Installation PC ROS2

ROS est un Middleware Open Source pour développer des applications robotiques. Originellement développé sous Linux (Ubuntu), il est maintenant disponible sur plusieurs systèmes d'exploitation dont Debian et Windows.

Installation des prérequis et liens importants

Pour des raisons de stabilité et légèreté du système, il y a tout à penser que les déploiements de ROS dans des milieux industriels se font (robotique autonome et mobile) et se feront à l'avenir sur Ubuntu et de plus en plus Debian. L'industrie des serveurs a déjà largement adopté Debian pour sa stabilité et sa modularité. C'est pourquoi plutôt que d'apprendre la ligne de commande Windows, nous recommandons d'apprendre la ligne de commande Bash, utilisée dans Ubuntu/Debian. Pour cela, il faut installer un système (noyau) Linux, plusieurs options s'offrent à nous:

- Machine virtuelle
  - Windows subsystem for Linux (WSL2)
  - Machine virtuelle Linux, par exemple via VirtualBox
- Machine physique
  - dual-boot Windows-Ubuntu -> Installation en quelques clics via une clé USB Live
  - PC sous Ubuntu 22.04
    - Pour une tour : Branchement d'un SSD SATA dédié au lieu du SSD Windows
    - Branchement d'un SSD USB3 type Transcend ESD310C

Notes importantes pour les installations virtuelles (deux premières options d'installation) :

- Ces installations sont suffisantes pour effectuer des simulations et du développement tant qu'il n'y a pas de Hardware à tester. VirtualBox fonctionne à peu près pour des TPs avec une VM URSim mais c'est loin d'être optimal (plantages,...)
- L'accélération graphique n'est pas supportée par la carte graphique (GPU) mais par le processeur (CPU) (voir ce bug)
- un PC avec 32Go de RAM est recommandé si des composants imposants de ROS doivent être compilés, par exemple pour utiliser la version de développement MoveIt 2 Rolling. En effet Windows consomme à lui seul près de 4-8Go, Ubuntu >2Go et la compilation >4Go, on peut vite atteindre la saturation. 16Go peuvent suffire mais il faudra compiler sans

parallélisation, et fermer des applications lourdes dans Windows comme Firefox.

Ubuntu via Windows SubSystem for Linux (WSL2)

WSL2 installe une machine virtuelle avec le noyau Linux complet, supporté et managé par Microsoft Windows. **Il n'y a pas besoin de droits administrateur car le logiciel est disponible dans le store Windows**.

Prérequis :

- Depuis le menu démarrer Windows, rechercher "A propos de", "Spécifications de Windows"
  - Version >22H2
  - Build >19041 (testé avec 19045.2486)
  - Si votre version est inférieure, demandez à votre administrateur de màj vers 22H2 et Build 19045.2486
  - Si vous ne pouvez màj, optez pour l'option d'installation d'Ubuntu via VirtualBox
- Exécuter Windows PowerShell en mode administrateur (connectez-vous avec un compte administrateur si vous n'avez pas les droits)
- Lancer `wsl --install` (si ça ne fonctionne pas, votre Windows n'est probablement pas à la bonne version)
- `wsl --update`
- Redémarrer l'ordinateur

Installation de Ubuntu 22 :

- Ouvrir Windows Store
- Rechercher et installer `Ubuntu` (c'est la version LTS actuelle qui sera installée, en ce moment 22.04.X)
- Depuis le menu démarrer Windows, Lancer l'application `Ubuntu`. Une Terminal s'ouvre (ligne de commande Linux Bash)
- Définir l'utilisateur principal, par exemple `ros2` et un mot de passe (8 caractères mini, majuscule, minuscule, chiffre, caractère spécial).
- Mettre à jour Ubuntu

```
sudo apt update
sudo apt upgrade
```

Depuis Windows, pour éteindre les Machines Virtuelles Ubuntu et ainsi libérer la mémoire RAM affectée :

- Lancer l'application `Windows PowerShell`
- `wsl --shutdown` Autres commandes WSL depuis `Windows PowerShell` :
- `wsl --status` : devrait retourner `Distribution par défaut : Ubuntu`, `Version par défaut : 2` (WSL2)
- `wsl --list` (ou `wsl -l -v`) : liste les Machines Virtuelles Linux installées via WSL (et la version WSL utilisée)

Docker dans une VM WSL2

Pour utiliser docker dans une VM WSL2, par exemple Ubuntu :

