ROS2 – 10 เขียน Node ภาษา C++ เบื้องต้น



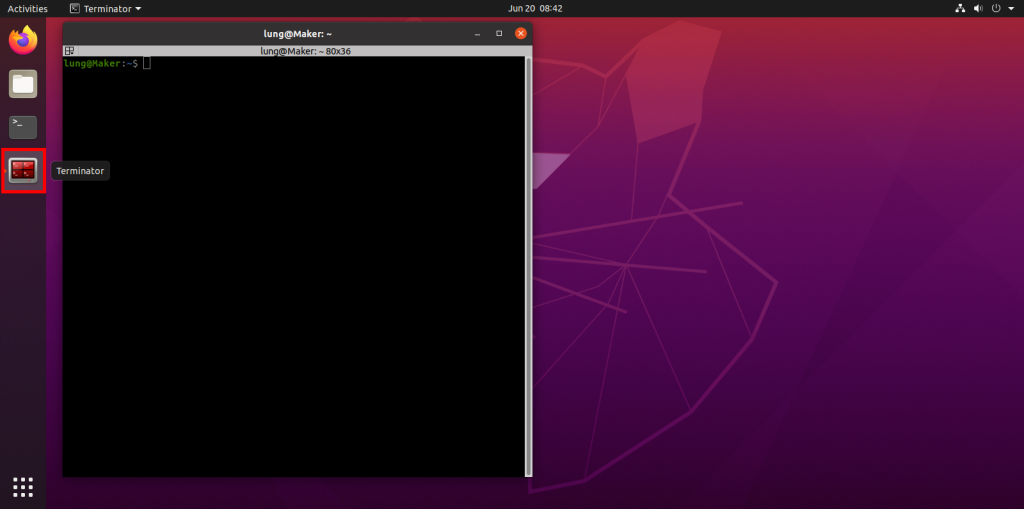
เขียน Node ภาษา C++ เบื้องต้น

การทำงานของ ROS จะแบ่งเป็น Node แต่ละ Node จะทำหน้าที่เฉพาะเช่น Node เซนเซอร์อุณหภูมิ, Node เซนเซอร์ความเร็ว, Node ควบคุมมอเตอร์, Node Lidar เป็นต้น แต่ละ Node จะประสานข้อมูลกันผ่าน Topic เช่นเซนเซอร์ Encoder จะ Publish ข้อมูลความเร็ว เข้าไปใน Topic Pool อุปกรณ์พวก Actuator เช่นมอเตอร์จะ Subscribe เพื่อดึงข้อมูลความเร็วไปประมวลผลและใช้งาน อาจจะเป็นการสั่งให้มอเตอร์หมุนเร็วหรือช้าขึ้น แต่ละวินาทีจะมีข้อมูล Publish หรือ Subscribe จำนวนมาก

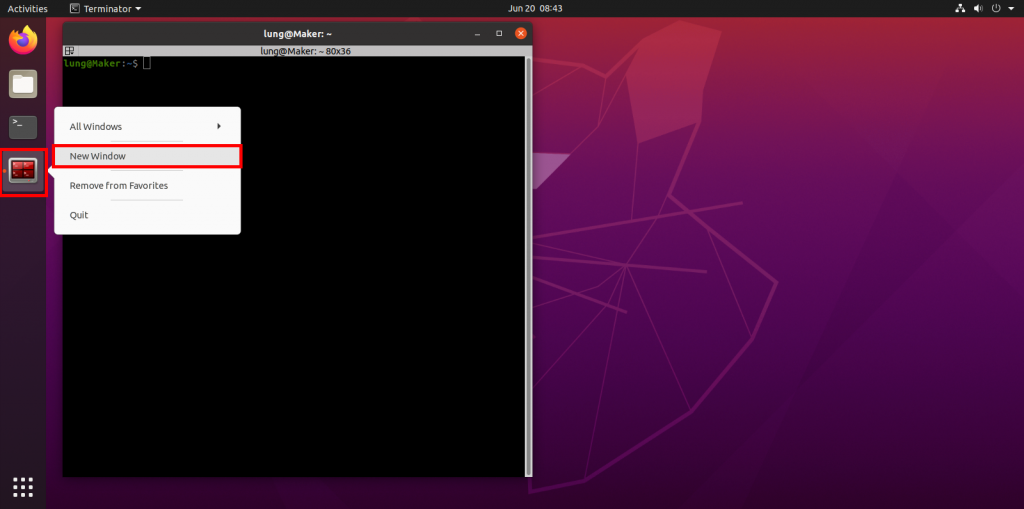
**1 : สร้าง Package ภาษา C++**  
สร้าง Package ภาษา C++ ตามขั้นตอนลิงค์ด้านล่าง

* [สร้าง Package ภาษา C++](http://www.lungmaker.com/ros2-7-%e0%b8%aa%e0%b8%a3%e0%b9%89%e0%b8%b2%e0%b8%87-package-%e0%b8%a0%e0%b8%b2%e0%b8%a9%e0%b8%b2-c/)

**2 : เขียน Node ภาษา C++ เบื้องต้น**  
เริ่มการทำงานโดย เปิด Terminator หน้าต่างที่ 1



คลิกขวาที่ Terminator -> New Windows



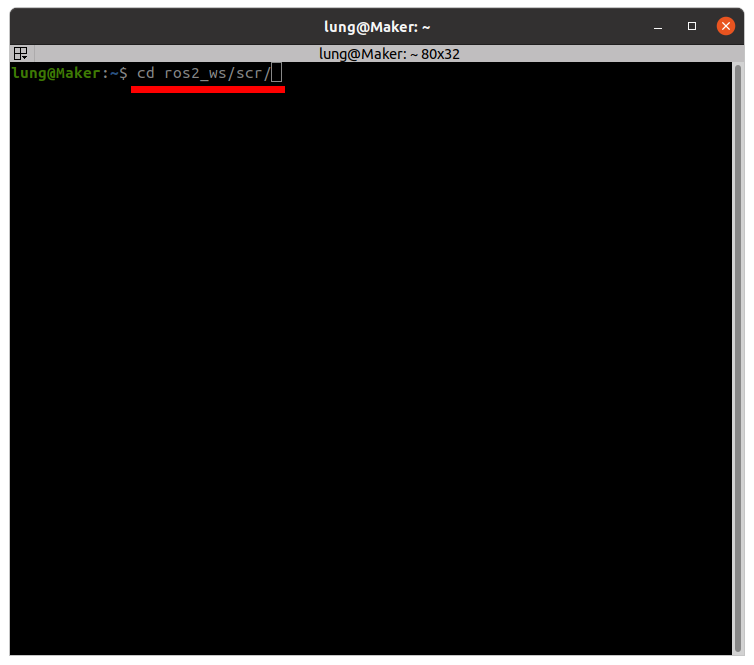
จะได้หน้าต่างที่ 2



เริ่มเขียนคำสั่ง ที่หน้าต่างที่ 1

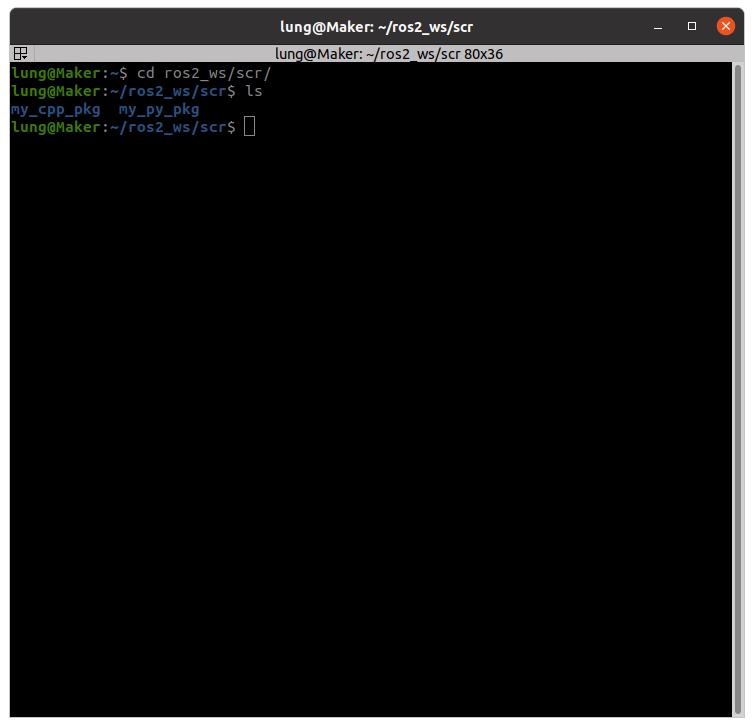
เริ่มสร้าง Node โดยเข้าไปที่โฟลเดอร์ของ Workspace ของ ROS2 โดยเราจะสร้าง Node ไว้ใน ros2\_ws/scr/

cd ros2\_ws/scr/



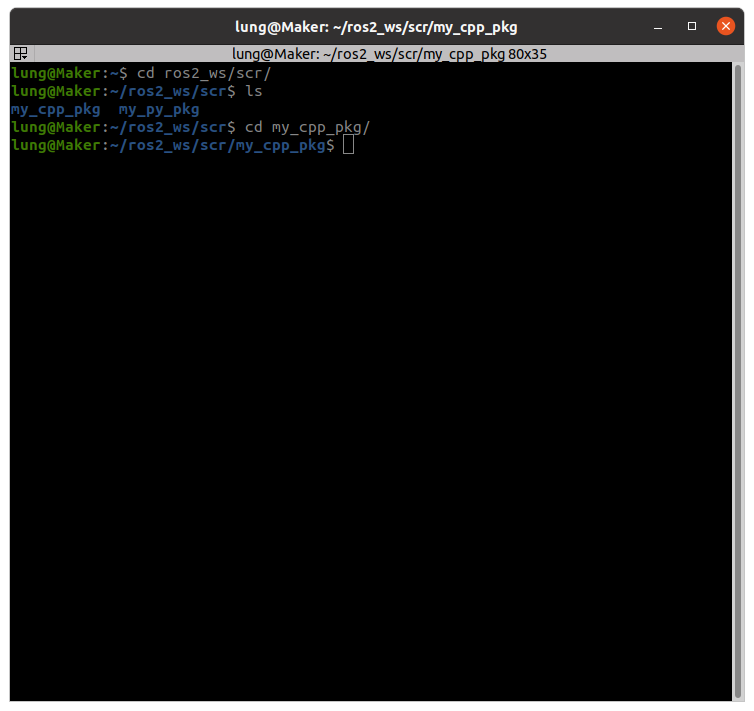
ดูรายชื่อไฟล์และ โฟลเดอร์ จะพบโฟลเดอร์ my\_py\_pkg และ my\_cpp\_pkg ซึ่งเป็น Package ที่ได้สร้างไว้ก่อนหน้านี้

