

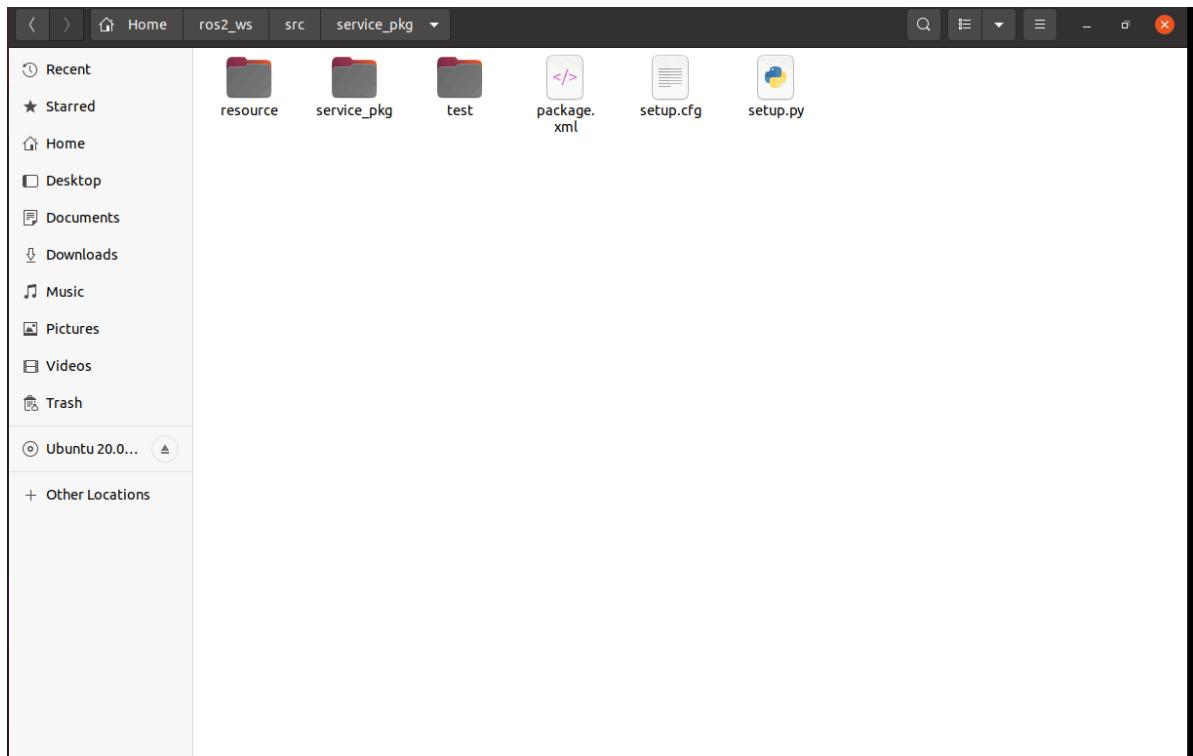
4. ROS2 Service Communication

Service communication is one of the ROS2 node communication methods, it is different from topic communication, service communication has a feedback mechanism, it will feedback the result of the service. Therefore, service communication is mostly used in programs that need feedback results, such as the small turtle routine, which calls the Spawn service to generate a turtle, and prints out the turtle's name after the service is completed (after the turtle is generated). The next step is to explain how to use the Python language to implement the service communication between nodes.

1. Create a new function package

Create a new function package in the src of the previously created workspace directory, and enter, in the terminal.

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



3. Writing server-side python files

3.1 Programme source code

Create a new file named server_demo.py.

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

Copy the following section into the file.

```
#导入相关的库文件
```

```

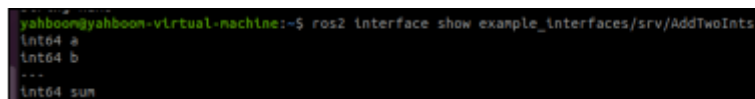
# Import the relevant library files
import rclpy
from rclpy.node import Node
from example_interfaces.srv import AddTwoInts

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：
        #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）
        # Create a server, using the create_service function, with the
parameters passed in being:
        # 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 together 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)

```

Focus on the service callback function, Add2Ints_callback, here you need to pass in the parameters in addition to self, there is the request and response, request is the parameters needed by the service, response is the service feedback results. request.a and request.b is the request part of the content, response.sum is the response part of the content, here first look at the AddTwoInts this type of data is how, you can use the following command to see. content, response.sum is the response part of the content, here first look at the AddTwoInts this type of data is how, you can use the following command to view.

