POLYTECH NICE-SOPHIA

UNIVERSITÉ CÔTE D'AZUR

INFORMATIQUE POUR LA ROBOTIQUE 2

Cours n °2 bis : Python avec Docker

Enseignante : Maëva LECAVELIER - maeva.lecavelier@gmail.com

Sommaire

01

Rappel

Cours précédent et problématiques

Installation de la carte SD

Pourquoi : espace disque limité. Bloquant pour l'installation de librairies, l'enregistrement d'images, la création de modèles d'IA, etc.

Retour rapide sur le tuto :
https://wiki.seeedstudio.com/reComputer_Jetson_Memory_Expansion/
extension USB

Problèmes rencontrés : espace disque insuffisant, configuration boot

Problèmes de versions...

Python3

Python 3.6 : vieille version de Python qui a nativement une vieille version de pip

pip

pip 9.x : vieille version de pip qui n'est pas compatible avec les bibliothèques de traitement d'images

Environnements virtuels

Se base sur la version Python du système, et non une version de Python spécifiée

Deux solutions possibles

Changer notre système :

- Installer une nouvelle version de Python
- Changer tous les paramètres système pour appeler la nouvelle version Python au lieu de Python3.6
- Trouver quelles versions de pip, des bibliothèques python etc sont compatibles avec ce Python...

Risques :

- Oublier de changer des pointeurs à la modification du système
- Se perdre dans les compatibilités de versions

Isoler notre environnement :

- Créer et configurer un environnement léger et avec juste ce qu'il nous faut
- Déployer cet environnement
- S'y connecter

Risques :

- Nouveau concept à appréhender
- Téléchargement de quelques paquets

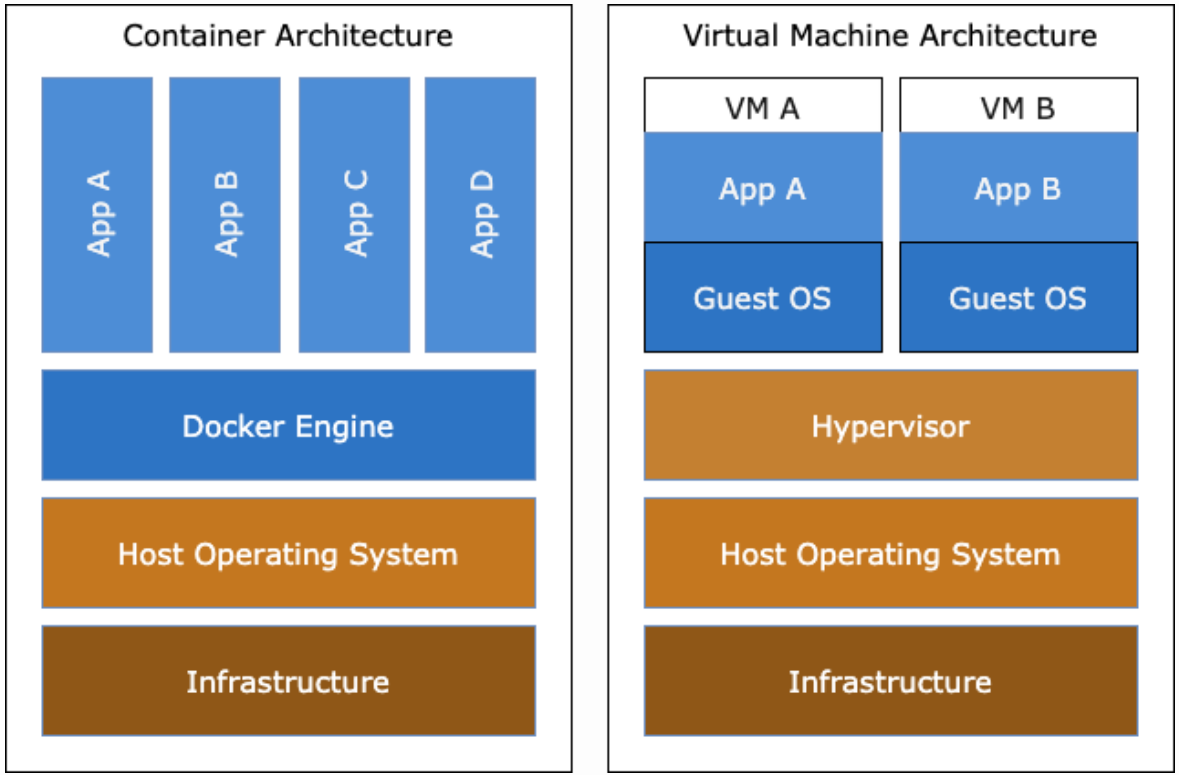
Avantage : on est maître de notre environnement, et on peut le redéployer à l'infini sur n'importe quelle plateforme

2

La virtualisation et conteneurisation

Des grands mots pour de beaux concepts

Virtualisation vs conteneurisation



En bref

Aspect	Virtualisation (VM)	Conteneurisation (Docker)
Taille	Lourd (plusieurs Go, car inclut un OS complet)	Léger (quelques dizaines de Mo, car partage du noyau)
Performances	Plus lent (le démarrage d'une VM prend plusieurs minutes)	Très rapide : un conteneur démarre en quelques secondes
Isolation	Complète, chaque VM a son propre OS. Seul le matériel est partagé	Isolation au niveau de l'application uniquement
Ressources utilisées	Nécessite beaucoup de CPU, RAM et espace disque	Très efficace en terme de ressources
Portabilité	Dépend du système hôte et de l'hyperviseur	Très portable : fonctionne sur presque toutes les machines

Et les environnements virtuels Python ?

