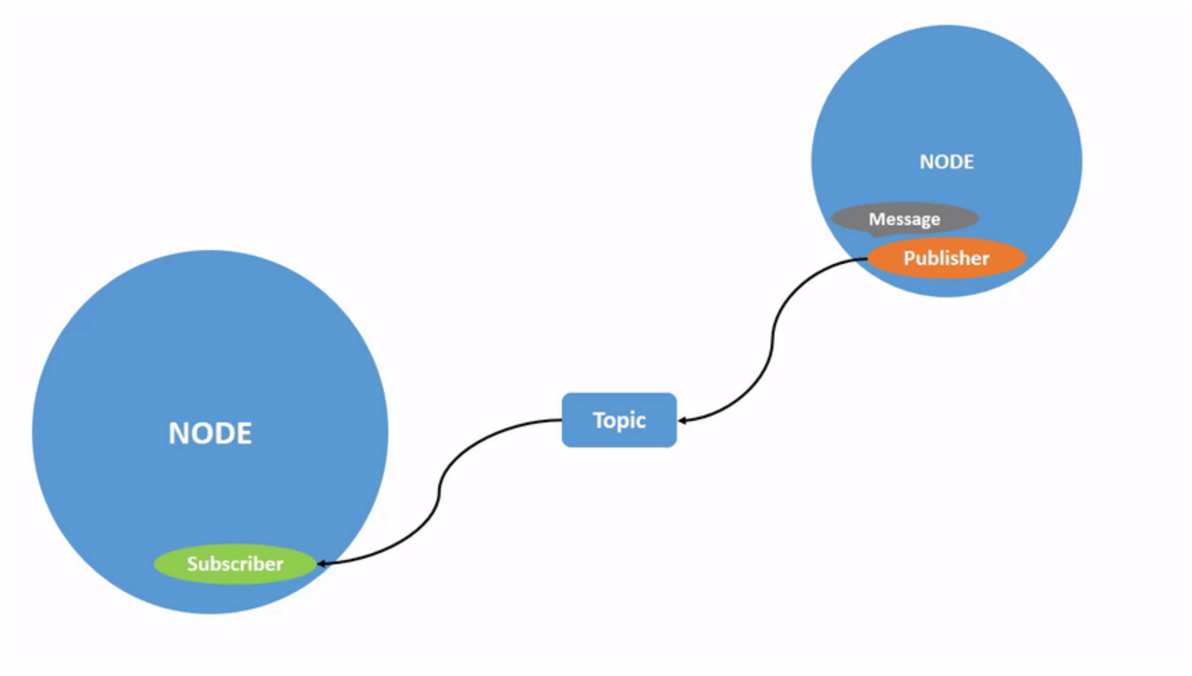
7. ROS2 Topic Communication

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 who subscribe to that topic receive the data.

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



Topic data transmission is a process where the data is transmitted from one node to another. The object sending data is called a publisher, and the object receiving data is called a subscriber. Each topic must have a name, and the data transmitted must have a fixed data type.

