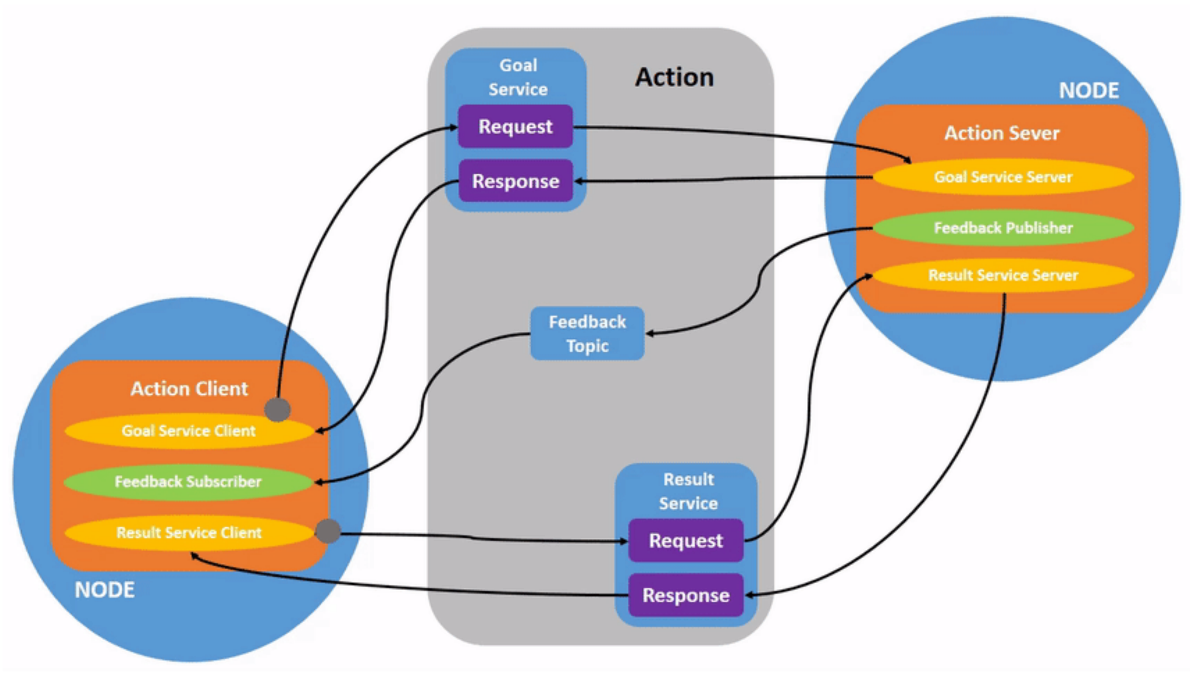
9.ROS2 action communication

1. Introduction to Action Communication

Action communication is a communication model with continuous feedback. Between the two communicating parties, the client sends request data to the server, and the server responds to the client. However, from the time the server receives the request to the time it generates the final response, it sends continuous feedback to the client.

The action communication client/server model is as follows:



2. Example Introduction

The action client submits an integer data type N. The action server receives the request data and accumulates all integers between 1 and N, returning the final result to the action client. Each addition calculates the current computation progress and provides feedback to the action client.

3. Create a new function package

3.1. Create an action communication interface function package

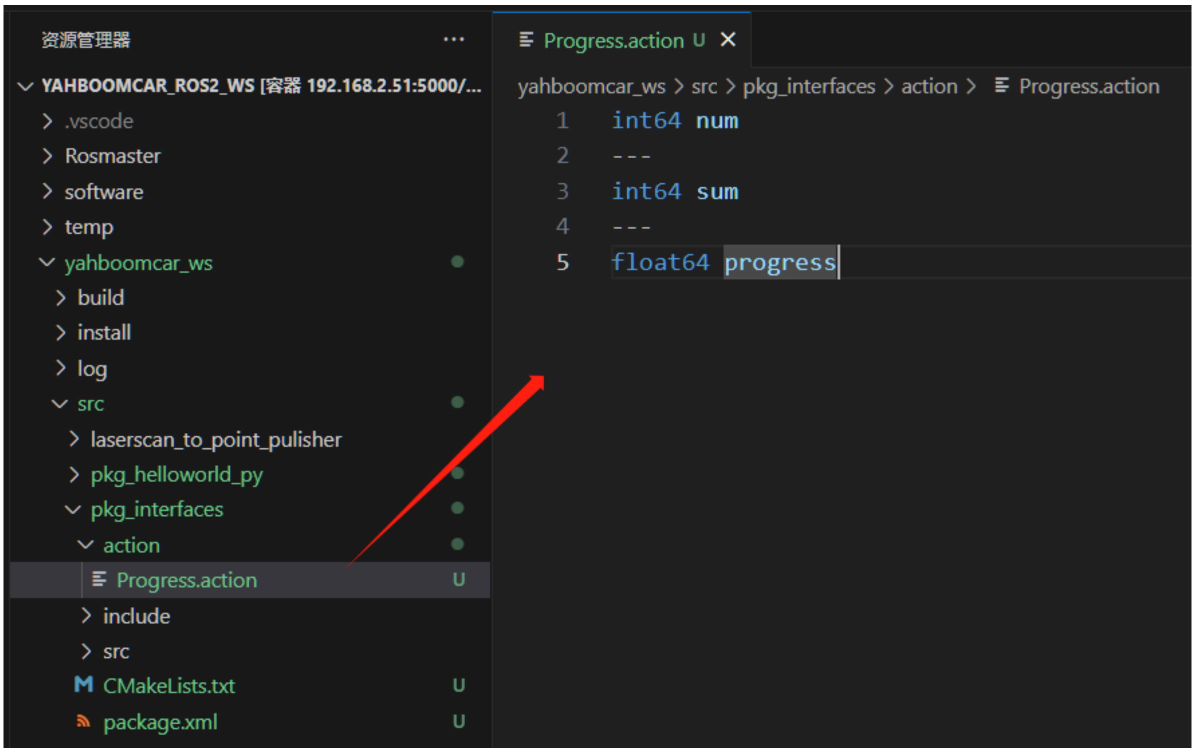
1. Action communication requires creating the action communication interface first.