- Désinstaller toute version précédente de docker installée sur votre VM Ubuntu. Dans Terminal(Ubuntu) :
  - `sudo apt remove docker*`
- Ajouter votre utilisateur au groupe docker
  - `sudo groupadd docker`
  - ```
    sudo usermod -aG docker $USER
    ```
- Passer sur une session administrateur_windows (.\install)
- Installer Docker Desktop for Windows https://docs.docker.com/desktop/windows/wsl/#turn-on-docker-desktop-wsl-2
  - Cocher WSL2 (devrait être coché par défaut si votre config WSL2 est OK)
- Ajouter votre utilisateur_windows (nom_de_famille) Windows au groupe docker
  - Dans CMD/Powershell :
    ```
    net localgroup docker-users "utilisateur_windows" /ADD
    ```
- Repasser sur votre session utilisateur_windows
- L'intégration Docker-WSL est activée sur la distribution WSL par défaut, normalement Ubuntu (22)
  - pour s'en assurer, `wsl --set-default ubuntu`
  - Au besoin il est possible de l'activer sur une distro spécifique dans **Settings** > **Resources** > **WSL Integration**
- Démarrer Terminal(Ubuntu)

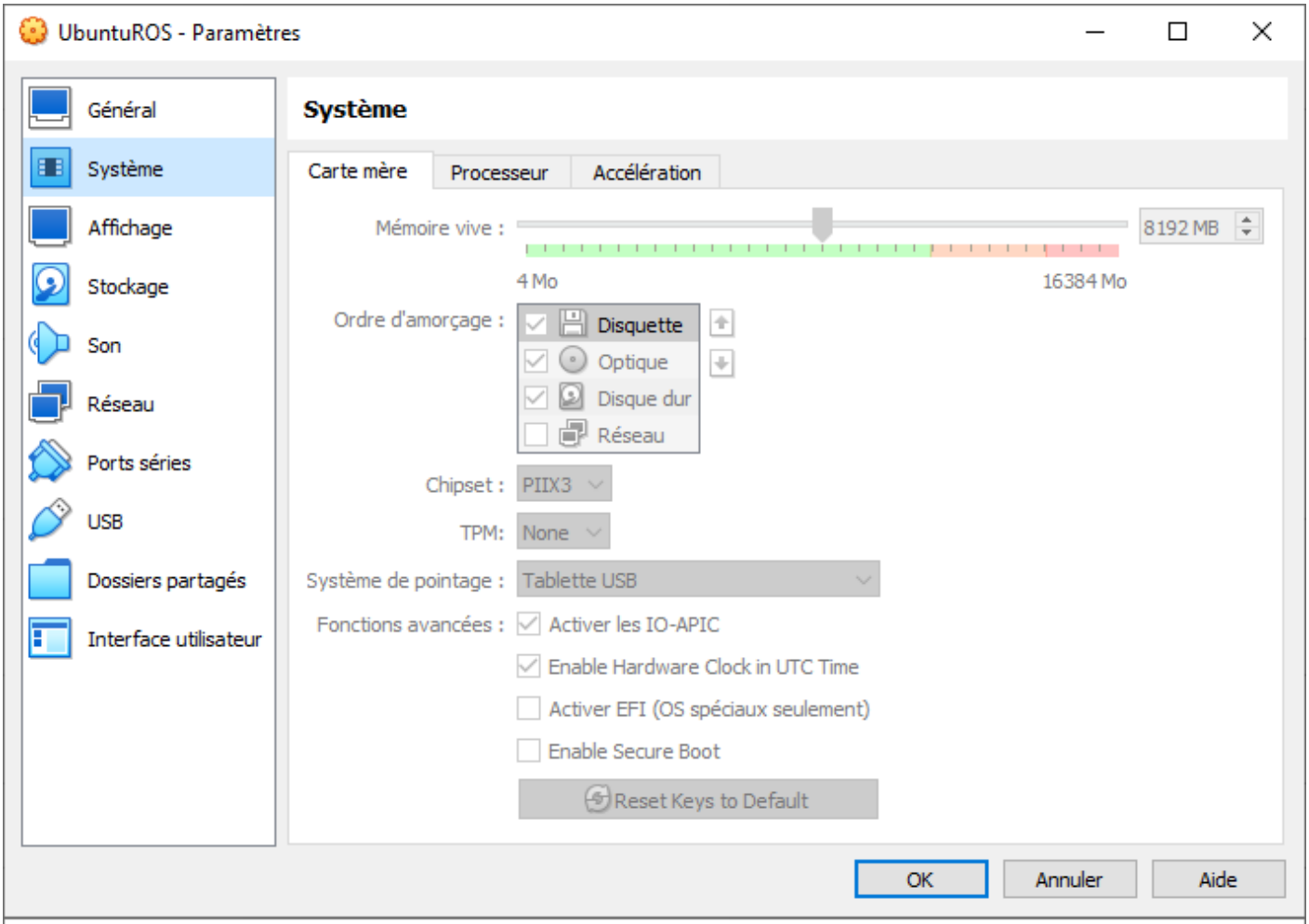
Ubuntu via VirtualBox

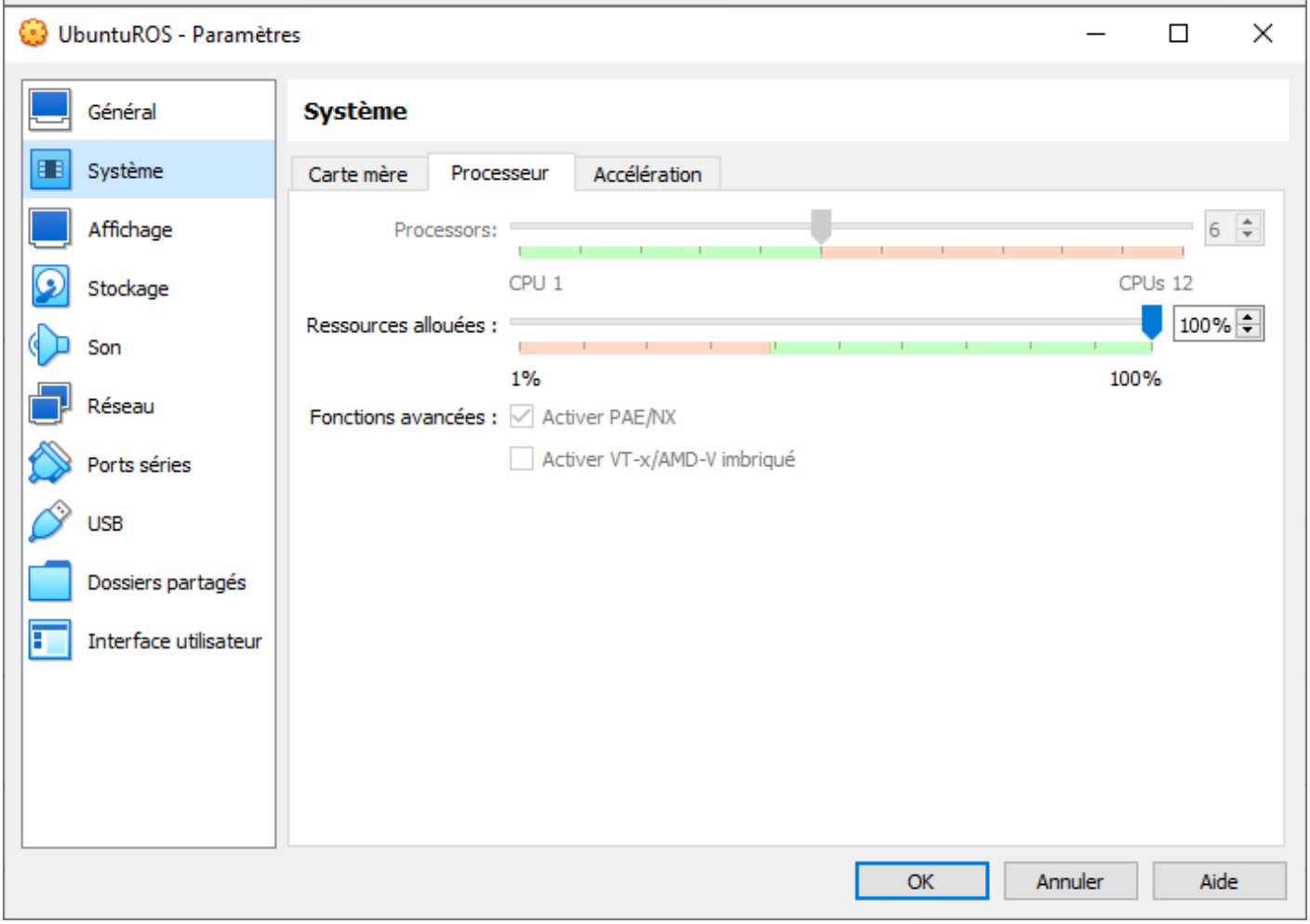
Télécharger et installer VirtualBox pour Windows :

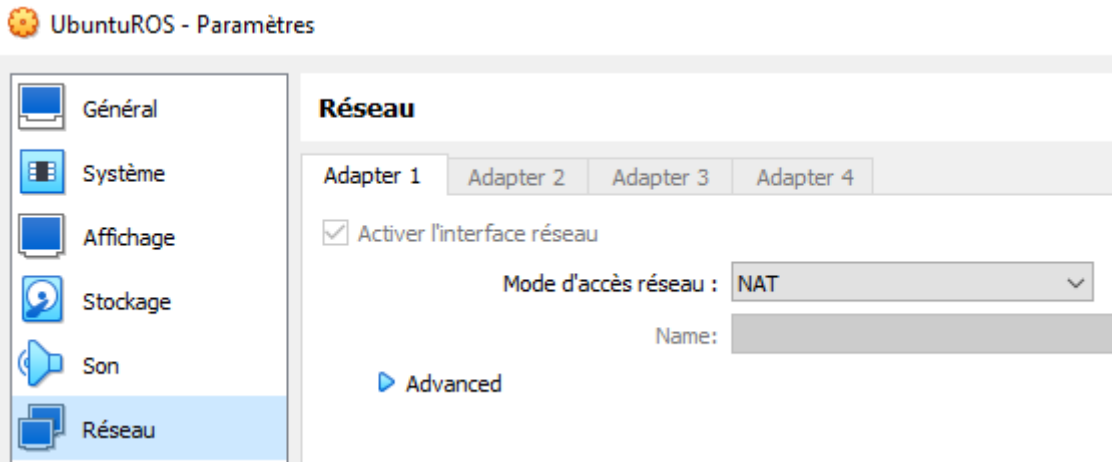
https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.html

- Télécharger la VM depuis seafile (\Seafile\IHA-IDF\Smart_Prod\Formation_ROS2\UbuntuROS.ova)
  - Lien public de téléchargement
    ```

- Lancer VirtualBox
- Importer la VM : Outils -> Importer -> Rechercher le fichier UbuntuROS.ova
- Vérifier et adapter la configuration de la VM en ressources RAM, CPU, GPU et Réseau selon la configuration de votre PC


UbuntuROS - Paramètres
Général
Système
Affichage
Stockage
Son
Réseau
Ports séries
USB
Dossiers partagés
Interface utilisateur
Système
Carte mère
Processeur
Accélération
Mémoire vive :
8192 MB
4 Mo
16384 Mo
Ordre d'amorçage :
Disquette
Optique
