

# Tutoriels de base

Pour comprendre les concepts de ROS2 par la pratique, il existe des tutoriels pour débutant. Ils reposent sur la simulation d'un robot mobile à deux roues principales développé par les développeurs de ROS en 2010 : [Turtlebot](#). Le [TurtleBot 3 est vendue par Robotis](#) et peut être [couplé à un bras manipulateur 5 axes OpenMANIPULATOR-X](#). Il est possible de [simuler des applications de manipulation mobile avec Gazebo](#).

Pour réaliser les tutoriels de base, il nous faut un environnement de développement, deux options :

- Une machine virtuelle (VM) [VirtualBox](#), ou disponible au téléchargement ici
  - Le plus simple et rapide pour démarrer
  - Si on n'a pas de Hardware ou qu'on ne souhaite travailler qu'en simulation
- Une installation simple ou dual-boot sur un PC
  - Il faut alors installer de zéro
  - Indispensable si on veut travailler avec du Hardware

Supposons que vous avez [installé et testé votre environnement comme celui de la VM](#).

Connexion à la VM

- nom : ubuntu22ros2
- utilisateur :
- mdp : département de l'IUT

Nous pouvons directement passer aux tutoriels sur les outils ROS 2 en ligne de commande :

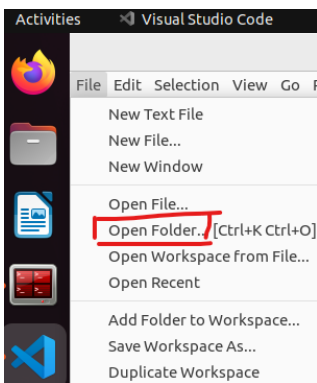
- [Configuring environment](#)
- [Using turtlesim, ros2, and rqt](#)
- [Understanding nodes](#)
- [Understanding topics](#)
- [Understanding services](#)
- [Understanding parameters](#)

Ces premiers tutoriels ne nécessitent qu'une installation basique de ROS 2, on n'y regarde pas le code source.

Ensuite on passe aux tutoriels sur les bibliothèques clientes de ROS 2 en C++ et Python :

- Using `colcon` to build packages
- Creating a workspace
- Creating a package
- Writing a simple publisher and subscriber (C++) --> Correction
- Writing a simple publisher and subscriber (Python)
- Writing a simple service and client (C++)
- Writing a simple service and client (Python)
- Creating custom msg and srv files

Ces tutoriels vous engagent à copier et analyser du code source en C++ et Python. Les fichiers créés sont placés dans le dossier de travail (workspace) `/home/etudiant/ros2_ws/src`, à ouvrir avec Visual Studio Code :



Vous trouverez des fichiers de correction commentés dans `ros2_ws/src/cpp_pubsub/src/`, en particulier :

- `publisher_member_function.cpp`
- `subscriber_member_function.cpp`

Pour les tester il faut lancer :

```
cd ~/ros2_ws
colcon build --packages-select cpp_pubsub
source install/setup.bash
ros2 run cpp_pubsub talker
```

Le noeud se met à publier/parler :

```
[INFO] [minimal_publisher]: Publishing: "Hello World: 0"
[INFO] [minimal_publisher]: Publishing: "Hello World: 1"
[INFO] [minimal_publisher]: Publishing: "Hello World: 2"
[INFO] [minimal_publisher]: Publishing: "Hello World: 3"
[INFO] [minimal_publisher]: Publishing: "Hello World: 4"
```

Puis dans un second Terminal :

```
ros2 run cpp_pubsub listener
```

Le noeud se met à écouter :

```
[INFO] [minimal_subscriber]: I heard: "Hello World: 10"  
[INFO] [minimal_subscriber]: I heard: "Hello World: 11"  
[INFO] [minimal_subscriber]: I heard: "Hello World: 12"  
[INFO] [minimal_subscriber]: I heard: "Hello World: 13"  
[INFO] [minimal_subscriber]: I heard: "Hello World: 14"
```

Tapez `Ctrl+C` dans chaque Terminal pour arrêter les noeuds ("stop spinning").

-----  
[Auteur: Gauthier Hentz](#), sur le [wiki de l'innovation de l'IUT de Haguenau](#)

Attribution-NonCommercial-PartageMemeConditions 4.0 International (CC BY-NC-SA 4.0)

---

Revision #3

Created 24 November 2023 14:43:47 by admin\_idf

Updated 11 September 2024 15:31:09 by admin\_idf