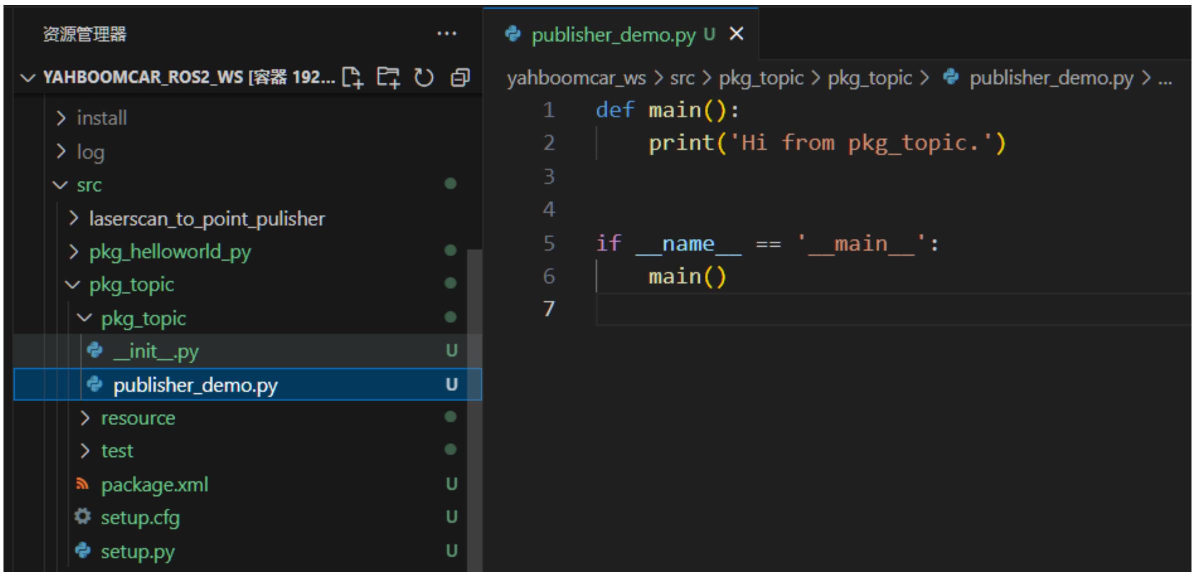
Next, we will explain how to implement topic communication between nodes using Python.

2. Create a New Package

- Switch to the src directory of the workspace
- Create a new pkg_topic package

```
ros2 pkg create pkg_topic --build-type ament_python --dependencies rclpy --node-name publisher_demo
```

After executing the above command, the pkg_topic package will be created, along with a publisher_demo node and the relevant configuration files.



```
def main():
    print('Hi from pkg_topic.')


if __name__ == '__main__':
    main()
```