Disque dur
Réseau
Chipset :
PIIX3
TPM:
None
Système de pointage :
Tablette USB
Fonctions avancées :
Activer les IO-APIC
Enable Hardware Clock in UTC Time
Activer EFI (OS spéciaux seulement)
Enable Secure Boot
Reset Keys to Default
OK
Annuler
Aide


UbuntuROS - Paramètres
Général
Système
Affichage
Stockage
Son
Réseau
Ports séries
USB
Dossiers partagés
Interface utilisateur
Système
Carte mère
Processeur
Accélération
Processors:
6
CPU 1
CPUs 12
Ressources allouées :
100%
1%
100%
Fonctions avancées :
Activer PAE/NX
Activer VT-x/AMD-V imbriqué
OK
Annuler
Aide

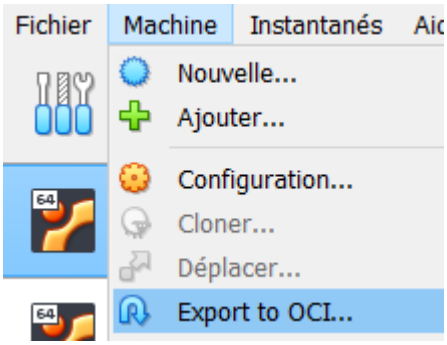


- Démarrer la VM
- Ignorer l'erreur sur le dossier partagé Linux-Windows

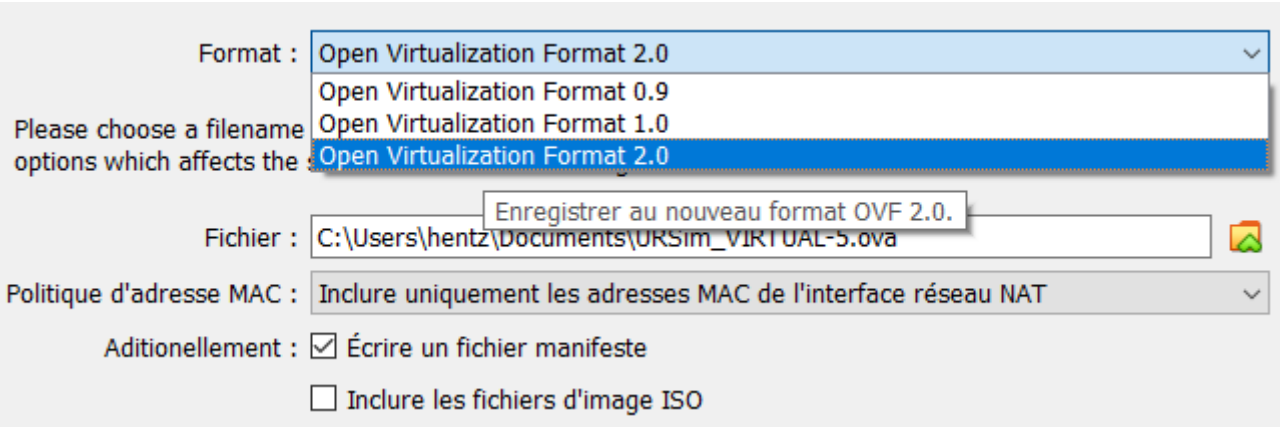
Exportation de VM au format OVF

Le système du TP est maintenu à jour et testé sur un PC Windows. Pour l'exporter sur les PC de salle TP, on veut avoir une image la plus petite possible.

- On commence par nettoyer Ubuntu puis on exporte un fichier `.ova`



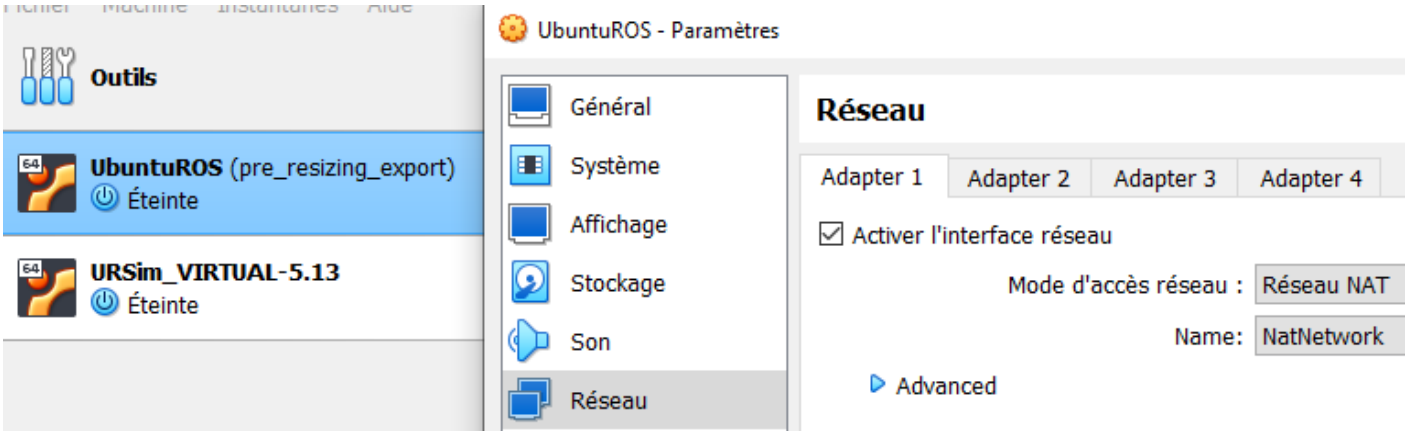
- Sélectionner le format `OVF 2.0` pour une meilleure compressions



Setup pour TP MoveIt2+URSim à l'IUT de Haguenau

La première année j'ai expérimenté avec des PC Windows et VirtualBox :

- Une VM contient Ubuntu 22, ROS et MoveIt
- Une seconde contient Xubuntu 14/16 avec URSim
- Les deux VMs en Réseau NAT



- Voir : https://innovation.iha.unistra.fr/books/robotique-open-source/page/programmer-un-robot-avec-moveit2-jumeau-numerique#bkmrk-sous-windows---virtu
- Il faut des machines de guerre, régler finement la quantité de RAM et de coeurs alloués aux VM et à Windows, et malgré cela les VM plantent.

En 2025 je change donc de fusil d'épaule et utilise la salle réseau de l'IUT :

- Avec des vieilles tours de 2013 : i5, 8G de RAM, petite carte graphique double écran, 60G de SSD
- Réseau isolé donc possibilité de mettre OS au choix sur les PC, d'isoler ou non les PC et d'éventuels robots à piloter

