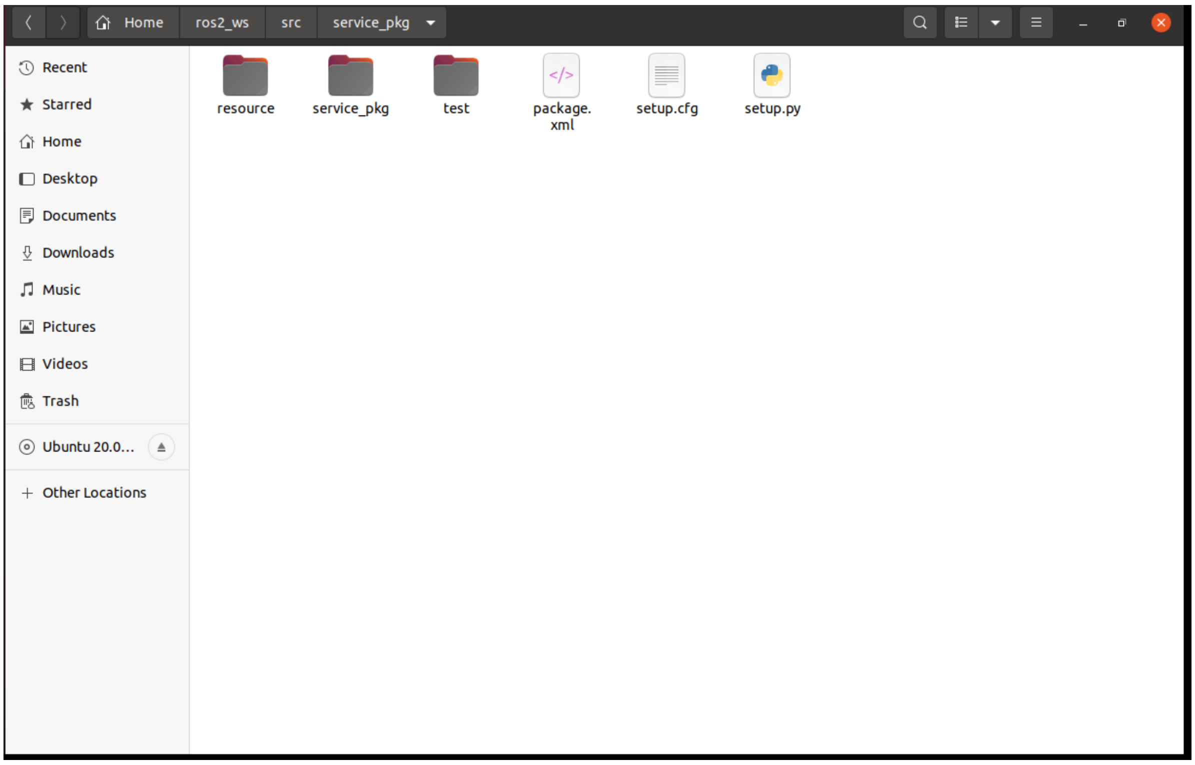
4. ROS2 service communication

Service communication is one of the communication modes of ROS2 nodes. Different from topic communication, service communication has a feedback mechanism, which will feedback the result of service. Therefore, service communication is mostly used in programs that need feedback on the result, such as the baby turtle routine, which calls the Spawn service to generate a turtle, and when the service is completed (after the turtle is generated), the turtle name is printed. Next, I will explain how to use Python language to implement service communication between nodes.

1. Create a feature package

Create a new function pack in the src of the workspace directory created earlier, and enter the terminal,

```
cd ~/ros2_ws/src
ros2 pkg create --build-type ament_python service_pkg
```



3. Write a server-side python file

3.1 Program source code

Create a new file called server_demo.py,

```
cd ~/ros2_ws/src/service_pkg/service_pkg
gedit server_demo.py
```

