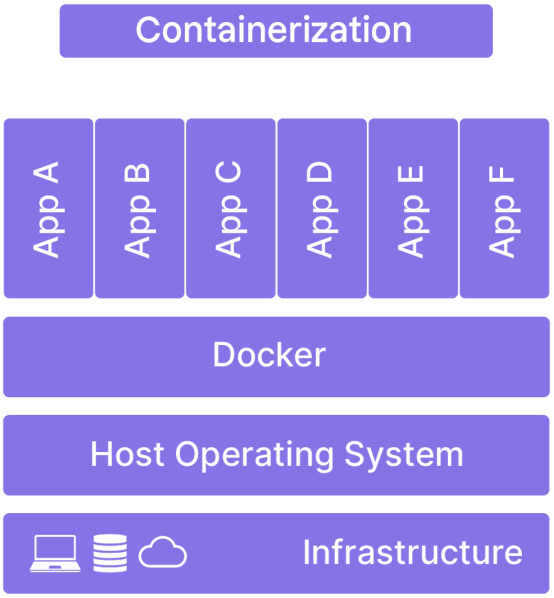
Docker permet de créer des conteneurs spécialisés pour les applications facilitant leur déploiement, leur gestion et leur sécurité tout comme un porte-conteneurs optimise le transport de différentes marchandises. chaque application nécessite un conteneur spécifique

docker permet de créer déployer et exécuter des conteneurs de manière intuitive (cad naturellement sans connaissance approfondie peut être à cause de l'accumulation d'expérience)

La virtualisation est une technologie qui permet de créer plusieurs machines virtuelles sur un seul serveur physique chaque machine virtuelle fonctionne comme une entité indépendante avec son propre OS , ses applications et ses ressources dédiés : cette isolation consomme plus de ressources . La clé de la virtualisation est l'hyperviseur, un logiciel qui permet de créer et de gérer des machines virtuelles.

La conteneurisation est un technique qui permet d'encapsuler des processus (programme en cours d'exécution) dans un ou plusieurs conteneurs qui partage le même noyau du système d'exploitation



fonctionnement des conteneurs à travers les cgroups et les namespace

Les conteneurs sont gérés par des moteurs de conteneurisation comme Docker. Ces moteurs utilisent des fonctionnalités du noyau Linux telles que les **cgroups** et les **namespaces** pour isoler et contrôler les ressources.

- les cgroups (control group) permettent de contrôler les ressources attribués à chaque conteneur (exemple limiter l'utilisation du cpu ou de la ram par un conteneur pour éviter qu'il consomme toutes les ressources
- les namespaces garantissent l'isolation du processus (programme en cours d'exécution) du système de fichier et des réseaux entre les conteneurs (chaque conteneur ne voit que ses ressources ce qui améliore l'efficacité et la sécurité de l'infrastructure

Concernant l'isolation, et contrairement aux machines virtuelles, **les conteneurs partagent le même noyau système** mais sont isolés en termes de processus, de réseau et de système de fichier.
dockerhub est une plateforme pour partager et découvrir des images docker

Architecture de docker

Docker Engine est le cœur de la plateforme Docker. Il est responsable de la création, de l'exécution et de la gestion des conteneurs. Il est lui-même découpé en plusieurs éléments :

- **Docker Daemon (dockerd) :** Processus principal (serveur) qui gère les conteneurs Docker sur l'hôte. Il expose une API REST.
- **Docker Client (docker) :** Interface en ligne de commande permettant aux utilisateurs d'interagir avec le daemon Docker pour exécuter des commandes de gestion des conteneurs.
- **Docker image** :ce sont des archives immuables contenant tout ce dont une application a besoin pour fonctionner, y compris le code, les bibliothèques et les dépendances.
- **Docker hub** : est une plateforme pour stocker, partager et découvrir des images Docker.Registre par défaut utilisé par le Docker Engine, il facilite le partage et la collaboration des utilisateurs de solutions de conteneurisation.
- **Docker swarm** : est un outil d'orchestration pour gérer des clusters de machines Docker. Il permet de regrouper plusieurs machines hôte en un cluster unique, facilitant ainsi la gestion de l'orchestration des conteneurs à grande échelle.

les autres outils d'exécution de conteneur : podman , run c

Résumé : La conteneurisation utilise des conteneurs légers partageant le noyau de l'OS hôte, tandis que la virtualisation crée des machines virtuelles avec leur propre OS complet.
Les cgroups gèrent les ressources allouées aux conteneurs et les namespaces assurent leur isolation au niveau des processus et des ressources système.
Docker simplifie la conteneurisation avec des composants comme Docker Engine, Docker Images, Docker Containers, Docker Compose, et Docker Swarm pour une gestion et une orchestration efficaces.