ls



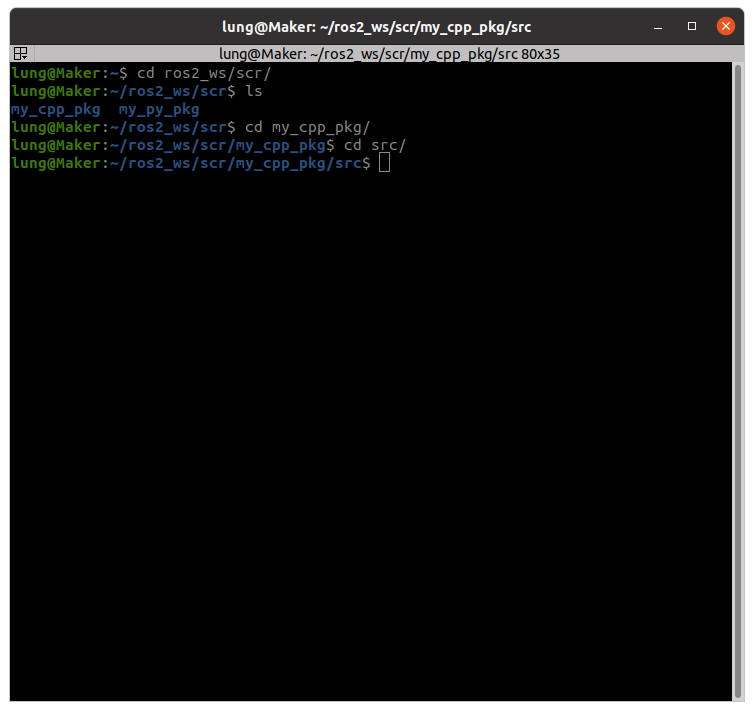
เข้าไปใน โฟลเดอร์ my\_cpp\_pkg

cd my\_cpp\_pkg/



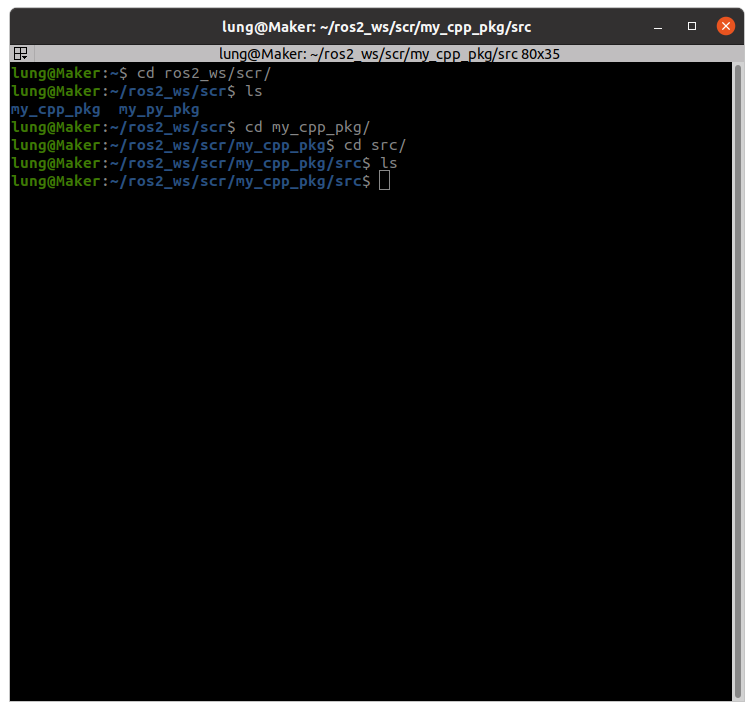
เข้าไปใน โฟลเดอร์ src

cd src/



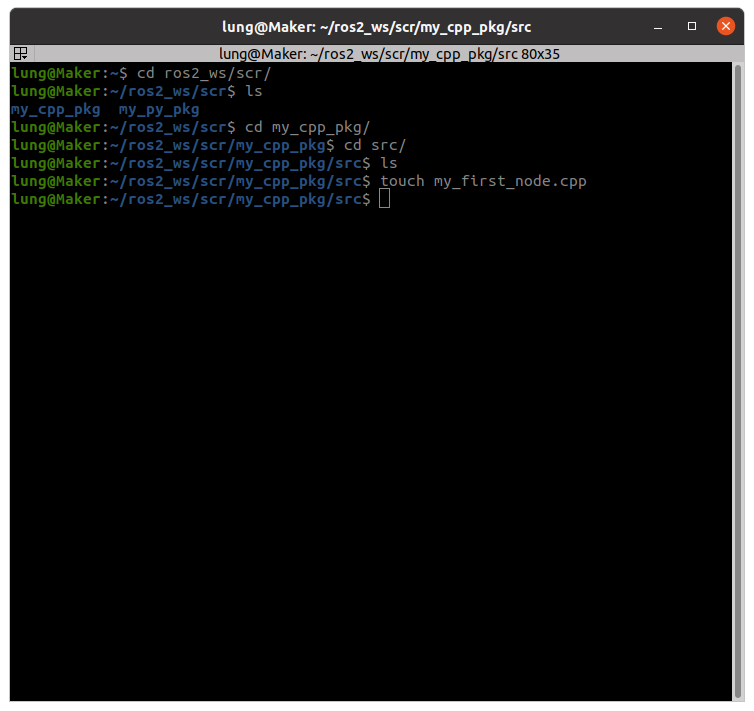
ดูรายชื่อไฟล์และ โฟลเดอร์ จะไม่พบ โฟลเดอร์หรือไฟล์ใดๆ

ls



สร้าง Node ภาษา C++ ชื่อ my\_first\_node.cpp

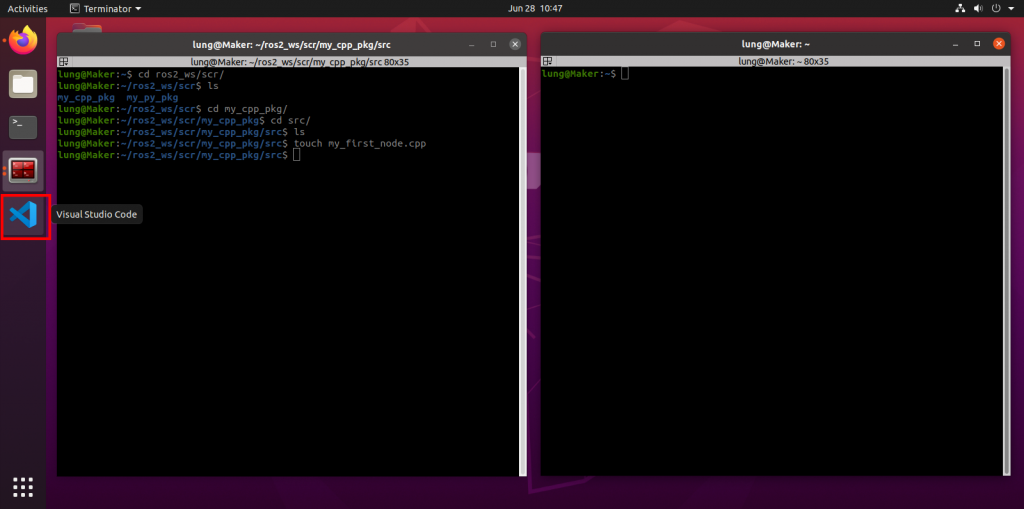
touch my\_first\_node.cpp



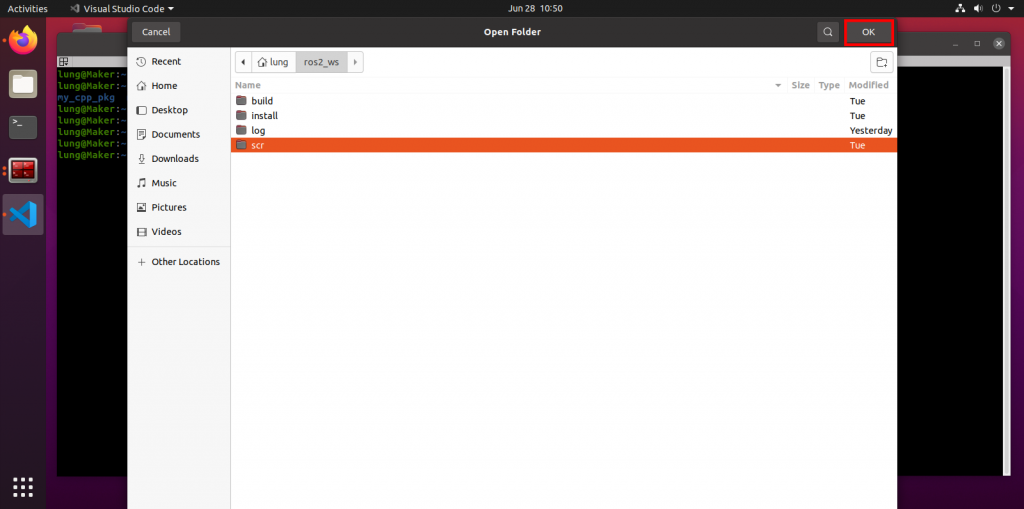
**3 : ติดตั้ง Visual Studio Code**  
ติดตั้ง Visual Studio Code บน Ubuntu 20.04 และ ทดสอบ โปรแกรมแรก ภาษา C++ ตามขั้นตอนลิงค์ด้านล่าง

* [ติดตั้ง Visual Studio Code บน Ubuntu 20.04](http://www.lungmaker.com/ros2-5-%e0%b8%95%e0%b8%b4%e0%b8%94%e0%b8%95%e0%b8%b1%e0%b9%89%e0%b8%87-visual-studio-code-%e0%b8%9a%e0%b8%99-ubuntu-20-04/)