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



```

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

```

-- part of the data of this type is divided into two parts, the upper part represents the request, the lower part represents the response. and then their respective fields and their respective variables, such as int64 a, int64 b, all in the then pass the parameter is, you need to specify the value of a, b is how much is it. Similarly, the result of the feedback also needs to specify the value of sum.

3.2 Modify the setup.py file.

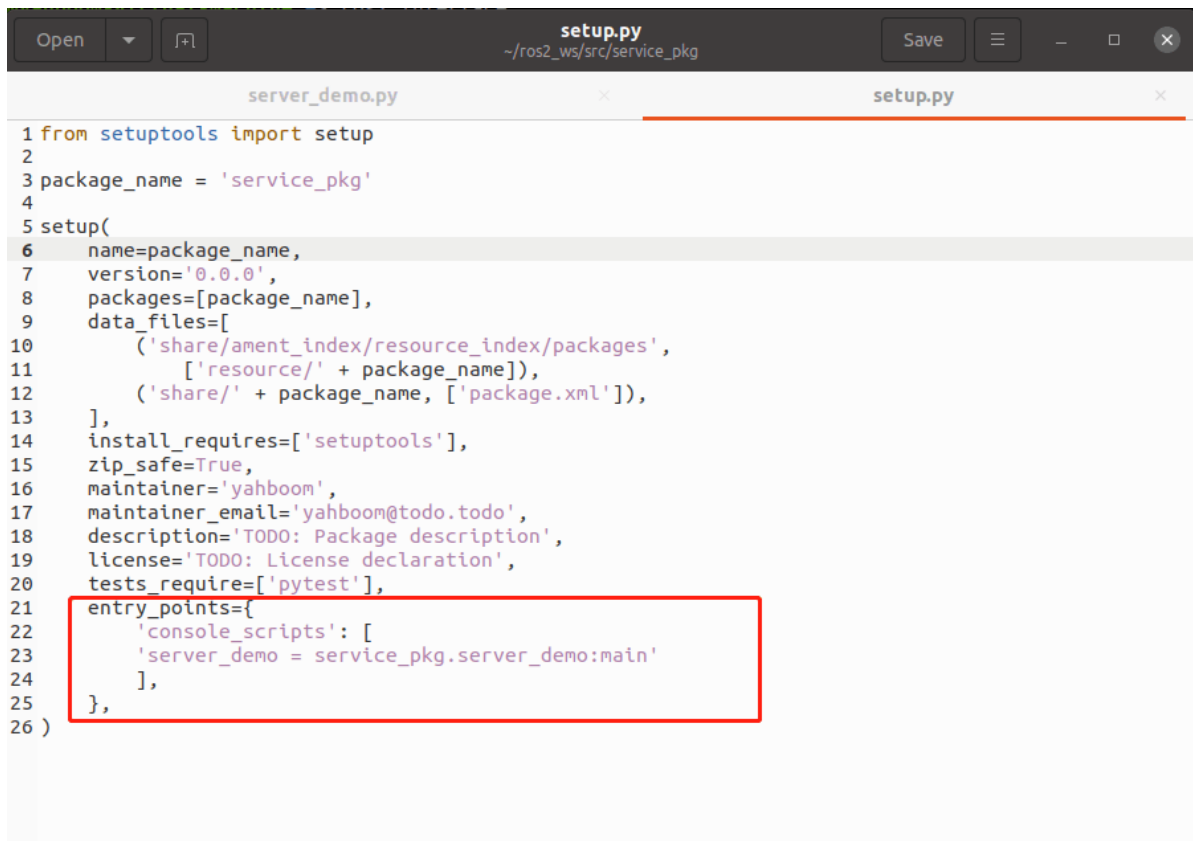
Terminal input.

```

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

```

Find the location shown below.



```
1 from setuptools import setup
2
3 package_name = 'service_pkg'
4
5 setup(
6     name=package_name,
7     version='0.0.0',
8     packages=[package_name],
9     data_files=[
10         ('share/ament_index/resource_index/packages',
11          ['resource/' + package_name]),
12         ('share/' + package_name, ['package.xml']),
13     ],
14     install_requires=['setuptools'],
15     zip_safe=True,
16     maintainer='yahboom',
17     maintainer_email='yahboom@todo.todo',
18     description='TODO: Package description',
19     license='TODO: License declaration',
20     tests_require=['pytest'],
21     entry_points={
22         'console_scripts': [
23             'server_demo = service_pkg.server_demo:main'
24         ],
25     },
26 )
```

Add the following to 'console_scripts': [].