L'Open Container Initiative (OCI) établit des standards ouverts pour l'interopérabilité des conteneurs, avec Docker adoptant ces standards pour éviter l'enfermement technologique.

Partie II :
docker repose sur une architecture client (son interface en ligne de commande utilisé par l'utilisateur pour interagir avec docker) serveur (son docker daemon)

les commandes docker et leur rôle :

- **Les volumes**: Ils sont utilisés pour stocker des données de manière persistante, voire pour partager des données entre plusieurs conteneurs.
- **Les networks**: Ils permettent la communication réseau entre les conteneurs et avec l'extérieur. Les networks sont associés à un type, chacun permettant de créer différentes topologies réseau.

- Lister les images : **docker images**
- Télécharger une image :**docker pull <image>**
- En mode interactif avec la commande : **docker run -it <image>**
- En tâche de fond avec la commande : **docker run -d <image>**
- Supprimer une image :**docker rmi <image>**
- Lister les volumes :**docker volume ls**
- Créer un volume :**docker volume create <volume_name>**
- Supprimer un volume :**docker volume rm <volume_name>**
- Lister les networks :**docker network ls**
- Créer un network :**docker network create <network_name>**
- Supprimer les networks :**docker network rm <network_name>**
- Avec un volume avec la commande : **docker run -v <volume_name>:/path/to/dir -it <image>**
- En exposant des ports du conteneur avec la commande : **docker run -p <host_port>:<container_port> -it <image>**
- Voir les processus en cours dans un conteneur : **docker top <container>**
- Afficher les logs du conteneur pour débugger ou surveiller : **docker logs <container>**
- Arrêter un conteneur : **docker stop <container>**
- Arrêter brutalement un conteneur : **docker kill <container>**
- Supprimer un conteneur : **docker rm <container>**
- Copier des fichiers depuis un conteneur vers l'hôte et vice versa : **docker cp <container>:/path/to/file /local/path**

Démarrer un serveur nginx : serveur http bien connu du monde du web : docker run nginx:latest

mapper un port donc le port 80 de nginx sur le port 8080 de la machine hôte :
commande :

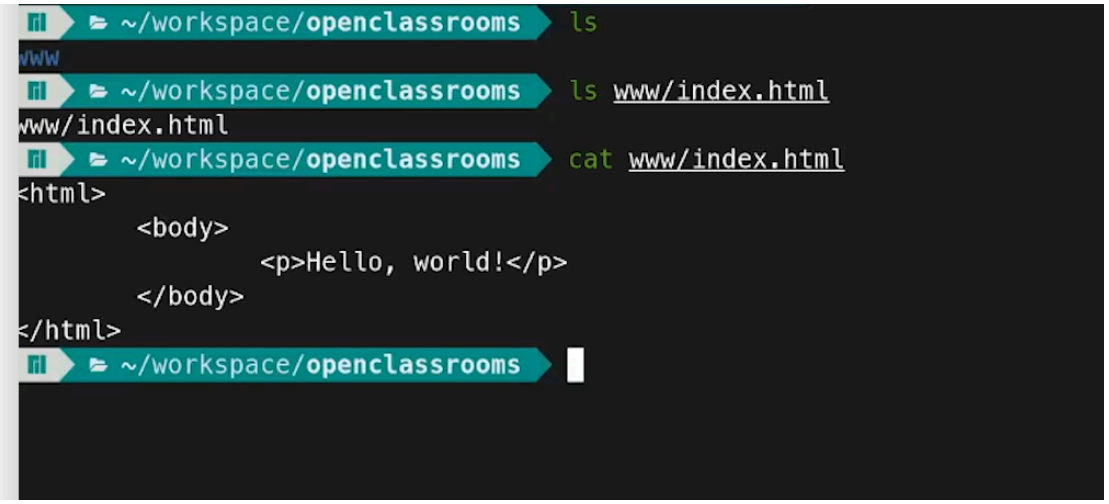
- **docker run nomImage -p portHote:portConteneur = docker run -p 8080:80 nginx:latest .**
- Pour tester on fait un curl **: curl http:localhost:8080** : on verra une page retourner par curl **.**

