# ROS Noetic. Basic tutorial

## 1- SETTING UP THE ROS ENVIRONMENT

a. **Open a terminal in Ubuntu, you can use the key shortcut Ctrl + Alt + T, if wanted.**

b. **Type the following instruction:**

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

**Step 1 should be made only once per ROS installation.**

## 2- CREATE A WORKSPACE

a. **Open a terminal in Ubuntu, you can use the key shortcut Ctrl + Alt + T, if wanted (Ctrl + Command + T, for Mac).**

b. **Create a new folder (directory) using the following terminal command:**

```
mkdir –p class_ws/src
```

c. **Enter the directory using the terminal command:**

```
cd class_ws
```

d. **Type the terminal command**

```
catkin_make
```

**The catkin_make instruction would compile the directory and define it as a ROS workspace.**

## 3- CREATE A ROS PACKAGE

a. **Move to the src folder of the *class_ws* workspace**

   i. **If you are still in the terminal where you created the workspace, type:**

   ```
   cd src
   ```

   ii. **Otherwise, type:**

cd ~/class_ws/src

b. **Create a new ROS package with the following command:**
   catkin_create_pkg my_first_pkg roscpp std_msgs geometry_msgs

***NOTE: catkin_create_pkg is the command that tells the system you will create a new ROS package, my_first_pkg is the name of the package, while roscpp, std_msgs and geometry_msgs are the package's dependencies, where:**

***- roscpp states that the nodes or codes of the project are implemented in C++ (you can add rospy if using python instead).***

***- std_msgs states that we are using ROS messages from the std_msgs library.***

***- geometry_msgs states that we are using ROS messages from the geometry_msgs library.***

c. **Move back to the class_ws location using the instruction:**
   cd ..
d. **Compile your new ROS package typing:**
   catkin_make
e. **Move again to the src folder of the workspace**
   cd src
f. **Type the command to see all the files of the src folder**
   ls
g. **You will see my_first_pkg in dark blue. Enter to the package with**
   cd my_first_pkg
h. **See the files within the ROS package folder. You should see:**
   i. **The include folder, which is used to store the local header files of the project.**

ii. **The src folder, where you must store the designed codes (ROS nodes)**
iii. **The CMakeLists.txt and package.xml files, which are the configuration documents of the ROS Package.**

i. **Close the terminal**

**4- Setup the ROS Package environment**

a. **Open a terminal in Ubuntu, you can use the key shortcut Ctrl + Alt + T, if wanted.**
b. **Type the following instruction:**

```
echo "source /home/UserName/class_ws/devel/setup.bash" >> ~/.bashrc
```

***NOTE: UserName is the username you typed when installing Ubuntu.***

```
source ~/.bashrc
```

c. **Close your terminal and open a new one. Then type:**

```
rospack list
```

**You will see a list in alphabetical order showing all the available packages to work with. If everything is OK, you should be able to find the ROS package you just created, (my_first_pkg, for this tutorial).**

