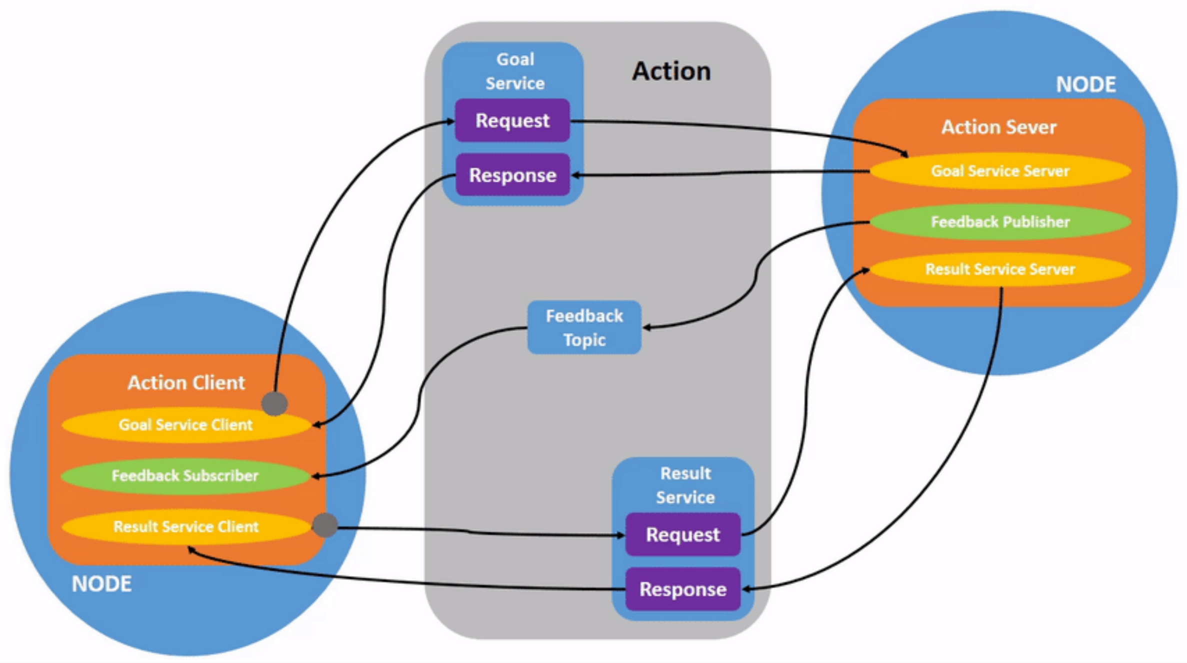
9、ROS2动作通讯

1、动作通讯简介

动作通信是一种带有连续反馈的通信模型，在通信双方中，客户端发送请求数据到服务端，服务端响应结果给客户端，但是在服务端接收到请求到产生最终响应的过程中，会发送连续的反馈信息到客户端。

