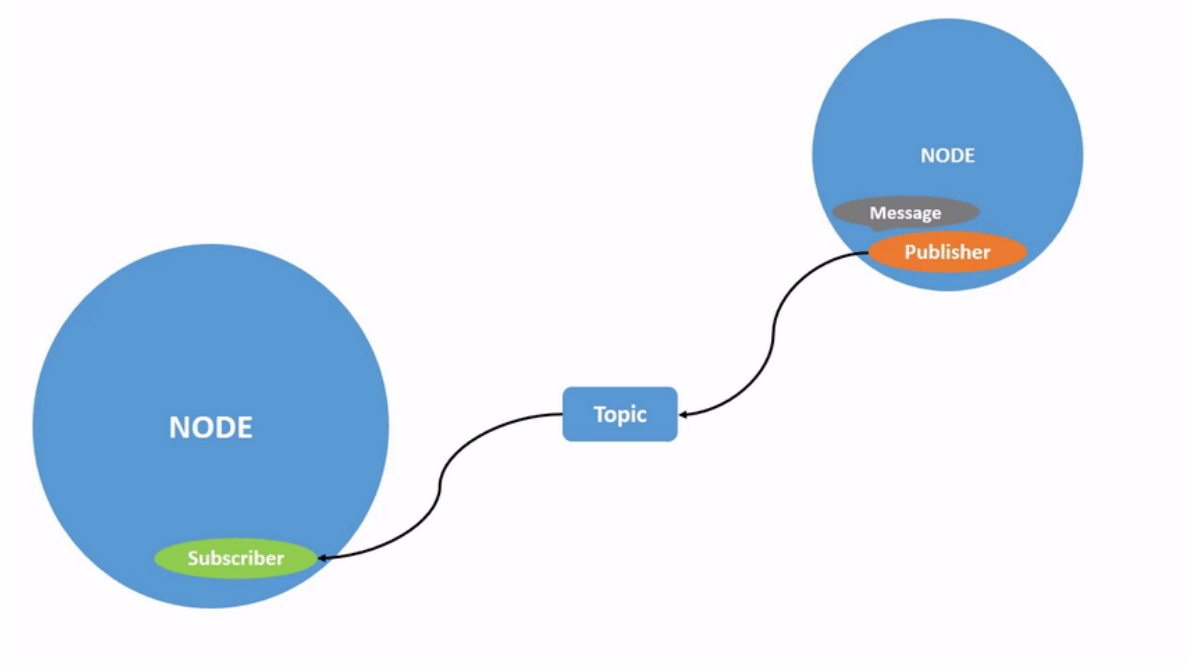
7.ROS2 topic communication publisher

1. Introduction to topic communication

Topic communication is the most frequently used communication method in ROS2. A publisher publishes data on a specified topic, and subscribers can receive the data as long as they subscribe to the data on the topic.

Topic communication is based on the publish/subscribe model, as shown in the figure:



The characteristics of topic data transmission are from one node to another node. The object sending data is called **Publisher**, and the object receiving data is called **Subscriber**. Each topic needs to have a name, the transmitted data also needs to have a fixed data type.

Next, we will explain how to use Python language to implement topic communication between nodes.

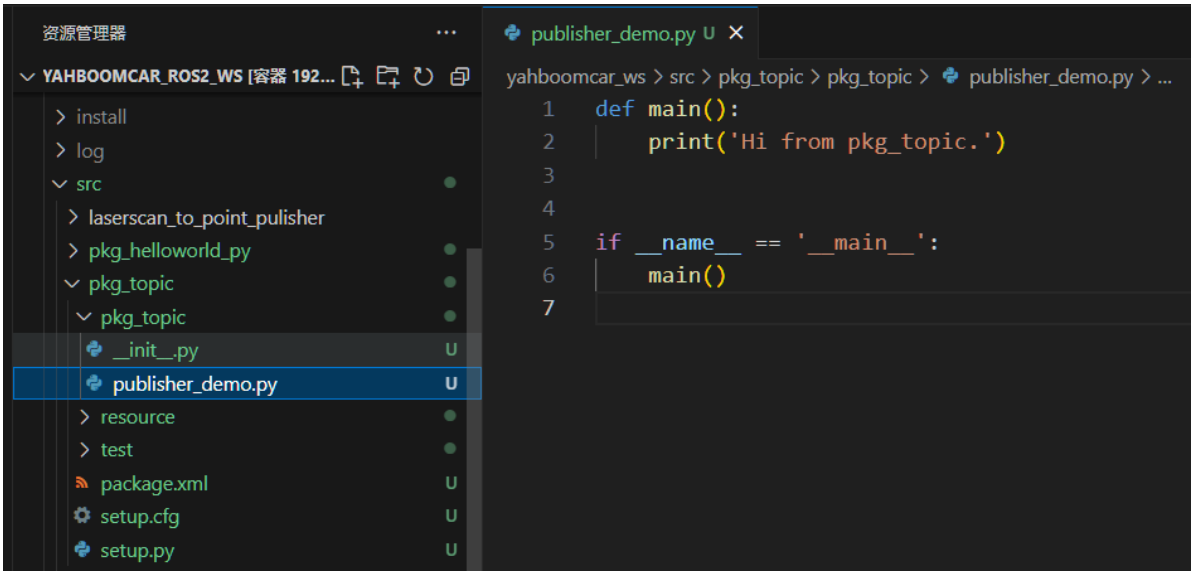
This case is located in the factory docker container. The source code location is:

