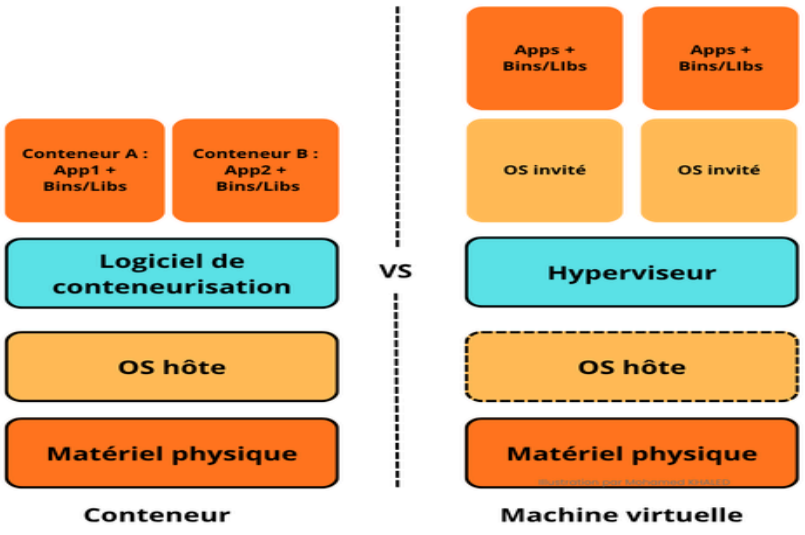
Les Fondamentaux Docker

1. Histoire de la conteneurisation

Évolution des déploiements applicatifs

- Années 1960-2000 : Applications déployées directement sur serveurs physiques:
    - Meilleure performance grâce à l'accès direct au matériel, sans aucune surcharge de virtualisation.
    - Conflits de dépendances.
    - Gaspillage de ressources.
- Années 2000-2010 : Virtualisation :
    - Chaque VM a son propre système d'exploitation, ce qui entraîne une redondance dans les systèmes et les processus communs
    - Démarrages et arrêts plus rapides.
    - Reproductibilité : possibilité de cloner ou de revenir dans le temps pour une VM donnée.
- Conteneurisation:
    - Un conteneur venant à faillir peut être très facilement remplacé
    - Partage du noyau de l'OS hôte
    - Démarrage rapide (secondes)

2. Machines Virtuelles vs Conteneurs



Aspect	VM	Conteneur
Partage du Noyau	**Non** (Chaque VM a son propre OS/Noyau invité)	**Oui** (Partage le noyau de l'OS hôte)
Démarrage	Minutes	Secondes
Isolation	Complète (hardware virtuel)	Moins isolée (Namespaces et Cgroups)
Taille	GB (OS complet)	MB (application + dépendances)
Portabilité	Lourde (images volumineuses)	Excellente

Alors que les *Namespaces* gèrent ce que le conteneur **peut voir**, les *Cgroups* contrôlent les ressources qu'il **peut utiliser.**

3. Architecture Docker

Les fondations techniques qui permettent la conteneurisation existent en effet dans le **noyau Linux** (*Linux Kernel*) depuis des années, bien avant l'arrivée de Docker.
L'outil **Linux Containers (LXC)**, apparu en **2008**, a été l'un des premiers à utiliser ces capacités. Cependant, c'est **Docker**, lancé en **2013**,a créé une plateforme simple et standardisée pour construire, distribuer et exécuter ces conteneurs.

3.1 Les composants de Docker

1. Docker Daemon

- Service backend qui tourne en arrière-plan
- Gère les images, conteneurs, networks, volumes
- Écoute les requêtes de l'API Docker

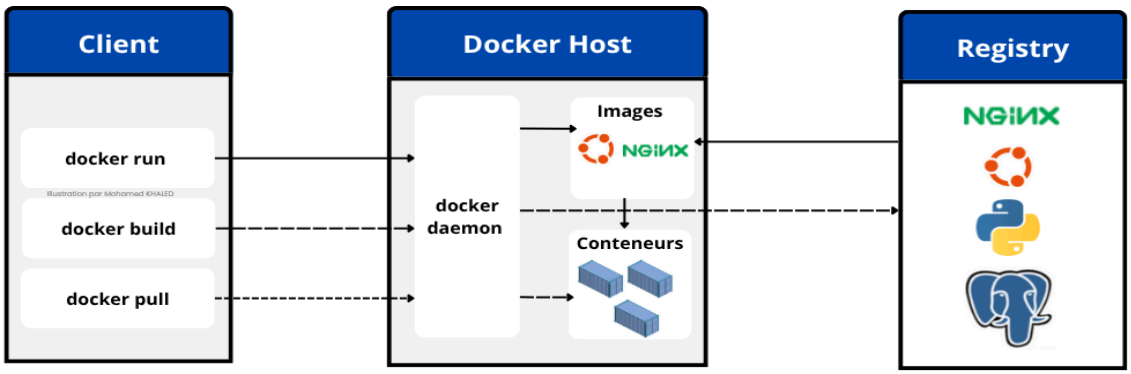
2. Docker Client

- Interface en ligne de commande
- Envoie les commandes au daemon via REST API
- Peut communiquer avec des daemons distants