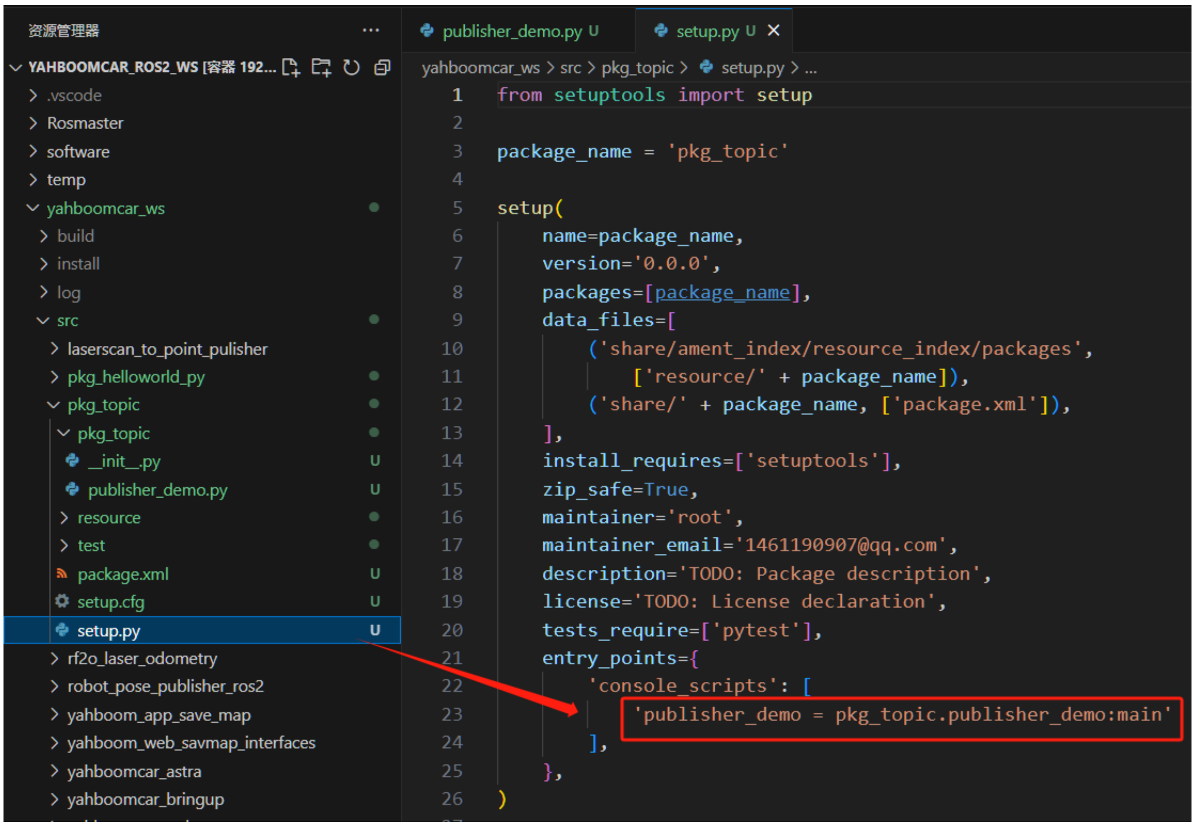
3. Publisher Implementation

3.1 Create a Publisher

Next, edit [publisher_demo.py] to implement the publisher functionality and add the following code:

```
#Import the rclpy library
import rclpy
from rclpy.node import Node
#Import String messages
from std_msgs.msg import String
#Create a Topic_Pub node subclass that inherits from the Node base
class Topic_Pub(Node):
    def __init__(self,name):
        super().__init__(name)
        #Create a publisher using the create_publisher function. The passed
parameters are:
        #Topic data type, topic name, and queue length for storing messages
        self.pub = self.create_publisher(String,"/topic_demo",1)
        #Create a timer that enters the interrupt handler every 1 second. The
passed parameters are:
        #Interval between interrupt executions, interrupt handler function
        self.timer = self.create_timer(1,self.pub_msg)
    #Define the interrupt handler function
    def pub_msg(self):
        msg = String()  #Create a String variable, msg
        msg.data = "Hi,I send a message." #Assign data to 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,
passing in the node name as a parameter
    rclpy.spin(pub_demo)     #Execute the rclpy.spin function, passing in the
Topic_Pub class object just created as a parameter
    pub_demo.destroy_node()  #Destroy the node object
    rclpy.shutdown()         #Shut down the ROS2 Python interface
```

3.2 Editing the Configuration File



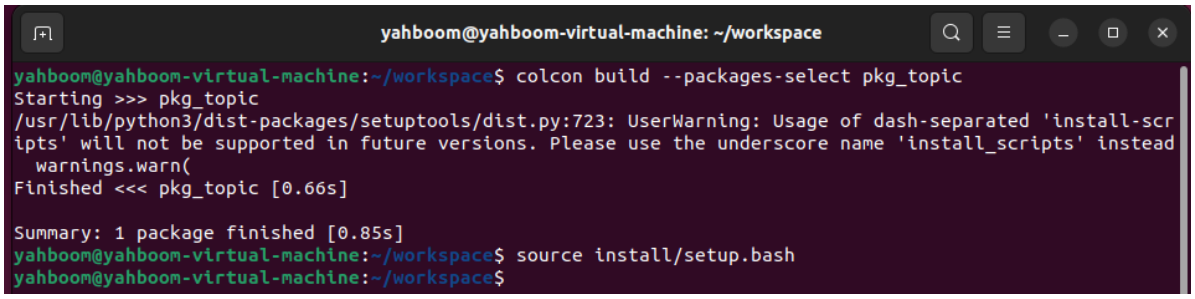
```python
from setuptools import setup

package_name = 'pkg_topic'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='root',
    maintainer_email='1461190907@qq.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'publisher_demo = pkg_topic.publisher_demo:main'
        ],
    },
)
```

