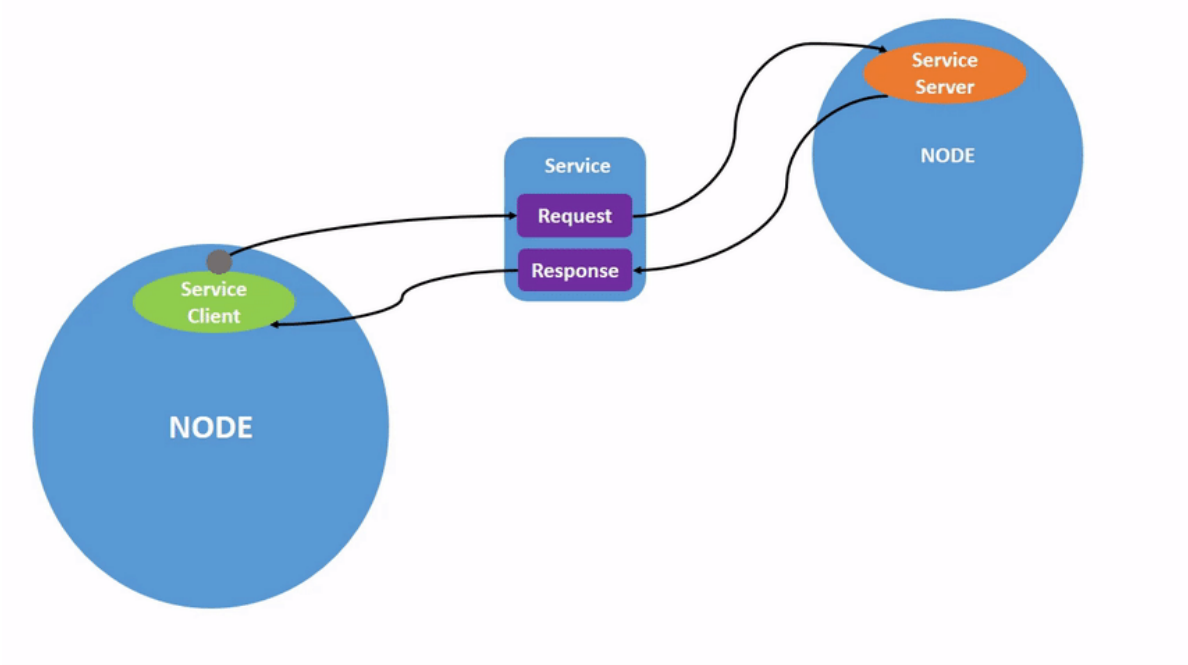
9.ROS2 service communication server

1. Introduction to service communication

Service communication is a communication model based on request and response. Between the two communicating parties, the client sends request data to the server, and the server responds to the client.

The client/server model is as follows:



From the perspective of the service implementation mechanism, this form of question-and-answer is called the client/server model, or CS model for short.When the client needs certain data, it sends request information for a specific service. After the server receives the request, it will process it and feedback the response information.

This communication mechanism is also very common in life, such as the various web pages we often browse. At this time, your computer browser is the client.Send a request to the website server through domain name or various operations, and the server will return the page data that needs to be displayed after receiving it.