**5- Create your first ROS node.**

a. **Move to the ROS package folder**

```
cd ~/class_ws/src/my_first_pkg
```

b. **Open Visual Studio Code from terminal**

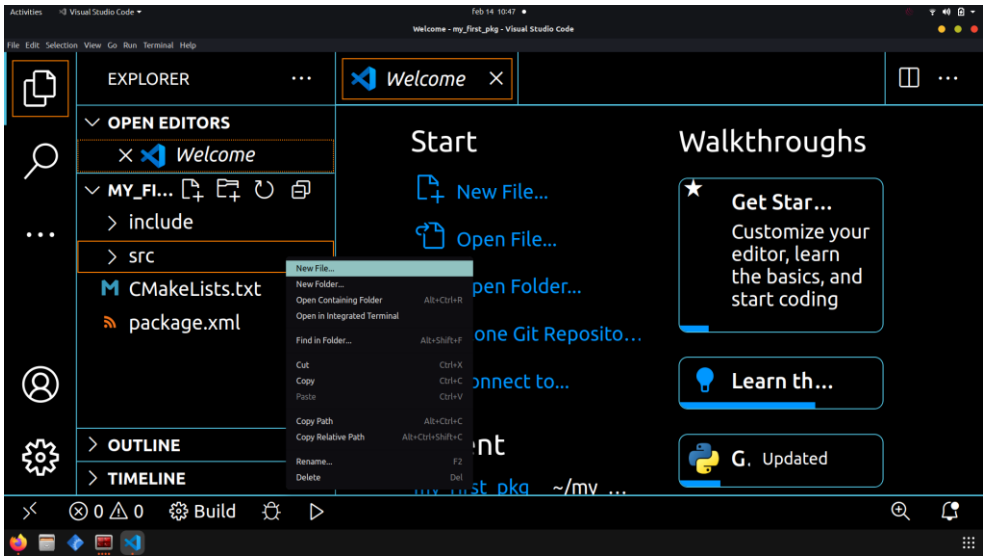
```
code .
```

c. **Create a new file doing the following:**

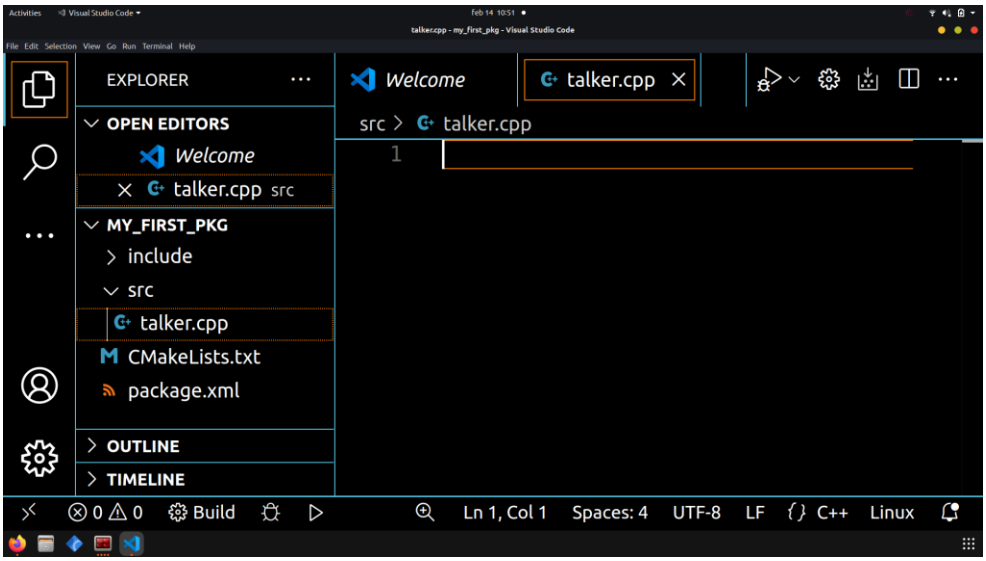
i. **Move your cursor to src**

ii. **Right-click**

iii. **Select New File**




d. **Name the file as talker.cpp**




**e. Code your program and save it with Ctrl + S .**

**6- Compiling a C++ Code**

a. **In VS Code, open the CMakeLists.txt file that is within the src folder of the ROS package.**

b. **Go to the last line of the file and press the ENTER key**
c. **Type the following instructions:**

```
add_executable(talker_node src/talker.cpp)
target_link_libraries(talker_node ${catkin_LIBRARIES})
```

**where:**

**talker_node is the Node ID of the talker.cpp code, that is, the ROS environment will recognize the talker.cpp program as talker_node.**

d. **Open a new terminal in Ubuntu, you can use the key shortcut Ctrl + Alt + T, if wanted (Ctrl + Command + T, for Mac).**
e. **Go to the workspace folder and compile:**

```
cd ~/class_ws
catkin_make
```

**If everything is correct, you'll see the 100% Built message. Otherwise, if there is an error in your code, DO NOT PANIC!!! The terminal will show you a message telling you what and where the error is.**

**7- Executing a single ROS Node**

a. **Initialize the ROS Master by typing in a terminal:**

```
roscore
```

b. **In a new terminal, execute the following command:**

```
rosrun my_first_pkg talker_node
```

c. **In a new terminal run the command:**

```
rostopic list
```

**You will see all the available topics in the ROS environment**

d. **Now, execute the following instruction in the terminal**

```
rostopic echo text2show
```

**You will see the message allocated at the data space, that is being published from talker.cpp**

e. **Move back to the terminal that is executing the node, and stop the process by pressing Ctrl + C.**

**8- Install the Eigen 3 library**

a. **Open a new terminal and type the following commands:**

sudo apt update

sudo apt install libeigen3-dev

**9- Setup the Eigen 3 library in your ROS project**

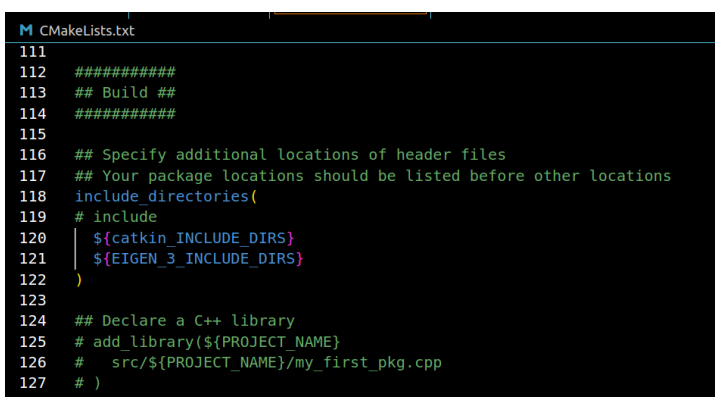
a. **Go to VSCode and open the CMakeLists.txt file within the package folder.**

b. **Go to line 118 where the function** include_directories(...) **is located.**

c. **Paste the following instruction within this function:**

${EIGEN_3_INCLUDE_DIRS}

**Example**




**10- Install Plotjuggler**

a. **Open a new terminal and type the following commands:**

sudo apt update

sudo apt install ros-noetic-plotjuggler

**b. In the same terminal, open plotjuggler using the instruction:**

rosrun plotjuggler plotjuggler