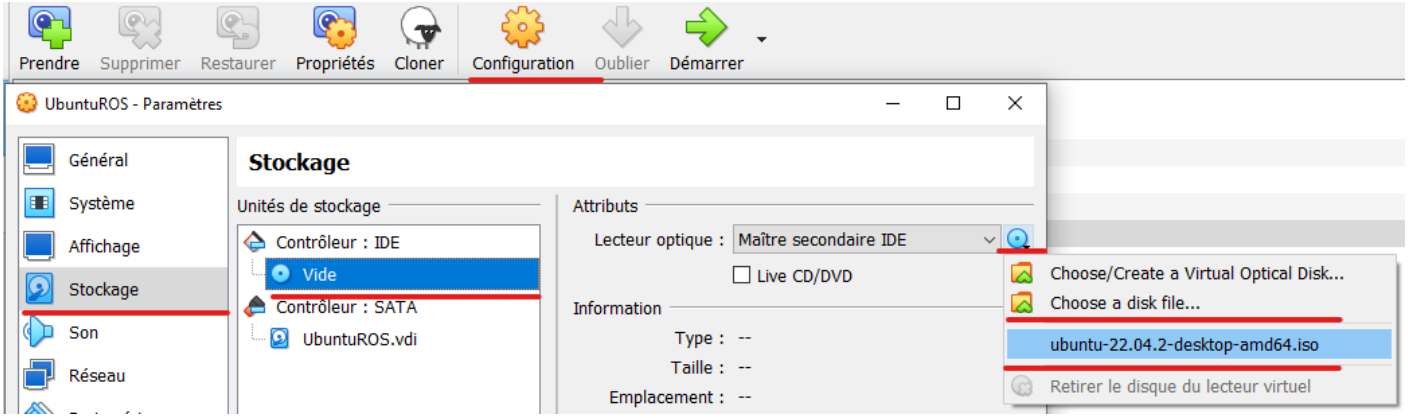
Migration VM vers disque physique

Entre 2024 et 2025 je suis passé de TPs en VM VirtualBox vers des PC physiques. Dans les deux cas, je maintiens l'environnement de TP sur VirtualBox de mon PC Windows. Ceci présente l'avantage de pouvoir maintenir des états de machines en fonction du type de TP.

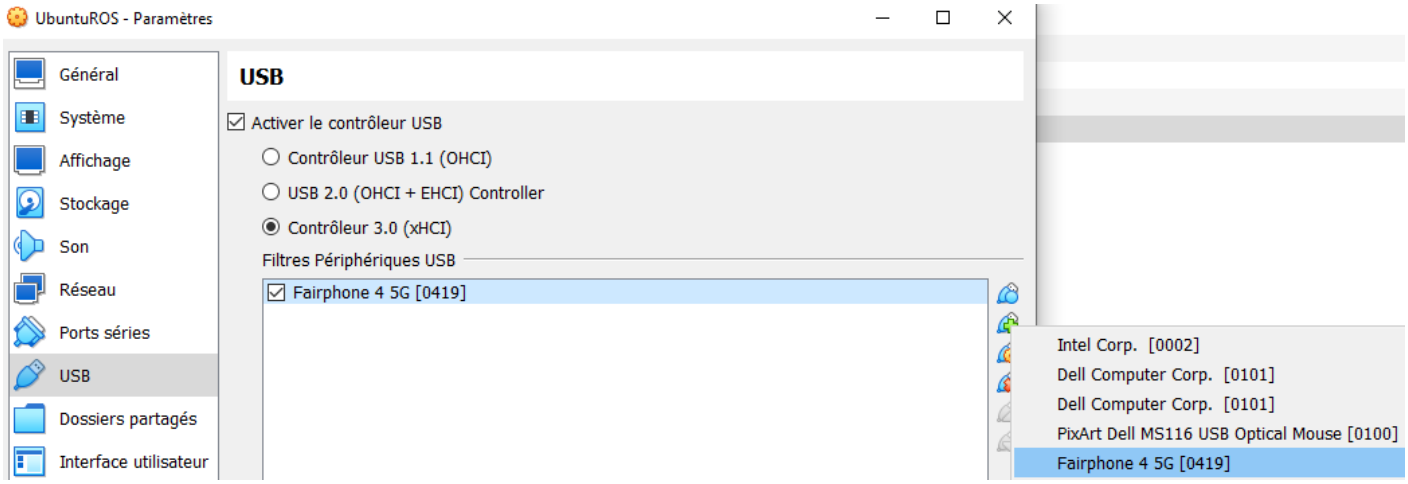
On peut déployer un disque virtuel de VM vers un disque physique :

- Nettoyer l'OS et éventuellement désactiver le SWAP pour encore gagner de l'espace

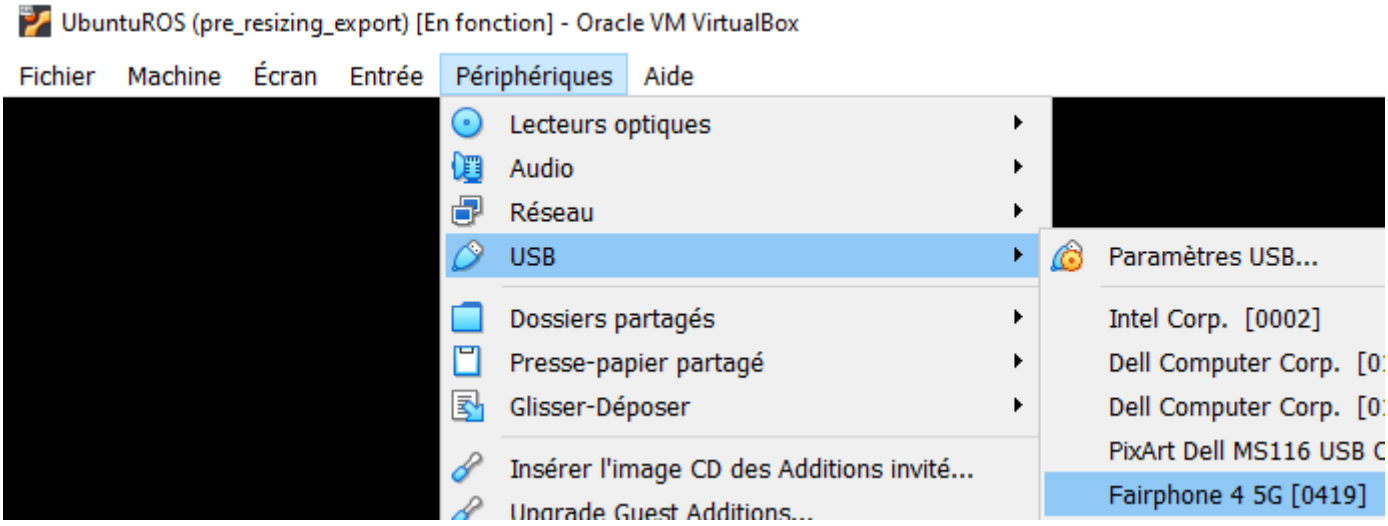
- Réduire la taille de la partition via l'application Disks. Celle-ci pourra être agrandie après copie de la VM. Garder quelques Go de marge.
- Démarrer la VM sur une `.iso` Live Ubuntu



- Brancher un SSD en USB3 au PC (utiliser un adaptateur SATA-USB3 si nécessaire)
- Passer le périphérique USB à la VM
  - Avant le démarrage



- ◦ Pendant que la VM tourne



- Ouvrir l'application Disks pour identifier les disques, en général :
  - disque virtuel de la VM : `/dev/sda`
  - SSD branché en USB : `/dev/sdb`
- Ouvrir un Terminal et lancer la commande de copie du disque virtuel vers le SSD physique :
- `sudo dd if=/dev/sda of=/dev/sdb bs=4096 status=progress && sync`
- Ouvrir Gparted (depuis une Live USB avec le SSD branché) pour vérifier que la partition principale, généralement `sdb3` est bien identifiée comme formatée en `ext4`. Agrandir la partition à la taille désirée.
- Si le SSD ne boot pas sur un PC, essayer de réparer le grub avec `boot-repair` depuis une Live USB

Windows 10/11

Une installation native sous Windows 10 avec Visual Studio 2019 (Version Community gratuite) est possible :

- ROS 1
- ROS 2

Installation de ROS2 Humble

Les distributions stables publiées (pré-compilées) de ROS2 sont nommées par ordre alphabétique.

Début 2023, on va installer **ROS 2 Humble** :