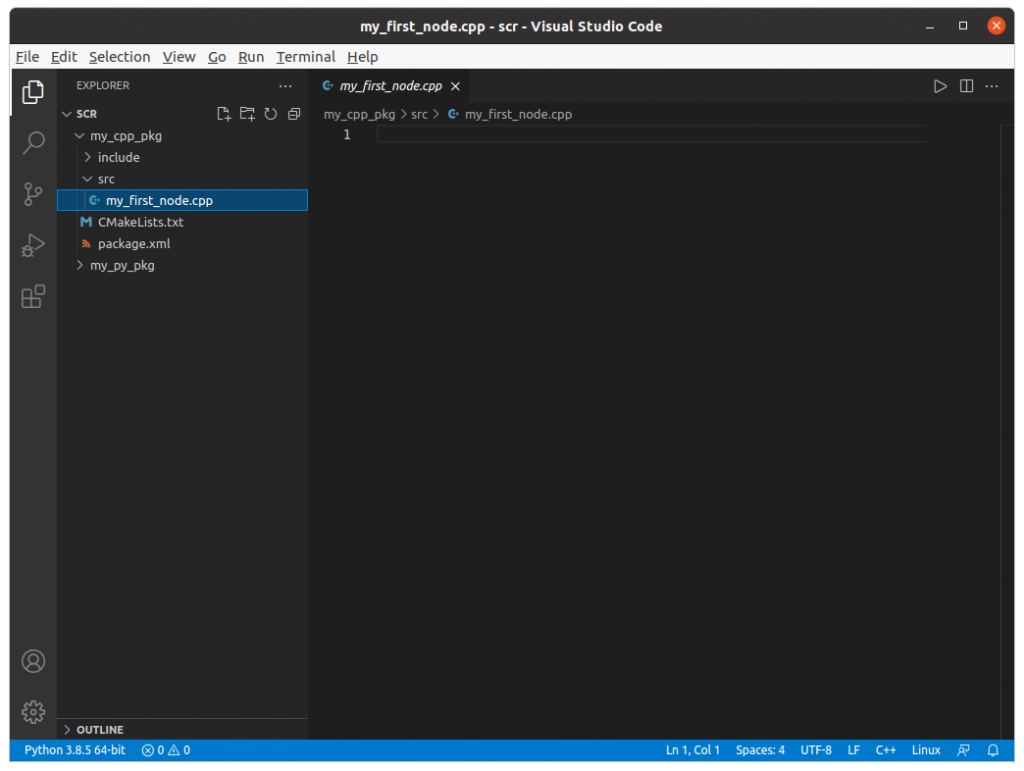
**4 : เขียนโค้ดด้วย Visual Studio Code**  
เปิด Visual Studio Code



เปิดโฟลเดอร์ scr -> OK



คลิกเปิดไฟล์ my\_first\_node.cpp



เริ่มเขียนโค้ด

#include "rclcpp/rclcpp.hpp"

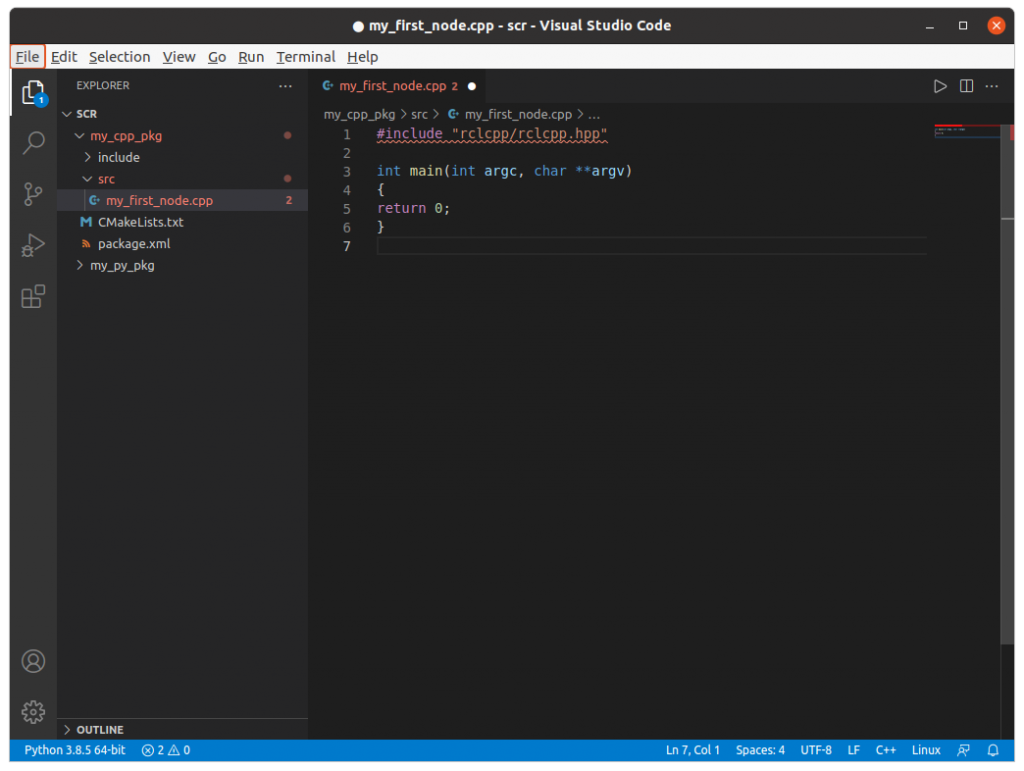
int main(int argc, char \*\*argv)

{

return 0;