动作通讯客户端/服务器模型如下：



2、案例介绍

动作客户端提交一个整型数据N，动作服务端接收请求数据并累加1-N之间的所有整数，将最终结果返回给动作客户端，且每累加一次都计算当前运算进度并反馈给动作客户端。

3、新建功能包

3.1、创建动作通讯接口功能包

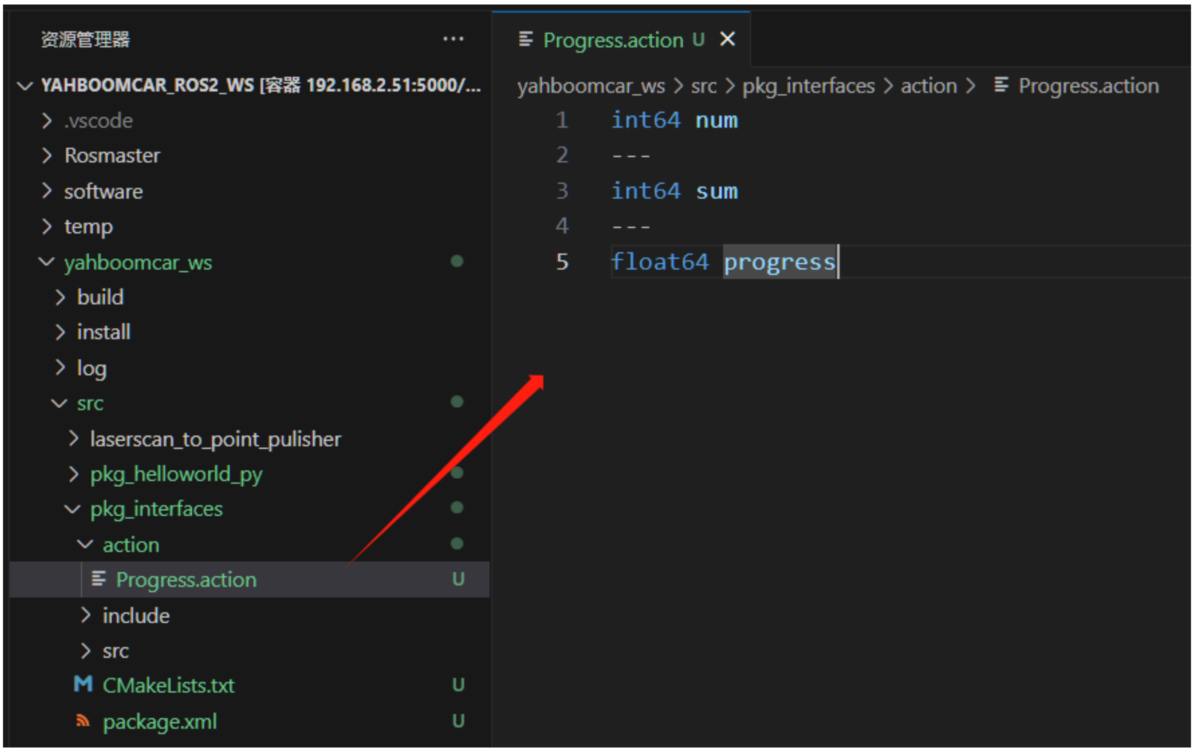
1、动作通讯需要先创建动作通讯接口

- 在工作空间的src目录下新建pkg_interfaces功能包

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

2、接着在pkg_interfaces功能包下面创建一个action的文件夹，并在action文件夹内新建【Progress.action】文件，文件内容如下：

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



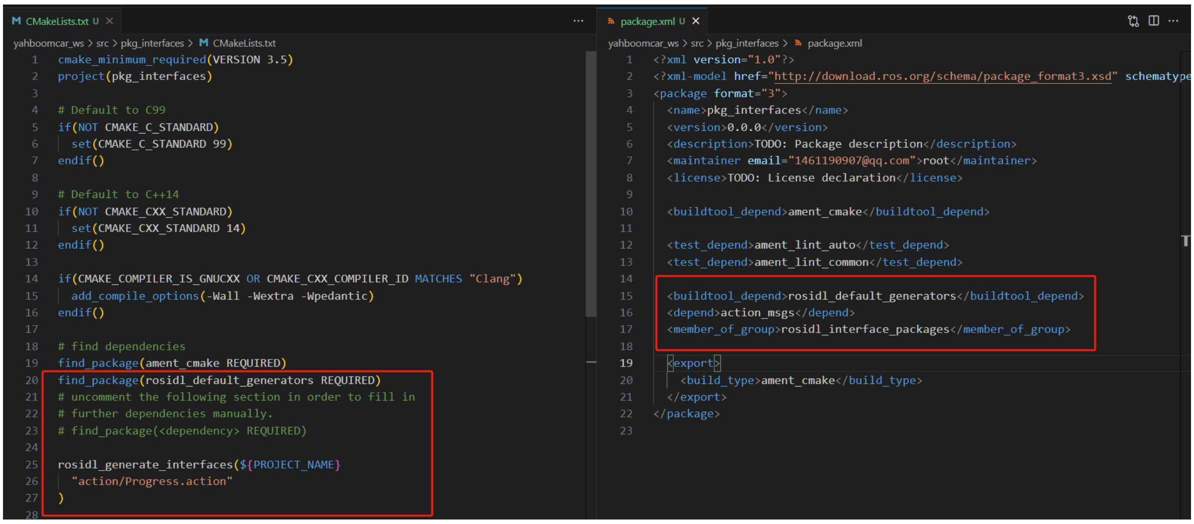
```
int64 num
---
int64 sum
---
float64 progress
```