```
sudo apt update && sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8
sudo apt install software-properties-common
sudo add-apt-repository universe
sudo apt update && sudo apt install curl
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o
/usr/share/keyrings/ros-archive-keyring.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-
keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME)
main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
sudo apt update && sudo apt upgrade
```

```
sudo apt install ros-humble-desktop-full
source /opt/ros/humble/setup.bash
echo 'source /opt/ros/humble/setup.bash' >> ~/.bashrc
```

Tester l'installation

https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html#try-some-examples

- Ouvrir un premier Terminal : `ros2 run demo_nodes_cpp talker`
- Ouvrir un second Terminal : `ros2 run demo_nodes_cpp listener`

Installation de Jazzy pour la Navigation et Manipulation

avec UR, Turtlebot3, Nav2, MoveIt2, etc.

```
  1  sudo apt update && sudo apt install locales
    2  sudo locale-gen en_US en_US.UTF-8
    3  sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
    4  export LANG=en_US.UTF-8
    5  sudo apt install software-properties-common
    6  sudo add-apt-repository universe
    7  sudo apt update && sudo apt install curl
    8  sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o
/usr/share/keyrings/ros-archive-keyring.gpg
    9  echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-
keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME)
main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
   10  sudo apt update && sudo apt upgrade
   11  sudo apt install ros-dev-tools
   12  exit
   13  locale
   14  sudo apt install ros-jazzy-desktop-full
   15  echo 'source /opt/ros/jazzy/setup.bash' >> ~/.bashrc
   16  exit
   17  ros2 run demo_nodes_cpp talker~
   18  ros2 run demo_nodes_cpp talker
```