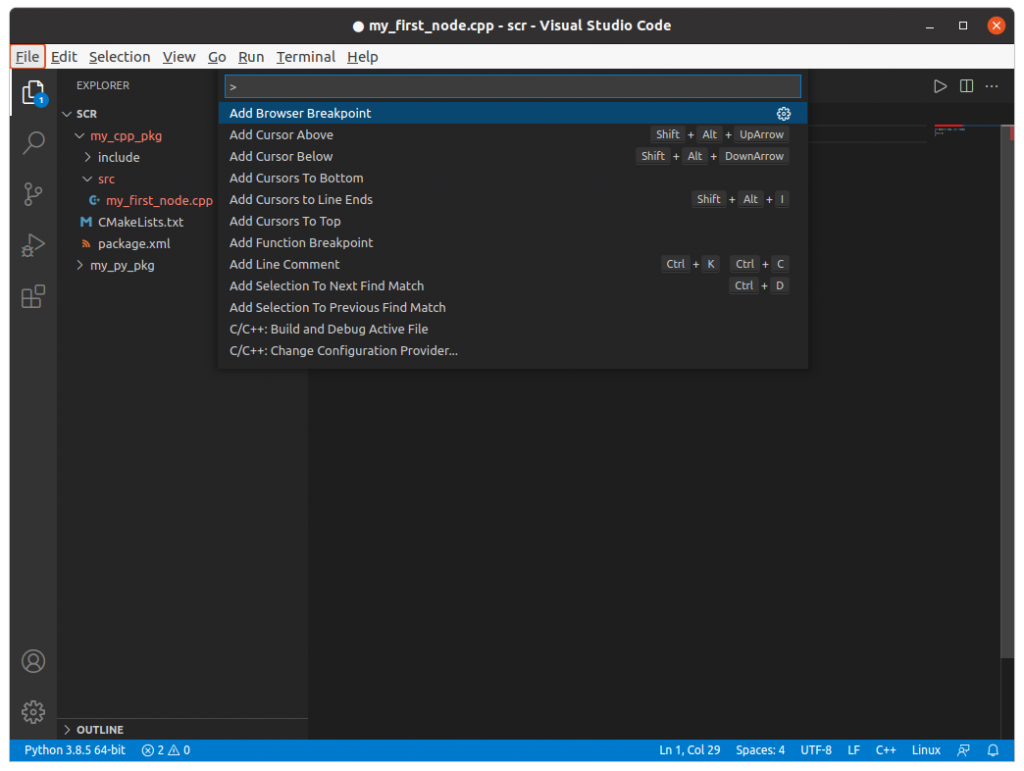
}

จะพบการแจ้งเตือน ว่าไม่พบไฟล์ rclcpp/rclcpp.hpp

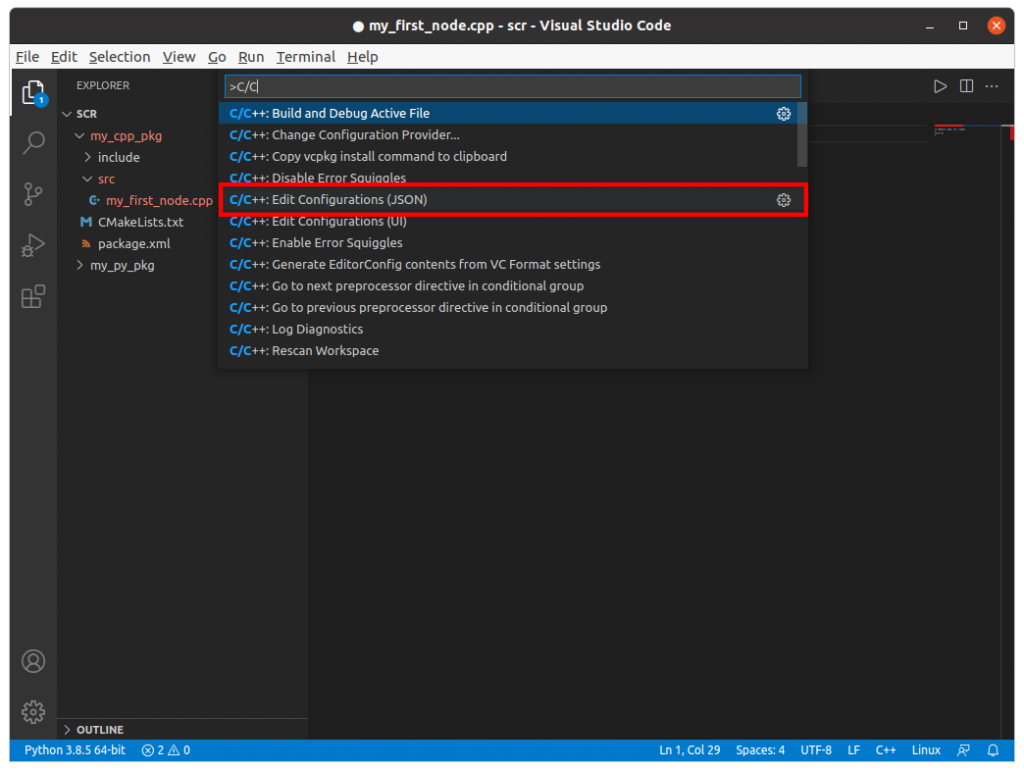


ให้แก้ไขดังนี้

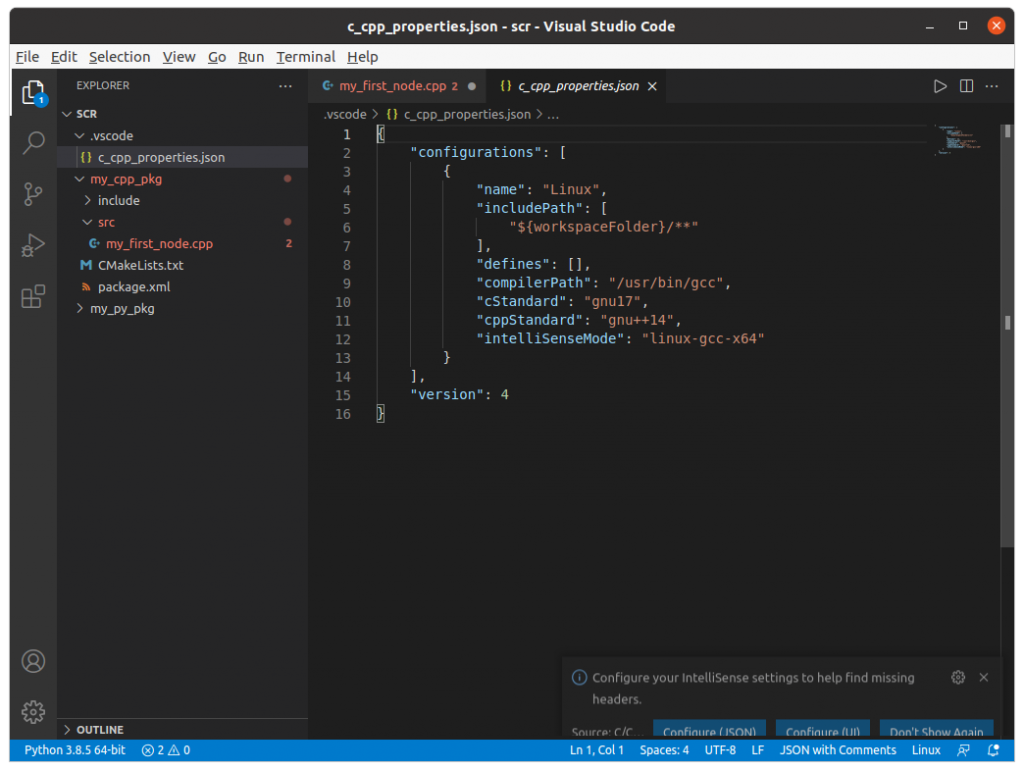
Edit Configurations : ที่คีย์บอร์ด กด Ctrl+Shift+P



ที่ช่องค้นหา ป้อน C/C แล้วเลือก C/C++: Edit Configurations (JSON)

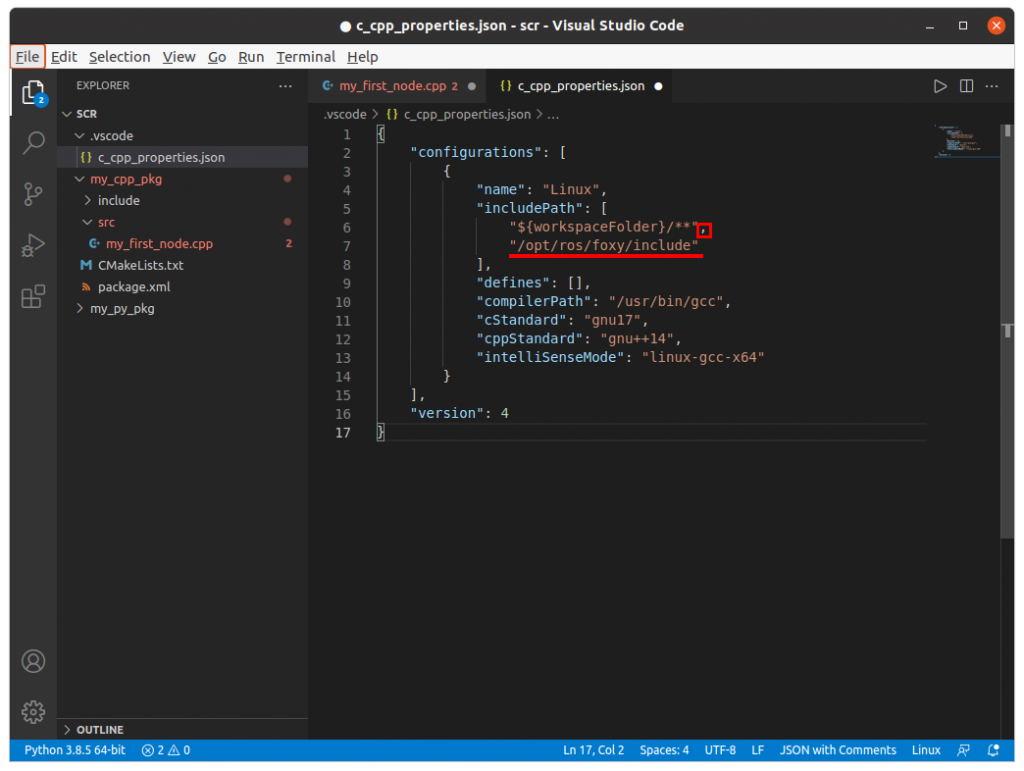


จะเปิดไฟล์ c\_cpp\_properties.json

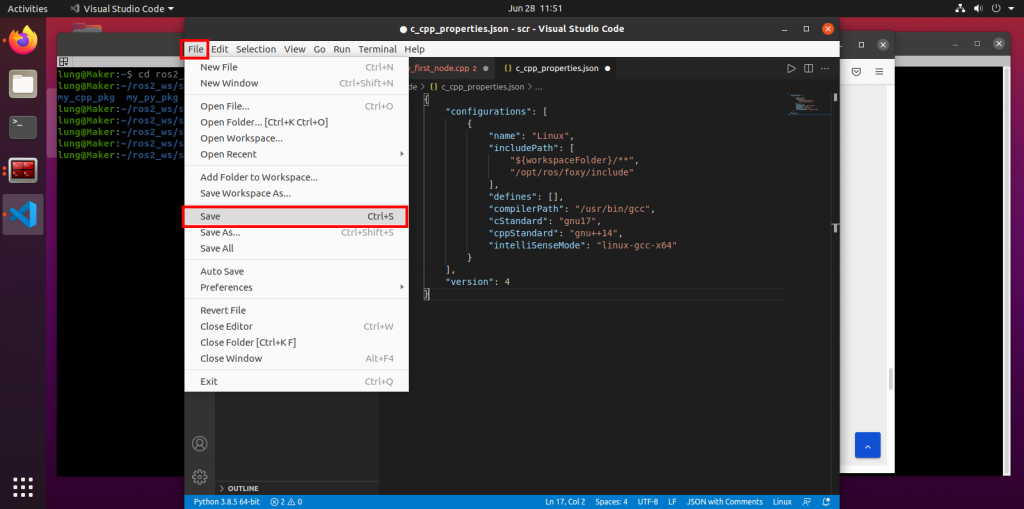


แก้ไข โดย เพิ่ม เครื่องหมาย comma หลังบรรทัดที่ 6 และ บรรทัดที่ 7 เพิ่มโค้ดตามด้านล่าง

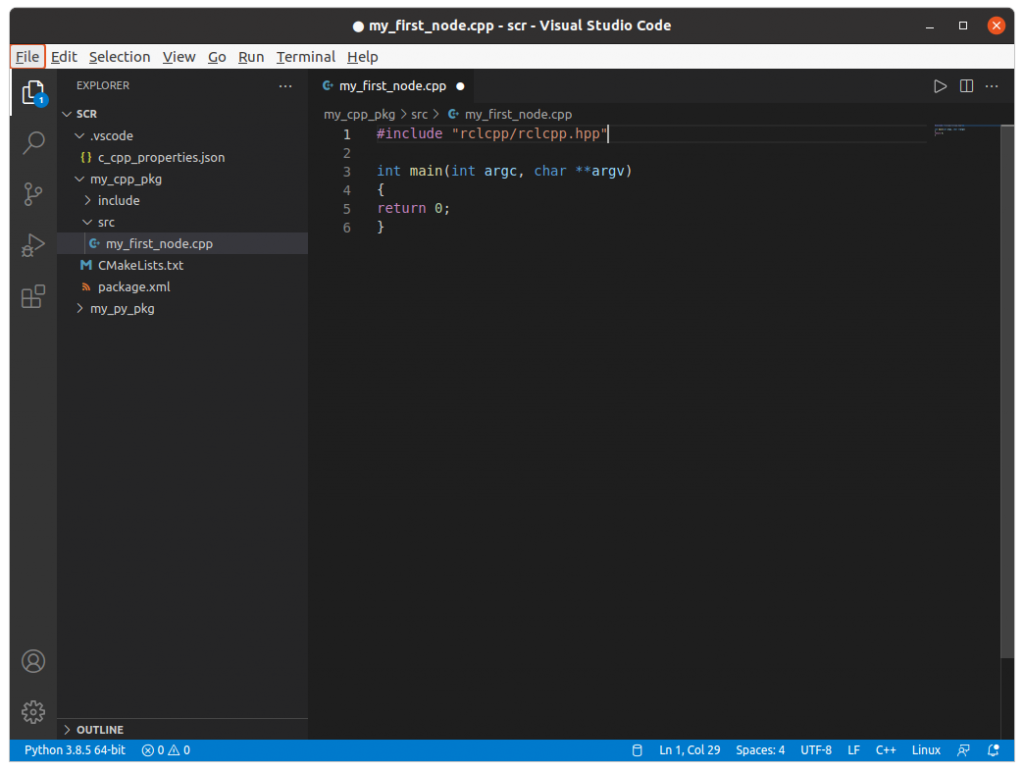
"/opt/ros/foxy/include"



