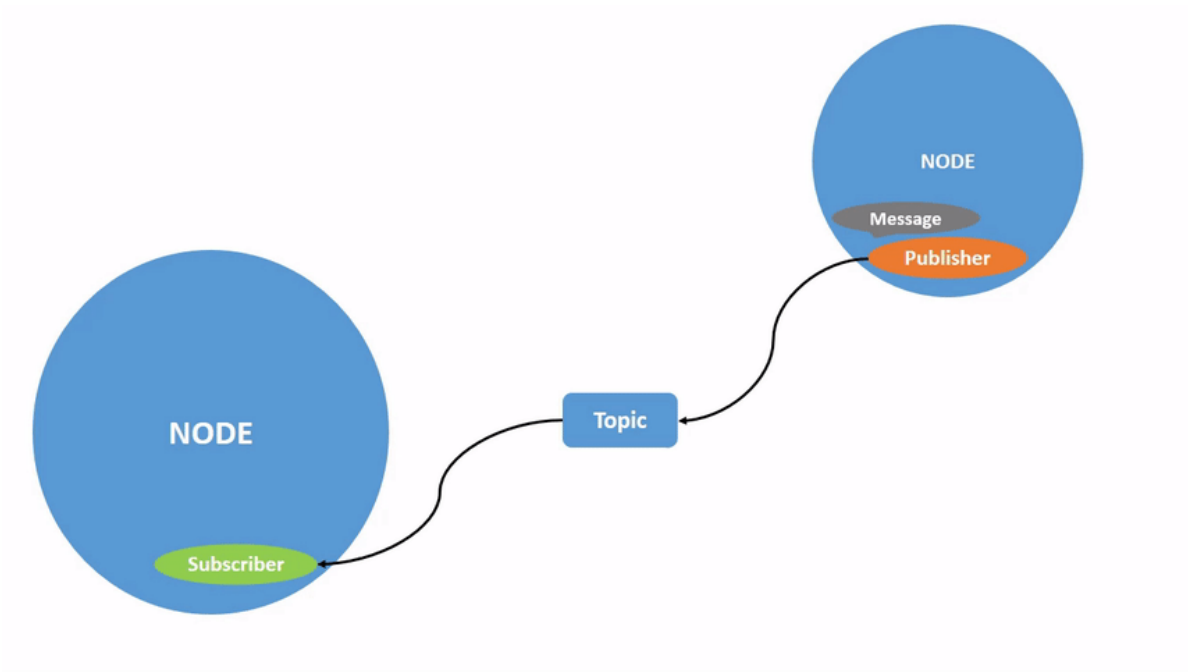
7、ROS2话题通讯

1、话题通讯简介

话题通讯是ROS2使用频率最高的一种通信方式，有发布者发布指定话题的数据，订阅者只要订阅了该话题的数据，就可以接收到数据。

话题通信是基于发布/订阅模型，如图：



话题数据传输的特性是从一个节点到另外一个节点，发送数据的对象称之为**发布者**，接收数据的对象称之为**订阅者**，每一个话题都需要有一个名字，传输的数据也需要有固定的数据类型。