```
   19  exit
   20  ros2 run demo_nodes_cpp listener
   21  exit
   22  sudo apt install docker-compose
   23  sudo usermod -aG docker $USER
   24  sudo service docker start
   25  docker run hello-world
   26  sudo service docker status
   27  docker run hello-world
   28  exit
   29  docker run hello-world
   30  sudo service docker restart
   31  docker run hello-world
   32  sudo usermod -aG docker robot
   33  docker run hello-world
   34  docker pull universalrobots/ursim_e-series
   35  exit
   36  docker run hello-world
   37  sudo service docker restart
   38  docker run hello-world
   39  sudo service docker start
   40  sudo service docker status
   41  docker run hello-world
   42  docker pull universalrobots/ursim_e-series
   43  sudo usermod -aG docker robot
   44  exit
   45  sudo apt install python3-argcomplete python3-colcon-common-extensions libboost-system-
dev build-essential
   46  sudo apt install ros-jazzy-hls-lfcd-lds-driver ros-jazzy-turtlebot3-msgs ros-jazzy-
dynamixel-sdk libudev-dev
   47  mkdir -p ~/turtlebot3_ws/src && cd ~/turtlebot3_ws/src
   48  git clone -b humble-devel https://github.com/ROBOTIS-GIT/turtlebot3.git
   49  git clone -b ros2-devel https://github.com/ROBOTIS-GIT/ld08_driver.git
   50  cd ~/turtlebot3_ws/src/turtlebot3
   51  rm -r turtlebot3_cartographer turtlebot3_navigation2
   52  cd ~/turtlebot3_ws/
   53  echo 'source /opt/ros/humble/setup.bash' >> ~/.bashrc
   54  source ~/.bashrc
   55  cd ..
```