File -> Save แล้วปิดหน้าต่างไฟล์นี้ลงไป



เมื่อกลับไปที่ไฟล์ my\_first\_node.cpp จะไม่พบการแจ้งเตือนแล้ว



เขียนโค้ดเพิ่มดังนี้

#include "rclcpp/rclcpp.hpp"

int main(int argc, char \*\*argv)

{

rclcpp::init(argc, argv);

auto node = std::make\_shared<rclcpp::Node>("cpp\_test");

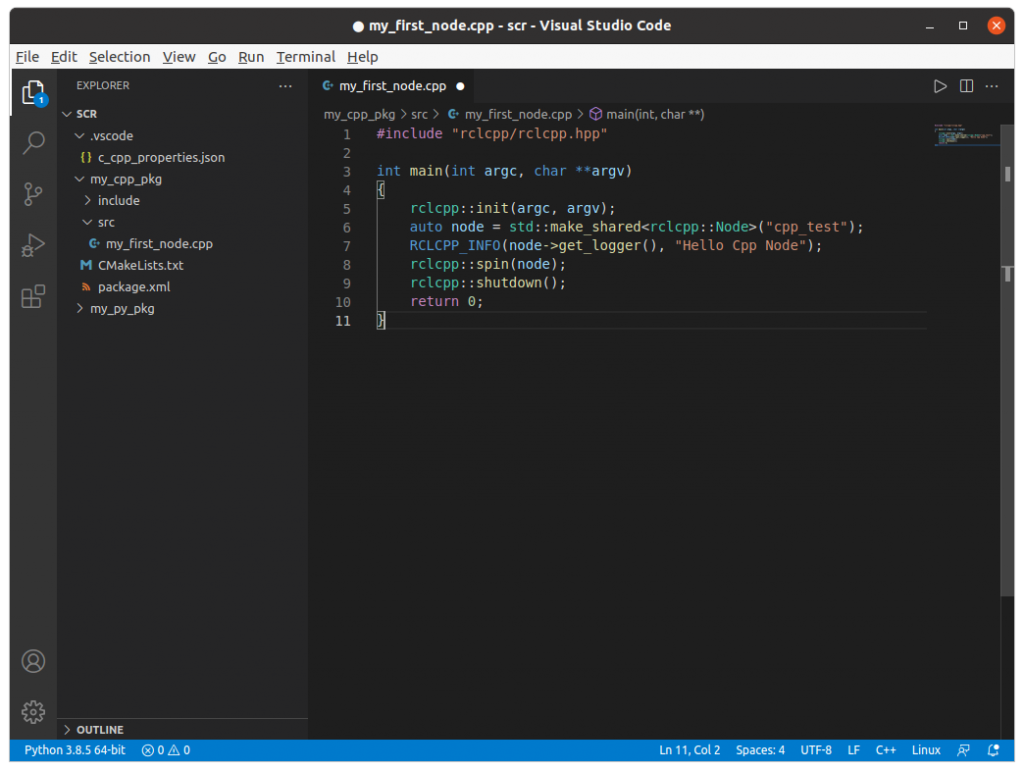
RCLCPP\_INFO(node->get\_logger(), "Hello Cpp Node");

rclcpp::spin(node);

rclcpp::shutdown();

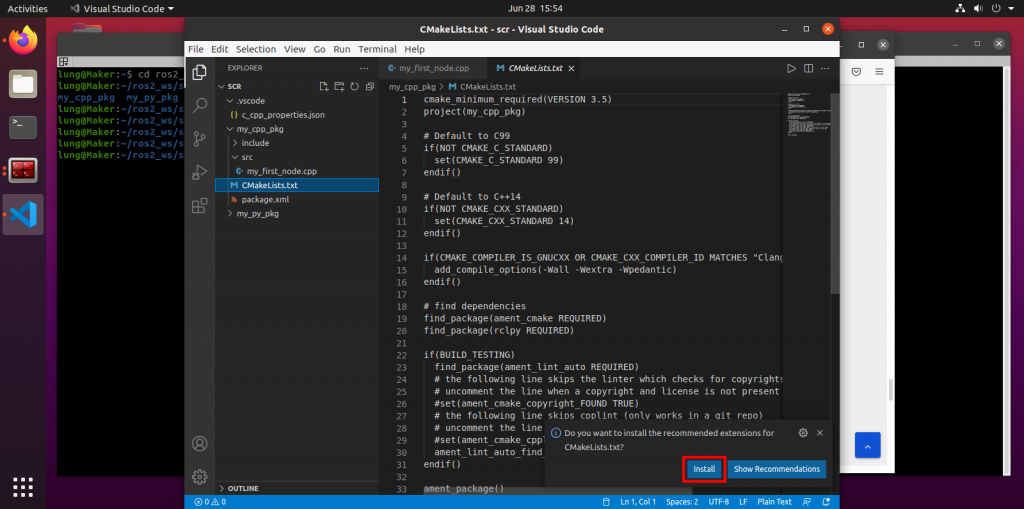
return 0;

}

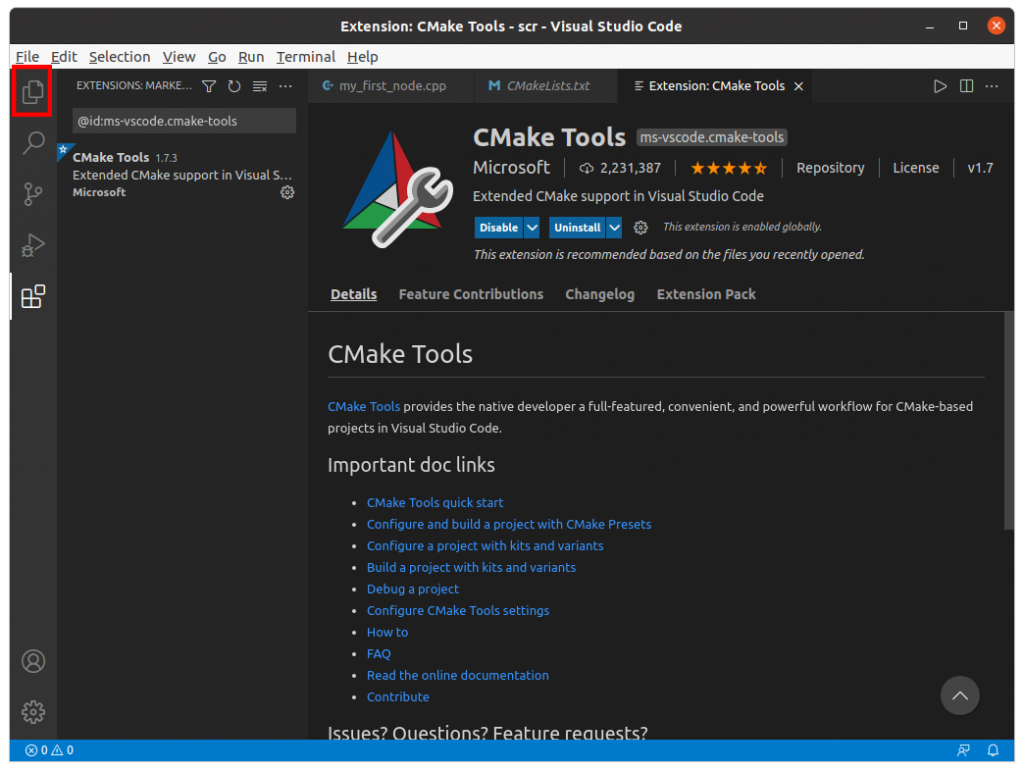


**5 : แก้ไขไฟล์ CMakeLists.txt**  
CMake คือการอ่านพิมพ์เขียวของโปรเจคจากไฟล์ CMakeLists.txt แล้วสร้าง build script แต่มันไม่ได้ทำการ build โปรแกรมด้วยตัวเอง การใช้ CMake ช่วยให้เรากำหนดซอฟต์แวร์ที่เราต้องการสร้างเพียงครั้งเดียว หลังจากนั้นเราสามารถพัฒนามันโดยใช้เครื่องมือที่เราชอบบนแพลตฟอร์มใดๆ ก็ได้ CMake จึงอำนวยช่วยความสะดวกให้กับโปรแกรมเมอร์ภาษา C++ ที่ต้องการสร้างซอฟต์แวร์ที่รันบนหลายๆ แพลตฟอร์มอย่างมาก

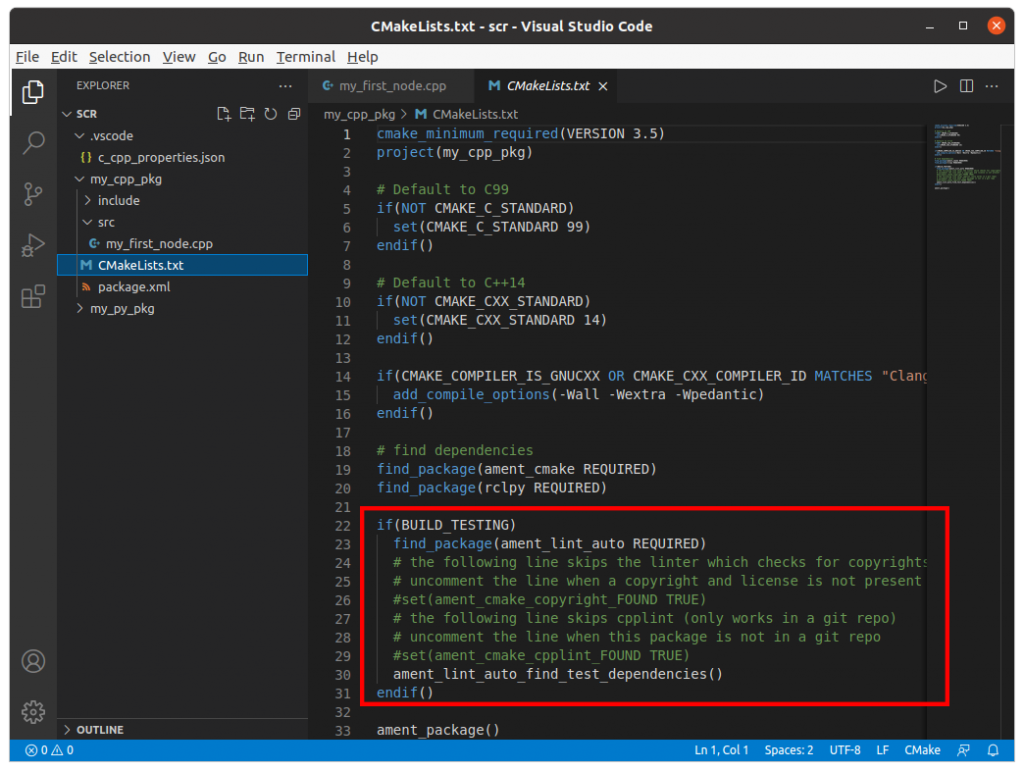
คลิกที่ไฟล์ CMakeLists.txt และคลิก install ตามคำแนะนำ ด้านล่าง ขวามือ