3. Docker Hub

- Stocke les images Docker
- Registry public le plus populaire
- Images officielles maintenues
- Images communautaires
- Gratuit pour images publiques
- Plans payants pour privé

Navigation sur https://hub.docker.com/search



3.2 Concepts clés Docker

1. Image Docker (La recette)

- Modèle en lecture seule
- Contient le système de fichiers et les paramètres d'exécution
- Construite en couches (layers)
- Immuable une fois créée

2. Conteneur Docker (Le gâteau)

- Instance exécutable d'une image
- Ajoute une couche en lecture/écriture au-dessus de l'image
- Éphémère par défaut
- Isolé des autres conteneurs

3. Volumes

- Stockage persistant
- Survit à la suppression du conteneur
- Partageable entre conteneurs

4. Networks

- Isolation réseau entre conteneurs
- Communication inter-conteneurs
- Exposition de ports vers l'hôte

4. Installation et premiers pas

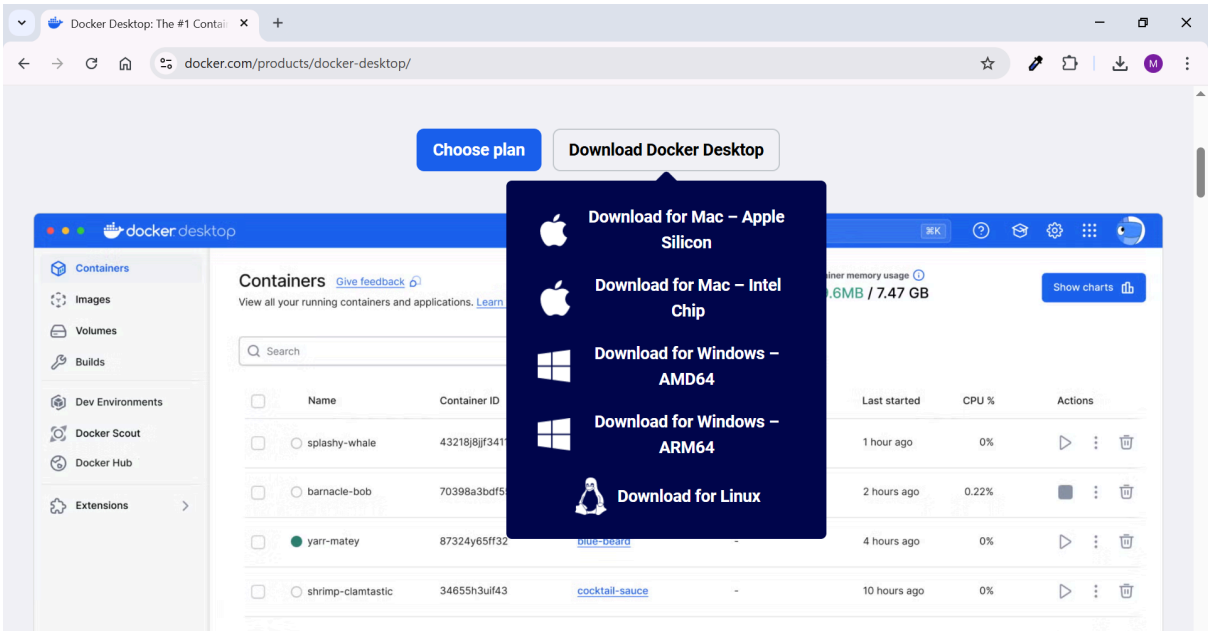
4.1 Installation selon l'OS

L'installation de Docker Desktop varie en fonction de votre système d'exploitation. Docker Desktop est l'application graphique recommandée car elle inclut le moteur Docker (Docker Engine), l'interface en ligne de commande (Docker CLI).

Pour garantir que vous obteniez la version la plus récente et la plus sécurisée, le téléchargement de Docker Desktop doit s'effectuer **uniquement** sur le site officiel de Docker.

Lien officiel : https://www.docker.com/products/docker-desktop/

- Sur cette page, choisissez la version correspondant à votre système d'exploitation (**Windows, macOS (Apple Silicon ou Intel), ou Linux**).
- Suivez ensuite les instructions spécifiques fournies sur la documentation officielle de Docker pour votre installation.