Bind Mount : Vous mappez un chemin spécifique depuis votre système hôte directement dans le conteneur. Par exemple :

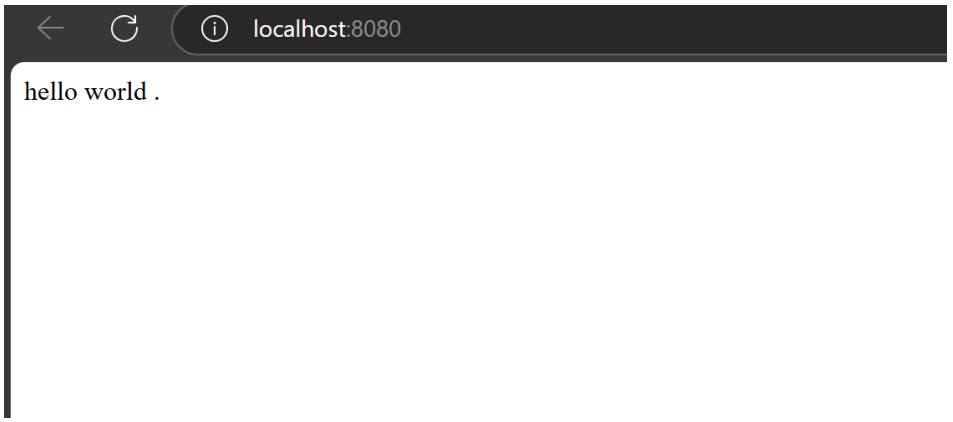
- On souhaite alors modifier le contenu de la page html servi par le serveur nginx alors pour cela il faut monter un volume (utiliser le flag -v qui permet de monter un volume au sein d'un conteneur avec du bind mount) . Il faut au préalable avoir créer un dossier comment dans l'image en dessous et ensuite exécuter :
- **docker run -p 8080:80 -v $PWD/WWW:/usr/share/nginx/html nginx (sur linux) ou**

docker run -p 8080:80 -v "C:\Users\nom\Documents\docker\course\www":/usr/share/nginx/html nginx ' sur windows) :

dans un shell linux la variable dollar pwd est un alias vers le répertoire courant pour ce qui est du répertoire **/usr/share/nginx/html** c'est le répertoire servi par le serveur nginx dans sa configuration par défaut



Résultat la page de nginx a été remplacé par notre fichier index.html en local



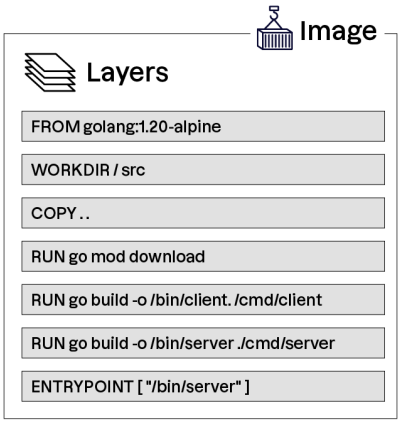

```
# Récupérer la dernière version de l'image Nginx officielle
docker pull nginx:latest
# Démarrer votre conteneur en mode interactif (Ctrl+C pour le stopper)
# en exposant le port 80 du conteneur sur le port 8080 de l'hôte
docker run -it -p 8080:80 nginx:latest
# Une fois stoppé, lister les conteneurs en incluant ceux stoppés
docker ps -a
# Supprimer le conteneur (où <container_id> est à remplacer par l'identifiant trouvé avec la com
docker rm <container_id>
```

docker run -it –rm merveil:lastest : va démarrer l'image dans un conteneur en mode interactive donc sous forme de shell et une fois qu'on quite il sera automatiquement supprimer

pour créer un tag : **docker tag merveil:lastest merveil:v1 .** et pour checker simplement voir que l'identifiant de merveil:latest qui sera le mm que merveil:v1

Partie III

Anatomie d'une image docker



QCM

1 **La virtualisation nécessite plus de ressources que la virtualisation conteneurisation** . La conteneurisation partage le noyau du système d'exploitation hôte, tandis que la virtualisation utilise un hyperviseur pour exécuter plusieurs systèmes d'exploitation.