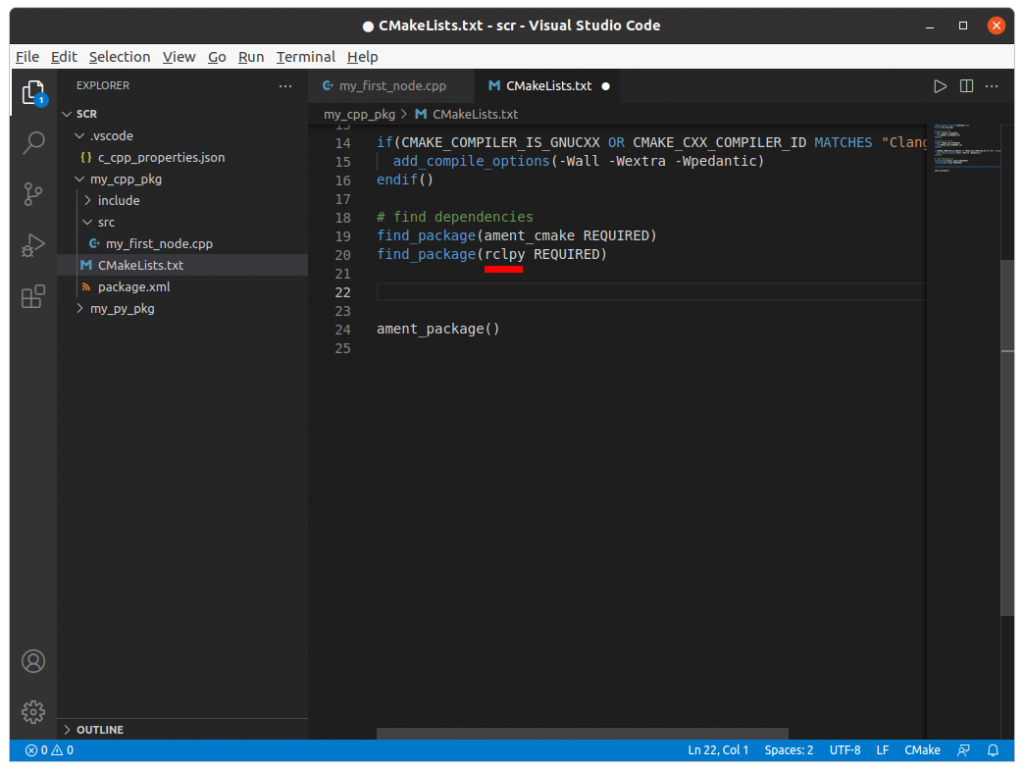
แสดงการติดตั้งสำเร็จ แล้วคลิกไอคอน Explorer ด้านซ้ายบน



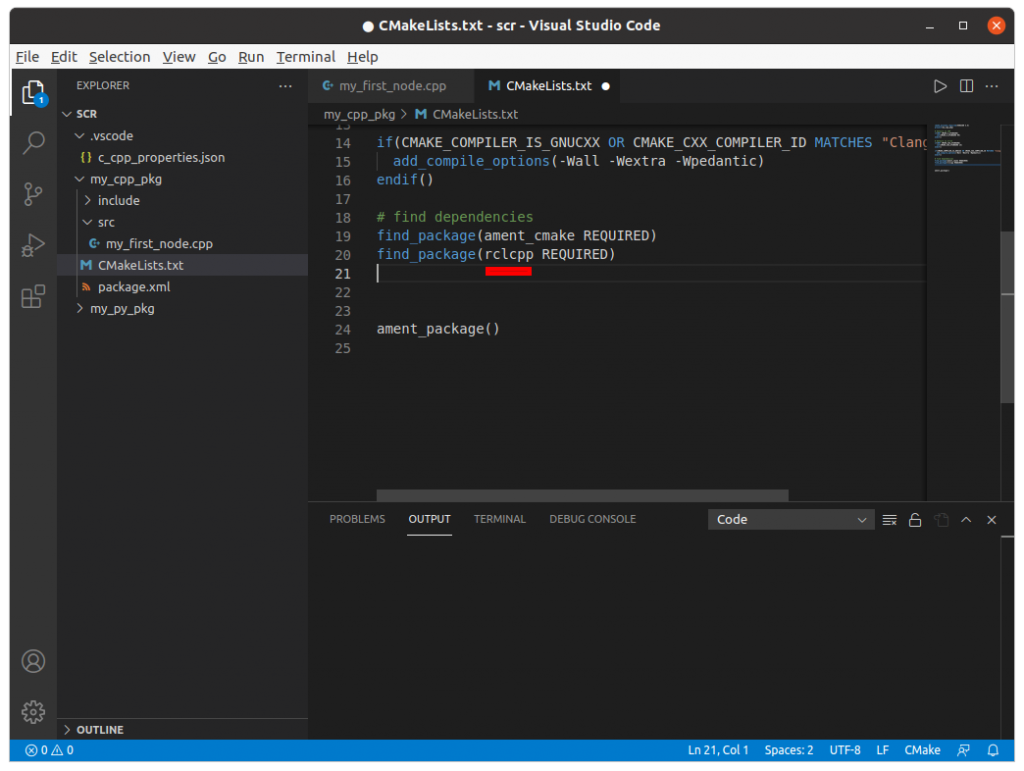
กลับไปที่ไฟล์ CMakeLists.txt ค้นหา และลบส่วนนี้ออก เพื่อเขียนโค้ด บริเวณนี้



ที่ find\_package(rclpy



แก้ไขเป็น rclcpp



เพิ่มโค้ดที่ บรรทัด 22 โค้ดดังนี้

add\_executable(cpp\_node src/my\_first\_node.cpp)

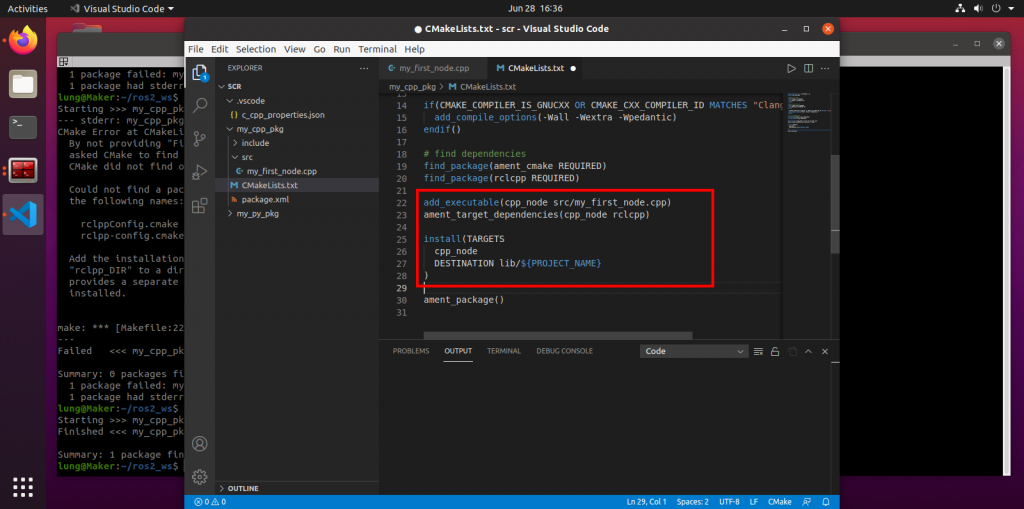
ament\_target\_dependencies(cpp\_node rclcpp)

install(TARGETS

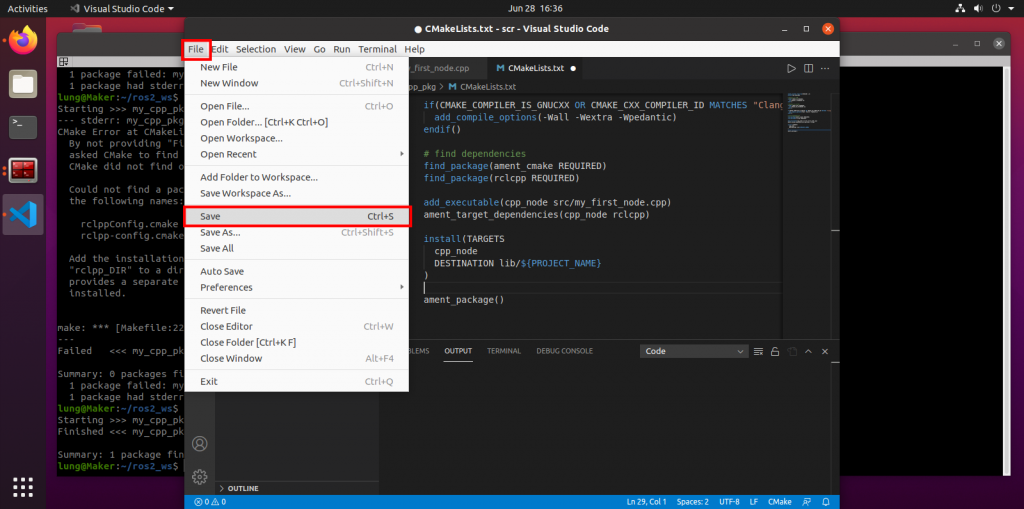
cpp\_node

DESTINATION lib/${PROJECT\_NAME}

)