3.3 Compiling the Package

- Compiling the Package

```
colcon build --packages-select pkg_topic
```

- Refresh the environment variables in the workspace



```
yahboom@yahboom-virtual-machine: ~/workspace
yahboom@yahboom-virtual-machine:~/workspace$ colcon build --packages-select pkg_topic
Starting >>> pkg_topic
/usr/lib/python3/dist-packages/setuptools/dist.py:723: UserWarning: Usage of dash-separated 'install-scr
ipts' will not be supported in future versions. Please use the underscore name 'install_scripts' instead
  warnings.warn(
Finished <<< pkg_topic [0.66s]

Summary: 1 package finished [0.85s]
yahboom@yahboom-virtual-machine:~/workspace$ source install/setup.bash
yahboom@yahboom-virtual-machine:~/workspace$
```

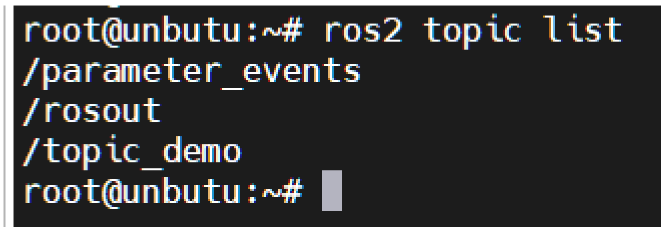
3.4 Running the Program

- After refreshing the environment variables, run the command

```
ros2 run pkg_topic publisher_demo
```

After the program successfully runs, nothing is printed. We can use the ros2 topic tool to view the data. First, check if there are any topics being published. Open another terminal and enter:

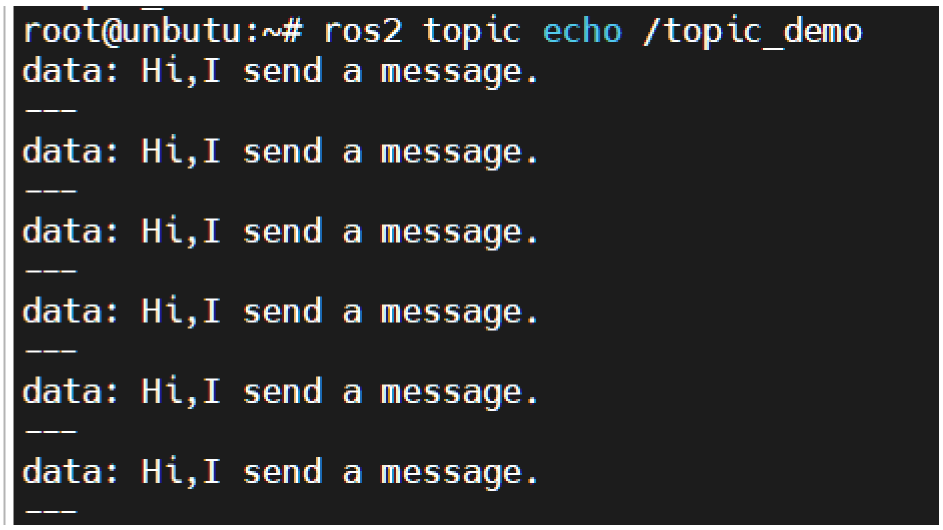
```
ros2 topic list
```



```
root@unbutu:~# ros2 topic list
/parameter_events
/rosout
/topic_demo
root@unbutu:~#
```

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

```
ros2 topic echo /topic_demo
```



