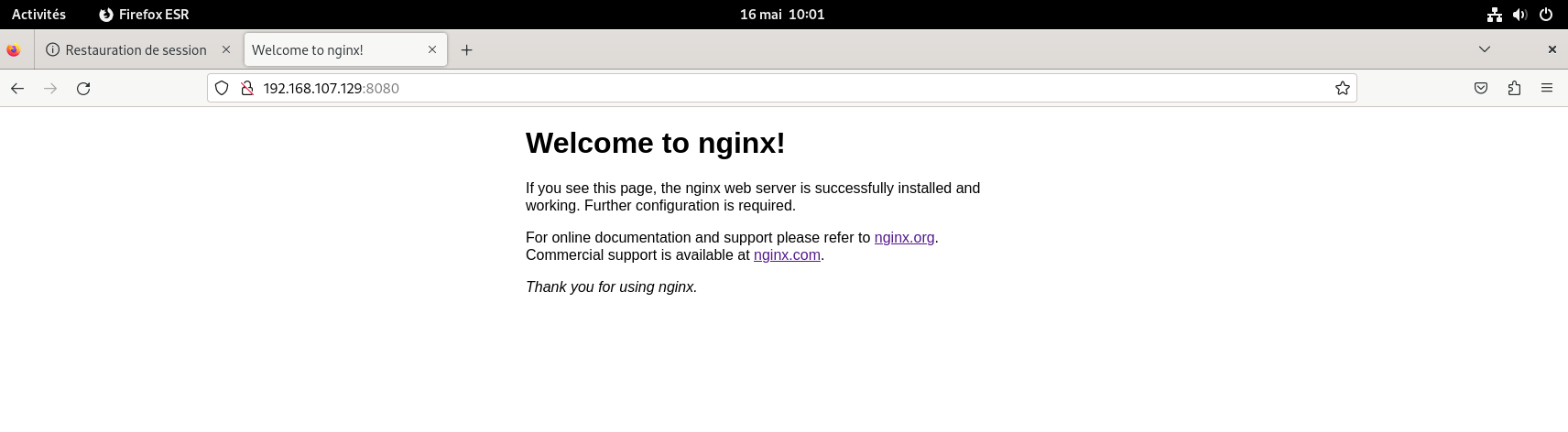


On part maintenant sur docker pour voir si elle a été bien cree en faisant docker image depuis notre debian nous constactons q u’elle à été bien crée





Et nous voyons que nous pouvons nous connecter en ssh sans problème par ce que nous avons configurer le mode ssh sinon ça ne peut pas fonctionner

Maintenat nous allons cree depuis notre portainer une image nginx pui se connecter