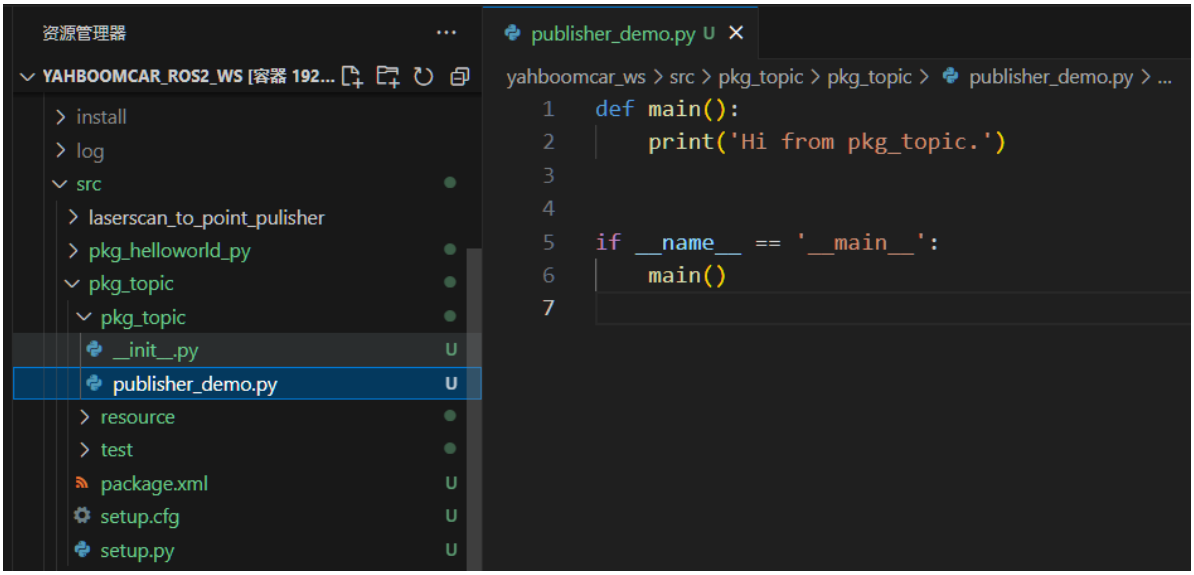
接下来就说明下如何使用Python语言实现节点之间的话题通讯。

2、新建功能包

- 切换到工作空间src目录下
- 新建pkg_topic功能包

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

执行完上述命令，会创建pkg_topic功能包，同时会创建一个publisher_demo的节点，并且已经配置好相关的配置文件



3、发布方实现

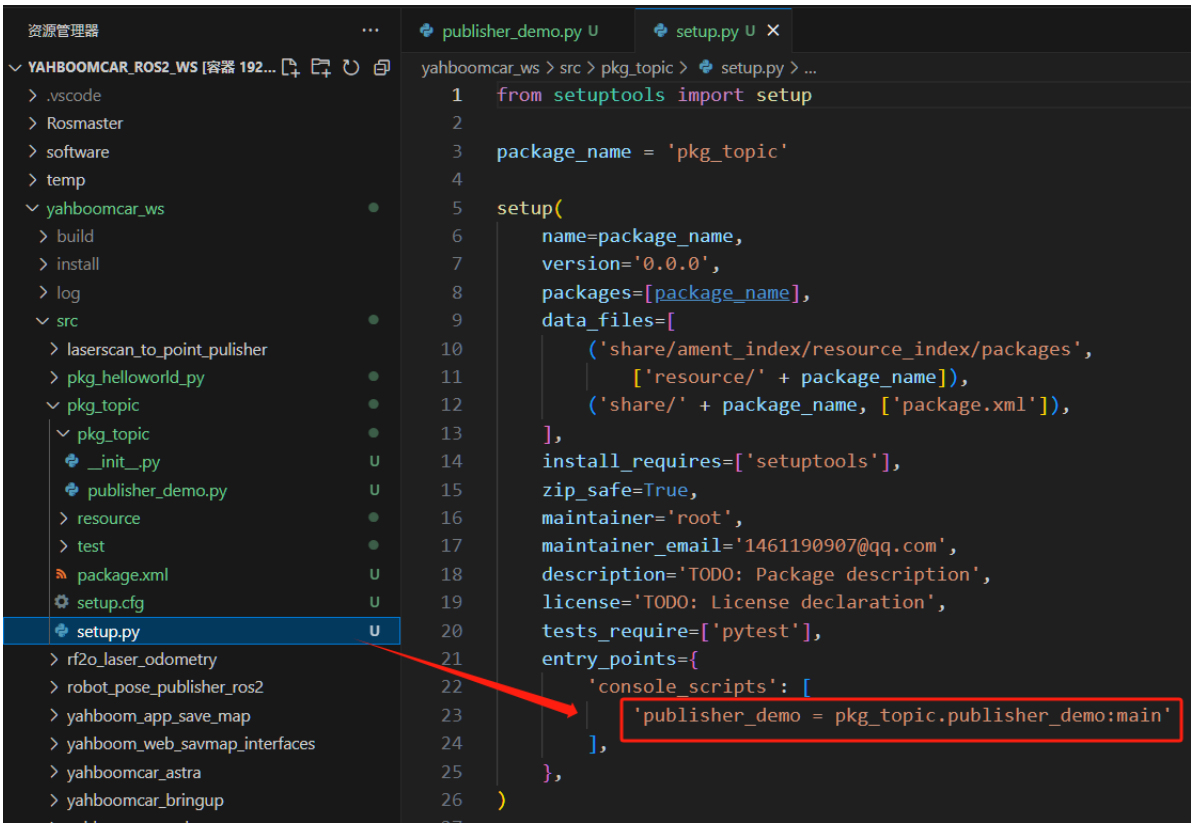
3.1 创建发布方

接下来编辑【publisher_demo.py】实现发布方的功能，添加如下代码：

```
#导入rclpy库
import rclpy
from rclpy.node import Node
#导入String字符串消息
from std_msgs.msg import String
#创建一个继承于Node基类的Topic_Pub节点子类 传入一个参数name
class Topic_Pub(Node):
    def __init__(self,name):
        super().__init__(name)
        #创建一个发布者，使用create_publisher的函数，传入的参数分别是：
        #话题数据类型、话题名称、保存消息的队列长度
        self.pub = self.create_publisher(String,"/topic_demo",1)
        #创建一个定时器，间隔1s进入中断处理函数，传入的参数分别是：
        #中断函数执行的间隔时间，中断处理函数
        self.timer = self.create_timer(1,self.pub_msg)
    #定义中断处理函数
    def pub_msg(self):
        msg = String()  #创建一个String类型的变量msg
        msg.data = "Hi,I send a message." #给msg里边的data赋值
        self.pub.publish(msg) #发布话题数据

#主函数