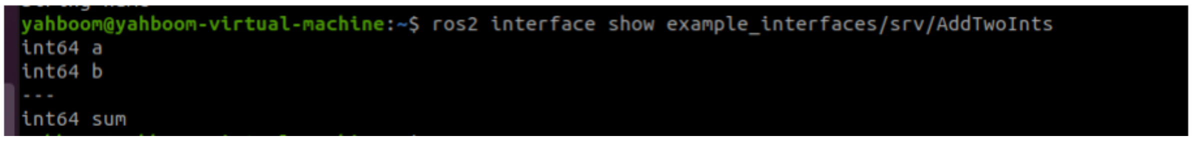
Copy the following sections into the file,

```
#导入相关的库文件
```

```
# Import related library files
import rclpy
from rclpy.node import Node
from example_interfaces.srv import AddTwoInts

class Service_Server(Node):
    def __init__(self,name):
        super().__init__(name)
        #创建一个服务端，使用的是create_service函数，传入的参数分别是：
        # Create a server using the create_service function with the following
parameters:
        #服务数据的数据类型、服务的名称，服务回调函数（也就是服务的内容）
        # The data type of the service data, the name of the service, the
service callback function (that is, the content of the service)
        self.srv = self.create_service(AddTwoInts, '/add_two_ints',
self.Add2Ints_callback)
    #这里的服务回调函数的内容是把两个整型数相加，然后返回相加的结果
    # The content of the service callback function here is to add two integers
and return the result of the addition
    def Add2Ints_callback(self,request,response):
        response.sum = request.a + request.b
        print("response.sum = ",response.sum)
        return response
def main():
    rclpy.init()
    server_demo = Service_Server("publisher_node")
    rclpy.spin(server_demo)
```

Take a look at the service callback function, Add2Ints_callback, where the required parameters are not only self, but also request and response. request is the parameter required by the service, and response is the feedback result of the service. Request. a and request.b are the content of the request part, and response.sum is the content of the response part. Here, first take a look at the data of this type AddTwoInts, which can be viewed by using the following command.

```
ros2 interface show example_interfaces/srv/AddTwoInts
```



```
yahboom@yahboom-virtual-machine:~$ ros2 interface show example_interfaces/srv/AddTwoInts
int64 a
int64 b
---
int64 sum
```

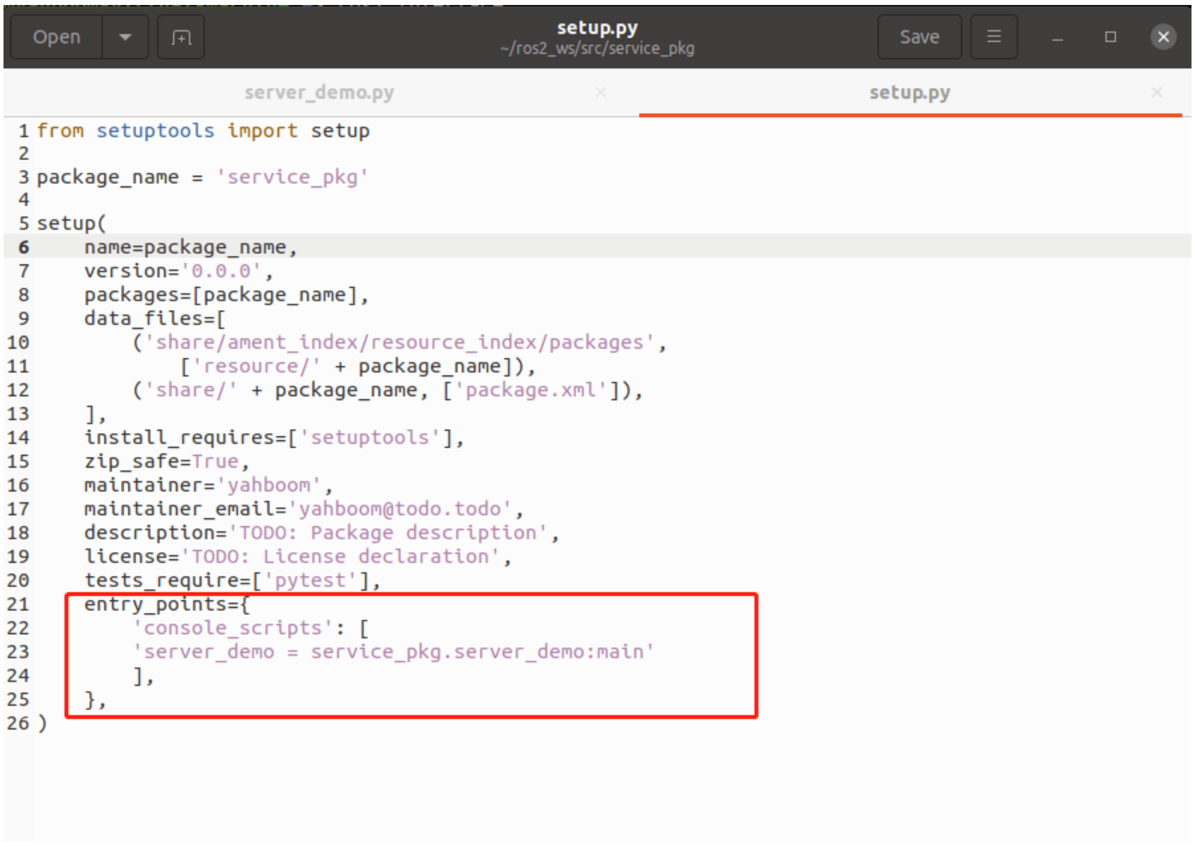
The section divides this type of data into two parts, with the top representing request and the bottom representing response. Then the respective fields and their own variables, such as int64a, int64b, all in the parameters are passed, need to specify the value of a, b is what. Again, the feedback needs to specify the value of sum.