3、在package.xml中需要添加一些依赖包，具体内容如下：

```
<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、在CMakeLists.txt 中添加如下配置：

```
find_package(rosidl_default_generators REQUIRED)

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

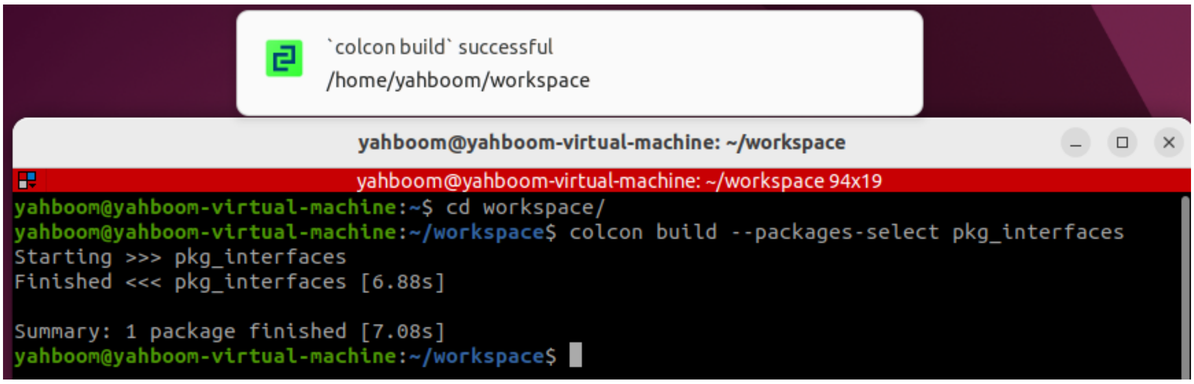


```
cmake_minimum_required(VERSION 3.5)
project(pkg_interfaces)

# 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、编译功能包：

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



`colcon build` successful
/home/yahboom/workspace

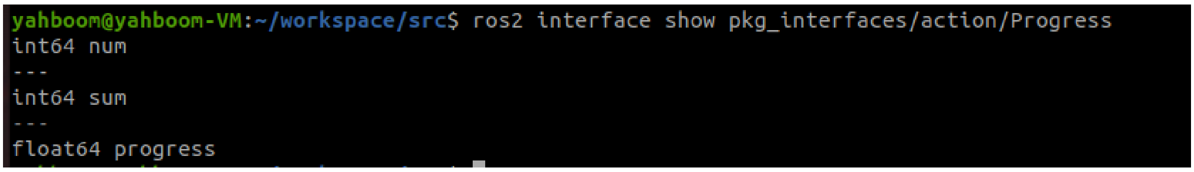
yahboom@yahboom-virtual-machine: ~/workspace
yahboom@yahboom-virtual-machine: ~/workspace 94x19

```
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、编译完成之后，在工作空间下的 install 目录下将生成`Progress.action`文件对应的C++和Python文件，我们也可以在终端下进入工作空间，通过如下命令查看文件定义以及编译是否正常：