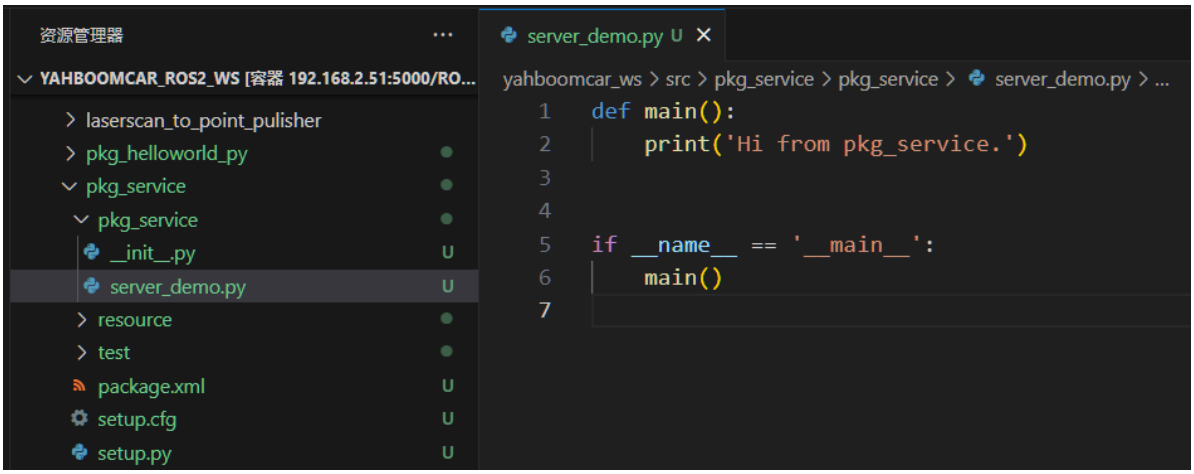
This case is located in the factory docker container. The source code location is:

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/pkg_service
```

2. Create a new function package

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

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



3. Server-side implementation

Next edit [server_demo.py] to implement server-side functions and add the following code:

```
#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)
        #To create a server, use the create_service function, and the parameters
passed in are:
         #The data type of service data, the name of the service, and the
service callback function (that is, the content of the service)
        self.srv = self.create_service(AddTwoInts, '/add_two_ints',
self.Add2Ints_callback)
    #The content of the service callback function here is to add two integers and
then return the added result.
    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)
    server_demo.destroy_node()                    # Destroy node object
    rclpy.shutdown()                              # Close the ROS2 Python
interface
```

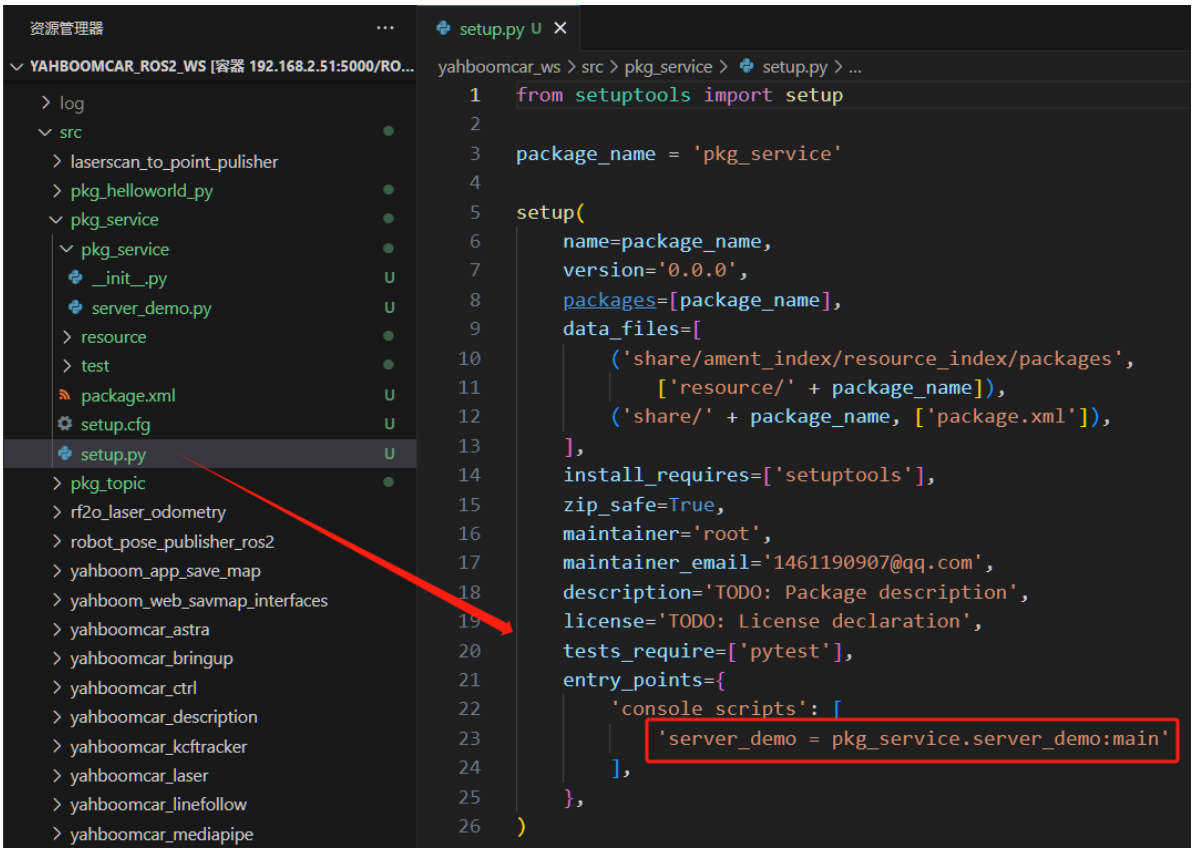
Focus on the service callback function, Add2Ints_callback. In addition to self, the parameters that need to be passed in here are request and response. Request is the parameter required by the service, and response is the feedback result of the service. request.a and request.b are the content of the request part, and response.sum is the content of the response part. Let's first take a look at what the data of the AddTwoInts type looks like. You can use the following command to view it.

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

