

SOMMAIRE

I.	Introduction : Qu'est-ce que Docker ?	2
	Prérequis	3
II.	Installation de Docker	3
III.	Vérification du bon fonctionnement de docker	5
VI.	Premier pas avec docker	5
	1.1. Interroger le "docker registry" en CLI	5
	1.2. Télécharger une image.....	6
	1.3. Connaître les images disponibles sur le système.....	6
	1.4. Supprimer une image	6
	2.1. Lancer un conteneur	6

I. Introduction : Qu'est-ce que Docker ?

Docker est une plateforme **open source** qui automatise le **déploiement, la gestion et l'exécution d'applications dans des conteneurs légers**. Un conteneur est une unité logicielle qui encapsule une application et ses dépendances, garantissant ainsi sa portabilité et son exécution cohérente quel que soit l'environnement.

Docker est une **technologie de conteneurisation** reposant sur le noyau Linux et ses fonctionnalités de virtualisation par conteneurs (LXC pour Linux Containers), notamment :

- Le composant *cgroups* pour contrôler et limiter l'utilisation des ressources pour un processus ou un groupe de processus (utilisation de la RAM, CPU entre autres) associé au système d'initialisation *systemd* qui permet de définir l'espace utilisateur et de gérer les processus associés ;
- Les *espaces de noms* ou *namespaces* qui permettent de créer des environnements sécurisés de manière à isoler les conteneurs et empêcher par exemple qu'un groupe puisse « voir » les ressources des autres groupes.

Docker offre des outils pour utiliser ces fonctionnalités de manière simplifiée pour permettre, entre autres :

- la duplication et la suppression des conteneurs ;
- l'accessibilité des conteneurs à travers la gestion des API et CLI ;
- la migration (à froid ou à chaud) de conteneurs.

Un conteneur Docker se construit à partir d'une image. Une image Docker est un package léger, autonome et exécutable d'un logiciel qui inclut tout ce qui est nécessaire pour l'exécuter : code de l'application, environnement d'exécution (runtime), outils système et librairies, etc.

De nombreuses images comme *Nextcloud*, *Debian* sont disponibles sur :

- le registre officiel (appelé *hub*): <https://hub.docker.com> ;
- de nombreux dépôts initiés par de « simples » utilisateurs.

Il est bien sûr possible de proposer des images, d'en modifier d'autres et de déposer la modification sur le dépôt officiel.

Limites des conteneurs Docker

Les conteneurs Docker souffrent tout de même de quelques limites :

- Sécurité : les conteneurs peuvent être plus vulnérables, car ils partagent un noyau et des composants systèmes et leur fonctionnement exige déjà un niveau d'autorisation élevé (généralement l'accès root dans les environnements Linux) : si toute l'architecture est basée sur Docker et si le système hôte est attaqué tous les services seront « accessibles

» et exposés plus facilement et rapidement aux attaques. À noter que les plateformes de conteneurs évoluent dans le sens d'une plus grande sécurisation en matière d'isolement et de séparation des droits des OS ;

- Complexité : la multiplication facile des conteneurs rend possible une consommation d'une grande quantité de ressources sans s'en rendre compte. Par ailleurs, la gestion d'un grand nombre de conteneurs peut devenir complexe. L'orchestration avec des outils tels que Kubernetes est souvent nécessaire pour simplifier le déploiement et la gestion en production ;
- Non idéal pour toutes les applications : certaines applications avec des exigences spécifiques peuvent ne pas être adaptées aux conteneurs. Il est important de comprendre les besoins de l'application et du contexte avant de choisir Docker.

Prérequis

Machine Debian

Nom : deb-docker

Réseau : VMBR0

Stockage : 32Go

RAM : 4Go

Processeur : 2 | cœurs : 2

II. Installation de Docker

Lien officiel de l'installation : <https://docs.docker.com/engine/install/debian/>

Se positionner en root.

Installation des paquets nécessaires à l'utilisation du dépôt docker en https

```
root@deb-docker:/home/harani# Apt update
```

```
root@deb-docker:/home/harani# Apt install ca-certificates curl gnupg
```

Importation de la clé du dépôt docker :

```
root@deb-docker:/home/harani# install -m 0755 -d /etc/apt/keyrings
```

```
root@deb-docker:/home/harani#
```

```
curl -fsSL https://download.docker.com/linux/debian/gpg -o  
/etc/apt/keyrings/docker.asc
```

```
root@deb-docker:/home/harani# chmod a+r /etc/apt/keyrings/docker.asc
```

Intégration du dépôt docker dans le fichier source.list et MAJ des dépôts :

```
root@deb-docker:/home/harani# echo \
```

```
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]  
https://download.docker.com/linux/debian \
```

```
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
```