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

正常情况下，终端将会输出与`Progress.action`文件一致的内容



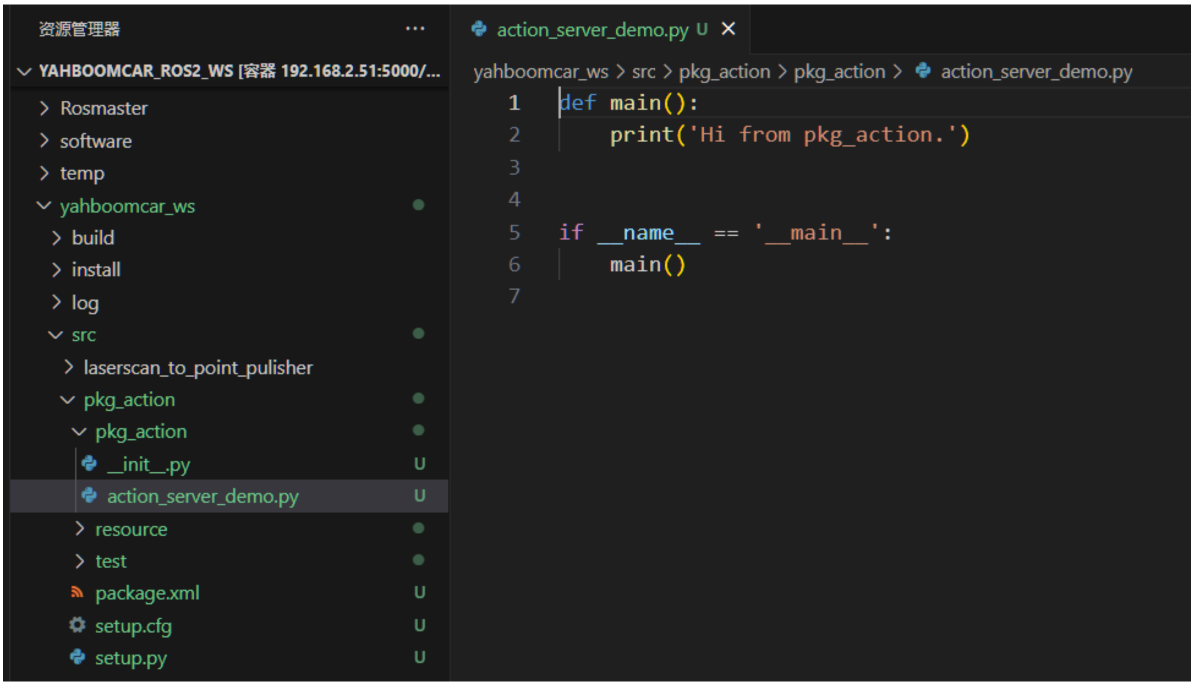
```
yahboom@yahboom-VM:~/workspace/src$ ros2 interface show pkg_interfaces/action/Progress
int64 num
---
int64 sum
---
float64 progress
```

3.2、创建动作通讯功能包

- 在工作空间的src目录下新建pkg_action功能包

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

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



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


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

4、服务端实现

4.1 创建服务端

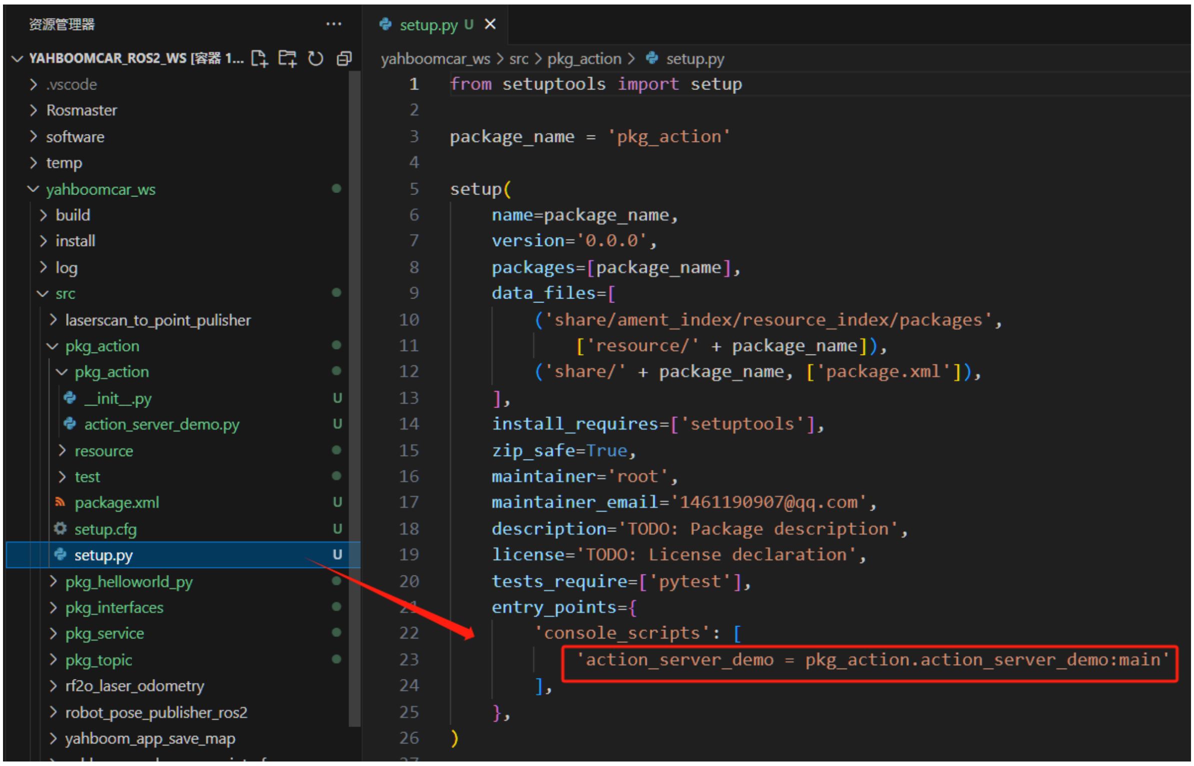
接下来编辑【action_server_demo.py】实现服务端的功能，添加如下代码：

```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')
        # 创建动作服务端
        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....')


        # 生成连续反馈；
        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)

        # 生成最终响应。
        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)
    # 调用spin函数，并传入节点对象
    Progress_action_server = Action_Server()
    rclpy.spin(Progress_action_server)
    Progress_action_server.destroy_node()
    # 释放资源
    rclpy.shutdown()
```