La conteneurisation partage le noyau du système d'exploitation hôte, ce qui permet aux conteneurs de démarrer rapidement et de consommer moins de ressources par rapport aux machines virtuelles, qui utilisent un hyperviseur pour exécuter plusieurs systèmes d'exploitation indépendants. Contrairement à la virtualisation, la conteneurisation ne nécessite pas d'hyperviseur et repose plutôt sur l'isolation au niveau du système d'exploitation, ce qui en fait une solution plus légère. De plus, la conteneurisation permet une utilisation plus efficace des ressources, contrairement à l'idée fausse selon laquelle elle en nécessiterait davantage.

2 **Quel intérêt principal lui soulignez-vous à propos de Docker** ?

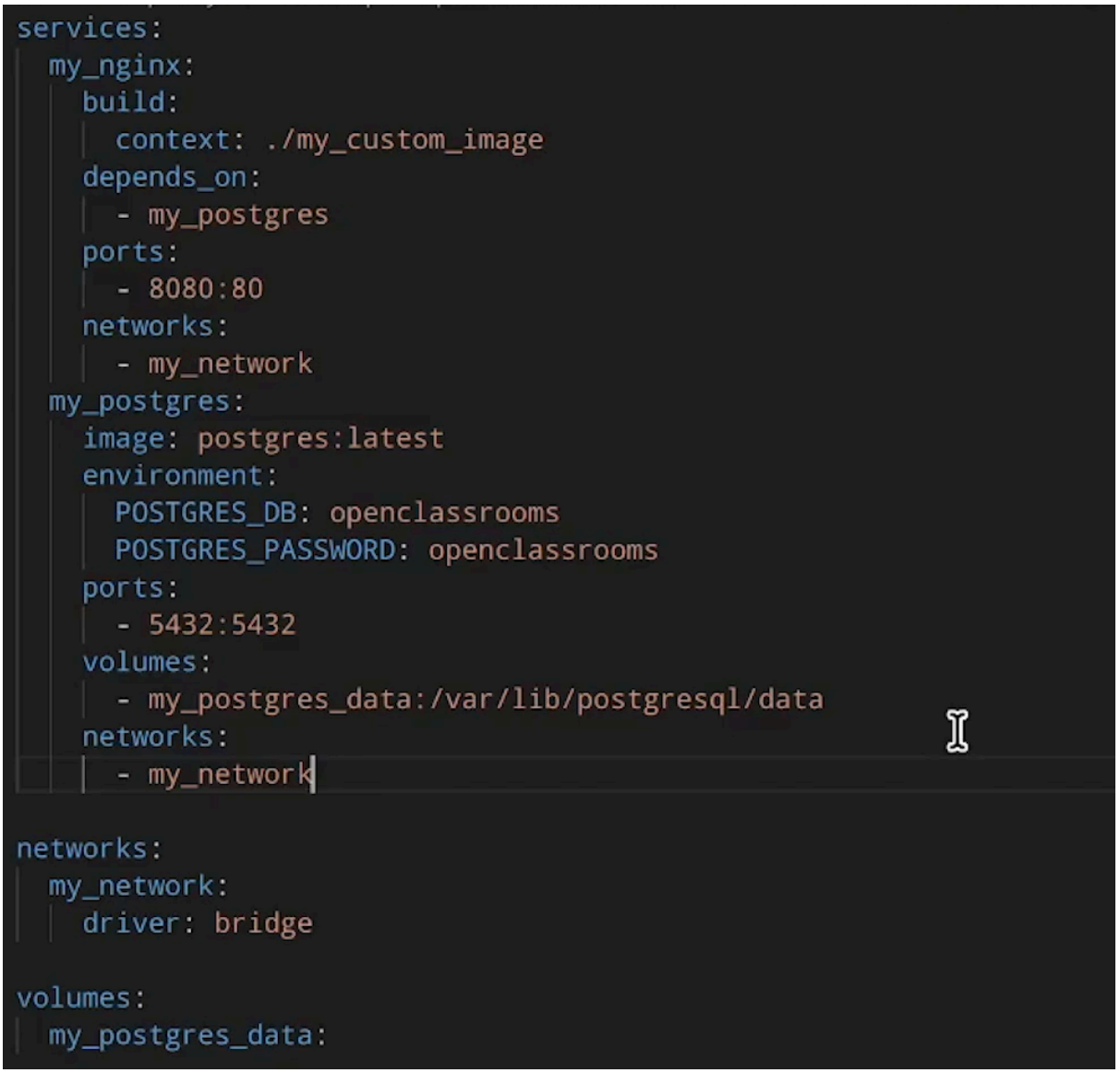
Gérer les conteneurs en fournissant une API pour interagir avec eux.