- Create a new pkg_interfaces function package in the src directory of the workspace.

```
ros2 pkg create --build-type ament_cmake pkg_interfaces
```

2. Next, create an action folder under the pkg_interfaces function package and create a new file called [Progress.action] within the action folder. The file contents are as follows:

```
int64 num
---
int64 sum
---
float64 progress
```



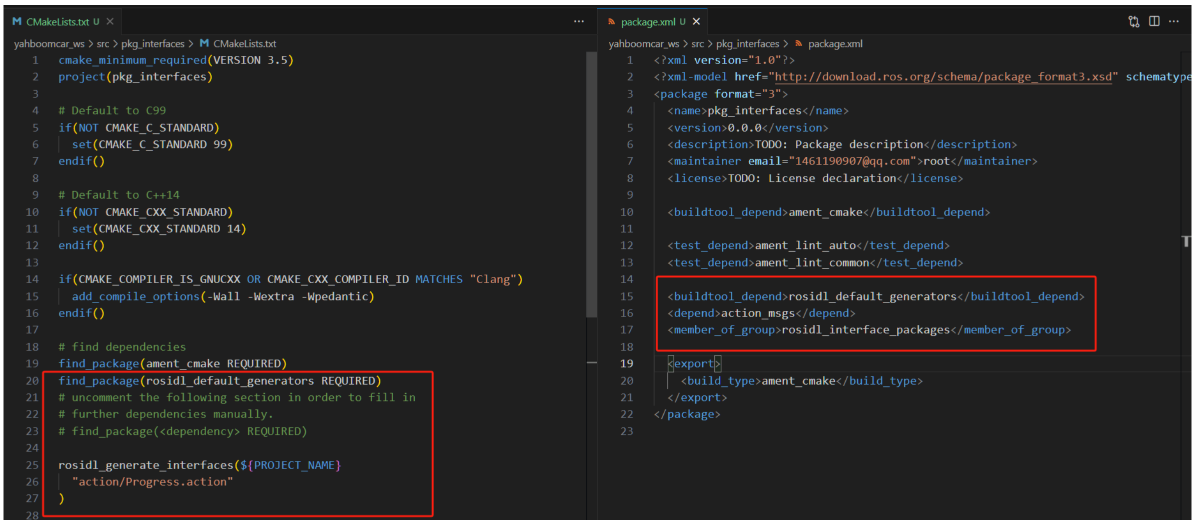
```
int64 num
---
int64 sum
---
float64 progress
```

3. Add some dependency packages to package.xml. The specific content is as follows:

```
<buildtool_depend>rosidl_default_generators</buildtool_depend>
<exec_depend>rosidl_default_runtime</exec_depend>
<depend>action_msgs</depend>
<member_of_group>rosidl_interface_packages</member_of_group>
```

4. Add the following configuration to CMakeLists.txt:

```
find_package(rosidl_default_generators REQUIRED)

rosidl_generate_interfaces(${PROJECT_NAME}
  "action/Progress.action"
)
```

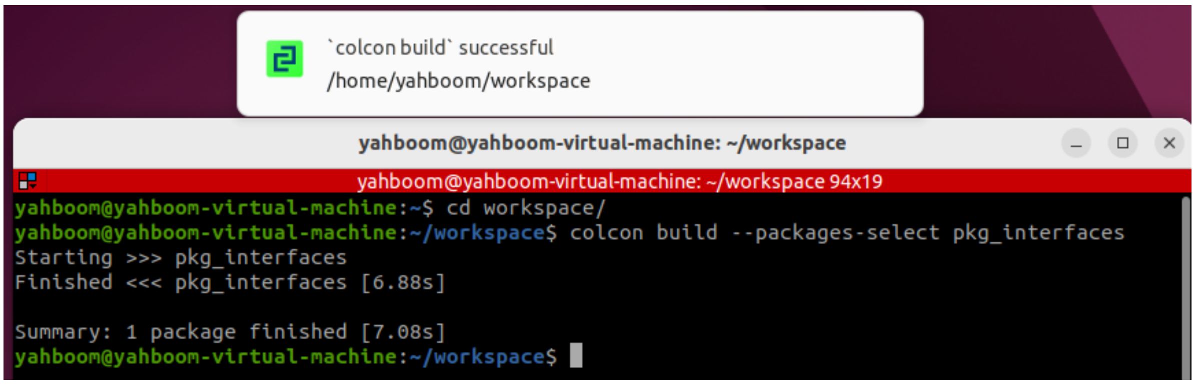

CMakeLists.txt U
yahboomcar_ws > src > pkg_interfaces > CMakeLists.txt
package.xml U
yahboomcar_ws > src > pkg_interfaces > package.xml

```
cmake_minimum_required(VERSION 3.5)
project(pkg_interfaces)

# Default to C99
if(NOT CMAKE_C_STANDARD)
  set(CMAKE_C_STANDARD 99)
endif()

# Default to C++14
if(NOT CMAKE_CXX_STANDARD)
  set(CMAKE_CXX_STANDARD 14)
endif()

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

# find dependencies
find_package(ament_cmake REQUIRED)
find_package(rosidl_default_generators REQUIRED)
# uncomment the following section in order to fill in
# further dependencies manually.
# find_package(<dependency> REQUIRED)

rosidl_generate_interfaces(${PROJECT_NAME}
  "action/Progress.action"
)
```