4.2 编辑配置文件

- 打开setup.py,在console_scripts列表中添加

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



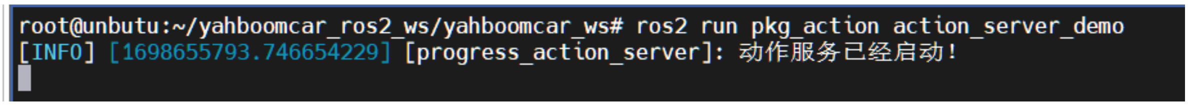
```
from setuptools import setup

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 编译功能包

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

4.4 运行程序

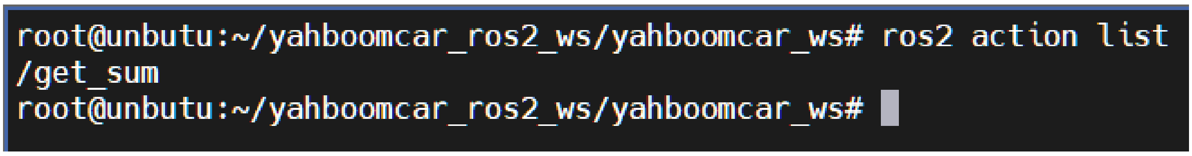
```
ros2 run pkg_action action_server_demo
```