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

3.3. Compilation workspace

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

After compiling, refresh the workspace environment variables.

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

3.4. Run the programme

Terminal input.

```
ros2 run service_pkg server_demo
```

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

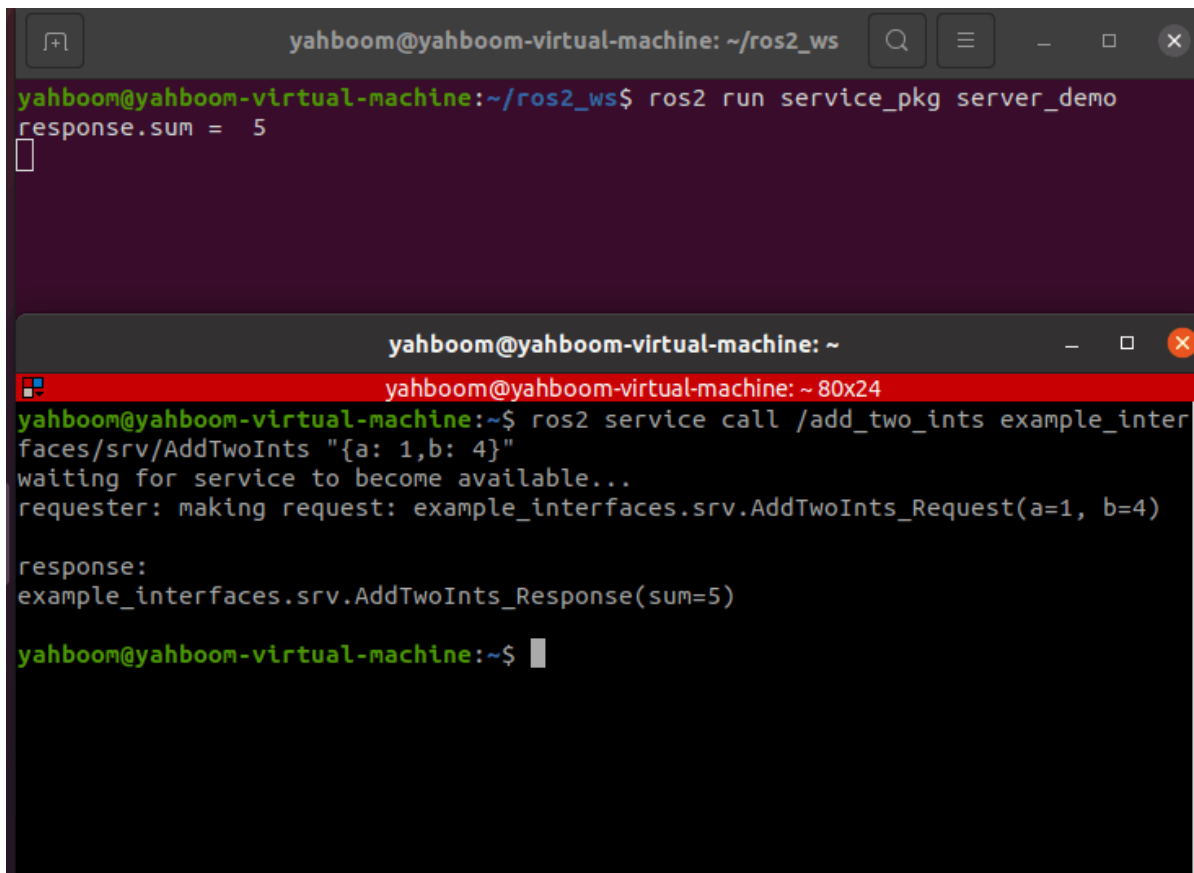
```
ros2 service list
```

```
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 invoke, which is done by the following command, entered in the terminal.

```
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.



