Cd into your ros2_ws/src directory

You may need to use if ros2 is not found.

Remember to set your domain id,

And you may need to source your terminal with the following

source /opt/ros/humble/setup.bash

And run the install bash upon building a package

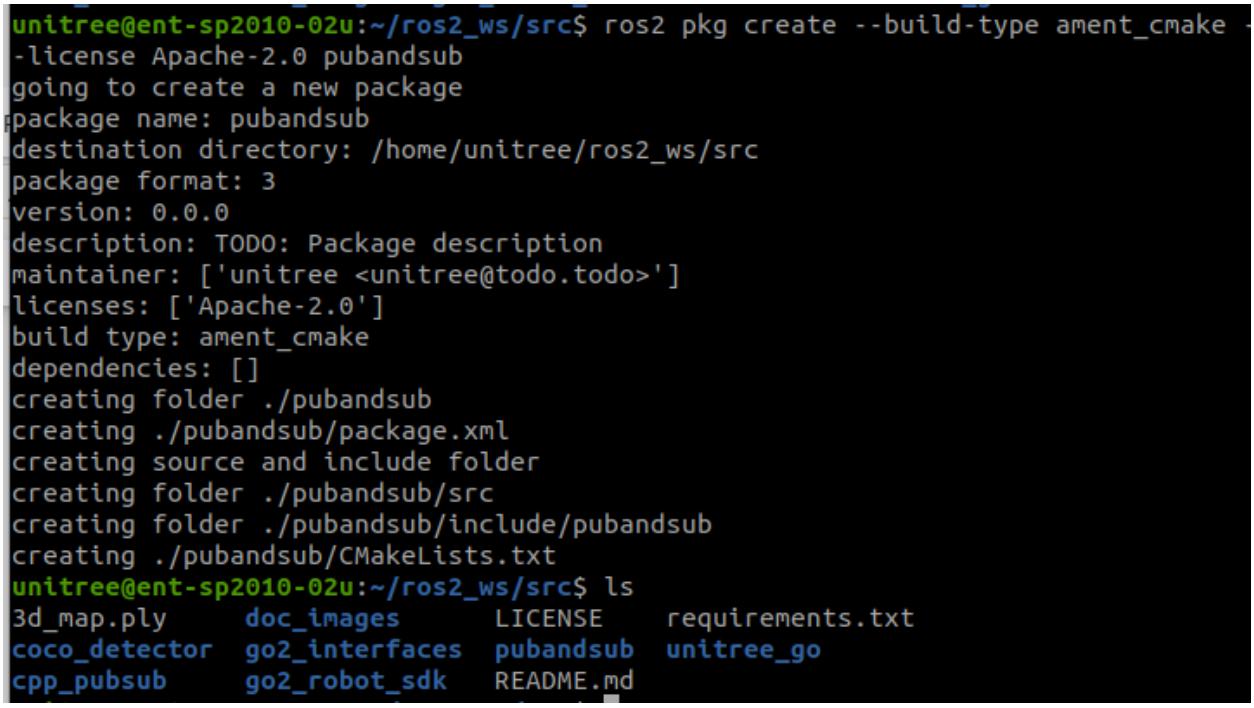
. Install/setup.bash

ros2 pkg create --build-type ament_cmake --license Apache-2.0 pubandsub

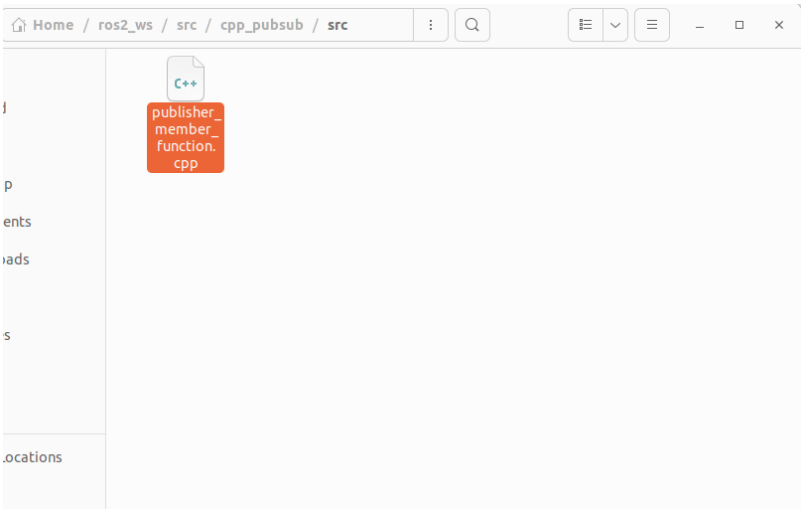
If you type ls you should see the following

If you get an error source the terminal with

source /opt/ros/humble/setup.bash



Put the publisher member file from canvas into the pubsub src folder



Have a look at the code

The top of the code includes the C++ headers . rclcpp/rclcpp.hpp allows us to use the common pieces of the ROS 2 system. Last is std_msgs/msg/string.hpp, which includes the built-in message type you will use to publish data. These lines represent the node's dependencies. The timer_callback function is where the message data is set and sent. The RCLCPP_INFO macro ensures every published message is printed to the console.

Our main fucntion, is where the node is actually executed. rclcpp::init initializes ROS 2, and rclcpp::spin starts processing data from the node, including callbacks from the timer.

We need to add these dependencies to the package.xml and CMakeLists.txt.

Before we do this look through the code and try to work out what is going on. You can add this to your logic book.

Find the package.xml in the folder structure open it and add the following lines immediatly after <build_type>ament_cmake</build_type> (still within the <export> tags)

<depend>rclcpp</depend>
<depend>std_msgs</depend>

Next open the CMakeLists.txt. We now need to add our packages to be built along with our main code, so under “find_package(ament_cmake REQUIRED)” add the following lines.