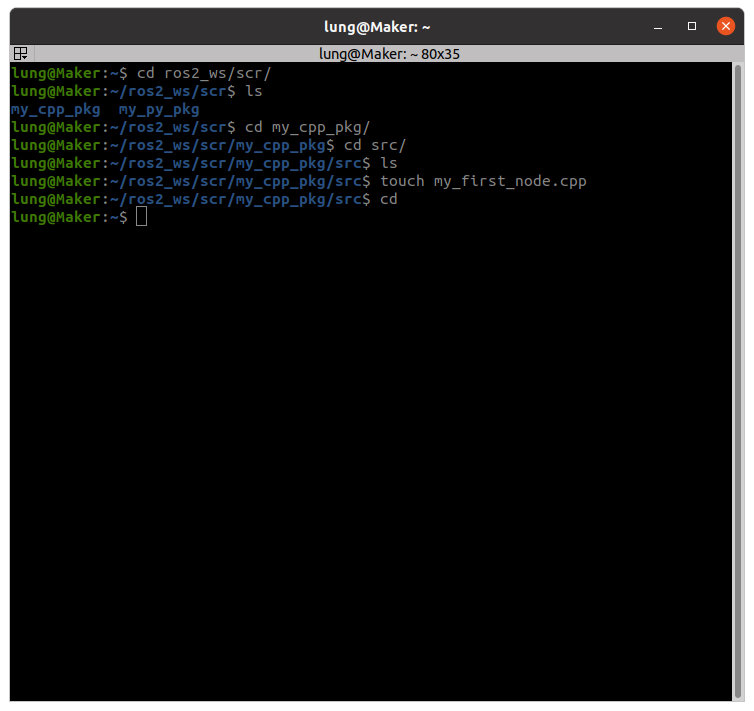


File -> Save



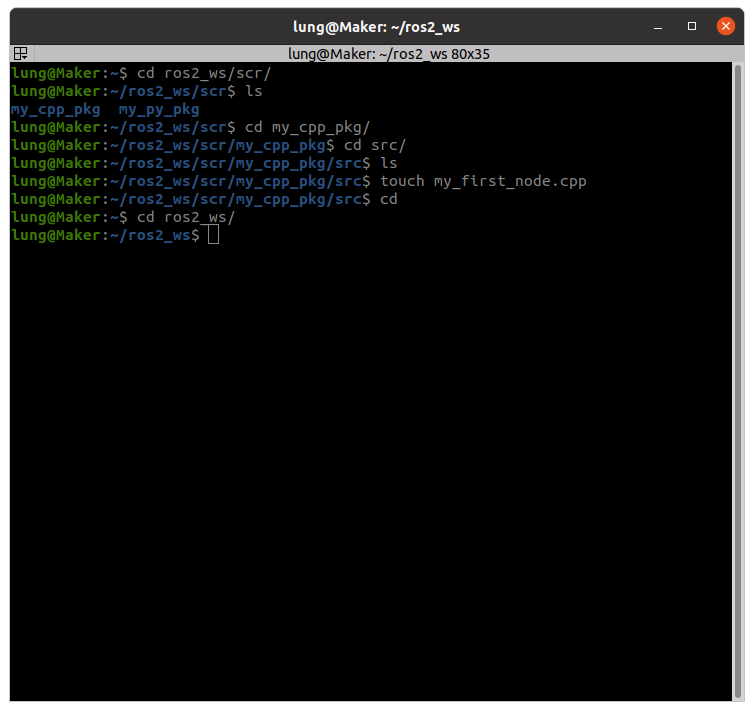
**6 : เรียกใช้งาน cpp\_node**  
กลับไปที่ Terminator เขียนคำสั่ง cd กลับสู่ Home

