Programmation Python
DOCKER

SAÏDI MASSINISSA

Qu’est ce que docker ?

Qu’est ce que Docker ?

- Docker est une plateforme de conteneurisation

- **conteneur** : environnements isolés contenant une application et toutes ses dépendances (bibliothèques, outils, configuration, OS…)

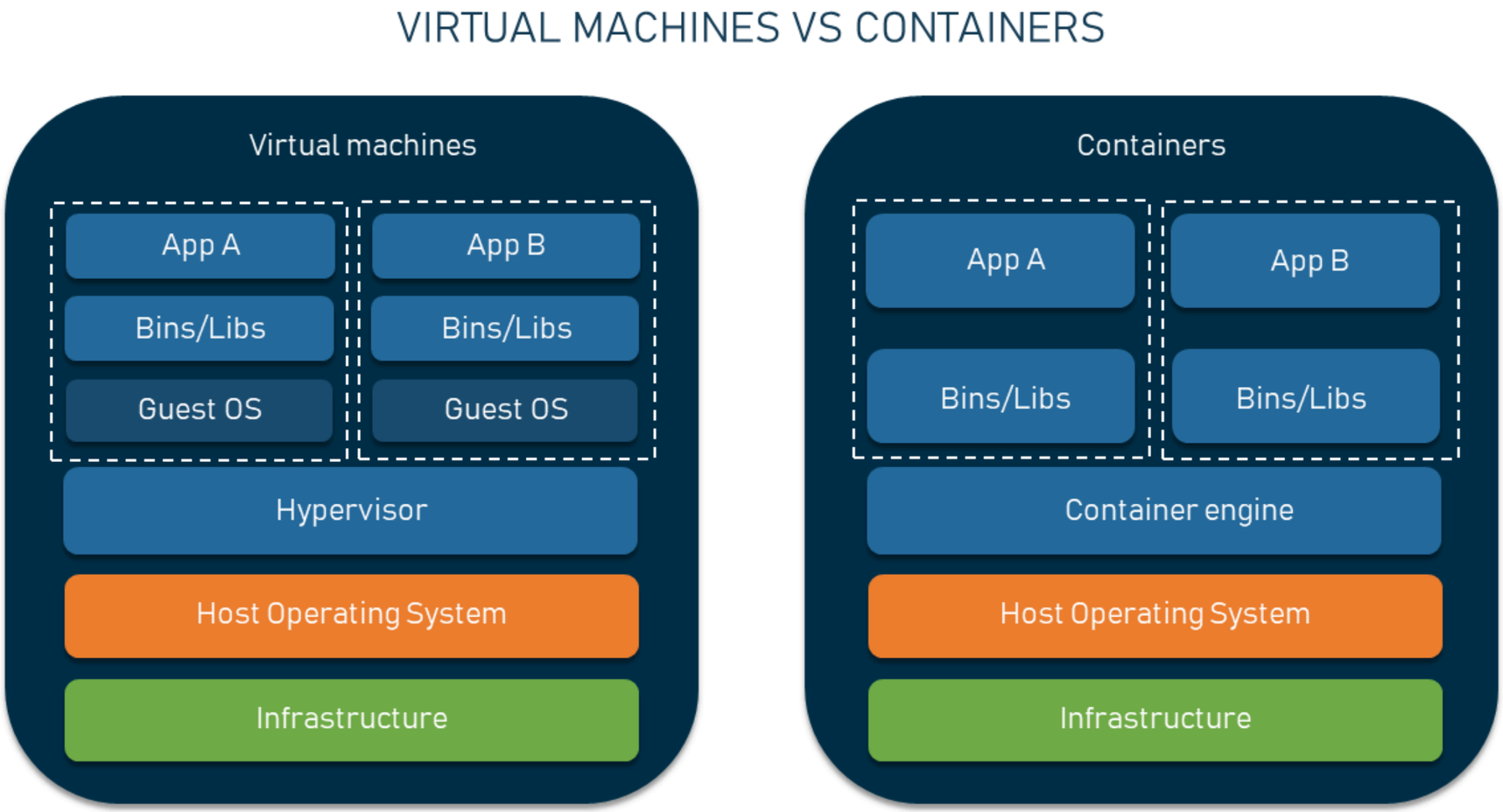
- Docker assure que ton application fonctionne partout de la même manière, que ce soit sur ton PC, un serveur, ou le cloud.

Comparaison avec Conda

	DOCKER	CONDA
Portabilité	Tout type d'apps et langages	Principalement Python
Isolation	Niveau système (OS, libraires python, libraries OS...)	Libraries python
Déploiement prod	Oui (cloud, serveurs, Kubernetes)	Usage en local
Objectif	Reproduire tout un environnement	Reproduire un environnement Python

Pourquoi Docker ?

Pourquoi Docker ?



altexsoft

Pourquoi Docker ?

- **Portabilité** : un conteneur fonctionne sur n'importe quel OS.
- **Reproductibilité** : plus d'incompatibilités entre environnements.
- **Isolation** : chaque application tourne dans sa boîte, sans conflit avec les autres.
- **Efficacité** : plus léger et rapide qu'une machine virtuelle.
- **Scalabilité** : facile à multiplier dans le cloud.

Concepts de base

Concepts de base

- **Dockerfile** : recette pour construire une image.

build

- **Image** : modèle figé (ex : python:3.13, mysql:8)
  NB : toute image est définie par une image de base

run

- **Conteneur** : instance vivante d'une image.

- **Volume** : stockage persistant (si tu veux garder des fichiers en dehors du conteneur).

- **DockerHub** : dépôt d'images prêtes à l'emploi.

Concepts de base

```
FROM python:3.13

WORKDIR /usr/src/app

COPY requirements.txt .
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt

COPY . /usr/src/app/

EXPOSE 80

CMD [“uvicorn“, “api:app“, “--host“, “0.0.0.0“, “--port“, “80“]
```

Concepts de base

IMAGES

- docker images : liste les images
- docker pull mon-image : pull l'image en local
- docker rmi mon-image : supprimer l'image en local
- docker build -t mon-image . : build mon image à partir de mon dockerfile

CONTENEURS

- docker run mon-image : lancer un conteneur simple
- docker run -it mon-image : lancer un conteneur en mode intéractif
- docker run -d -p 8080:80 mon-image : Lancer en arrière-plan et mapper un port

- docker ps : voir les conteneurs actifs
- docker ps -a : voir tous les conteneurs (même arrêtés)
- docker rm <id> : supprimer un conteneur

LOGS

- docker logs <id> : voir les logs d'un conteneur
- docker exec -it <id> bash : ouvrir un shell dans le conteneur

DOCKER COMPOSE

Docker Compose

- Docker permet de lancer un conteneur à la fois avec docker run.
- Mais dans un vrai projet il y a plusieurs services (ex : API + BDD + app).

- Docker Compose est un outil qui permet de :
  - Décrire toute l'application (plusieurs conteneurs) dans un fichier YAML
  - Lancer tout d'un coup avec une seule commande (docker-compose up).

Pourquoi utiliser Docker Compose ?

- **Simplifie le déploiement** : une seule commande pour tout lancer.
- **Décrit la stack complète** (API, BDD, front, etc.) dans un fichier clair.
- **Évite les conflits** : chaque service tourne dans son conteneur isolé.

Concepts de base

```
version: "3"

services:
  service1:
    image: ...
    ports:
      - "hote:conteneur"
    environment:
      - VAR=valeur
  service2:
    build: .
    depends_on:
      - service1
```