```
   56  sudo nano .bashrc
   57  exit
   58  cd turtlebot3_ws/
   59  source install/setup.bash
   60  ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
   61  ls src/
   62  colcon build --symlink-install
   63  rosdep update && rosdep install --ignore-src --from-paths src -y
   64  vcs --help
   65  vcs status
   66  sudo apt list ros-jazzy-gazebo-ros-pkgs
   67  sudo apt list ros-jazzy-ros-gz
   68  sudo apt install ros-jazzy-ros-gz
   69  colcon build --symlink-install
   70  ros2 launch nav2_bringup tb3_simulation_launch.py slam:=True nav:=True headless:=False
use_sim_time:=True
   71  sudo apt install ros-jazzy-navigation2 ros-jazzy-nav2-bringup
   72  ros2 launch nav2_bringup tb3_simulation_launch.py headless:=False
   73  exit
   74  cd ..
   75  cd turtlebot3_ws/
   76  colcon build --symlink-install --parallel-workers 1
   77  cd src/
   78  git clone -b humble https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
   79  cd ..
   80  colcon build --symlink-install --parallel-workers 1
   81  sudo nano .bashrc
   82  sudo nano ~/.bashrc
   83  source install/setup.bash
   84  ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
   85  sudo apt list ros-jazzy-turtlebot3-*
   86  sudo apt install ros-jazzy-turtlebot3-fake-node
   87  sudo apt install ros-jazzy-gazebo-msgs
   88  cd src/
   89  sudo apt update && rosdep install -r --from-paths . --ignore-src --rosdistro
$ROS_DISTRO -y
   90  rosdep update
   91  sudo apt update && rosdep install -r --from-paths . --ignore-src --rosdistro
$ROS_DISTRO -y
```

```
   92  sudo curl https://packages.osrfoundation.org/gazebo.gpg --output
/usr/share/keyrings/pkgs-osrf-archive-keyring.gpg
   93  echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/pkgs-osrf-
archive-keyring.gpg] http://packages.osrfoundation.org/gazebo/ubuntu-stable $(lsb_release -cs)
main" | sudo tee /etc/apt/sources.list.d/gazebo-stable.list > /dev/null
   94  sudo apt-get update
   95  sudo apt-get install gz-harmonic
   96  cd ..
   97  colcon build --symlink-install --parallel-workers 1
   98  cd ..
   99  ls
  100  mkdir -p ur_ws/src
  101  cd ur_ws/src
  102  cd ..
  103  git clone https://github.com/UniversalRobots/Universal_Robots_ROS2_Tutorials.git
src/ur_tutorials
  104  rosdep update && rosdep install --ignore-src --from-paths src -y
  105  git clone -b ros2
https://github.com/UniversalRobots/Universal_Robots_ROS2_GZ_Simulation.git
src/ur_simulation_gz
  106  rosdep update && rosdep install --ignore-src --from-paths src -y
  107  colcon build --symlink-install
  108  exit
  109  docker run hello-world
  110  docker pull universalrobots/ursim_e-series
  111  ros2 run ur_robot_driver start_ursim.sh -m ur5e
  112  sudo apt install ros-jazzy-ur
  113  sudo apt list python3-rosdep
  114  sudo rosdep init
  115  rosdep update
  116  sudo apt update
  117  sudo apt dist-upgrade
  118  ros2 run ur_robot_driver start_ursim.sh -m ur5e
  119  sudo apt list python3-colcon*
  120  colcon mixin add default https://raw.githubusercontent.com/colcon/colcon-mixin-
repository/master/index.yaml
  121  colcon mixin update default
  122  sudo apt list python3-vcstool
  123  mkdir -p ~/ws_moveit/src
```