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd" schematype
<package format="3">
  <name>pkg_interfaces</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="1461190907@qq.com">root</maintainer>
  <license>TODO: License declaration</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

  <test_depend>ament_lint_auto</test_depend>
  <test_depend>ament_lint_common</test_depend>

  <buildtool_depend>rosidl_default_generators</buildtool_depend>
  <depend>action_msgs</depend>
  <member_of_group>rosidl_interface_packages</member_of_group>

  <export>
    <build_type>ament_cmake</build_type>
  </export>
</package>
```

5. Compile the package:

```
colcon build --packages-select pkg_interfaces
```

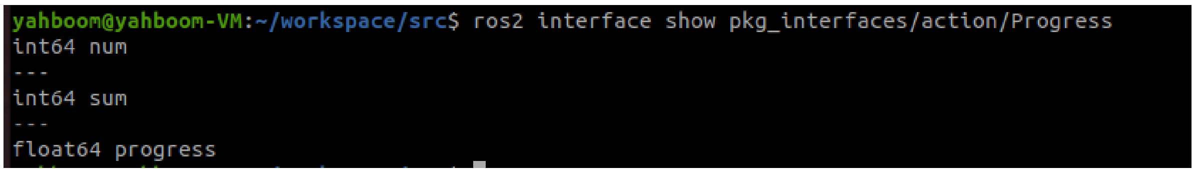

`colcon build` successful
/home/yahboom/workspace
yahboom@yahboom-virtual-machine: ~/workspace
yahboom@yahboom-virtual-machine: ~/workspace 94x19

```
yahboom@yahboom-virtual-machine:~$ cd workspace/
yahboom@yahboom-virtual-machine:~/workspace$ colcon build --packages-select pkg_interfaces
Starting >>> pkg_interfaces
Finished <<< pkg_interfaces [6.88s]

Summary: 1 package finished [7.08s]
yahboom@yahboom-virtual-machine:~/workspace$
```

6. After compilation is complete, the C++ and Python files corresponding to the `Progress.action` file will be generated in the `install` directory of the workspace. You can also access the workspace in a terminal and verify the file definitions and compilation by running the following command:

```
source install/setup.bash
ros2 interface show pkg_interfaces/action/Progress
```

Under normal circumstances, the terminal will output the same content as the `Progress.action` file.



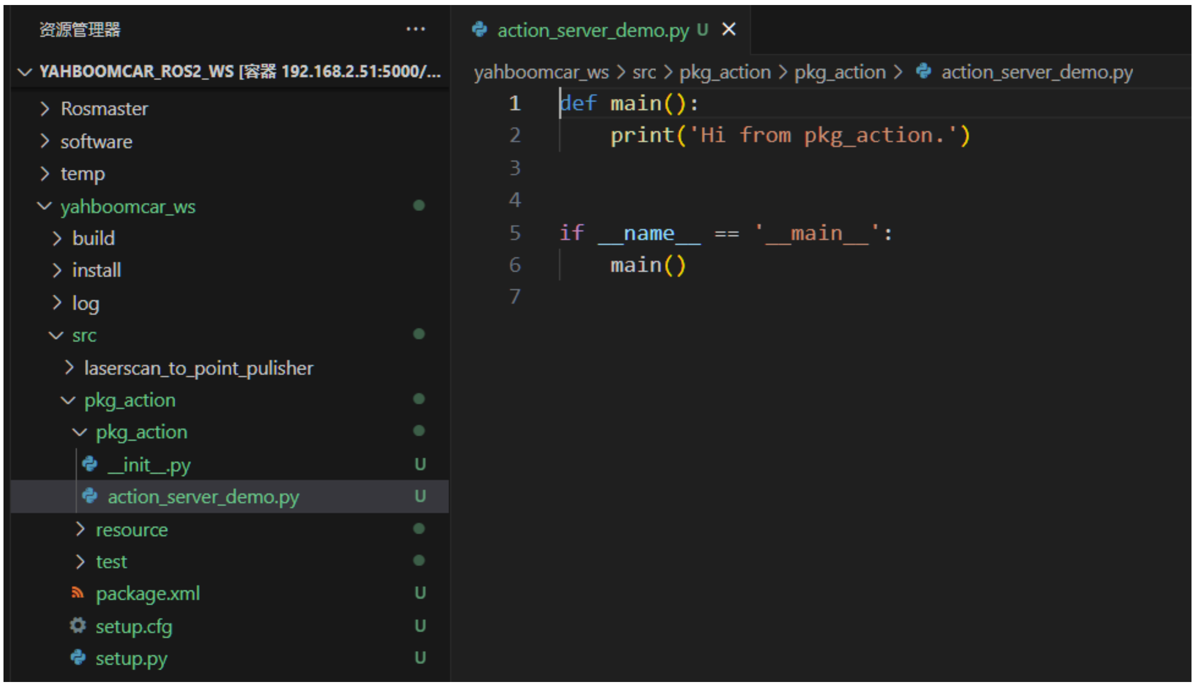
```
yahboom@yahboom-VM:~/workspace/src$ ros2 interface show pkg_interfaces/action/Progress
int64 num
---
int64 sum
---
float64 progress
```

3.2. Creating the Action Communication Function Package

Create a new package called pkg_action in the src directory of the workspace.