3.2. Modify the setup.py file

Terminal input,

```
cd ~/ros2_ws/src/service_pkg
gedit setup.py
```

Find the location shown below,



```python
from setuptools import setup

package_name = 'service_pkg'

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='yahboom',
    maintainer_email='yahboom@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
        'server_demo = service_pkg.server_demo:main'
        ],
    },
)
```

Add the following in 'console_scripts': [].

```
'server_demo = service_pkg.server_demo:main'
```

3.3. Compile workspace

```
cd ~/ros2_ws
colcon build
```

After compiling, refresh the environment variables in the workspace,

```
source ~/ros2_ws/install/setup.bash
```

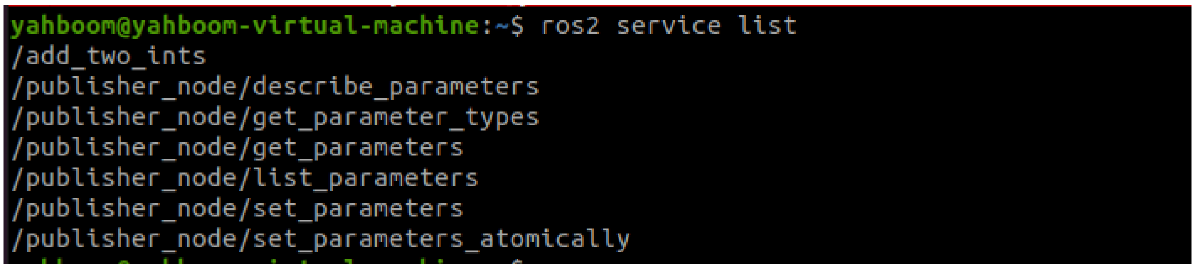
3.4. Run the program

Terminal input,

```
ros2 run service_pkg server_demo
```

After running, the service is not called, so there is no feedback data, you can call the service through the command line, first query what services are currently available, terminal input,

```
ros2 service list
```