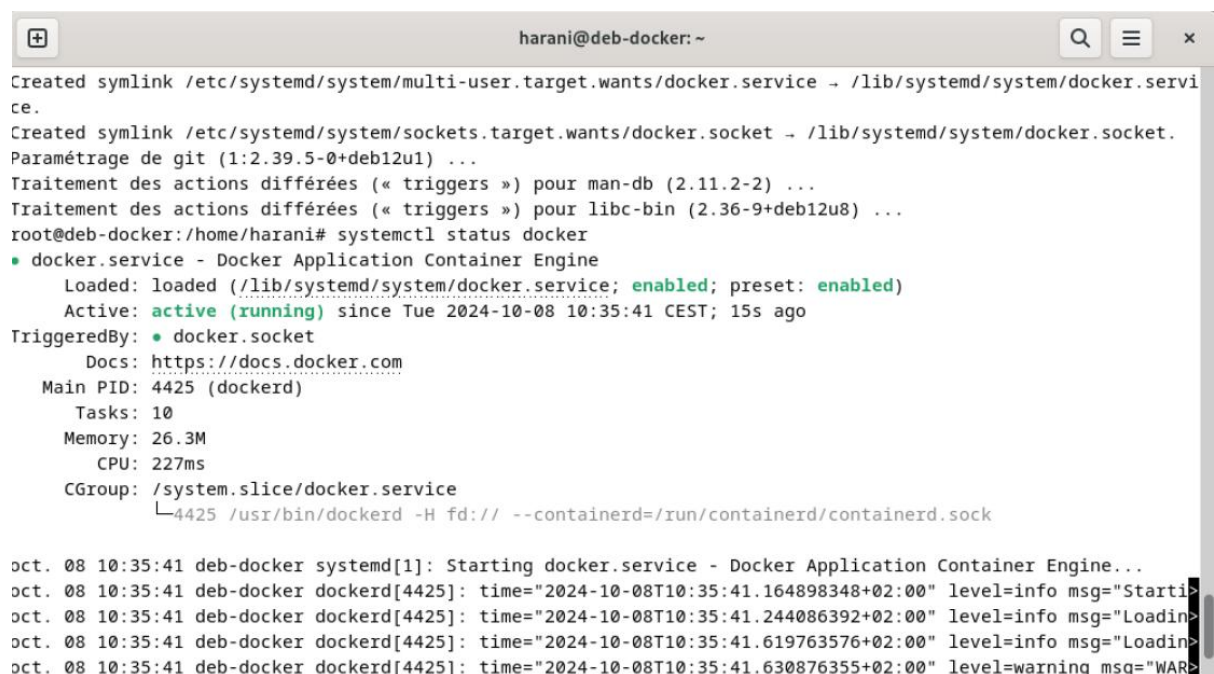
```
tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
root@deb-docker:/home/harani# apt-get update
```

Installation de docker

```
root@deb-docker:/home/harani# apt-get install docker-ce docker-ce-cli containerd.io  
docker-buildx-plugin docker-compose-plugin
```

```
root@deb-docker:/home/harani# Systemctl status docker
```



```
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.  
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.  
Paramétrage de git (1:2.39.5-0+deb12u1) ...  
Traitement des actions différées (« triggers ») pour man-db (2.11.2-2) ...  
Traitement des actions différées (« triggers ») pour libc-bin (2.36-9+deb12u8) ...  
root@deb-docker:/home/harani# systemctl status docker  
● docker.service - Docker Application Container Engine  
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; preset: enabled)  
   Active: active (running) since Tue 2024-10-08 10:35:41 CEST; 15s ago  
TriggeredBy: ● docker.socket  
   Docs: https://docs.docker.com  
  Main PID: 4425 (dockerd)  
    Tasks: 10  
  Memory: 26.3M  
     CPU: 227ms  
   CGroup: /system.slice/docker.service  
           └─4425 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock  
  
oct. 08 10:35:41 deb-docker systemd[1]: Starting docker.service - Docker Application Container Engine...  
oct. 08 10:35:41 deb-docker dockerd[4425]: time="2024-10-08T10:35:41.164898348+02:00" level=info msg="Starti>  
oct. 08 10:35:41 deb-docker dockerd[4425]: time="2024-10-08T10:35:41.244086392+02:00" level=info msg="Loadin>  
oct. 08 10:35:41 deb-docker dockerd[4425]: time="2024-10-08T10:35:41.619763576+02:00" level=info msg="Loadin>  
oct. 08 10:35:41 deb-docker dockerd[4425]: time="2024-10-08T10:35:41.630876355+02:00" level=warning msg="WAR>
```

```
root@deb-docker:/home/harani# Gpasswd -a harani docker
```

III. Vérification du bon fonctionnement de docker

```
root@deb-docker:/home/harani# Docker version
```

Lancement du conteneur de test :

```
root@deb-docker:/home/harani# Docker run --rm hello-world
```

l'option --rm sera effacé après l'affichage.