4.2 Vérification de l'installation:

- La version du client Docker:
  `docker --version`
- Informations détaillées:
  `docker info`

Cette commande fournit des informations détaillées sur l'état du client Docker (l'outil que vous utilisez) et le serveur Docker (Docker Engine) qui exécute les conteneurs:

- Client Docker:
  - Version: La version de votre client Docker.
  - Context: définit où vos commandes Docker (docker run, docker ps, etc.) seront exécutées.
  - Debug Mode: Le mode de débogage.
  - Plugins: Liste tous les plugins installés et disponibles, qui étendent les fonctionnalités de Docker.
    - buildx: Le plugin moderne pour construire des images Docker.
    - compose: Le plugin docker compose pour définir et exécuter des applications multi-conteneurs.
    - ai, debug, desktop, extension, scout, etc.: Divers plugins fournis par Docker Inc. pour des fonctionnalités avancées (gestion, sécurité, etc.).
- Server (Le Moteur Docker):

- État Actuel des Conteneurs et Images
  - Containers: nombre de conteneurs au total.
  - Running: nombre de conteneurs en cours d'exécution.
  - Stopped:nombre de conteneurs arrêtés.
  - Images:nombre des images Docker stockées localement.
- Configuration du Stockage et du Système
  - Server Version: La version du moteur Docker.
  - Storage Driver: le système de fichiers utilisé par Docker pour stocker les images et les couches de conteneurs sous Linux.
  - Backing Filesystem: Le système de fichiers sous-jacent à Storage Driver
  - Logging Driver: le format pour stocker les journaux (logs) des conteneurs.
  - Cgroup Driver: indique comment le moteur qui gère l'allocation des ressources (CPU, mémoire) aux conteneurs.
  - Kernel Version: indique le noyau sur lequel le moteur Docker s'exécute.
  - Operating System / OSType: Le moteur s'exécute bien sous quel OS au sein de Docker Desktop.
  - Architecture: L'architecture du processeur du moteur.
  - CPUs / Total Memory: Les ressources (CPU et RAM) allouées au moteur Docker par Docker Desktop (ces ressources peuvent être configurées dans les paramètres de Docker Desktop).
  - Docker Root Dir: le chemin dans le système de fichiers où Docker stocke toutes ses données (images, conteneurs, volumes).
- Composants et Runtimes
  - Plugins: Liste les drivers disponibles pour les volumes, les réseaux (bridge, host, overlay, etc.) et les logs.
  - Runtimes: Les technologies utilisées pour exécuter les conteneurs.
  - containerd version / runc version: Les versions des composants sous-jacents utilisés par le moteur Docker.

Le mode débogage nécessite un abonnement Pro, Team ou Business . Le mode débogage présente plusieurs avantages, tels que :

- Une boîte à outils personnalisable. Cette boîte à outils inclut de nombreux outils Linux standard préinstallés, tels que vim, nano htop, et curl.
- La possibilité d'accéder aux conteneurs qui n'ont pas de shell, par exemple les conteneurs slim ou sans distribution.

4.3 Exécuter une image depuis Docker hub:

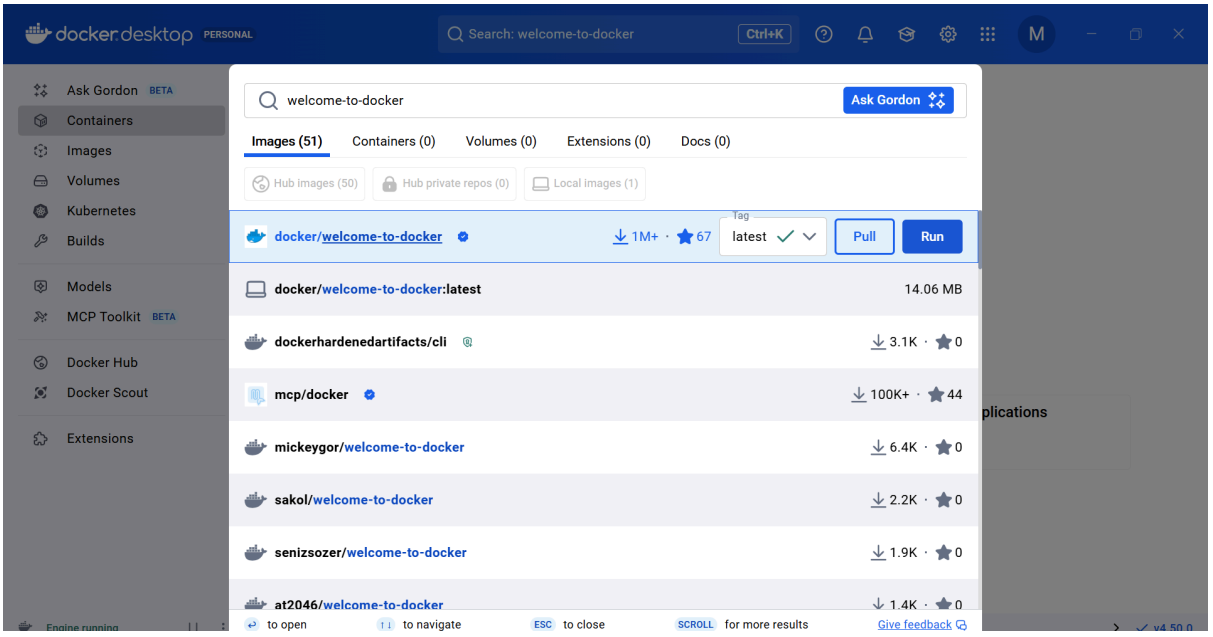
- **Exécuter l'image via le Terminal:**
  - Ouvrez votre interface de ligne de commande
  - Utilisez la commande docker run hello-world pour télécharger l'image et démarrer un conteneur à partir de celle-ci
    Ce qui se passe en détail:
    1. Docker client contacte le Docker daemon.
    2. Daemon cherche l'image 'hello-world' localement.