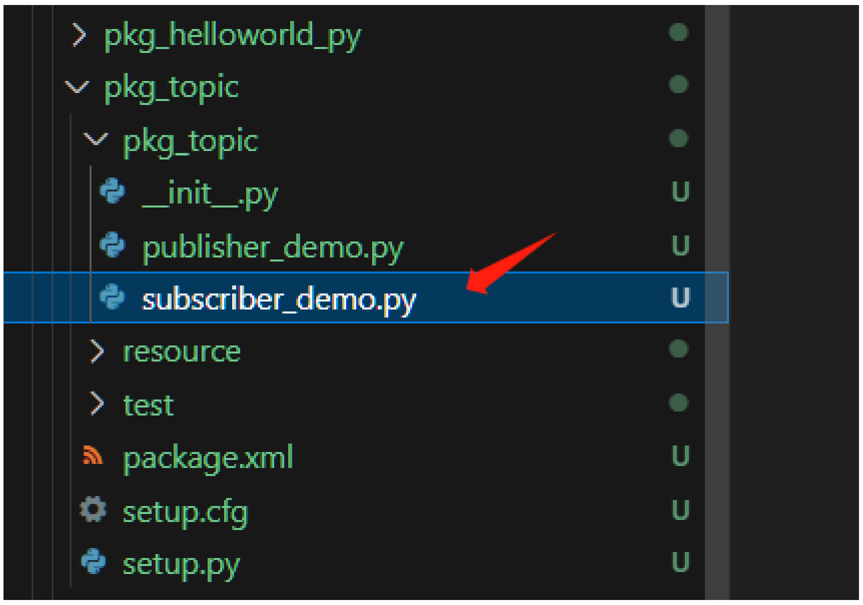
```
root@unbutu:~# ros2 topic echo /topic_demo
data: Hi,I send a message.
---
data: Hi,I send a message.
---
data: Hi,I send a message.
---
data: Hi,I send a message.
---
data: Hi,I send a message.
---
data: Hi,I send a message.
---
```

As you can see, the output "Hi, I send a message." from the terminal matches the line msg.data = "Hi, I send a message." in our code.

4. Subscriber Implementation

4.1 Creating a Subscriber

Create a new file, [subscriber_demo.py], in the same directory as [publisher_demo.py].

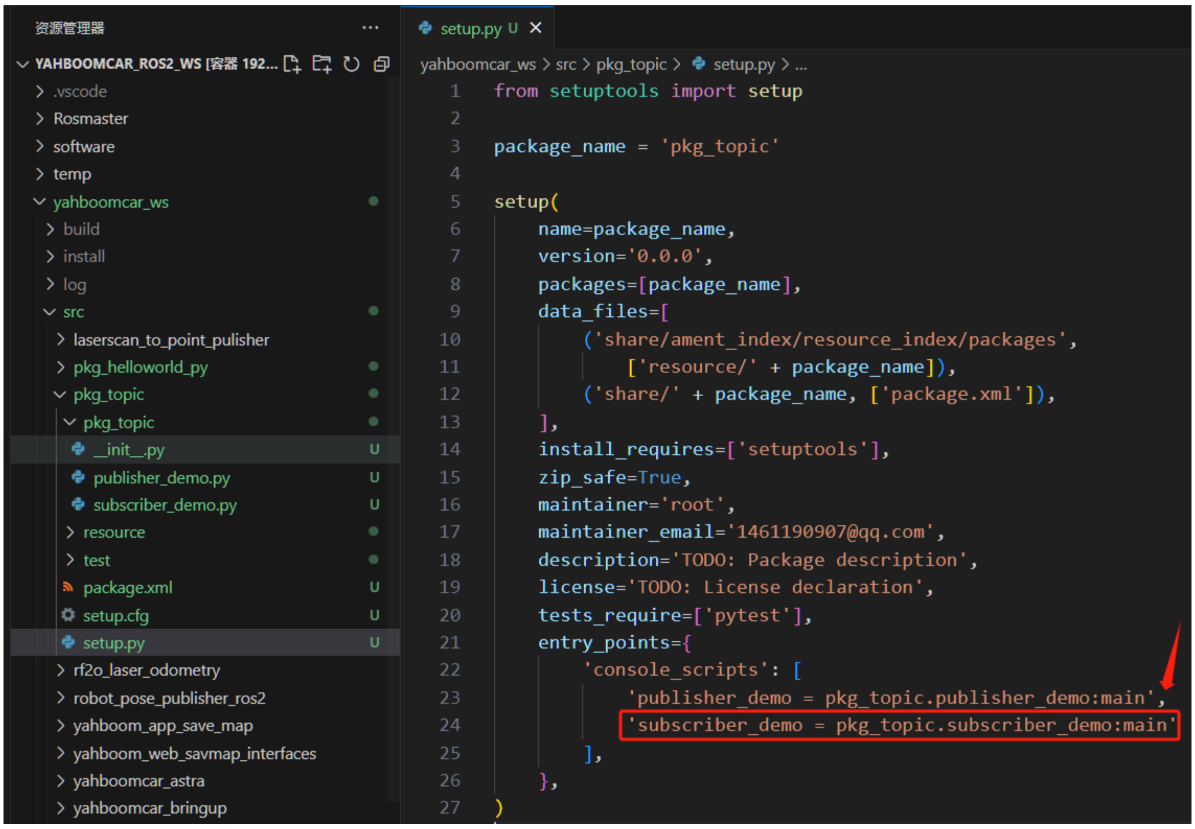


```
> pkg_helloworld_py
v pkg_topic
  v pkg_topic
      __init__.py          U
      publisher_demo.py    U
      subscriber_demo.py   U
    > resource
    > test
    package.xml            U
    setup.cfg              U
    setup.py               U
```