```
ros2 pkg create pkg_action --build-type ament_python --dependencies rclpy
pkg_interfaces --node-name action_server_demo
```

After executing the above command, the pkg_action function package will be created, along with an action_server_demo node and the relevant configuration files.


资源管理器

YAHBOOMCAR_ROS2_WS [容器 192.168.2.51:5000/...

```
> Rosmaster
> software
> temp
v yahboomcar_ws
  > build
  > install
  > log
  v src
    > laserscan_to_point_pulisher
    v pkg_action
      v pkg_action
        __init__.py                U
        action_server_demo.py      U
      > resource
      > test
      package.xml                  U
      setup.cfg                    U
      setup.py                     U
```

action_server_demo.py U

yahboomcar_ws > src > pkg_action > pkg_action > action_server_demo.py

```python
def main():
    print('Hi from pkg_action.')


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

4. Server Implementation

4.1 Creating the Server

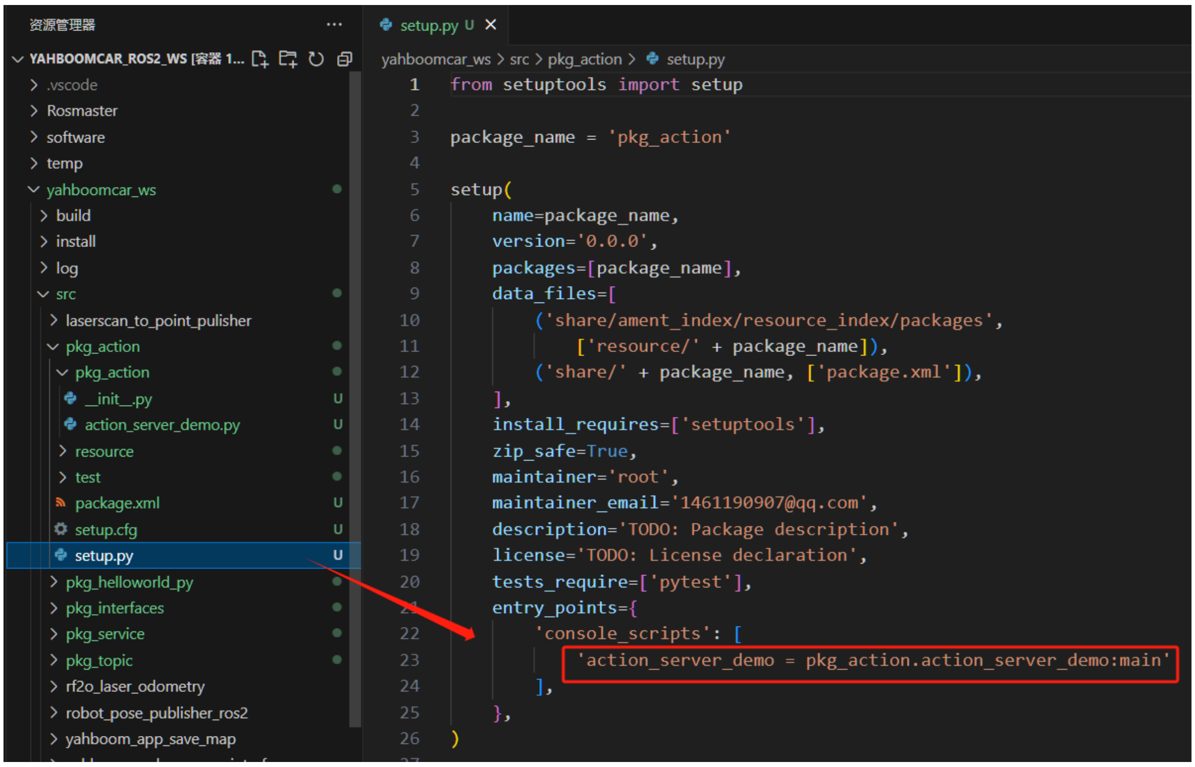
Next, edit [action_server_demo.py] to implement the server functionality and add the following code:

```python
import time
import rclpy
from rclpy.action import ActionServer
from rclpy.node import Node

from pkg_interfaces.action import Progress


class Action_Server(Node):

    def __init__(self):
        super().__init__('progress_action_server')
        # Creating an action server
        self._action_server = ActionServer(
            self,
            Progress,
            'get_sum',
            self.execute_callback)
        self.get_logger().info('动作服务已经启动！')

    def execute_callback(self, goal_handle):
        self.get_logger().info('开始执行任务....')


        # Generate continuous feedback;
        feedback_msg = Progress.Feedback()

        sum = 0
        for i in range(1, goal_handle.request.num + 1):
            sum += i
            feedback_msg.progress = i / goal_handle.request.num
            self.get_logger().info('连续反馈: %.2f' % feedback_msg.progress)
            goal_handle.publish_feedback(feedback_msg)
            time.sleep(1)

        # Generate the final response.
        goal_handle.succeed()
        result = Progress.Result()
        result.sum = sum
        self.get_logger().info('任务完成! ')

        return result


def main(args=None):

    rclpy.init(args=args)
    # Call the spin function and pass in the node object
    Progress_action_server = Action_Server()
    rclpy.spin(Progress_action_server)
    Progress_action_server.destroy_node()
    # Release resources
    rclpy.shutdown()
```