```
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!
```

另一个终端输入：

```
ros2 action list
```

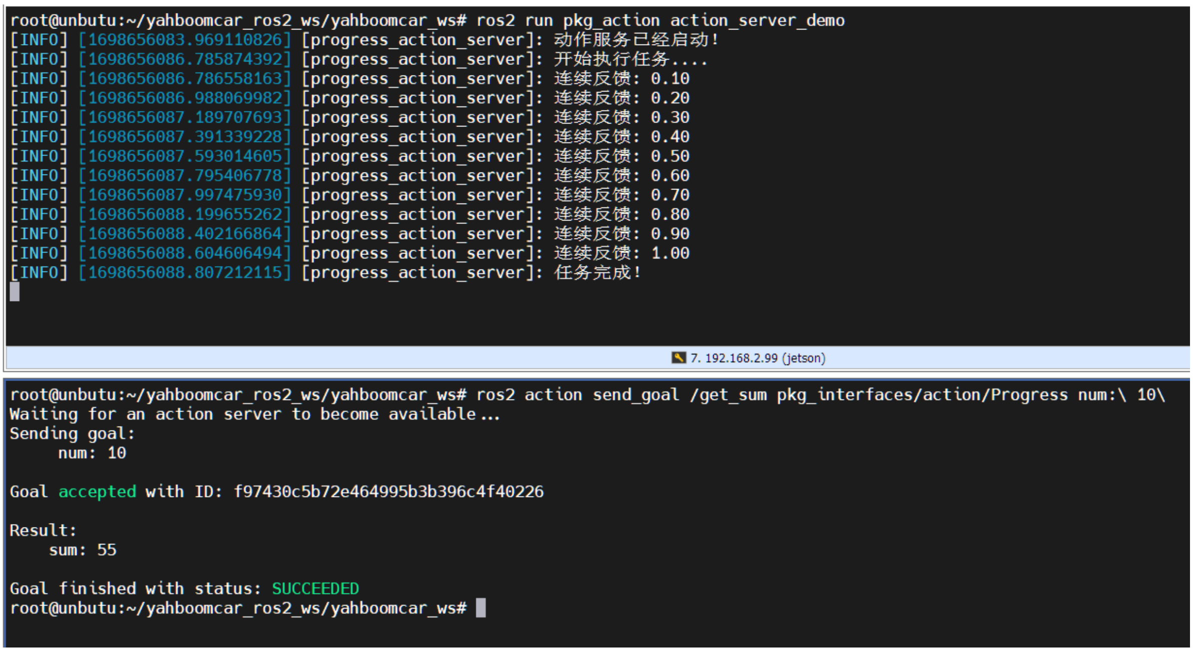


```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 action list
/get_sum
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# 
```

/get_sum就是我们需要调用的动作，通过以下命令进行调用，终端输入：

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

这里我们求1到10的和：



```
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!
```

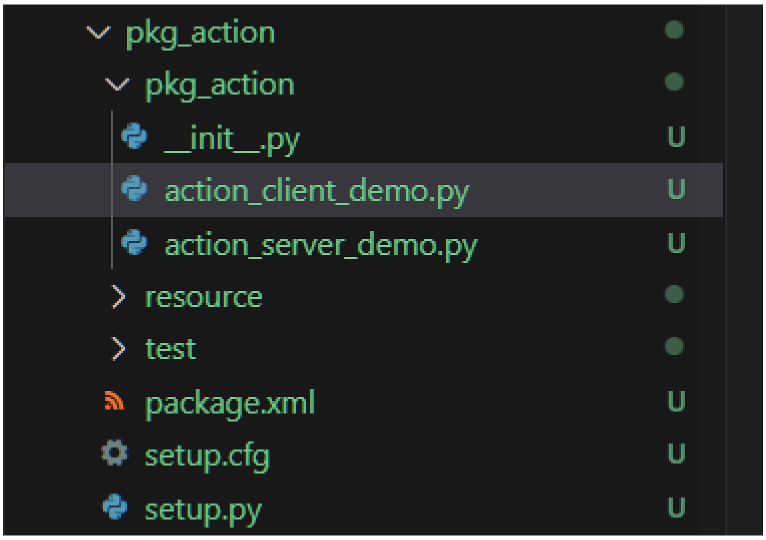
```
root@unbutu:~/yahboomcar_ros2_ws/yahboomcar_ws# ros2 action send_goal /get_sum pkg_interfaces/action/Progress num:\ 10\
Waiting for an action server to become available...
Sending goal:
     num: 10

Goal accepted with ID: f97430c5b72e464995b3b396c4f40226

Result:
    sum: 55

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

上图上面是服务端，下面是客户端。可以看到1到10的和计算的过程中有服务端一直在反馈计算的进度，最后显示任务完成，客户端也收到了反馈的和为55

5、客户端实现

5.1 创建客户端

在【action_server_demo.py】同级目录下新建文件【action_client_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