3. Ne la trouve pas, donc télécharge depuis Docker Hub.
4. Crée un nouveau conteneur depuis cette image.
5. Exécute l'application dans le conteneur.
6. Affiche le message (Hello from Docker!).
7. Le conteneur s'arrête.

Windows PowerShell

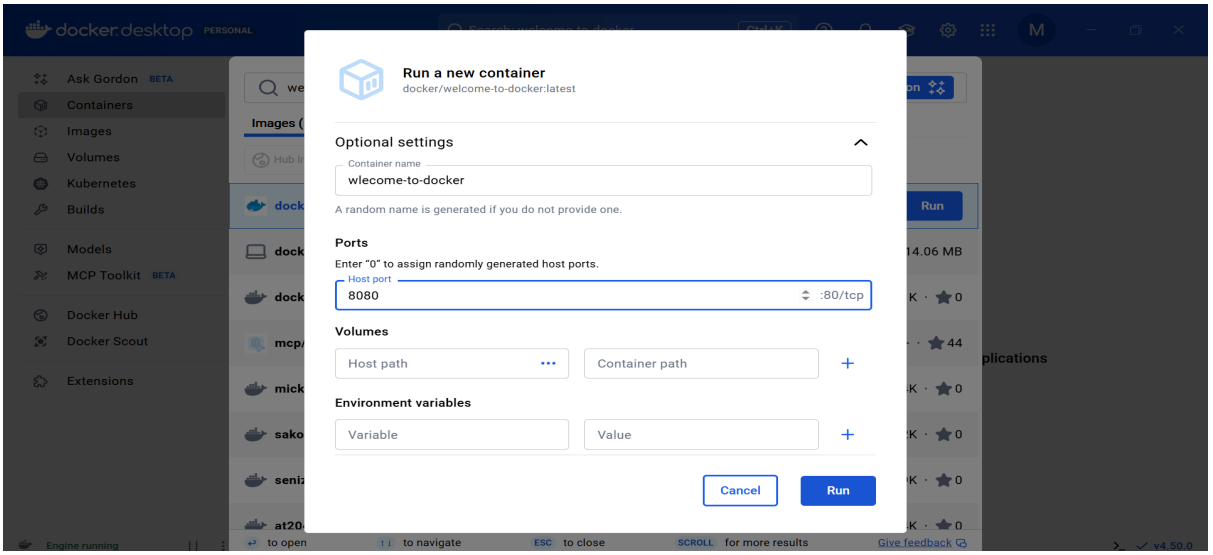
```
PS C:\Users\medkh> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:f7931603f70e13dbd844253370742c4fc4202d290c80442b2e68706d8f33ce26
Status: Downloaded newer image for hello-world:latest

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/
```

- **Exécuter l'Image depuis l'Interface Graphique:**
  - Ouvrez l'application Docker Desktop.
  - Cherchez l'image welcome-to-docker dans la barre du recherche.



  - Cliquez sur le bouton **"Run"** (Exécuter)

- Une fenêtre de configuration s'ouvrira. Vous pouvez y définir des paramètres (nom du conteneur, ports, etc.).



- Cliquez sur **"Run"** dans cette fenêtre pour démarrer le conteneur.
- Allez dans la section **Containers** dans la barre latérale de Docker Desktop.
- Localisez l'image **welcome-to-docker**.
- Cliquez sur le bouton ⋮ puis sur le bouton **"Open with browser"**