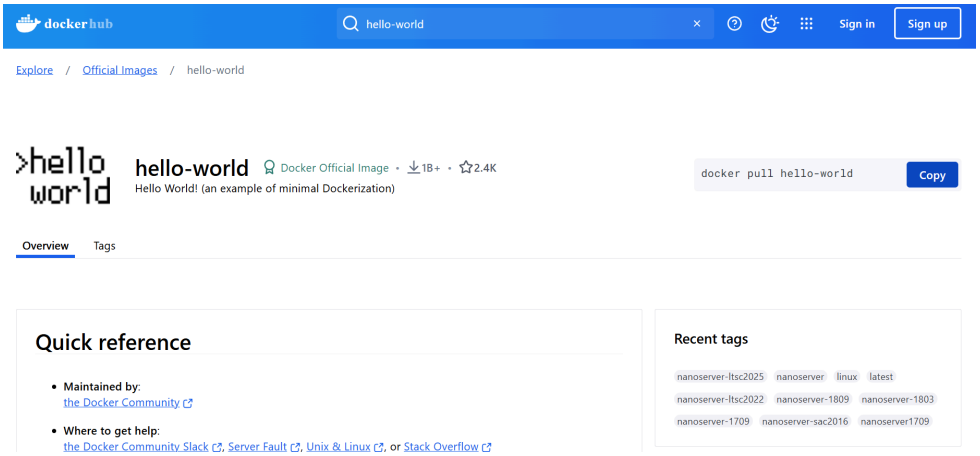
TP 2 : Prise en Main Docker

I. Premier exemple

Pour s'assurer que l'installation s'est bien déroulée, on va exécuter un premier exemple : Rendez-vous sur : ***https://hub.docker.com/*** pour chercher une image existante sur Docker Hub. soit par exemple : hello-world.



Sur la ligne de commande sur Docker Desktop, les commandes suivantes vous permettront de télécharger et d'exécuter l'image hello-world : ***docker pull hello-world*** , ***docker run hello-world***.

II. Manipulation des commandes de bases

a) Placez-vous sur Docker Hub.
b) Choisissez une image (exemple : redis).
c) Télécharger une version de cette image.
d) Lister l'ensemble d'images que vous avez sur votre machine.
e) Créer un conteneur à partir de cette image : docker run redis
f) Lister les conteneurs en cours d'exécution.
g) Exécuter le conteneur en mode détaché.
h) Stopper le conteneur.
i) Démarrer le conteneur.
j) Afficher l'historique de tous les conteneurs existants sur votre machine.
k) Télécharger une deuxième version de l'image que vous avez choisie différente de la première version.
l) Exécutez-la parallèlement avec la première
m) Qu'est-ce que vous remarquez?
n) Liez le port du conteneur de cette image à un port de la machine.

III. Créer une image à partir d'un conteneur

Dans cet exercice, nous allons créer une image Docker personnalisée à partir d'un conteneur sous Ubuntu.

a. Lancer la commande qui permet d'importer une image ubuntu depuis le Docker HUB et démarrer un shell bash en mode interactif dans le conteneur.

b. Personnaliser le conteneur en installant de nouveaux paquets à l'aide de la commande suivante : apt-get update / apt-get install -y figlet .
c. Récupérer l'ID du conteneur créé.
d. Créer une image locale à l'aide de la commande suivante :

docker container commit ID_de_vote_conteneur

Cette commande crée une nouvelle image Docker à partir d'un conteneur existant. Elle permet de capturer l'état actuel du conteneur, y compris toutes les modifications apportées (comme les paquets installés, les fichiers créés, etc.), et de le convertir en une image réutilisable.

e. Vérifier la création de l'image. Vous devez normalement voir apparaître dans la liste de vos images, l'image Ubuntu de votre conteneur ainsi que l'image que vous venez de créer. Cette dernière, comme vous pourrez le constater, ne possède ni de repository, ni de tag.
f. Attribution d'un tag à votre image à l'aide de la commande suivante :

docker image tag <ID_de_votre_image> <nom_de_l'image>

Vous pouvez ensuite vérifier que l'attribution du tag à bien été réalisée à l'aide de la commande docker image.

g. vous allez maintenant essayer de faire fonctionner un conteneur comportant l'image que vous venez de créer.

IV. Création d'une image à l'aide de Dockerfile

Pour générer une image, nous utilisons un fichier appelé Dockerfile. Ce fichier est particulièrement utile pour gérer les modifications, notamment lorsque les images deviennent volumineuses et complexes. Par exemple, si une nouvelle version de figlet est disponible, il ne serait pas nécessaire de reconstruire l'image entièrement. Le Dockerfile contient les commandes apt-get nécessaires à l'installation de figlet, permettant ainsi à quiconque l'utilisant de recréer facilement l'image en suivant ces instructions.

Dans cette partie, nous allons créer l'image d'une application Node.js, affichant un message "Hello world". Ne vous inquiétez pas si vous n'êtes pas familier avec Node.js : Docker (et cet exercice) ne nécessite pas une connaissance approfondie de ce sujet.

ATTENTION ! Si vous êtes encore dans le conteneur Ubuntu que vous avez lancé, vous devez le quitter pour revenir à l'invite de commande de base de Docker. Pour cela, entrez la commande exit.

a. Créer un fichier appelé index.js où on va récupérer le hostname et on l'affiche. Dans ce fichier, écrivez les éléments suivants :

```
var os = require("os") ;
var hostname = os.hostname() ; console.log(« hello from » + hostname) ;
```

b. Pour **dockeriser** cette application, nous allons créer un fichier **Dockerfile** en utilisant **Alpine Linux** comme image de base. Ce fichier inclura les commandes nécessaires pour :
   - ✔ Ajouter **Node.js**
   - ✔ Copier le code source dans le conteneur
   - ✔ Définir la commande par défaut exécutée lors de la création du conteneur
c. Construire une image nommée hello:v0.1 à partir du fichier Dockerfile que vous avez créé.
d. Pour vous assurer que votre application fonctionne correctement, vérifier en démarrant le conteneur de votre application. Vous devriez obtenir en sortie quelque chose de semblable à ce qui suit (excepté l'ID qui sera différente) : hello from 92d79b6de29f.
e. Qu'est-ce qui vient de se passer ?