4.2 Edit the Configuration File

- Open setup.py and add the following line to the console_scripts list:

```
'action_server_demo = pkg_action.action_server_demo:main',
```



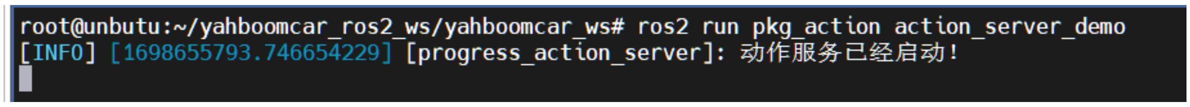
```
from setuptools import setup

package_name = 'pkg_action'

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': [
            'action_server_demo = pkg_action.action_server_demo:main'
        ],
    },
)
```

4.3 Compile the Package

```
colcon build --packages-select pkg_action
```

4.4 Run the Program

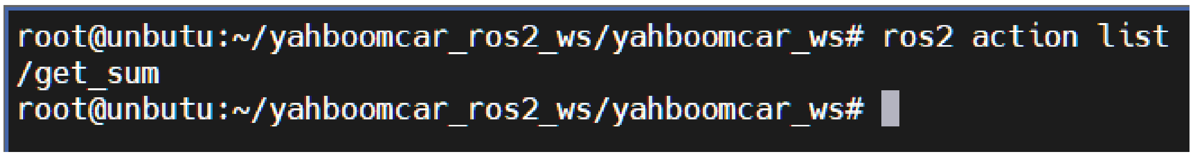
```
ros2 run pkg_action action_server_demo
```



```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 run pkg_action action_server_demo
[INFO] [1698655793.746654229] [progress_action_server]: 动作服务已经启动!
```

In another terminal, enter:

```
ros2 action list
```

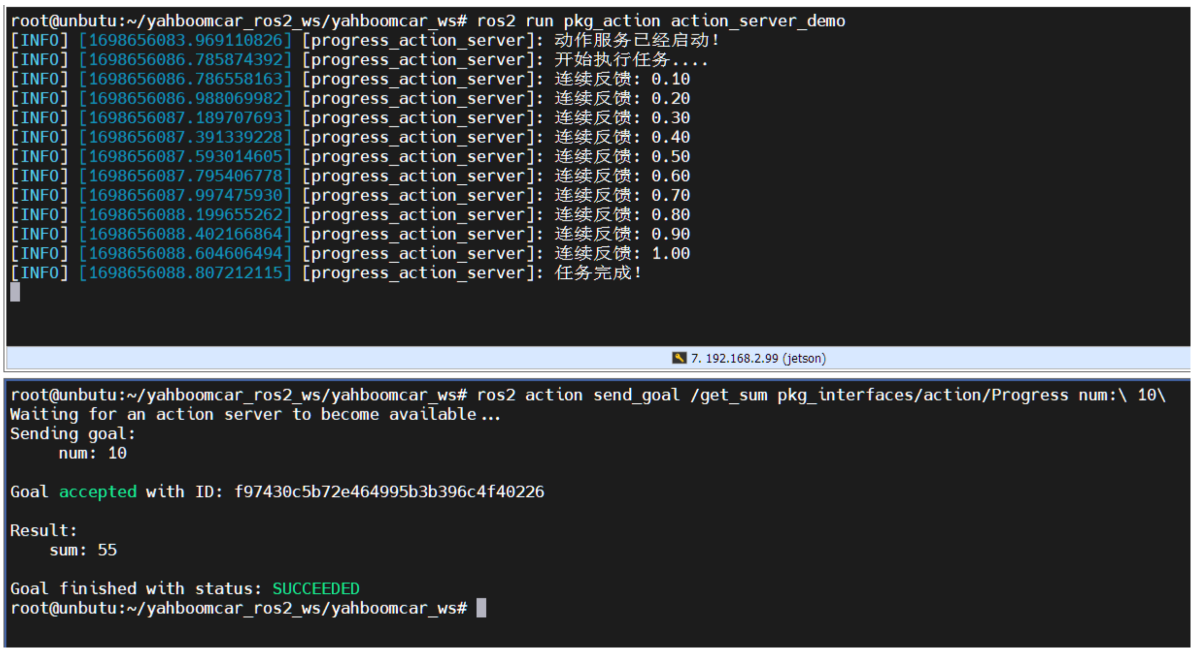


```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 action list
/get_sum
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws#
```

/get_sum is the action we need to call. To do so, enter the following command in the terminal:

```
ros2 action send_goal /get_sum pkg_interfaces/action/Progress "{num: 10}"
```

Here we calculate the sum of the values from 1 to 10:



```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 run pkg_action action_server_demo
[INFO] [1698656083.969110826] [progress_action_server]: 动作服务已经启动!
[INFO] [1698656086.785874392] [progress_action_server]: 开始执行任务....
[INFO] [1698656086.786558163] [progress_action_server]: 连续反馈: 0.10
[INFO] [1698656086.988069982] [progress_action_server]: 连续反馈: 0.20
[INFO] [1698656087.189707693] [progress_action_server]: 连续反馈: 0.30
[INFO] [1698656087.391339228] [progress_action_server]: 连续反馈: 0.40
[INFO] [1698656087.593014605] [progress_action_server]: 连续反馈: 0.50
[INFO] [1698656087.795406778] [progress_action_server]: 连续反馈: 0.60
[INFO] [1698656087.997475930] [progress_action_server]: 连续反馈: 0.70
[INFO] [1698656088.199655262] [progress_action_server]: 连续反馈: 0.80
[INFO] [1698656088.402166864] [progress_action_server]: 连续反馈: 0.90
[INFO] [1698656088.604606494] [progress_action_server]: 连续反馈: 1.00
[INFO] [1698656088.807212115] [progress_action_server]: 任务完成!
```