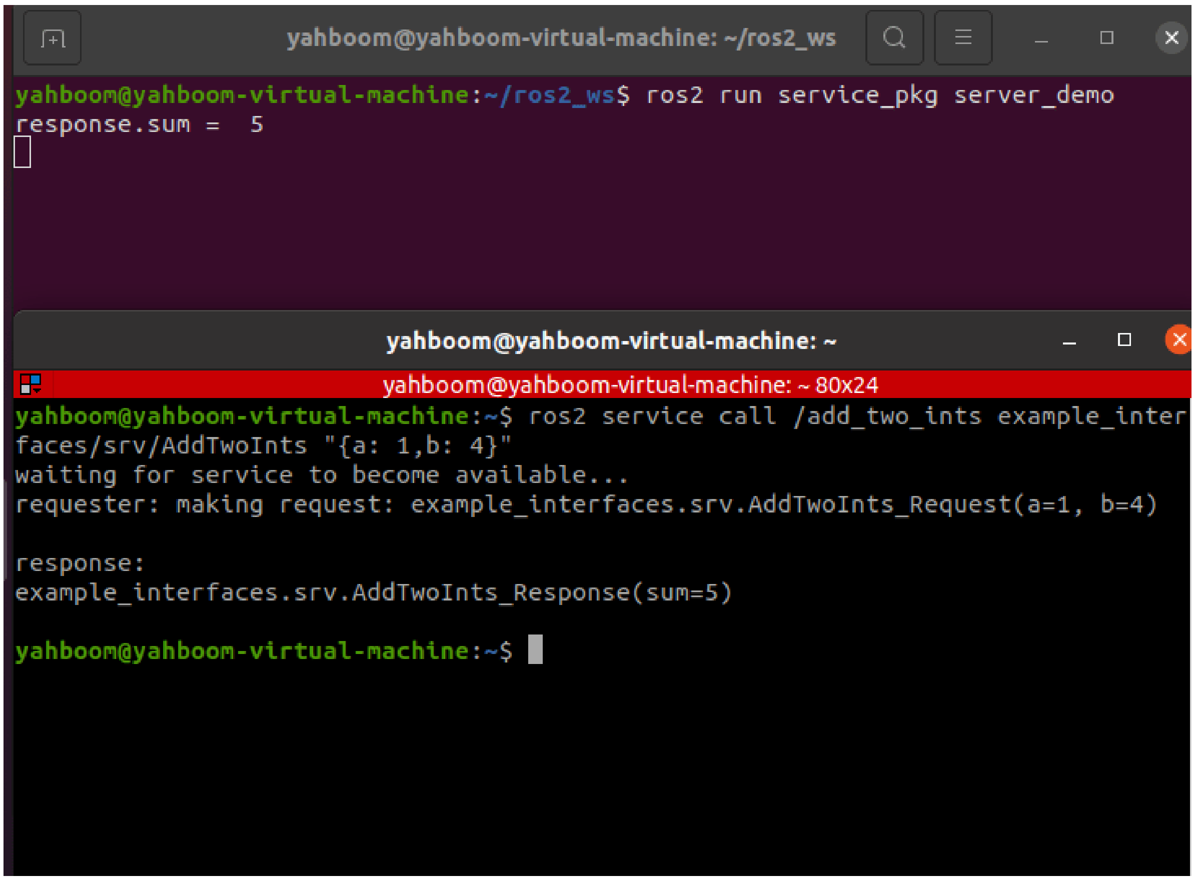


```
yahboom@yahboom-virtual-machine:~$ ros2 service list
/add_two_ints
/publisher_node/describe_parameters
/publisher_node/get_parameter_types
/publisher_node/get_parameters
/publisher_node/list_parameters
/publisher_node/set_parameters
/publisher_node/set_parameters_atomically
```

/add_two_ints is the service we need to call, by the following command, terminal input,

```
ros2 service call /add_two_ints example_interfaces/srv/AddTwoInts "{a: 1,b: 4}"
```

Here we assign the value of a to 1 and the value of b to 4, that is, we call the service to compute the sum of 1 and 4,



```
yahboom@yahboom-virtual-machine: ~/ros2_ws
yahboom@yahboom-virtual-machine:~/ros2_ws$ ros2 run service_pkg server_demo
response.sum =  5
```

```
yahboom@yahboom-virtual-machine: ~
yahboom@yahboom-virtual-machine: ~ 80x24
yahboom@yahboom-virtual-machine:~$ ros2 service call /add_two_ints example_inter
faces/srv/AddTwoInts "{a: 1,b: 4}"
waiting for service to become available...
requester: making request: example_interfaces.srv.AddTwoInts_Request(a=1, b=4)

response:
example_interfaces.srv.AddTwoInts_Response(sum=5)

yahboom@yahboom-virtual-machine:~$
```

As can be seen from the figure above, after calling the service, the feedback result is 5, and the terminal running the server also prints the feedback value.

4. Write client-side python files

4.1. Program source code

Create a new file named client_demo.py,

```
cd ~/ros2_ws/srcservice_pkg/service_pkg
gedit client_demo.py
```