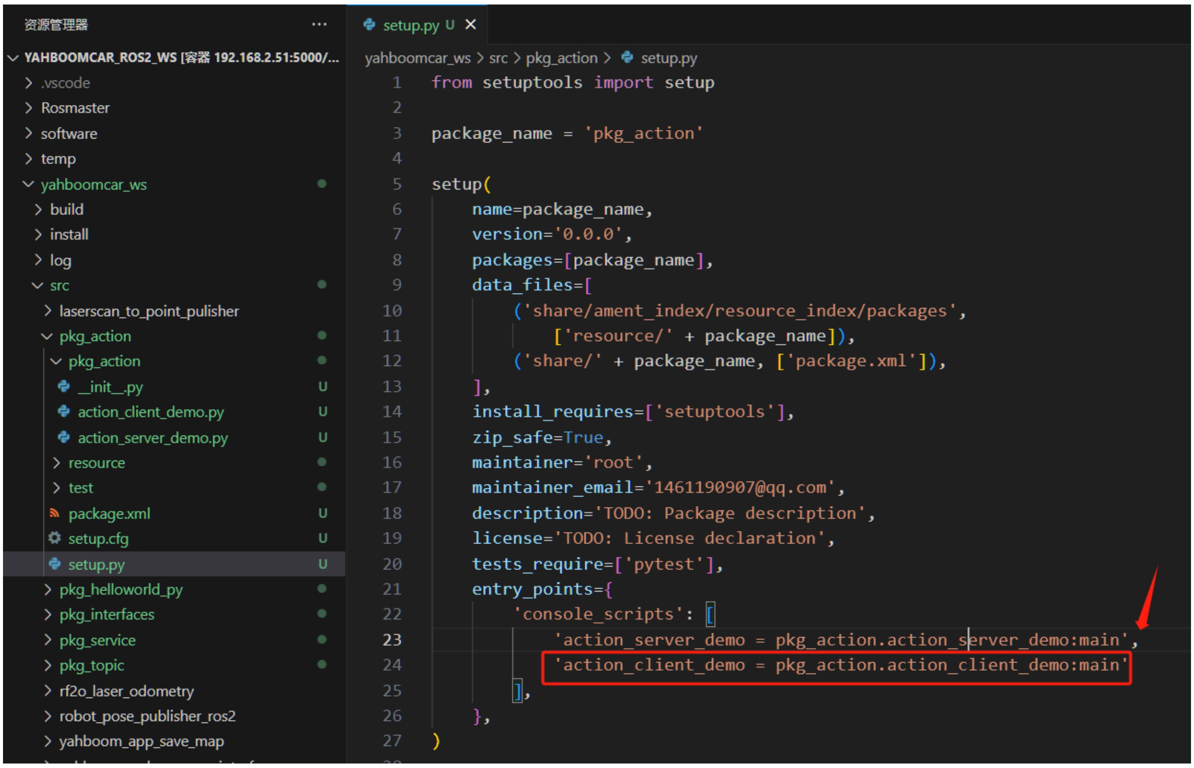
接下来编辑【action_client_demo.py】实现服务端的功能，添加如下代码：

```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')
        # 创建动作客户端;
        self._action_client = ActionClient(self, Progress, 'get_sum')

    def send_goal(self, num):
        # 发送请求;
        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):
        # 处理目标发送后的反馈;
        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)

    # 处理最终响应。
    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.释放资源。
        rclpy.shutdown()

    # 处理连续反馈；
    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 编辑配置文件

- 打开setup.py,在console_scripts列表中添加

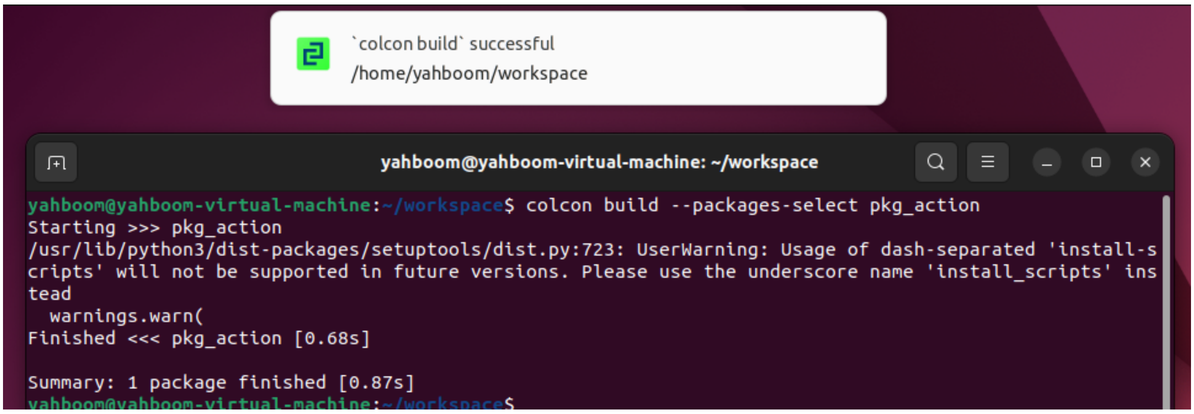
```
'action_client_demo = pkg_action.action_client_demo:main'
```