7. 192.168.2.99 (jetson)

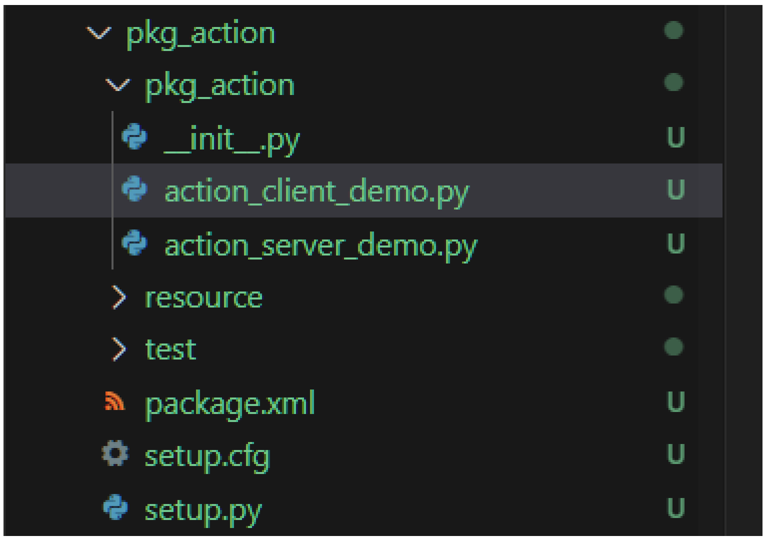
```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 action send_goal /get_sum pkg_interfaces/action/Progress num:\ 10\
Waiting for an action server to become available...
Sending goal:
     num: 10

Goal accepted with ID: f97430c5b72e464995b3b396c4f40226

Result:
    sum: 55

Goal finished with status: SUCCEEDED
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws#
```

The upper part of the image above shows the server, and the lower part shows the client. You can see that the server continuously reports progress while calculating the sum from 1 to 10. Finally, the task is completed, and the client receives feedback that the sum is 55.

5. Client Implementation

5.1 Creating the Client

Create a new file, action_client_demo.py, in the same directory as action_server_demo.py.


```
∨ pkg_action                    ●
  ∨ pkg_action                  ●
      __init__.py               U
      action_client_demo.py     U
      action_server_demo.py     U
  > resource                    ●
  > test                        ●
    package.xml                 U
    setup.cfg                   U
    setup.py                    U
```

Next, edit action_client_demo.py to implement the server functionality and add the following code:

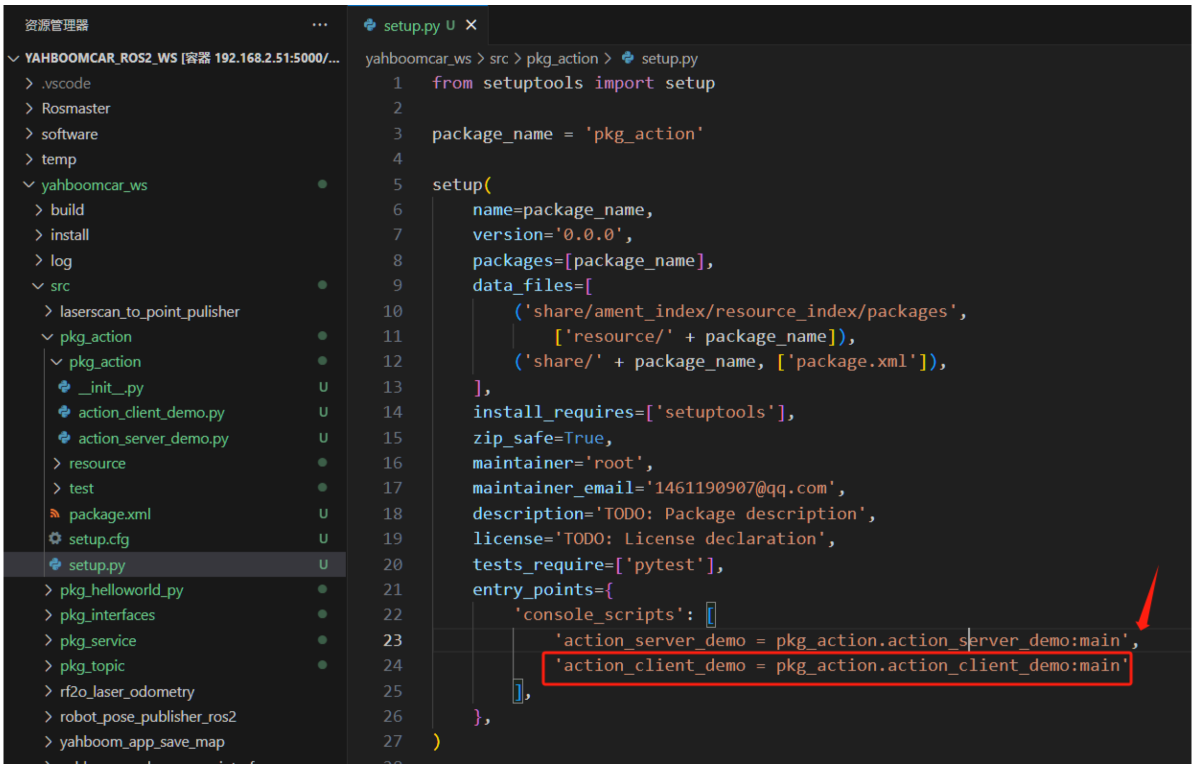
```python
import rclpy
from rclpy.action import ActionClient
from rclpy.node import Node
from pkg_interfaces.action import Progress

class Action_Client(Node):
    def __init__(self):
        super().__init__('progress_action_client')
        # Create an action client;
        self._action_client = ActionClient(self, Progress, 'get_sum')

    def send_goal(self, num):
        # Send a request;
        goal_msg = Progress.Goal()
        goal_msg.num = num
        self._action_client.wait_for_server()
        self._send_goal_future = self._action_client.send_goal_async(goal_msg,
feedback_callback=self.feedback_callback)
        self._send_goal_future.add_done_callback(self.goal_response_callback)

    def goal_response_callback(self, future):
        # Process the feedback after the target sends it;
        goal_handle = future.result()
        if not goal_handle.accepted:
            self.get_logger().info('请求被拒绝')
            return

        self.get_logger().info('请求被接收，开始执行任务！')

        self._get_result_future = goal_handle.get_result_async()
        self._get_result_future.add_done_callback(self.get_result_callback)

    #Process the final response.
    def get_result_callback(self, future):
        result = future.result().result
```

```
            self.get_logger().info('最终计算结果:sum = %d' % result.sum)
            # 5. Release resources.
            rclpy.shutdown()

    # Processing continuous feedback;
    def feedback_callback(self, feedback_msg):
        feedback = (int)(feedback_msg.feedback.progress * 100)
        self.get_logger().info('当前进度: %d%%' % feedback)


def main(args=None):
    rclpy.init(args=args)
    action_client = Action_Client()
    action_client.send_goal(10)
    rclpy.spin(action_client)
```

