Exercices ROS 2 – Workspace

1. Introduction

Le but de l'exercice est de créer un premier workspace et de voir l'utilisation des outils de compilation et le fonctionnement des workspace.

Il comprend :

- La création d'un workspace
- La création d'un package « vide » dans ce workspace
- L'utilisation de colcon pour compiler des sources
- La copie d'un package depuis les sources et sa compilation

- Les commandes à effectuer dans un terminal sont indiqué par le symbole $

```
$ example de commande      // un commentaire sur la ligne de commande
```

- Les **DISTRO** sont à remplacer par votre version de ROS2 :

```
Par exemple sur ROS2 Humble (Ubuntu 20.04):
$ source /opt/ros/**DISTRO**/setup.bash ### devient :
$ source /opt/ros/humble/setup.bash
```

2. Utilisation du workspace ROS « par défaut »

Un workspace est un ensemble de fichier et de variables d'environnement permettant de manipuler et d'utiliser les briques élémentaires du système ROS que sont les packages.

Il faut commencer par ouvrir un terminal : CTRL + T ou utiliser l'application « terminator »

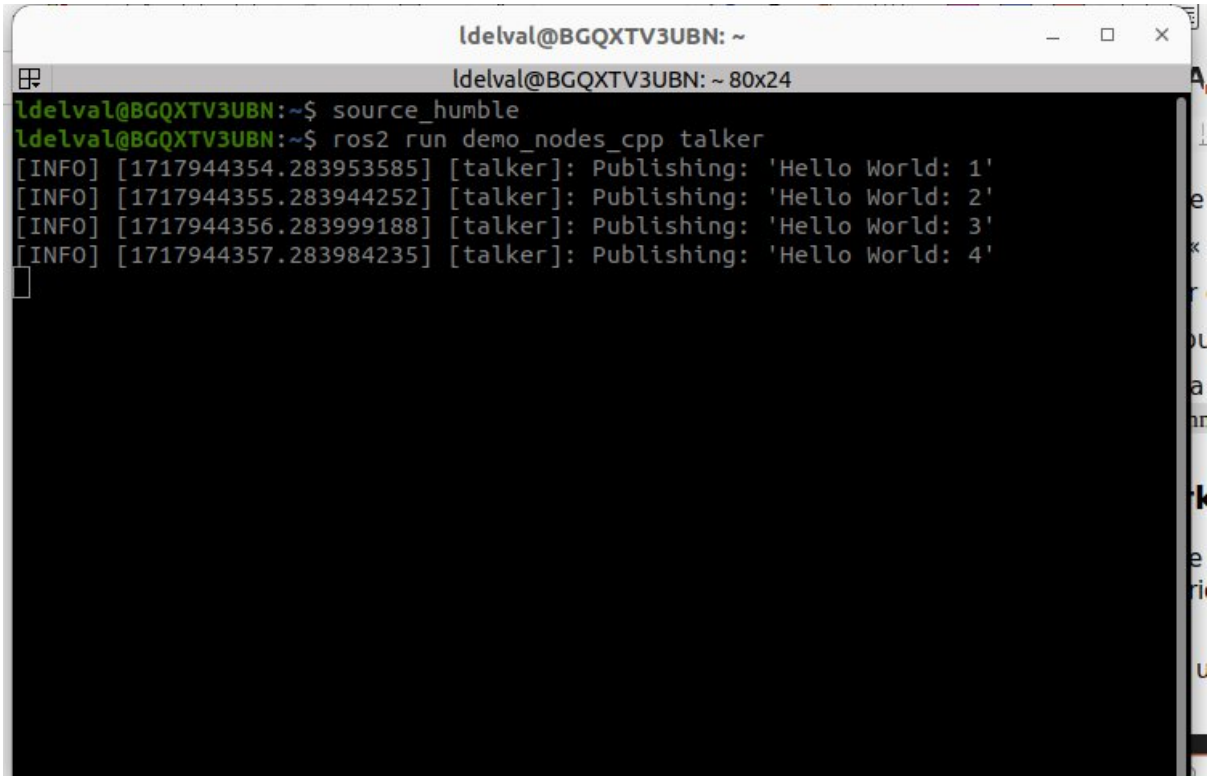
La première étape est de charger les variables d'environnement ROS « par défaut ».

```
$ source /opt/ros/**DISTRO**/setup.bash
```

Celà permet de charger les variables d'environnement du workspace de ROS qui contient tous les éléments de ROS installés « précompilés ».

Vous pouvez vérifier que vous avez accès au commande ROS en utilisant la commande :

```
$ ros2 run demo_nodes_cpp talker
```



Dans un autre terminal, utilisez la commande ROS :

```
$ ros2 run demo_nodes_cpp talker
```

La commande ne fonctionne pas car les variables d'envionnements ne sont pas charger dans ce terminal.

Exercice : Faite fonctionner le talker dans 2 terminaux.

Essayer maintenant un talker et un listener

```
$ ros2 run demo_nodes_cpp listener
```

3. Création d'un nouveau workspace

Ouvrez un nouveau terminal.