Copy the following into it,

```
#导入相关的库
# Import related libraries
```

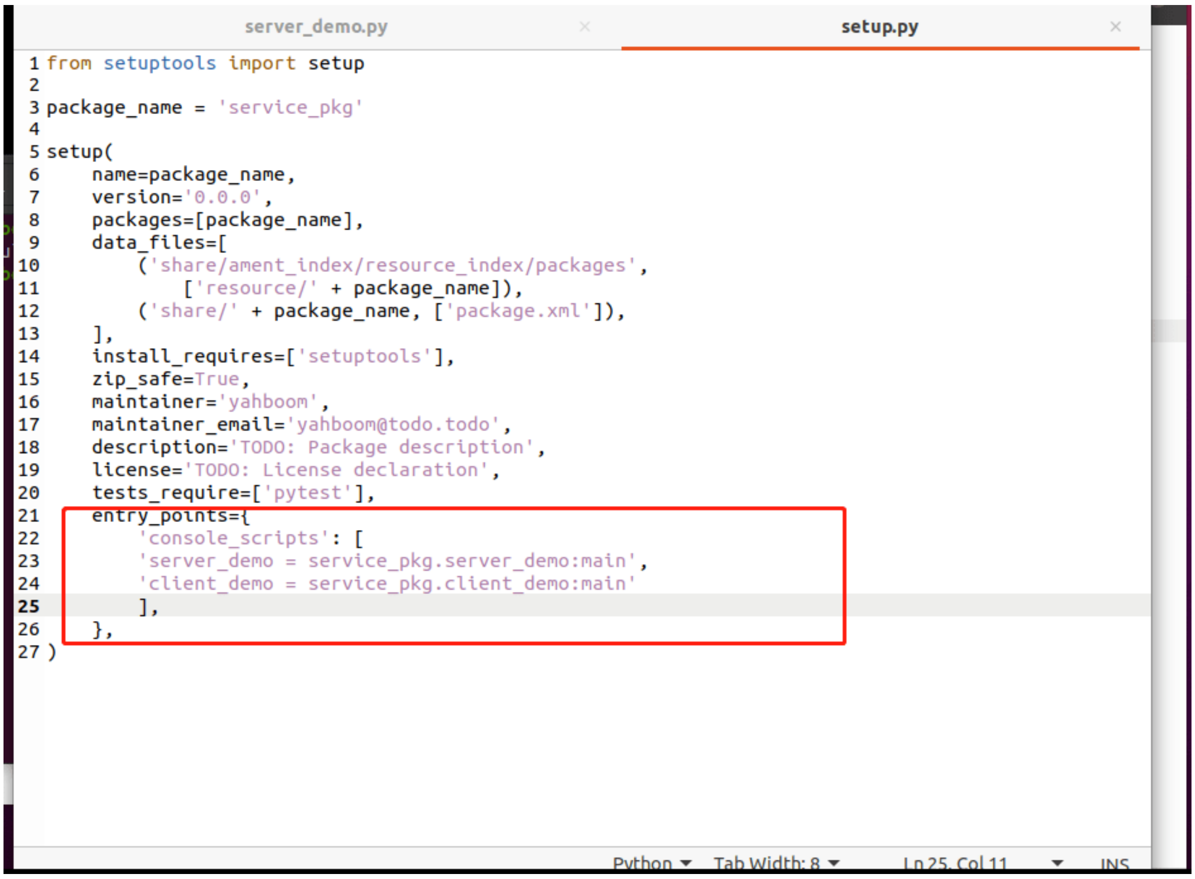
```
import rclpy
from rclpy.node import Node
from example_interfaces.srv import AddTwoInts

class Service_Client(Node):
    def __init__(self,name):
        super().__init__(name)
        #创建客户端，使用的是create_client函数，传入的参数是服务数据的数据类型、服务的话题名
称
        # Create client, using create_client function, passed in parameters are
the data type of the service data, the service topic name
        self.client = self.create_client(AddTwoInts,'/add_two_ints')
        # 循环等待服务器端成功启动
        # Loop waiting for server side to start successfully
        while not self.client.wait_for_service(timeout_sec=1.0):
            print("service not available, waiting again...")
        # 创建服务请求的数据对象
        # Create the data object for the service request
        self.request = AddTwoInts.Request()

    def send_request(self):
        self.request.a = 10
        self.request.b = 90
        #发送服务请求
        # Send a service request
        self.future = self.client.call_async(self.request)

def main():
    rclpy.init() #节点初始化  # Node initialization
    service_client = Service_Client("client_node") #创建对象# Create object
    service_client.send_request() #发送服务请求
    # Send a service request
    while rclpy.ok():
        rclpy.spin_once(service_client)
        #判断数据是否处理完成
        # Determine whether the data processing is complete
        if service_client.future.done():
            try:
                #获得服务反馈的信息并且打印
                # Get service feedback information and print it
                response = service_client.future.result()
                print("Result = ",response.sum)
            except Exception as e:
                service_client.get_logger().info('Service call failed %r' %
(e,))

        break
```