5.2 Edit the Configuration File

- Open setup.py and add the following line to the console_scripts list:

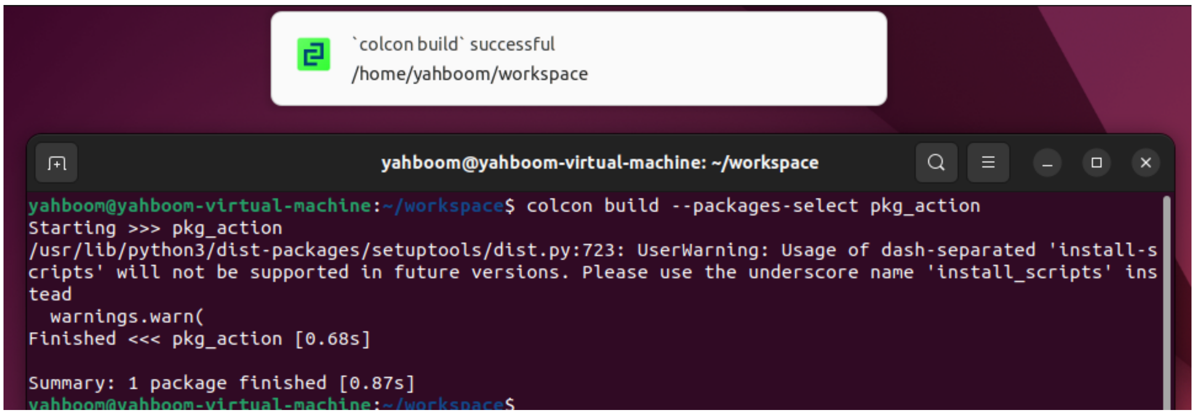
```
'action_client_demo = pkg_action.action_client_demo:main'
```



```
from setuptools import setup

package_name = 'pkg_action'

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': [
            'action_server_demo = pkg_action.action_server_demo:main',
            'action_client_demo = pkg_action.action_client_demo:main'
        ],
    },
)
```

