ROS Installation

Installation Instruction

- This Document assumes that the reader has installed *Ubuntu 18.04*. However, if you haven't installed *Ubuntu 18.04* yet make sure to install it before proceeding. There are tons of resources available on the Internet to get this done.
- You can download Ubuntu 18.04 ISO file from here.

ROS Melodic Installation

ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license.

Here the distribution compatible with *Ubuntu 18.04* is the *ROS Melodic Morenia*. Follow the steps below to install *ROS Melodic*

Installation Steps

1. Setup your computer to accept software from packages.ros.org.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
$(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

2. Set up your keys.

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80'
--recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

3. Make sure your Debian package index is up-to-date.

```
sudo apt update
```

4. Installing the ROS recommended configuration.

```
sudo apt install ros-melodic-desktop-full
```

Configuration Steps

1. Adding environment variables: To Automatically add ROS environment variables to your bash session every time a new shell (terminal) is launched, enter the following commands (this step is similar as adding environmental variable in windows):

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

2.Initialize rosdep: Before you can use many ROS tools, you will need to initialize rosdep. rosdep enables you to easily install system dependencies for sources you want to compile and is required to run some core components in ROS.

```
sudo apt install python-rosdep
```

```
sudo apt install python-rosdep
```

```
rosdep update
```

More Packages to install

1. Catkin Tools

```
sudo apt-get install ros-melodic-catkin python-catkin-tools
```

2. std_msg package

```
sudo apt install ros-melodic-std-msgs
```

3. turtlesim

```
sudo apt-get install ros-melodic-ros-tutorials
```

Catkin Workspace

- catkin is the official build system of ROS and the successor to the original ROS build system, rosbuild.

- catkin combines CMake macros and Python scripts to provide some functionality on top of CMake's normal workflow.

- catkin was designed to be more conventional than rosbuild, allowing for better distribution of packages, better cross-compiling support, and better portability.

src

- The src folder contains the source code of catkin packages. This is where you can extract/checkout/clone source code for the packages you want to build.

- Each folder within the src folder contains one or more catkin packages. This folder should remain unchanged by configuring, building, or installing.

- The root of the src folder contains a symbolic link to catkin's boiler-plate 'toplevel' CMakeLists.txt file. This file is invoked by CMake during the configuration of the catkin projects in the workspace. It can be created by calling catkin_init_workspace in the src folder directory. When we execute the catkin_make command from the workspace folder, it checks inside the src folder and builds each package.

build

- The build folder is where CMake is invoked to build the catkin packages in the src folder.

- CMake and catkin keep their cache information and other intermediate files here.

- The build folder does not have to be contained within the workspace nor does it have to be outside of the src folder, but this is recommended.

devel

- The development folder (or devel folder) is where built targets are placed before being installed.
- The way targets are organized in the devel folder is the same as their layout when they are installed.
- This provides a useful testing and development environment which does not require invoking the installation step.
- The location of the devel folder is controlled by a catkin specific CMake variable called CATKIN_DEVEL_PREFIX, and it defaults to build/devel folder.
- This is the default behavior because it might be confusing to CMake users if they invoked CMake in a build folder and that modified things outside of the current directory.
- It is recommended, however, to set the devel folder directory to be a peer of the build folder directory.

```
source ~/<workspace_name>/devel/setup.bash
```

Create a Catkin Workspace

1. Open up the terminal.

2. Create the root workspace directory. You can name your directory anything but by ROS convention we will use catkin_ws as the name.

```
cd ~/
```

```
mkdir --parents catkin_ws/src
```

```
cd catkin_ws
```

3. Initialize the catkin workspace.

```
catkin init
```

*Look for the statement "**Workspace configuration appears valid**", showing that your catkin workspace was created successfully. If you forgot to create the src directory, or did not run catkin init from the workspace root (both common mistakes), you'll get an error message like "**WARNING: Source space does not yet exist**".*

4. Build the workspace.

```
cd ~/catkin_ws
```

```
catkin build
```

5. Now your catkin workspace will have additional directories build, devel, logs.

```
ls
```

6. Now to make your workspace visible to ROS. Source the setup file in the devel directory.

```
source ~/catkin_ws/devel/setup.bash
```

By doing this, all the packages that you create inside the src folder will be visible to ROS.

7. This setup.bash file of your workspace must be source every time when you want to use ROS packages created inside this workspace.

To save typing, add this to your .bashrc,

1. gedit ~/.bashrc
2. Add to the end: source ~/catkin_ws/devel/setup.bash
3. Save and close the editor.