def main():
    rclpy.init() #初始化
    pub_demo = Topic_Pub("publisher_node") #创建Topic_Pub类对象，传入的参数就是节点的
名字
    rclpy.spin(pub_demo)      #执行rclpy.spin函数，里边传入一个参数，参数是刚才创建好的
Topic_Pub类对象
    pub_demo.destroy_node()  #销毁节点对象
    rclpy.shutdown()          #关闭ROS2 Python接口
```

3.2 编辑配置文件

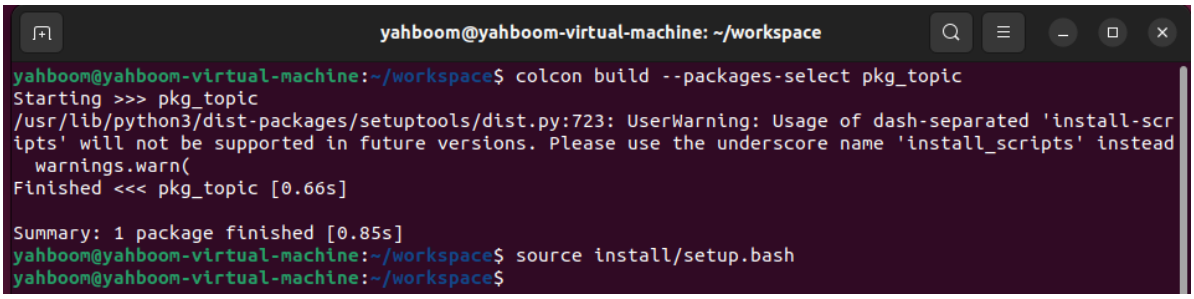


3.3 编译功能包

- 编译功能包

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

- 在工作空间下刷新环境变量



3.4 运行程序

- 刷新环境变量后运行命令

```
ros2 run pkg_topic publisher_demo
```

程序成功运行后是没有打印任何东西的，我们可以通过ros2 topic 工具来查看数据，首先，先查看这个是否有话题发布，开启另一个终端输入：

```
ros2 topic list
```

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

这个topic_demo就是程序里定义的话题数据了，接下来，我们用ros2 topic echo来打印下这个数据，终端输入：

```
ros2 topic echo /topic_demo
```

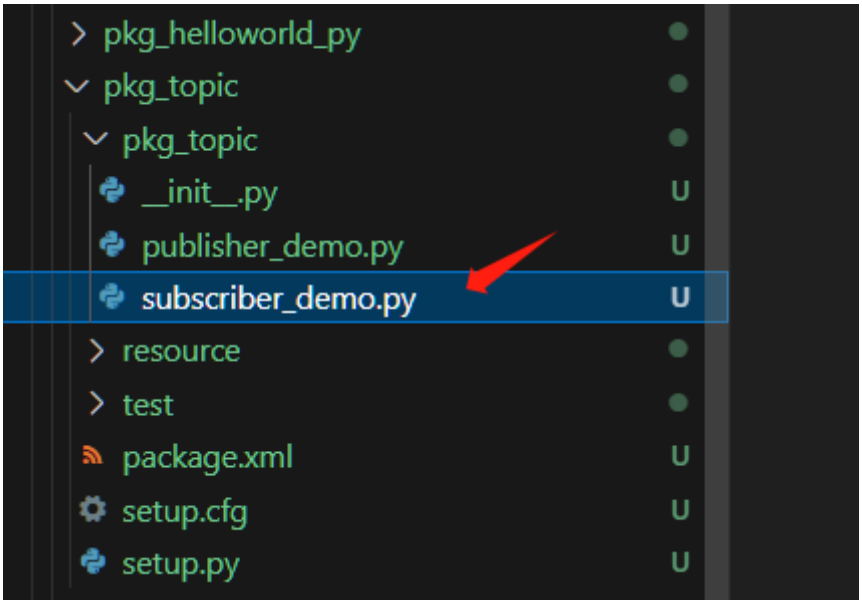
```
root@unbutu:~# ros2 topic echo /topic_demo
data: Hi,I send a message.
---
data: Hi,I send a message.
---
data: Hi,I send a message.
---
data: Hi,I send a message.
---
data: Hi,I send a message.
---
data: Hi,I send a message.
---
```