```python
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 编译功能包

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



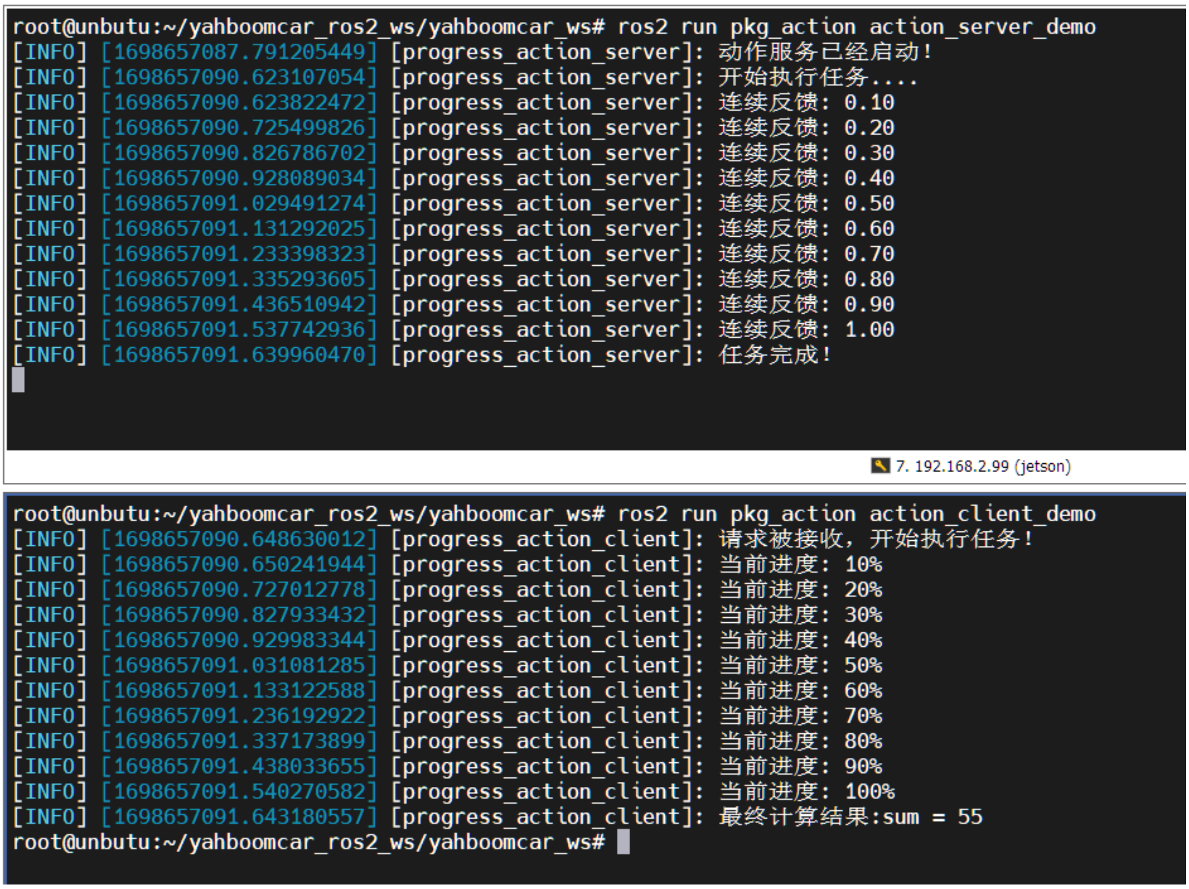
`colcon build` successful
/home/yahboom/workspace
yahboom@yahboom-virtual-machine: ~/workspace

```
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 运行程序

分终端执行如下：

```
#启动服务端节点
ros2 run pkg_action action_server_demo
#启动客户端节点
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！
```

```
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#
```

上图上面是服务端，下面是客户端。这里我们求1到10的和，可以看到1到10的和计算的过程中有服务端一直在反馈计算的进度，最后显示任务完成，客户端也收到了反馈的和为55