cd



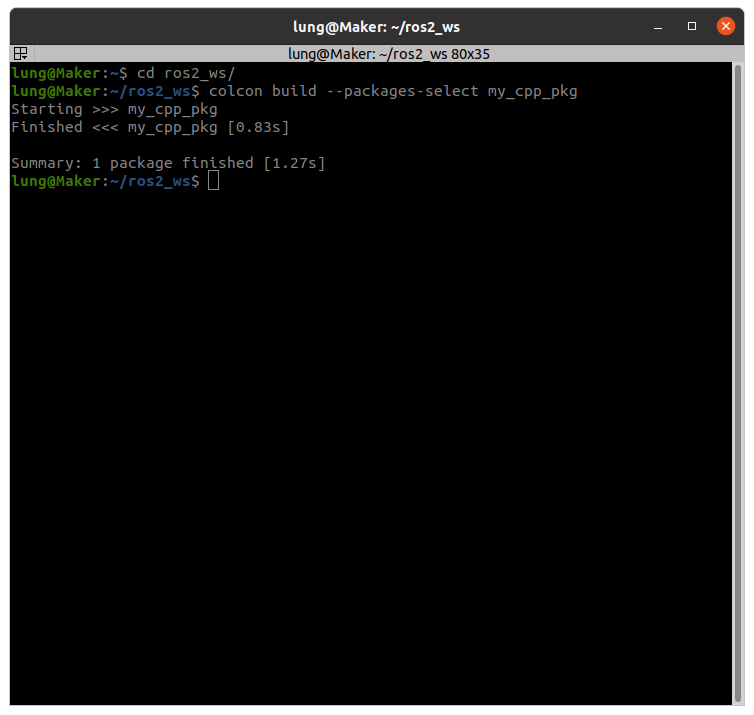
เข้าไปใน โฟลเดอร์ ros2\_ws

cd ros2\_ws/



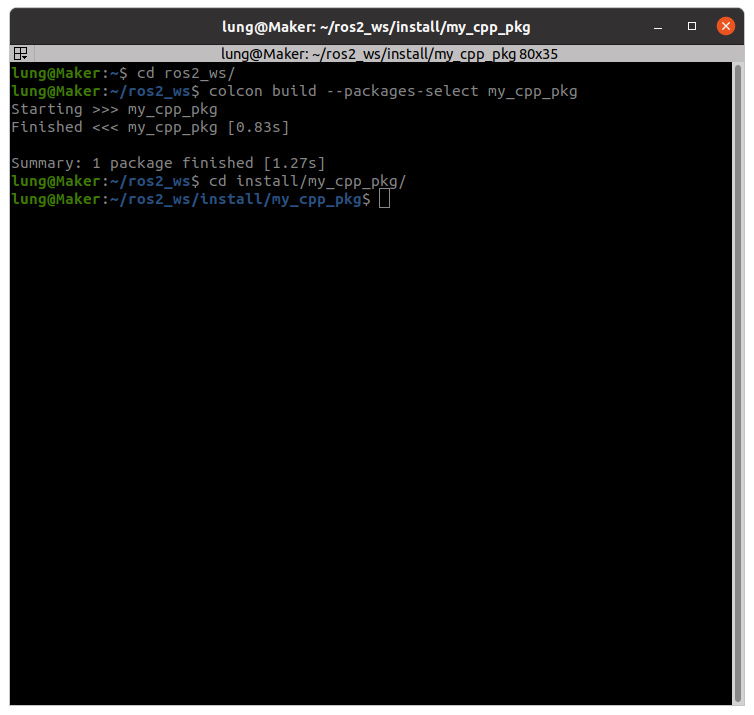
Build แพ็คเกจ my\_cpp\_pkg

colcon build --packages-select my\_cpp\_pkg



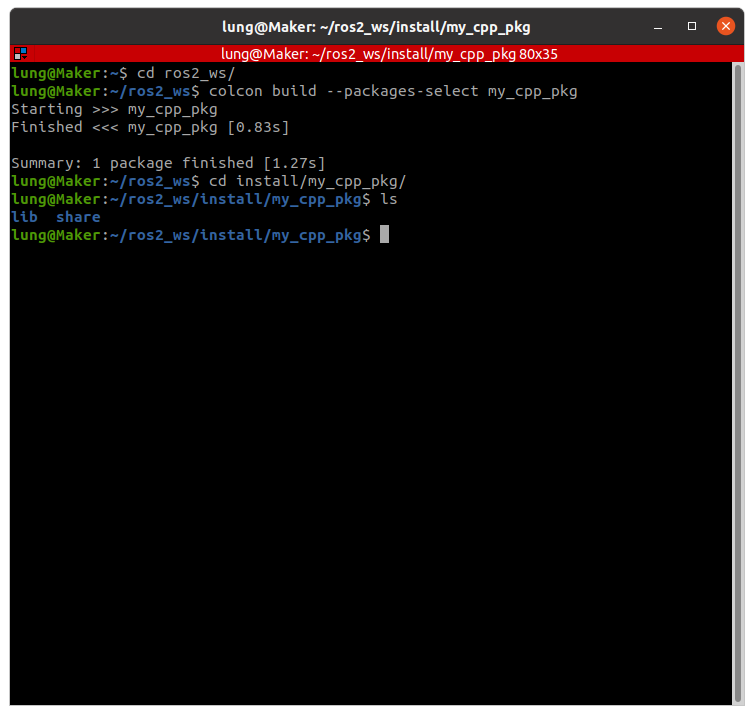
เข้าไปในโฟลเดอร์ install/my\_cpp\_pkg/

cd install/my\_cpp\_pkg/



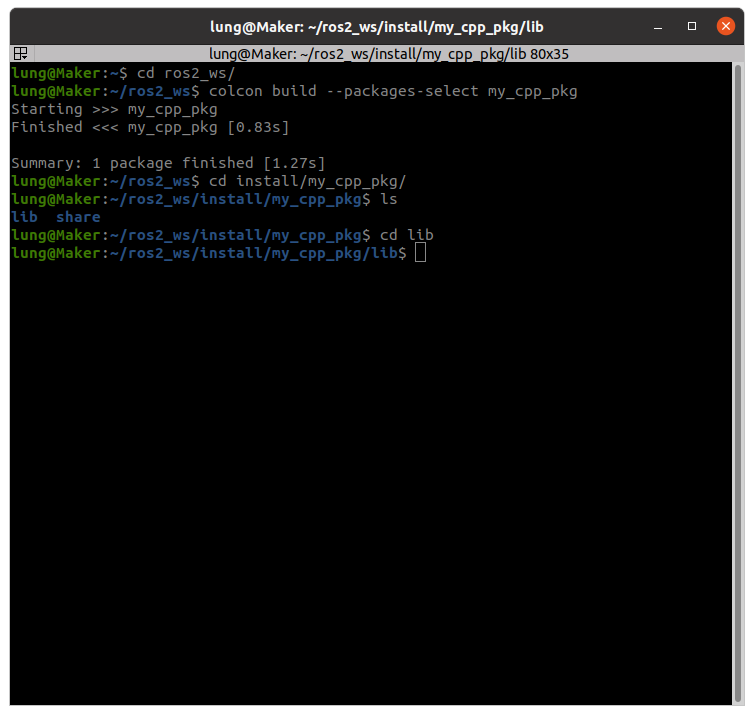
ลิสรายการ

ls



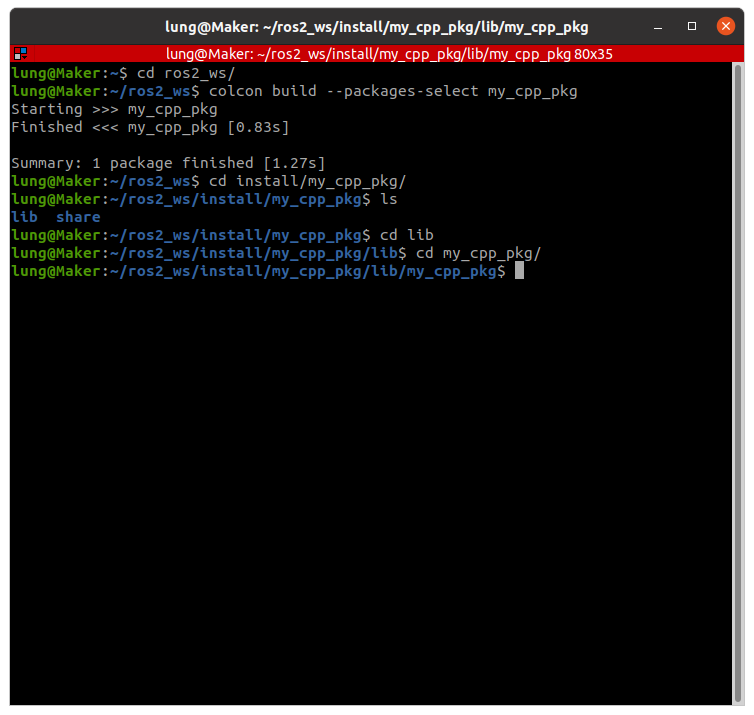
เข้าไปในโฟลเดอร์ lib

cd lib



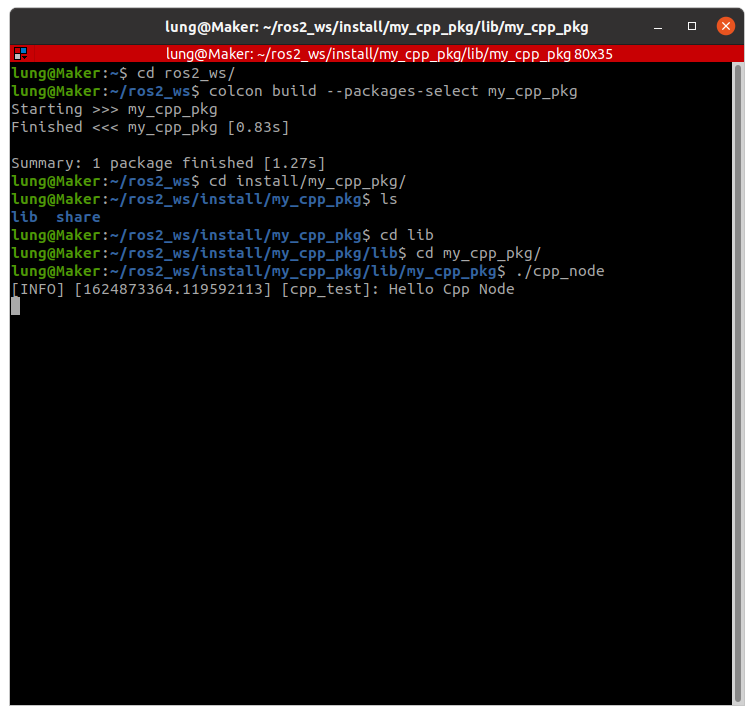
เข้าไปโฟลเดอร์ my\_pcc\_pkg

cd my\_cpp\_pkg/

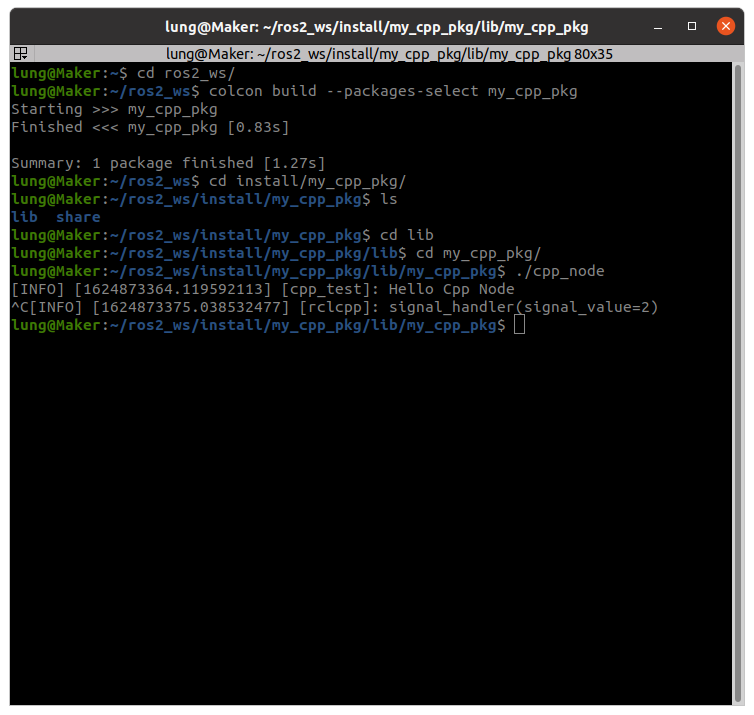


เรียกใช้งาน cpp\_node จะแสดงผลลัพธ์ การทำงาน

./cpp\_node

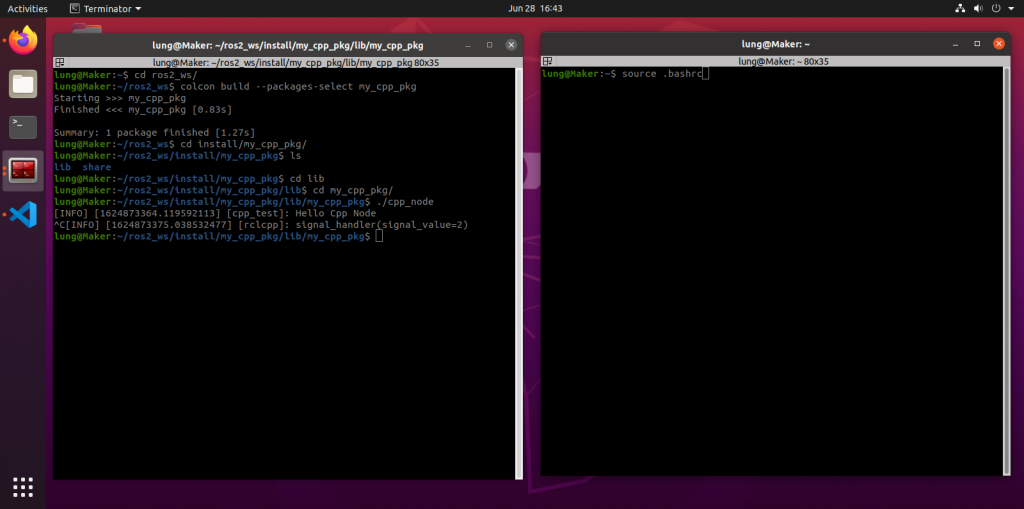


กด Ctrl + c เพื่อหยุดการทำงาน



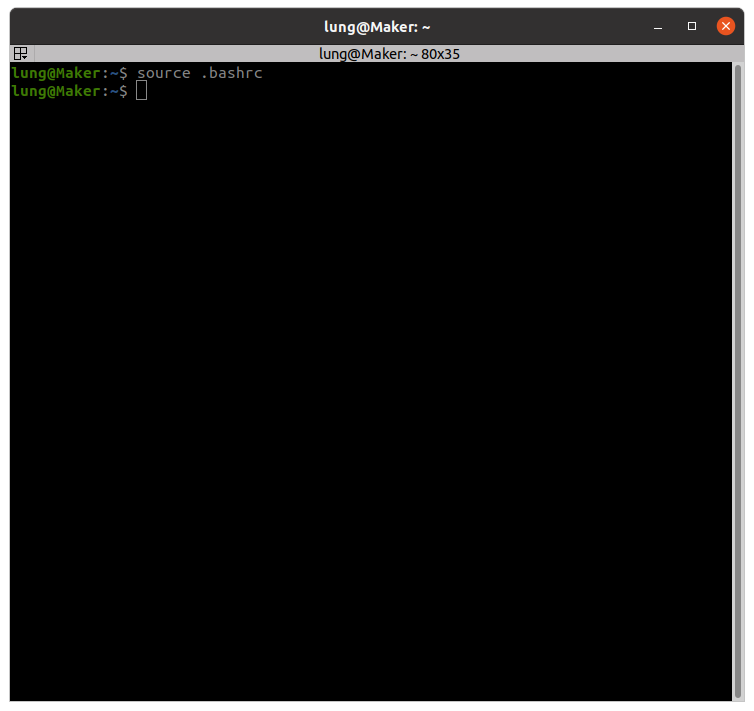
**7 : เรียกใช้งาน cpp\_node โดยใช้ ROS2**  
ซึ่งการเรียกใช้งาน cpp\_node แบบขั้นตอนที่ผ่านมา จะเห็นได้ว่ามีหลายขั้นตอน ที่ยุ่งยาก โดยการเรียกใช้งานด้วย ROS2 มีขั้นตอนดังนี้

ไปที่ Terminator หน้าต่างที่ 2



ใช้คำสั่ง

source .bashrc



เรียกใช้งานด้วยคำสั่ง ros2 run ชื่อPackag ชื่อNode จะแสดงผลลัพธ์ การทำงาน

ros2 run my\_cpp\_pkg cpp\_node