可以看出，终端打印的"Hi,I send a message."与我们代码里边的msg.data = "Hi,I send a message."一致。

4、订阅方实现

4.1 创建订阅方

在【publisher_demo.py】同级目录下新建文件【subscriber_demo.py】



接下来编辑【subscriber_demo.py】实现订阅方的功能，添加如下代码：

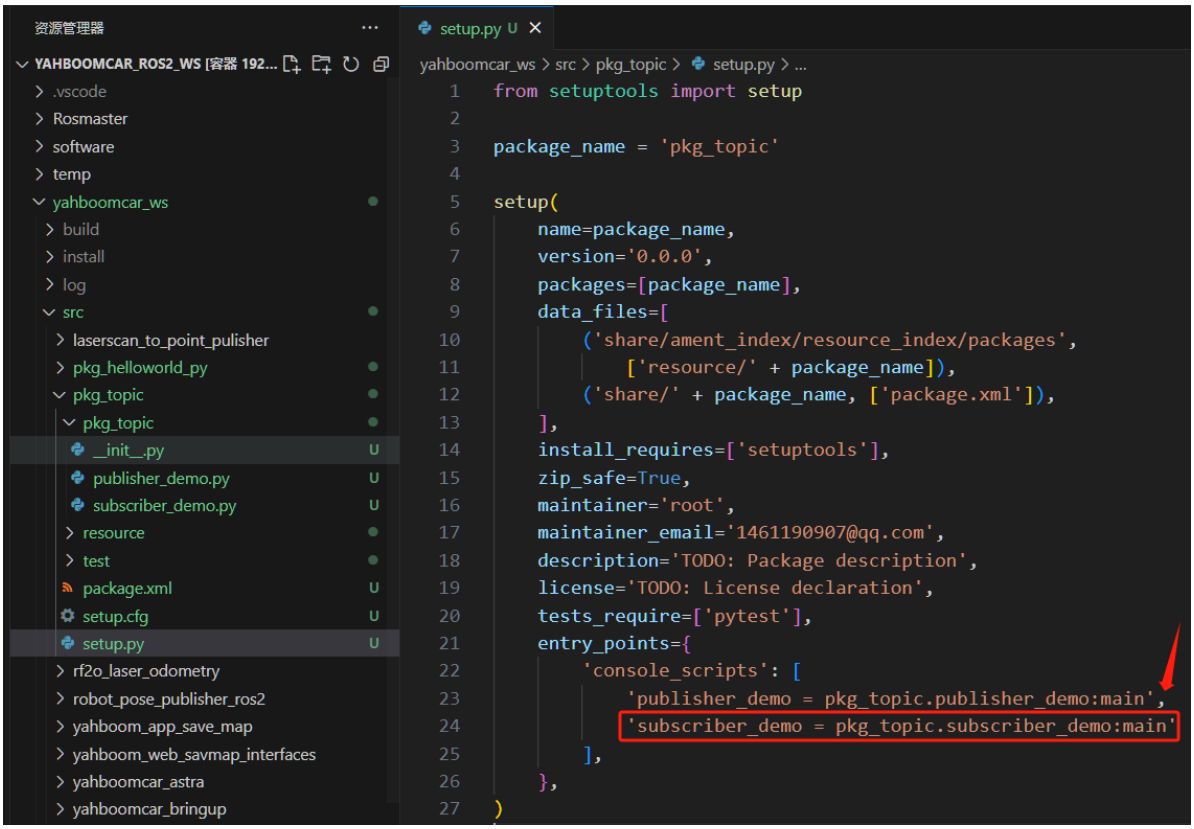
```
#导入相关的库
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class Topic_Sub(Node):
    def __init__(self,name):
        super().__init__(name)
        #创建订阅者使用的是create_subscription，传入的参数分别是：话题数据类型，话题名称，回调函数名称，队列长度
        self.sub = self.create_subscription(String,"/topic_demo",self.sub_callback,1)
    #回调函数执行程序：打印接收的到信息
    def sub_callback(self,msg):
        # print(msg.data,flush=True)
        self.get_logger().info(msg.data)



def main():
    rclpy.init() #ROS2 Python接口初始化
    sub_demo = Topic_Sub("subscriber_node") # 创建对象并进行初始化
    rclpy.spin(sub_demo)
    sub_demo.destroy_node()  #销毁节点对象
    rclpy.shutdown()         #关闭ROS2 Python接口
```