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/pkg_topic
```

2. Create a new function package

```
cd ~/yahboomcar_ros2_ws/yahboomcar_ws/src
ros2 pkg create pkg_topic --build-type ament_python --dependencies rclpy --node-
name publisher_demo
```

After executing the above command, the pkg_topic function package will be created, and a publisher_demo node will be created, and the relevant configuration files have been configured.



3. Publisher implementation

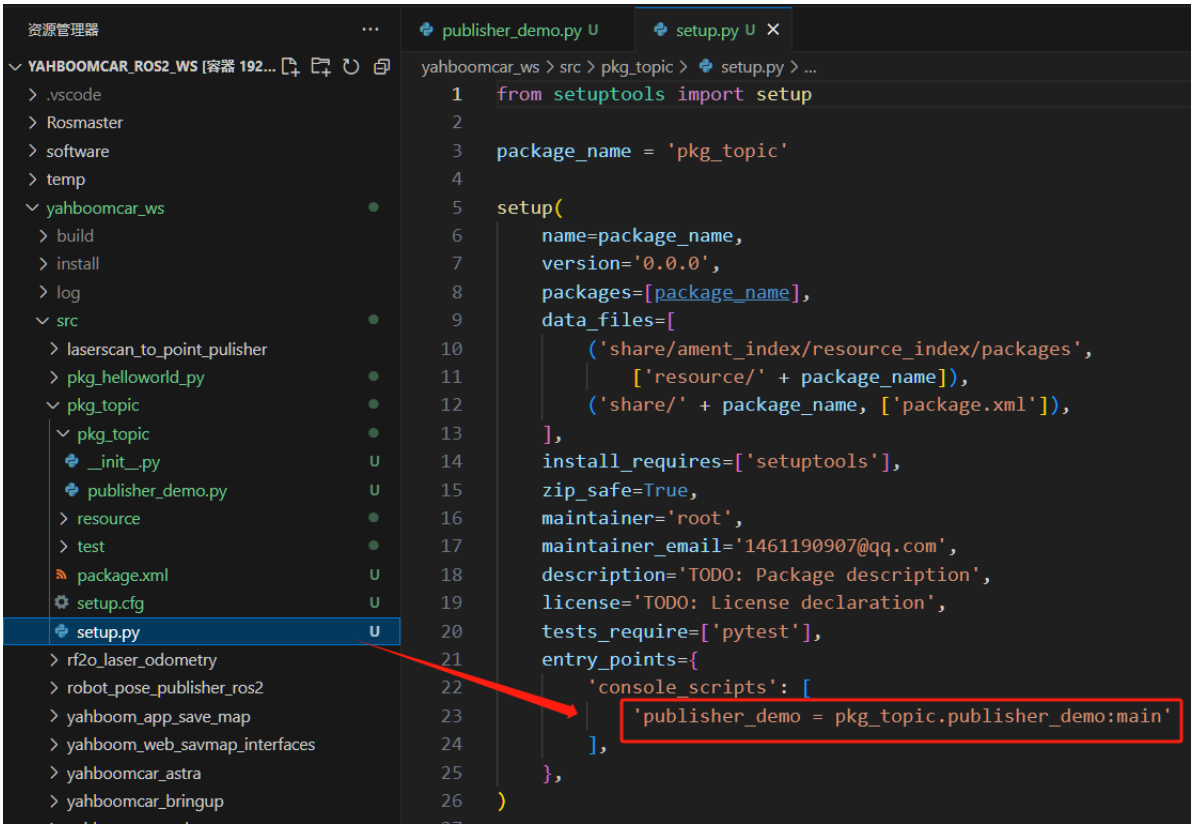
Next edit [publisher_demo.py] to implement the publisher's functions and add the following code:

```
#Import rclpy library
import rclpy
from rclpy.node import Node
#Import String message
from std_msgs.msg import String
#Create a Topic_Pub node subclass that inherits from the Node base class and pass
in a parameter name
class Topic_Pub(Node):
    def __init__(self,name):
        super().__init__(name)
        #To create a publisher, use the create_publisher function. The
parameters passed in are:
        #Topic data type, topic name, queue length to save messages
        self.pub = self.create_publisher(String,"/topic_demo",1)
        #Create a timer and enter the interrupt processing function at intervals
of 1s. The parameters passed in are:
        #Interval time between interrupt function execution, interrupt
processing function
        self.timer = self.create_timer(1,self.pub_msg)
    #Define interrupt handler function
    def pub_msg(self):
        msg = String()  #Create a variable msg of type String
        msg.data = "Hi,I send a message." #Assign a value to the data in msg
        self.pub.publish(msg) #Publish topic data

#Main function
def main():
    rclpy.init() #initialization
    pub_demo = Topic_Pub("publisher_node") #Create a Topic_Pub class object, and
the parameter passed in is the name of the node.
```

```
    rclpy.spin(pub_demo)     #Execute the rclpy.spin function and pass in a
parameter. The parameter is the Topic_Pub class object just created.
    pub_demo.destroy_node()  #Destroy node object
    rclpy.shutdown()         #Turn off the ROS2 Python interface
```

4. Edit configuration file



5. Compile workspace

```
cd ~/yahboomcar_ros2_ws/yahboomcar_ws
colcon build --packages-select pkg_topic
source install/setup.bash
```

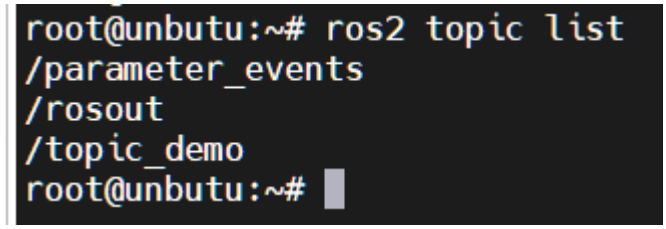
6.Run program

Open a terminal:

```
ros2 run pkg_topic publisher_demo
```

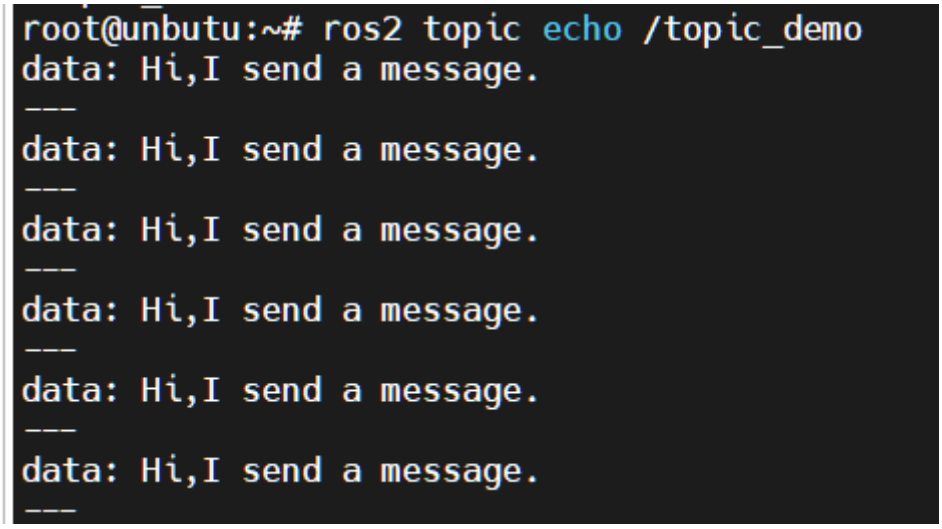
After the program runs successfully, nothing is printed. We can view the data through the ros2 topic tool. First, check whether there is a topic published, and open another terminal to input:

```
ros2 topic list
```



This topic_demo is the topic data defined in the program. Next, we use ros2 topic echo to print this data, and enter in the terminal:

```
ros2 topic echo /topic_demo
```



It can be seen that the "Hi,I send a message." printed by the terminal is consistent with the msg.data = "Hi,I send a message." in our code.