find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)

Next add

add_executable(talker src/publisher_member_function.cpp)
ament_target_dependencies(talker rclcpp std_msgs)

And then add

Install (TARGETS
talker
DESTINATION lib/${PROJECT_NAME})

```
4 if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
5   add_compile_options(-Wall -Wextra -Wpedantic)
6 endif()
7
8 # find dependencies
9 find_package(ament_cmake REQUIRED)
10 find_package(rclcpp REQUIRED)
11 find_package(std_msgs REQUIRED)
12 # uncomment the following section in order to fill in
13 # further dependencies manually.
14 # find_package(<dependency> REQUIRED)
15
16 add_executable(talker src/publisher_member_function.cpp)
17 ament_target_dependencies(talker rclcpp std_msgs)
18
19 install(TARGETS
20   talker
21   DESTINATION lib/${PROJECT_NAME})
22
23 ament_package()
```

Going back to our **SRC** folder add the subsribber code from canvas

Open up the cmake folder again and add the following line under the executable for the talker. But before the install targets!

add_executable(listener src/subscriber_member_function.cpp)
ament_target_dependencies(listener rclcpp std_msgs)

Then change the install targets to

install(TARGETS
talker
listener
DESTINATION lib/${PROJECT_NAME})

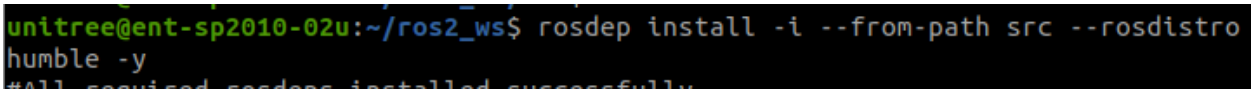
```
cmake_minimum_required(VERSION 3.5)
project(pubandsub)

#Default to C++14
if(NOT CMAKE_CXX_STANDARD)
  set(CMAKE_CXX_STANDARD 14)
endif()

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)

add_executable(talker src/publisher_member_function.cpp)
ament_target_dependencies(talker rclcpp std_msgs)

add_executable(listener src/subscriber_member_function.cpp)
ament_target_dependencies(listener rclcpp std_msgs)

install(TARGETS
  talker
  listener
  DESTINATION lib/${PROJECT_NAME})
```