VI. Premier pas avec docker

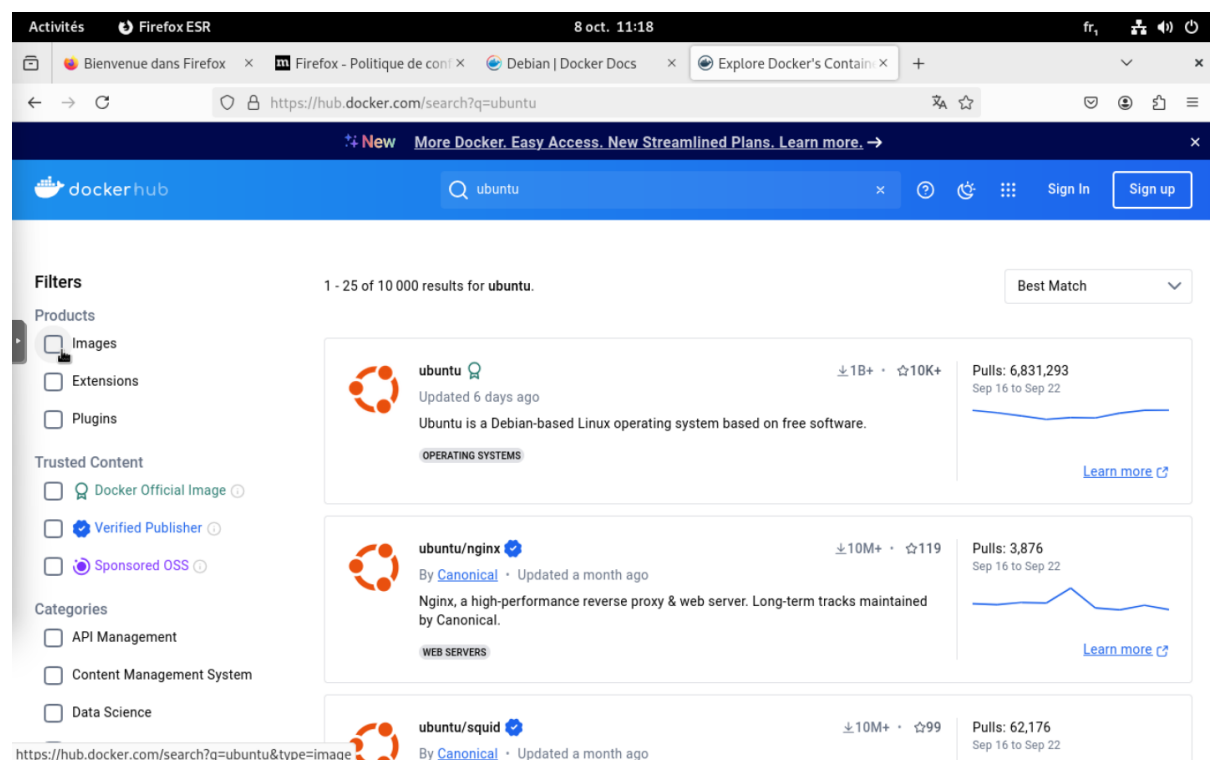
1. Gérer les images

1.1. Interroger le "docker registry" en CLI

Commande : `docker search <mot clé>`

```
root@deb-docker:/home/harani# Docker search ubuntu
```

Équivalent en interface graphique



Cliquer sur Ubuntu. Plusieurs images pour Ubuntu correspondant aux différentes versions. C'est le tag qui différencie les versions.

1.2. Télécharger une image

Commande : `Docker pull <nom image>`

```
root@deb-docker:/home/harani# Docker pull ubuntu
```

Pour récupérer une autre version de l'image, il faut associer le nom de l'image + récupérer la version :

```
root@deb-docker:/home/harani# Docker pull ubuntu:lunar
```

1.3. Connaître les images disponibles sur le système

```
root@deb-docker:/home/harani# Docker images
```

```
root@deb-docker:/home/harani# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    61b2756d6fa9   3 weeks ago   78.1MB
ubuntu        lunar     f4cdeba72b99   10 months ago 70.3MB
hello-world    latest    d2c94e258dcb   17 months ago 13.3kB
root@deb-docker:/home/harani#
```

1.4. Supprimer une image

Commande : `docker rmi <nom image>`

```
root@deb-docker:/home/harani# Docker rmi hello-world
```

2. Gérer les conteneurs

2.1. Lancer un conteneur

Commande : `docker run [option] <nom image> [commande]`