The image shows two terminal windows. The top window shows the command `ros2 run service_pkg server_demo` being executed, with the output `response.sum = 5`. The bottom window shows the command `ros2 service call /add_two_ints example_interfaces/srv/AddTwoInts "{a: 1,b: 4}"` being executed, with the output `waiting for service to become available...`, `requester: making request: example_interfaces.srv.AddTwoInts_Request(a=1, b=4)`, and `response: example_interfaces.srv.AddTwoInts_Response(sum=5)`.

```
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 you can see from the above figure, after calling the service, the result returned is 5, and the terminal running the server also prints the value returned.

4. Write the client-side python file

4.1 Programme source code

Create a new file named `client_demo.py`.

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

Copy the following into the inside.

```
#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 the client, using the create_client function, the parameters
        # passed in are the data type of the service data, the topic name of the service
        self.client = self.create_client(AddTwoInts, '/add_two_ints')
        # 循环等待服务器端成功启动
        # Loop to wait for the server side to start up 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 data objects for service requests
        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 initialisation
    service_client = Service_Client("client_node") #创建对象
    # Create Objects
    service_client.send_request() #发送服务请求# Send service request
    while rclpy.ok():
        rclpy.spin_once(service_client)
        #判断数据是否处理完成
        # Determine if 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 and print
                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 Modifying the setup.py file


Terminal input.

```

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

```

Find the location shown below.



```
server_demo.py  setup.py
1 from setuptools import setup
2
3 package_name = 'service_pkg'
4
5 setup(
6     name=package_name,
7     version='0.0.0',
8     packages=[package_name],
9     data_files=[
10         ('share/ament_index/resource_index/packages',
11          ['resource/' + package_name]),
12         ('share/' + package_name, ['package.xml']),
13     ],
14     install_requires=['setuptools'],
15     zip_safe=True,
16     maintainer='yahboom',
17     maintainer_email='yahboom@todo.todo',
18     description='TODO: Package description',
19     license='TODO: License declaration',
20     tests_require=['pytest'],
21     entry_points={
22         'console_scripts': [
23             'server_demo = service_pkg.server_demo:main',
24             'client_demo = service_pkg.client_demo:main'
25         ],
26     },
27 )
```

Add the following to 'console_scripts': [].

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

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

4.3. Compilation workspace

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

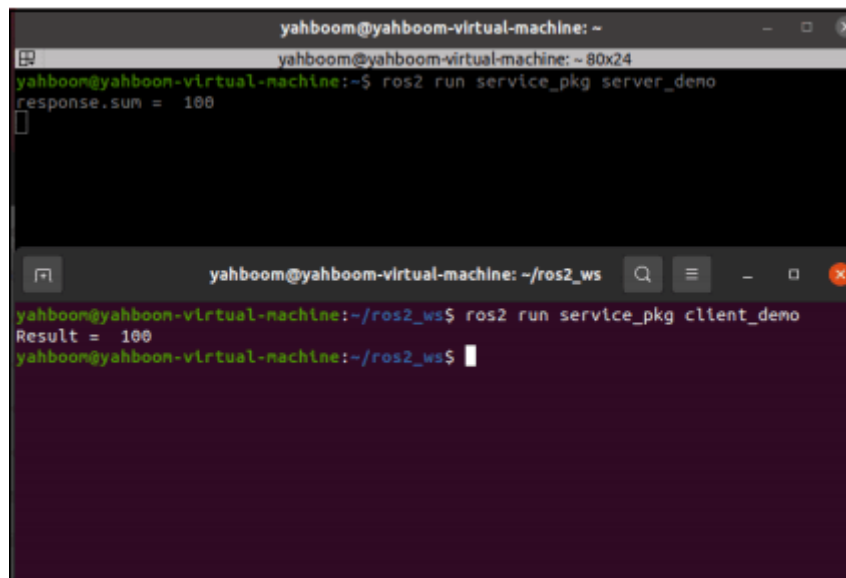
After compiling, refresh the workspace environment variables.

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

4.4. Run the programme

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
```



The image shows two terminal windows from a virtual machine. The top window, titled 'yahboom@yahboom-virtual-machine: ~', has a terminal icon in its title bar. It shows the command 'ros2 run service_pkg server_demo' being executed, followed by the output 'response.sum = 100'. The bottom window, titled 'yahboom@yahboom-virtual-machine: ~/ros2_ws', has a search icon, a menu icon, and window control buttons in its title bar. It shows the command 'ros2 run service_pkg client_demo' being executed, followed by the output 'Result = 100'.

```
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 does the summing and gets the result as 100, the result is printed in both terminals.