At this stage our base package is complete

Navigate to up a level to Ros_Ws and run

rosdep install -i --from-path src --rosdistro humble –y

(Note: If you have other ros2 files in your folder, the rosdep will try to install files needed for these. As we don't want to fix these issues, we can just navigate to our puband sub folder and run the command from their.



This line of code will look for any missing packages and install them for us.

Assuming you get no errors

Run the following

colcon build --packages-select pubandsub

Colcon build being the command to build our ros2 package, and pubandsub the package name. (should take about 5 seconds). Assuming you get no errors your package is built.

Open 2 new terminal source your terminals with. Navigate to ros2_ws in both and source them if needed

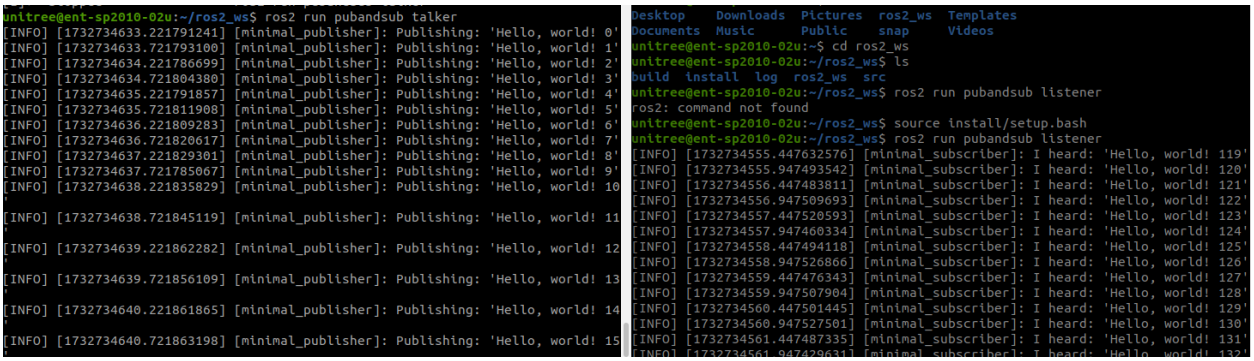
. Install/setup.bash

In one terminal run

ros2 run pubandsub talker

And in the other

ros2 run pubandsub listener



Kill both commands in the terminals (ctrl-c)

Tasks: (please note upon future build you may need to close the terminal and run colcon from a new sourced terminal.

Look at the current logic in publisher.cpp file

message.data = "Hello, world! " + std::to_string(count_++);

Change this code to only print a hello world + random number between 1-20

(hit section 2 of, https://isocpp.org/files/papers/n3551.pdf is a good shout)

Likely it will look something like

message.data = "Hello, world! " + std::to_string(roll_a_fair_die( ));

Build your code in the original terminal and run both the talker and listener again.

Okay lets get rid of the “hello world” so we only print a number.

In the listener node

Modifier the following lines

```
private:
  void topic_callback(const std_msgs::msg::String & msg) const
  {
    RCLCPP_INFO(this->get_logger(), "I heard: '%s'", msg.data.c_str());
  }
  rclcpp::Subscription<std_msgs::msg::String>::SharedPtr subscription_;
};
```

So that it only sends a message if a number between 16-20 is heard. The following line of code may help by converting a string to an int.

```
int vname = std::stoi(msg.data.c_str());
```

Advanced:

Next still in the listener node, at an integer which is set to 96.

Each time the listener hears a value between 16-18 subtract a value from the integer we just made = 7 +rand(1-6), double this value if we hear 19-20. Once our integer is less <=0 stop the publisher sending any more messages.