4.2. Modify the setup.py file

Terminal input,

```
cd ~/ros2_ws/src/topic_pkg
gedit setup.py
```

Find the location shown below,



```python
from setuptools import setup

package_name = 'service_pkg'

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='yahboom',
    maintainer_email='yahboom@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
        'server_demo = service_pkg.server_demo:main',
        'client_demo = service_pkg.client_demo:main'
        ],
    },
)
```

Add the following in 'console_scripts': [],

```
'client_demo = service_pkg.client_demo:main'
```

```
'client_demo = service_pkg.client_demo:main'
```

4.3. Compile workspace

```
cd ~/ros2_ws
colcon build
```

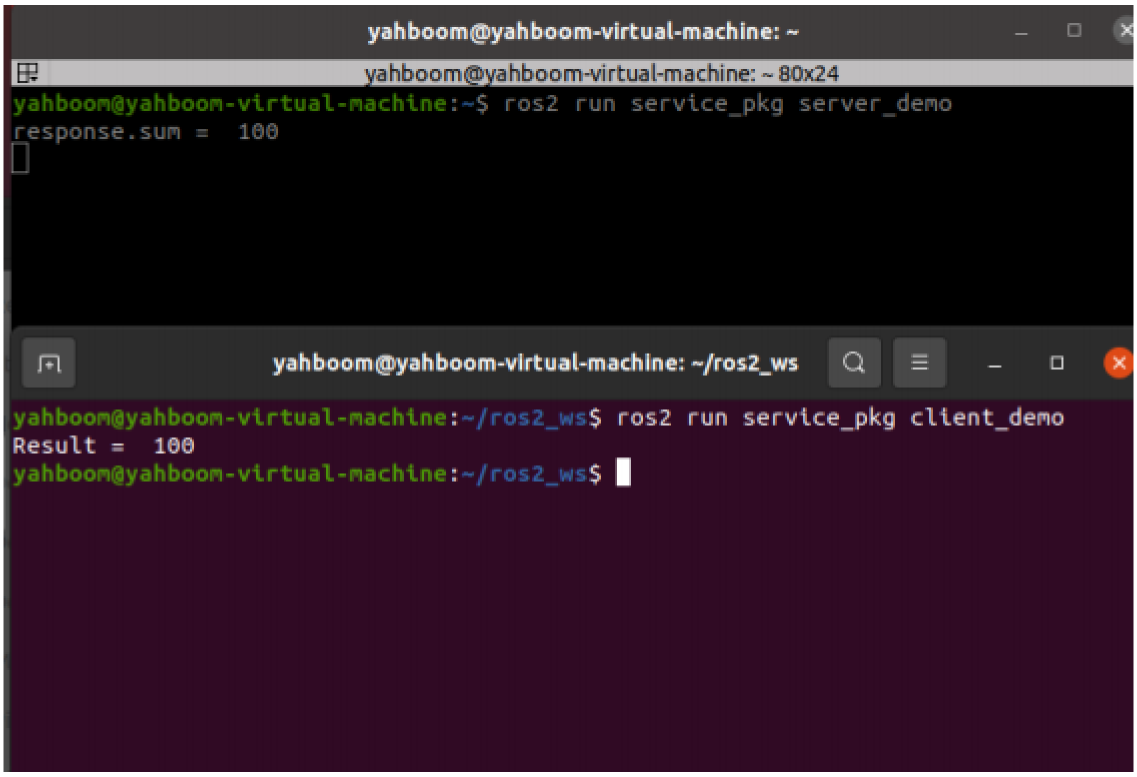
After compiling, refresh the environment variables in the workspace,

```
source ~/ros2_ws/install/setup.bash
```

4.4. Run the program

Terminal input,

```
#启动服务端
# Start the server
ros2 run service_pkg server_demo
#启动客户端
# Start the client
ros2 run service_pkg client_demo
```


yahboom@yahboom-virtual-machine: ~
yahboom@yahboom-virtual-machine: ~ 80x24

```
yahboom@yahboom-virtual-machine:~$ ros2 run service_pkg server_demo
response.sum =  100
```

yahboom@yahboom-virtual-machine: ~/ros2_ws

```
yahboom@yahboom-virtual-machine:~/ros2_ws$ ros2 run service_pkg client_demo
Result =  100
yahboom@yahboom-virtual-machine:~/ros2_ws$
```

First run the server, then run the client, the client provides a=10, b=90, the server summed, the result is 100, the result is printed in both terminals.