3 **L'OCI (Open Container Initiative)** est une organisation formée pour établir des standards ouverts pour les formats de conteneurs et les moteurs d'exécution, garantissant ainsi que les conteneurs peuvent être exécutés de manière uniforme sur différentes plateformes.

Docker compose

- Docker Compose est un outil essentiel pour définir et gérer des applications multi-conteneurs de manière cohérente et centralisée via un fichier docker-compose.yml
- Le fichier docker-compose.yml est composé de plusieurs sections, les principales étant services , définissant les conteneurs,volumes et networks.
- Les sous-commandes docker compose …permettent de gérer l'ensemble du cycle de vie de votre environnement conteneurisé, du démarrage à l'arrêt, en passant par la surveillance de celui-ci.

```
services:
# Section de définition des “services” qui constituent l’environnement
# applicatif
volumes:
# Section de définition des “volumes” (supports de stockage) utilisés
# par l’environnement applicatif
networks:
# Section de définition des “networks” (réseaux et interfaces virtuels)
# utilisés par l’environnement applicatif.
```



```yaml
services:
  my_nginx:
    build:
      context: ./my_custom_image
    depends_on:
      - my_postgres
    ports:
      - 8080:80
    networks:
      - my_network
  my_postgres:
    image: postgres:latest
    environment:
      POSTGRES_DB: openclassrooms
      POSTGRES_PASSWORD: openclassrooms
    ports:
      - 5432:5432
    volumes:
      - my_postgres_data:/var/lib/postgresql/data
    networks:
      - my_network

networks:
  my_network:
    driver: bridge

volumes:
  my_postgres_data:
```

Docker swarm

permet de gérer uine flotte de porte conteneur

Ansible

c’est un outil d’automatisation permettant de configurer et déployer un système il est basé sur python

Un **node** (ou ***managed node***, ou ***host***) est un poste connecté au node manager en SSH, et sur lequel Ansible viendra pousser les tâches d’automatisation.

Un **node manager**, ou ***control node***, est un poste qui contrôle les nodes grâce à sa connexion SSH. Il dispose d'une version Ansible d'installé pour leur pousser les tâches d'automatisation grâce aux commandes `ansible` et `ansible-playbook`

Un **rôle** est une **structure arborescente** constituée de **répertoires** et de **fichiers de configuration** YAML, qui vont avoir pour fonction d'installer tel ou tel système.

eg : role d'installer apache , role d'installer maria db

Une tâche est une instruction décrite en YAML dans un fichier de configuration.

De façon schématique, vous pouvez retenir que :

- un **rôle** contient un ou plusieurs **fichiers de configuration** (YAML) ;
- un **fichier de configuration** contient une ou plusieurs **tâches** ;
- une **tâche** fait appel à un **module**.

Un playbook est un **fichier de configuration YAML** contenant une suite de jeux d'instructions, ou *plays* en anglais. Chacun peut être constitué d'options, et fait appel à **un ou plusieurs rôles**. Il permet de décrire une **stratégie de déploiement**, ou de configuration, en **structurant** les actions nécessaires.

`ansible-galaxy init wordpress` créera toute l'arborescence nécessaire en une commande.

`ansible-vault` vous permettra de chiffrer une chaîne de caractères, par exemple : ansible-vault encrypt_string 'chaine1'.