```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 interface show example_interfaces/srv/AddTwoInts
int64 a
int64 b
---
int64 sum
```

"---" divides this type of data into two parts, the upper part represents the request, and the lower part represents the response. Then there are respective variables in their respective fields, such as int64 a, int64 b. When passing in parameters, you need to specify the values of a and b. Similarly, the feedback result also needs to specify the value of sum.

4. Edit configuration file



5. Compile workspace

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

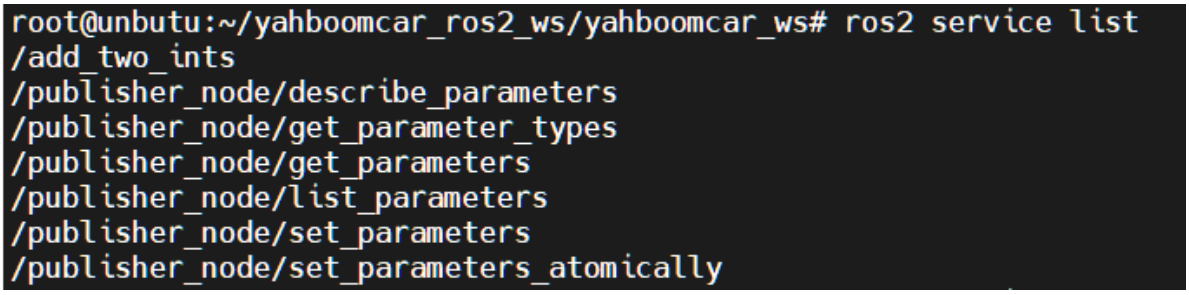
6.Run program

Open a terminal:

```
ros2 run pkg_service server_demo
```

After running, since the service is not called, there is no feedback data. You can call the service through the command line. First, query what services are currently available, and enter in another terminal:

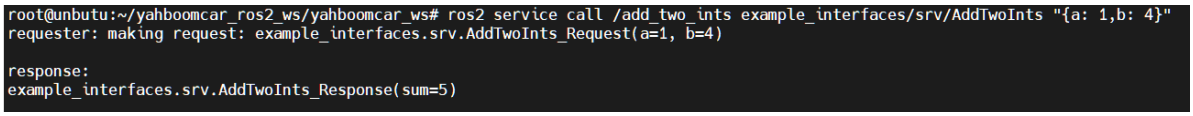
```
ros2 service list
```



/add_two_ints is the service we need to call. Call it through the following command and enter it 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, calling the service to calculate the sum of 1 and 4:



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