```
  124  cd ~/ws_moveit/src
  125  git clone -b main https://github.com/moveit/moveit2_tutorials
  126  vcs import --recursive < moveit2_tutorials/moveit2_tutorials.repos
  127  cd moveit2
  128  git checkout jazzy
  129  ls
  130  ls moveit2.repos
  131  cat moveit2.repos
  132  cd ..
  133  cd src/
  134  sudo apt update && rosdep install -r --from-paths . --ignore-src --rosdistro
$ROS_DISTRO -y
  135  cd ..
  136  colcon build --mixin releasee
```

Installation d'autres versions de ROS2

Pour avoir accès à toutes les dernières fonctionnalités en cours de développement (partiellement publiées), il faut installer **ROS2 Rolling**, qui est une distribution en développement continu "rolling release". Par exemple en Avril 2023, l'API Python de MoveIt2 et son tutoriel ne sont disponibles que sous rolling.

On peut installer plusieurs versions de ros en parallèle. Chaque version sera installée dans `/opt/ros/version`. Pour faire cohabiter les deux versions, il faut "sourcer" le bon répertoire avant de lancer un programme `ros2 launch ...` ou de compiler un workspace `colcon build ...`. Deux options s'offrent à nous :

- Si on bascule souvent de version : commenter les lignes `source /opt/ros/humble/setup.bash` en bas du fichier `~/.bashrc`
  - Il faudra alors lancer la commande `source /opt/ros/humble/setup.bash` **à chaque nouvelle ouverture de Terminal Bash**.
- Si on travaille principalement avec une version : commenter la ligne correspondant à la version principale `source /opt/ros/humble/setup.bash` en bas du fichier `~/.bashrc` lorsqu'on veut utiliser la version secondaire.

Gestion de version avec Ansible

L'idéal serait de gérer l'état des VM/PC de TP avec ansible plutôt que des snapshot VirtualBox

https://github.com/richlamdev/ansible-desktop-ubuntu

Outils utiles

Terminal multi-fenêtres Terminator

- Installer Terminator : c'est un logiciel de Ligne de commande pratique pour programmer avec ROS
  - Depuis Windows Store : Rechercher et installer `Terminator (Ubuntu)`
  - Depuis la ligne de commande Linux : `sudo apt install terminator`
- Depuis le menu démarrer Windows, Lancer `Terminator (Ubuntu)`

Visual Studio Codium

Pour éviter d'alourdir la VM avec de la télémétrie Microsoft, on installe la version sans tracker de Visual Studio Code depuis un dépôt debian :

- Lancer la VM VirtualBox ou WSL (`Terminator (Ubuntu)`)
- Dans Terminator, lancer les commandes suivantes :

```
wget https://gitlab.com/paulcarroty/vscodium-deb-rpm-repo/raw/master/pub.gpg
sudo mv pub.gpg /usr/share/keyrings/vscodium-archive-keyring.asc
echo 'deb [ signed-by=/usr/share/keyrings/vscodium-archive-keyring.asc ]
https://paulcarroty.gitlab.io/vscodium-deb-rpm-repo/debs vscodium main' \
    | sudo tee /etc/apt/sources.list.d/vscodium.list
sudo apt update
sudo apt install codium
```

- Lancer VSCodium dans la VM VirtualBox ou directement depuis Windows, lancer `VSCodium (Ubuntu)`
- Ouvrir le **dossier contenant le code source** `/src` du projet dont vous voulez étudier/modifier le code : `File --> Open Folder --> ~/ws_moveit/src`

Installer Firefox dans WSL

https://askubuntu.com/questions/1444962/how-do-i-install-firefox-in-wsl-when-it-requires-snap-but-snap-doesnt-work

```
sudo snap remove firefox
sudo apt remove firefox
sudo add-apt-repository ppa:mozillateam/ppa

# Create a new file, it should be empty as it opens:
sudo gedit /etc/apt/preferences.d/mozillateamppa

# Insert these lines, then save and exit
Package: firefox*
Pin: release o=LP-PPA-mozillateam
Pin-Priority: 501

# after saving, do
sudo apt update
sudo apt install firefox-esr
```

Alléger Ubuntu (pour VM ou clonage)

Désinstaller snap :

- Vérifier qu'on n'a pas de paquet snap important avec `snap list`
- Purger snap et tous ses paquets

```
sudo rm -rf /var/cache/snapd/

sudo apt autoremove --purge snapd gnome-software-plugin-snap

rm -fr ~/snap

# sudo apt-mark hold snapd
```

- Empêcher snap d'être réinstallé par Ubuntu

```
cat <<EOF | sudo tee /etc/apt/preferences.d/nosnap.pref
# To prevent repository packages from triggering the installation of Snap,
# this file forbids snapd from being installed by APT.
# For more information: https://linuxmint-user-guide.readthedocs.io/en/latest/snap.html
```

```
Package: snapd
Pin: release a=*
Pin-Priority: -10
EOF
```

Installer le magasin d'applications de gnome sans snap/flatpak `sudo apt install gnome-software --no-install-recommends`

Supprimer les paquets apt plus nécessaires `sudo apt autoremove --purge`

Supprimer le cache de compilation de VSCode `~/.cache/vscode-cpptools`

Supprimer les fichiers de compilation des workspaces qui ne seront pas utilisés en TP. Attention à conserver les paquets qui devront être compilés en TP (en utilisant `colcon build --package-select`).

Réduire la taille de la partition via l'application Disks. Celle-ci pourra être agrandie après copie de la VM.

Sources

- Installation de MoveIt2 Humble sur Ubuntu 22.04
- Tutoriels débutant

Auteur: Gauthier Hentz, sur le wiki de l'innovation de l'IUT de Haguenau

Revision #23
Created 31 March 2023 08:42:32 by admin_idf
Updated 2 February 2025 21:56:10 by admin_idf