Gestion des bibliothèques Python

Pratique pour avoir plusieurs projets sur une même machine hôte
Permet la gestion fine des bibliothèques **Python uniquement**

Isole un projet Python

Permet d'avoir plusieurs projets Python sur une même machine sans conflit sur les versions des bibliothèques

/!\ Se base sur la version Python de l'hôte

Cependant, la version Python est dépendante de l'hôte : si on change la version de Python, ça change pour tout le monde !

03

Docker

Simple et rapide

Qu’est-ce que Docker ?

- Permet d’exécuter des applications (ie: des programmes) dans des conteneurs. Un conteneur est un environnement léger, isolé et portable qui regroupe une application et tout ce dont elle a besoin pour fonctionner.
- Intérêt de docker : portabilité, isolation, efficacité (ressources, temps, espace disque)
- Facilite le développement et déploiement d’applications.

Comment utiliser Docker ?

Un Dockerfile : la recette de l'image à utiliser

```
docker build –t mon-docker .
```

```
docker run mon-docker
```

```dockerfile
# Utiliser une image officielle Python 3.10
FROM python:3.10-slim

# Installer des dépendances système nécessaires
RUN apt-get update && apt-get install -y --no-install-recommends \
    python3-pip \
    python3-dev \
    python3-venv \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# Installer les bibliothèques Python
RUN pip install --no-cache-dir \
    numpy \
    jupyterlab \
    flask \
    requests

# Créer un dossier pour les projets
WORKDIR /app

# Copier les scripts ou fichiers nécessaires (optionnel)
COPY . /app

# Définir le point d'entrée par défaut pour le conteneur
CMD ["/bin/bash"]
```

Dockerfile

- Exemple du Dockerfile utilisé dans ce TD : polytech-robo3-24-25/2b-Python_Docker/Dockerfile at main · MaevaLecavelier/polytech-robo3-24-25

Première partie du TD

- Installer et configurer Docker sur votre Jetson

Pour cela, des ressources sont prêtes sur Github :

- `$> git clone https://github.com/MaevaLecavelier/polytech-robo3-24-25.git`
- `$> cd polytech-robo3-24-25/2b-Python_Docker`

Et suivre le README ☺

Deuxième partie du TD

Utiliser l'IA dans ce Docker:

- Exercice sans aide : polytech-robo3-24-25/2-Introduction_Python/TD2_intro-IA.md at main · MaevaLecavelier/polytech-robo3-24-25

- Exercice guidé avec solution : polytech-robo3-24-25/2b-Python_Docker/app/TD02-bis/exo-tensorflow_correction.md at main · MaevaLecavelier/polytech-robo3-24-25

Des questions ?

Au travail !

4

Docker-compose

Allons plus loin

Pourquoi gérer Docker

Un seul Docker : facile mais limité

En utilisant un seul Docker (conteneur), il n'y a pas besoin d'orchestrer différentes applications et services. Mais s'il y en a plusieurs ? Ou si je veux accéder à un élément extérieur à mon Docker ?

Accéder à des périphériques physiques

Pour accéder à des périphériques (USB, série... une caméra ou une Arduino par exemple ☺) il faut donner le droit au Docker d'y accéder

Faire communiquer différents services (avancé)

Dans le cas d'une application plus complexe, il faut orchestrer cette architecture multi-services.

Un chef d’orchestre : docker-compose

Permet de faire le lien entre des conteneurs Docker, ou entre la machine physique et un conteneur

Comment utiliser docker-compose ?

Un fichier qui résume l'organisation du déploiement : **docker-compose.yml**

Pour rajouter la possibilité d'accéder à des appareils connecté à l'hôte :

```
devices:
  - /dev/ttyUSB0:/dev/ttyUSB0
  - /dev/video0:/dev/video0
```

Pour utiliser le fichier :

- `$> docker-compose build # construit le conteneur`
- `$> docker compose up -d # lancer le conteneur`
- `$> docker ps # permet de voir si le conteneur est lancé`

docker-compose.yml

- Exemple du fichier docker-compose.yml utilisé dans ce cours : polytech-robo3-24-25/2b-Python_Docker/docker-composer.yml at main · MaevaLecavelier/polytech-robo3-24-25

Troisième partie du TD

- Pour ceux qui n’avaient pas pu faire le TD de la semaine dernière avec l’utilisation de la caméra : faire ce TD en utilisant docker-compose pour pouvoir accéder à la caméra

- Faire le nouveau TD sur la communication série entre Python et Arduino dans un environnement Docker

Merci !

Des questions ?