Next, edit [subscriber_demo.py] to implement the subscriber functionality and add the following code:

```
#Import related libraries
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class Topic_Sub(Node):
    def __init__(self,name):
        super().__init__(name)
        #Create a subscriber using create_subscription. The passed parameters
are: topic data type, topic name, callback function name, and queue length.
        self.sub =
self.create_subscription(String,"/topic_demo",self.sub_callback,1)
    #Callback function executes the program: prints the received message.
    def sub_callback(self,msg):
        # print(msg.data,flush=True)
        self.get_logger().info(msg.data)



def main():
    rclpy.init() #Initialize the ROS2 Python interface.
    sub_demo = Topic_Sub("subscriber_node")#Create the object and initialize it.
    rclpy.spin(sub_demo)
    sub_demo.destroy_node()  #Destroy the node object
    rclpy.shutdown()         #Shut down the ROS2 Python interface
```

4.2 Editing the Configuration File



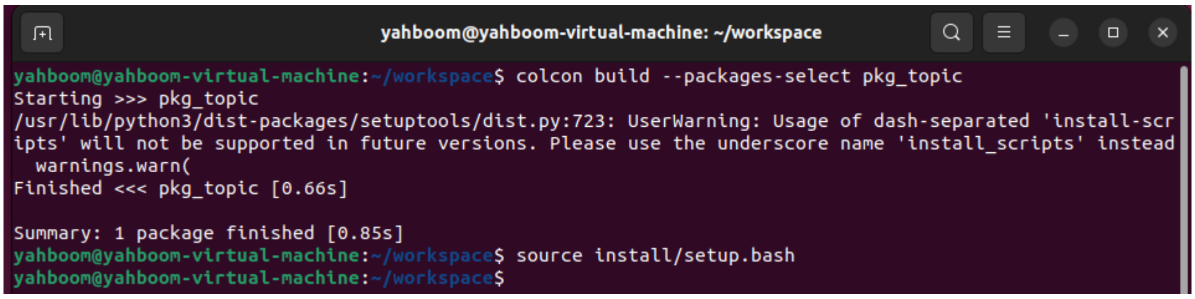
```
from setuptools import setup

package_name = 'pkg_topic'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='root',
    maintainer_email='1461190907@qq.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'publisher_demo = pkg_topic.publisher_demo:main',
            'subscriber_demo = pkg_topic.subscriber_demo:main'
        ],
    },
)
```