4.2 编辑配置文件

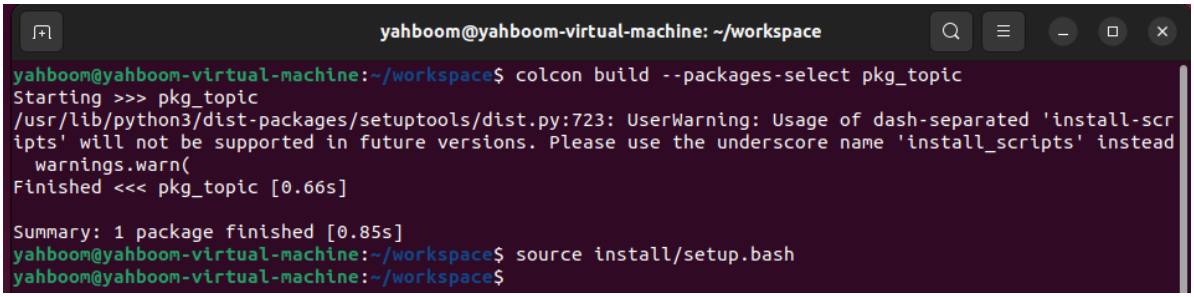


4.3 编译工作空间

- 编译功能包

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

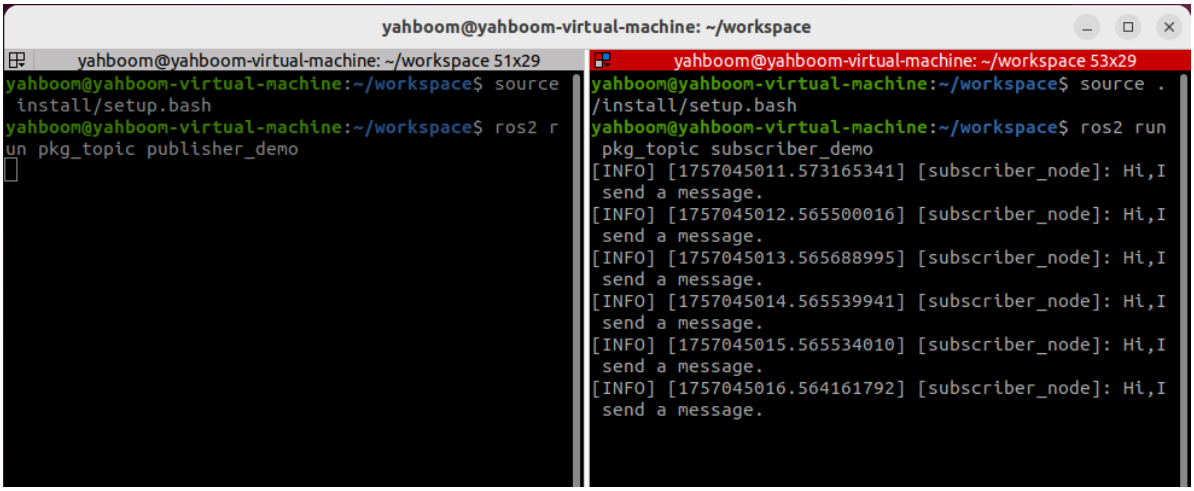
- 在工作空间下刷新环境变量



4.4 运行程序

分割终端执行如下：

```
#启动发布者节点
ros2 run pkg_topic publisher_demo
#启动订阅者节点
ros2 run pkg_topic subscriber_demo
```



如上图所示，运行订阅者这点的终端会打印发布者发布的/topic_demo的信息。