5.3 Compile the Package

```
colcon build --packages-select pkg_action
```



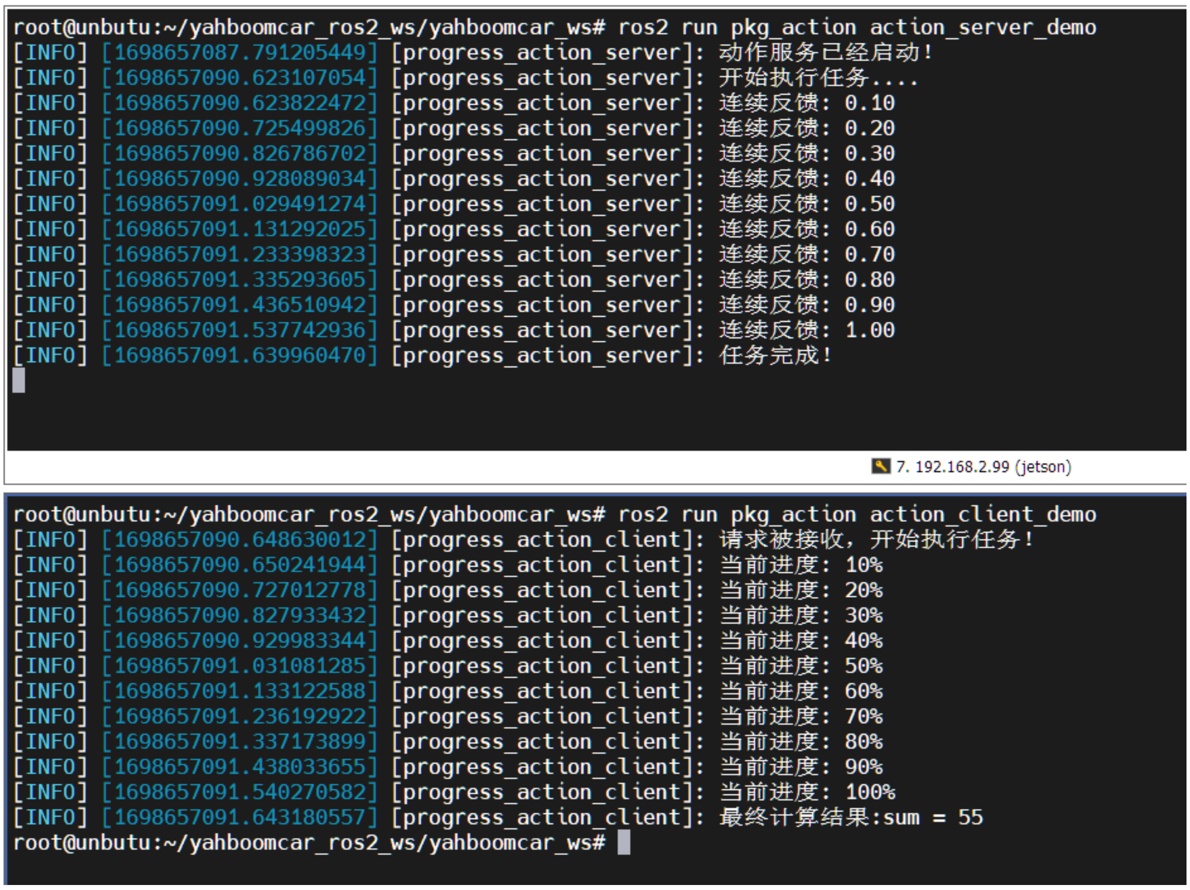
```
`colcon build` successful
/home/yahboom/workspace
```

```
yahboom@yahboom-virtual-machine: ~/workspace
yahboom@yahboom-virtual-machine:~/workspace$ colcon build --packages-select pkg_action
Starting >>> pkg_action
/usr/lib/python3/dist-packages/setuptools/dist.py:723: UserWarning: Usage of dash-separated 'install-s
cripts' will not be supported in future versions. Please use the underscore name 'install_scripts' ins
tead
  warnings.warn(
Finished <<< pkg_action [0.68s]

Summary: 1 package finished [0.87s]
yahboom@yahboom-virtual-machine:~/workspace$
```

5.4 Run the Program

Execute the following command in a separate terminal:

```
# Start the server node
ros2 run pkg_action action_server_demo
# Start the client node
ros2 run pkg_action action_client_demo
```



```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 run pkg_action action_server_demo
[INFO] [1698657087.791205449] [progress_action_server]: 动作服务已经启动!
[INFO] [1698657090.623107054] [progress_action_server]: 开始执行任务....
[INFO] [1698657090.623822472] [progress_action_server]: 连续反馈: 0.10
[INFO] [1698657090.725499826] [progress_action_server]: 连续反馈: 0.20
[INFO] [1698657090.826786702] [progress_action_server]: 连续反馈: 0.30
[INFO] [1698657090.928089034] [progress_action_server]: 连续反馈: 0.40
[INFO] [1698657091.029491274] [progress_action_server]: 连续反馈: 0.50
[INFO] [1698657091.131292025] [progress_action_server]: 连续反馈: 0.60
[INFO] [1698657091.233398323] [progress_action_server]: 连续反馈: 0.70
[INFO] [1698657091.335293605] [progress_action_server]: 连续反馈: 0.80
[INFO] [1698657091.436510942] [progress_action_server]: 连续反馈: 0.90
[INFO] [1698657091.537742936] [progress_action_server]: 连续反馈: 1.00
[INFO] [1698657091.639960470] [progress_action_server]: 任务完成!
```

```
7. 192.168.2.99 (jetson)
```

```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 run pkg_action action_client_demo
[INFO] [1698657090.648630012] [progress_action_client]: 请求被接收, 开始执行任务!
[INFO] [1698657090.650241944] [progress_action_client]: 当前进度: 10%
[INFO] [1698657090.727012778] [progress_action_client]: 当前进度: 20%
[INFO] [1698657090.827933432] [progress_action_client]: 当前进度: 30%
[INFO] [1698657090.929983344] [progress_action_client]: 当前进度: 40%
[INFO] [1698657091.031081285] [progress_action_client]: 当前进度: 50%
[INFO] [1698657091.133122588] [progress_action_client]: 当前进度: 60%
[INFO] [1698657091.236192922] [progress_action_client]: 当前进度: 70%
[INFO] [1698657091.337173899] [progress_action_client]: 当前进度: 80%
[INFO] [1698657091.438033655] [progress_action_client]: 当前进度: 90%
[INFO] [1698657091.540270582] [progress_action_client]: 当前进度: 100%
[INFO] [1698657091.643180557] [progress_action_client]: 最终计算结果:sum = 55
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws#
```

The image above shows the server on the top and the client on the bottom. Here, we're calculating the sum of 1 to 10. You can see that the server continuously reports progress during the calculation. Finally, the task is completed, and the client receives feedback that the sum is 55.