4.3 Compile the Workspace

- Compile the package

```
colcon build --packages-select pkg_topic
```

- Refresh the environment variables in the workspace

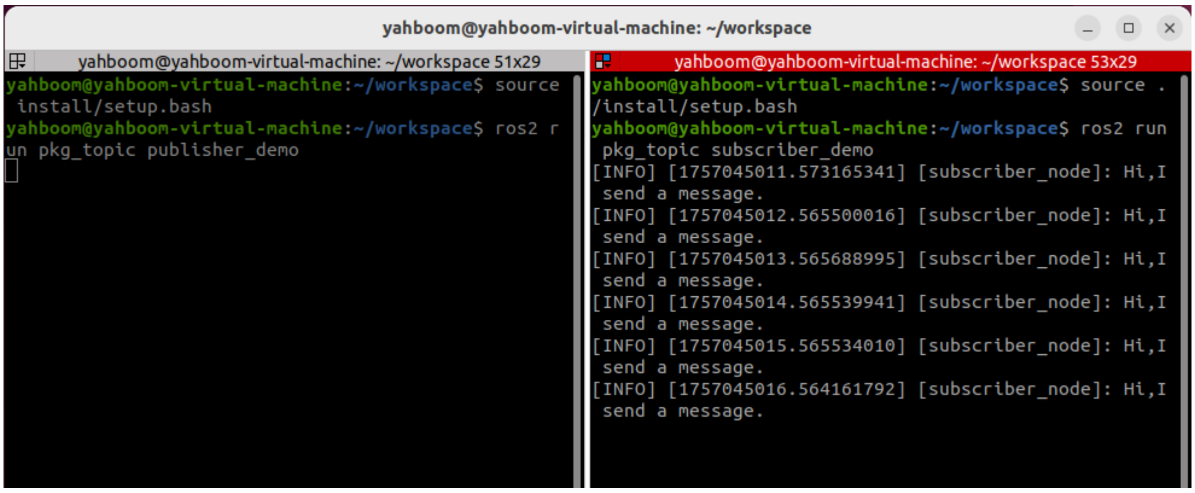


```
yahboom@yahboom-virtual-machine:~/workspace$ colcon build --packages-select pkg_topic
Starting >>> pkg_topic
/usr/lib/python3/dist-packages/setuptools/dist.py:723: UserWarning: Usage of dash-separated 'install-scr
ipts' will not be supported in future versions. Please use the underscore name 'install_scripts' instead
  warnings.warn(
Finished <<< pkg_topic [0.66s]

Summary: 1 package finished [0.85s]
yahboom@yahboom-virtual-machine:~/workspace$ source install/setup.bash
yahboom@yahboom-virtual-machine:~/workspace$
```

4.4 Run the Program

Execute the following command in a separate terminal:

```
# Start the publisher node
ros2 run pkg_topic publisher_demo
# Start the subscriber node
ros2 run pkg_topic subscriber_demo
```



```
yahboom@yahboom-virtual-machine:~/workspace$ source
 install/setup.bash
yahboom@yahboom-virtual-machine:~/workspace$ ros2 r
un pkg_topic publisher_demo
```

```
yahboom@yahboom-virtual-machine:~/workspace$ source .
/install/setup.bash
yahboom@yahboom-virtual-machine:~/workspace$ ros2 run
 pkg_topic subscriber_demo
[INFO] [1757045011.573165341] [subscriber_node]: Hi,I
 send a message.
[INFO] [1757045012.565500016] [subscriber_node]: Hi,I
 send a message.
[INFO] [1757045013.565688995] [subscriber_node]: Hi,I
 send a message.
[INFO] [1757045014.565539941] [subscriber_node]: Hi,I
 send a message.
[INFO] [1757045015.565534010] [subscriber_node]: Hi,I
 send a message.
[INFO] [1757045016.564161792] [subscriber_node]: Hi,I
 send a message.
```

As shown in the figure above, the terminal running the subscriber node will print the information published by the publisher, /topic_demo.