Un workspace ROS n'est qu'ensemble de dossier. Il y a deux dossiers à créer « à la main ». Le premier est le **dossier racine** :

```
$ cd                    // Pour revenir à la racine du dossier « HOME »
$ mkdir my_ros_ws // pour créer un dossier vide nommer my_ros_ws
$ cd my_ros_ws  // pour ce déplacer dans ce dossier
```

Le second est le dossier qui contiendra les sources des packages de ce workspace, il a pour nom **src** : (Vérifier bien que vous êtes dans le dossier **racine)**

```
$ mkdir src
```

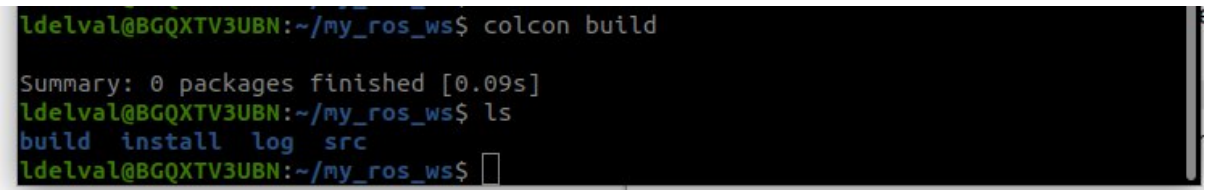
Félicitations, vous venez de créer votre premier workspace ROS.

Vous pouvez compiler l'ensemble des packages (à savoir aucun pour l'instant) avec la commande (vérifier bien que vous êtes dans le dossier racine et pas dans le dossier src ou ailleurs !!) :

```
$ source /opt/ros/**DISTRO**/setup.bash
$ colcon build            // compile et génére les fichier pour le workspace
```

On peut voir les fichiers générer par la commande :

```
$ ls       // liste les fichiers et dossiers du dossiers courant
```



4. Création d'un package "vide"

ROS 2 fournit un outil de génération de package vide. La première étape est de se rendre dans le dossier src du workspace.

```
$ cd ~/my_ros_ws/src       // on se rend dans le dossier src du workspace
```

Pour en savoir plus sur la commande de création de package la commande :

```
$ ros2 pkg create –help
```

Nous allons créer un package vide contenant un noeud simple :

```
$ ros2 pkg create --build-type ament_cmake --node-name my_node my_package
```

```
ldelval@BGQXTV3UBN:~/my_ros_ws/src$ ros2 pkg create --build-type ament_cmake --node-name my_node my_package
going to create a new package
package name: my_package
destination directory: /home/ldelval/my_ros_ws/src
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['ldelval <ludovic.delval@irt-jules-verne.fr>']
licenses: ['TODO: License declaration']
build type: ament_cmake
dependencies: []
node_name: my_node
creating folder ./my_package
creating ./my_package/package.xml
creating source and include folder
creating folder ./my_package/src
creating folder ./my_package/include/my_package
creating ./my_package/CMakeLists.txt
creating ./my_package/src/my_node.cpp

[WARNING]: Unknown license 'TODO: License declaration'.  This has been set in the package.xml, but no LICENSE file has been created.
It is recommended to use one of the ament license identitifers:
PickNik
Apache-2.0
BSL-1.0
BSD-2.0
BSD-2-Clause
BSD-3-Clause
GPL-3.0-only
LGPL-3.0-only
MIT
MIT-0
```

Si vous voulez regarder les différents fichiers générés, vous pouvez utiliser l'IDE, vscode avec la commande

```
$ code .
```

Il faut maintenant compiler le workspace pour utiliser l'exécutable dans le package.

```
$ cd ~/my_ros_ws     // On retourne à la racine du workspace
$ colcon build
```

```
ldelval@BGQXTV3UBN:~/my_ros_ws$ colcon build
Starting >>> my_package
Finished <<< my_package [0.87s]

Summary: 1 package finished [0.96s]
ldelval@BGQXTV3UBN:~/my_ros_ws$
```

Notre workspace est maintenant compilé.

Il ne reste plus qu'à charger les variables d'environnements de ce workspace.

```
$ cd ~/my_ros_ws     // On retourne à la racine du workspace
$  source install/setup.bash
```

Il est maintenant possible de lancer un nouveau nœud :

```
$ ros2 run my_package my_node
```

```
ldelval@BGQXTV3UBN:~/my_ros_ws$ ros2 run my_package my_node
hello world my_package package
ldelval@BGQXTV3UBN:~/my_ros_ws$
```

Nous allons maintenant copier un package (comme si nous le téléchagions de github ou autre) dans le workspace.

5. Copie d'un package

Pour copier les sources :

```
$ cd
$ git clone https://github.com/ros2/demos.git -b **DISTRO**
$ cp -r ~/demos/demo_nodes_py ~/my_ros_ws/src      // On copie tout le dossier dans un autre dossier
```

EXERCICE : Compilez le workspace et utiliser les nœuds depuis ce workspace

Talker et Listener (sur deux terminaux différents).

(N'oubliez pas les variables d'environnements)

Note : Pour la suite du workshop. Afin de ne pas oublier de charger les variables d'environnement, nous allons automatiquement ajouter la commande dans un script.

```
$ echo "source /opt/ros/**DISTRO**/setup.bash" >> ~/.bashrc
$ echo "source /my_ros_ws/install/setup.bash" >> ~/.bashrc
```