ROS2 Experience
James H Phelan MD
2020.08.08 -

Conventions in this document:
bold = my keyboard entries, significant terminal output, chapter titles, entry dates
[brackets] = side comments by me
italics = instructions or comments

Email from Vesa Pykala:
Team,
I saw a promising tutorial about ROS 2 with Ubuntu 20 / new Foxy.
It should be free for two weeks time, I subscribed.
I have currently done 10 lessons. It will take one week or what ever you want.
https://www.skillshare.com/classes/ROS2-For-Beginners-Build-Robotics-Applications-with-Robot-Operating-System-2/1870429910/projects

https://www.skillshare.com/classes/ROS2-For-Beginners-Build-Robotics-Applications-with-Robot-Operating-System-2/1870429910/projects

Wanted to use the Jetson NANO for the platform and install Ubuntu 20.04:
https://developer.nvidia.com/embedded/jetson-nano-developer-kit
BUT the NANO doesn't support Ubuntu 20.04 [at least not easily....] ;-((

So may try installing a Virtual Machine on my Windows 10 laptop. Does it have room? 890GB should do!

Lesson 6. Install Ubuntu 20.04 on a Virtual Machine:
Download Ubuntu 20.04 Desktop
https://ubuntu.com/download/desktop
Download VirtualBox
https://www.virtualbox.org/
Install VirtualBox
Install Ubuntu into VirtualBox
    CAUTION: VirtualBox can get hidden behind the browser window
Restart computer: restart (only restarts Virtual Machine)
Please remove the installation medium, then press ENTER:
    (just press ENTER, there is no CD-ROM)

Reconfig for dynamic sizing & cut/paste:
Devices / Insert Guest Additions CD Image
Devices / Shared Clipboard / Bidirectional
this may require resetting during use
Devices / Drag and Drip / Bidirectional
Devices / Network / Connect Network Adapter
tutorial says use bridge adapter but NAT worked better

Terminal / $ **sudo apt update**
Terminal / $ **sudo apt upgrade**
Eject Guest Additions CD

Lesson 7. Programming Tools I Use During This Course:
Install Terminator improved splitable terminal window:
$ **sudo apt install terminator**
failed, no network
Googled "VirtualBox lost network adapter"
suggested reinstall VirtualBox AND Ubuntu. [Right!! NOT!]
suggested bridge adapter > NAT. Worked!
Able to pull up Drudge Report
$ **sudo apt install terminator**
success
Google "ubuntu terminator keyboard shortcuts"
download .pdf by svschannak

Install Chromium to replace Firefox
$ **sudo apt install chromium-browser**

Install Visual Studio Code
Visual Studio Code is available from the official Microsoft Apt repositories.
To install it, follow the steps below:
$ **sudo apt update**
$ **sudo apt upgrade**
$ **sudo apt install software-properties-common apt-transport-https wget**
$ **wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add -**
$ **sudo add-apt-repository "deb [arch=amd64]**
https://packages.microsoft.com/repos/vscode stable main"
$ **sudo apt install python3-pip**
$ **sudo apt install code**
$ **sudo apt update**
$ **sudo apt upgrade**
$ **gedit** # to check that gedit text editor is installed (it is)

Using Ubuntu Activities search, find Visual Studio Code
Make favorite
open
install extensions for C/C++; Python; Cmake

Lesson 8. Install ROS2 Foxy Fitzroy on Ubuntu 20.04
Google “install ros2 foxy”
select - https://index.ros.org/doc/ros2/Installation/Foxy/
Installing ROS 2 Foxy Fitzroy
Binary packages
Linux (Ubuntu Focal(20.04)
Debian packages
Installing ROS 2 via Debian Packages
Setup Locale
$ **sudo locale-gen en_US en_US.UTF-8**
$ **sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8**
$ **export LANG=en_US.UTF-8**
Setup Sources
$ **sudo apt update && sudo apt install curl gnupg2 lsb-release**
$ **curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -**
$ **sudo apt autoremove** # optional - to clean up unused install files
$ **sudo sh -c 'echo "deb [arch=$(dpkg --print-architecture)] http://packages.ros.org/ros2/ubuntu $(lsb_release -cs) main" > /etc/apt/sources.list.d/ros2-latest.list'**
$ **sudo apt update**
$ **sudo apt install ros-foxy-desktop**
Environment setup
Sourcing the setup script
DEFER til later
Install argcomplete (optional)
ROS 2 command line tools use argcomplete to autocompletion.
So if you want autocompletion, installing argcomplete is necessary.
$ **sudo apt install python3-argcomplete**
Install additional RMW implementations (optional, do)
$ **sudo apt install ros-foxy-rmw-connext-cpp**
you may have to ENTER down to end of license and type in y-e-s
Sourcing the setup script
NOW do it
$ **source /opt/ros/foxy/setup.bash**
edit .bashrc file to auto-source--
$ **cd /opt/ros/foxy/**
$ gedit .bashrc
scroll to bottom of file and add (on line 119)
source /opt/ros/foxy/setup.bash
and save

Lesson 9. Launch a ROS2 Program:
james@JP-VM:~$ **ros2 run demo_nodes_cpp talker**
james@JP-VM:~$ **ros2 run demo_nodes_cpp listener**

Lesson 10. Write Your First ROS2 Node
Lesson 11. Create a ROS2 Workspace
james@JP-VM:~$ **mkdir ros2_ws**
james@JP-VM:~$ **cd ros2_ws/**
james@JP-VM:~/ros2_ws$ **mkdir src**
james@JP-VM:~/ros2_ws$ **ls**
src
Lesson 12. Add auto-completion for Colcon
james@JP-VM:~/ros2_ws$ **colcon build**
colcon: command not found
Google: “colcon: command not found ROS2 Ubuntu 20.04", tools/past year:
Index - ROS.orgindex.ros.org › doc › ros2 › Tutorials › Colcon-Tutorial
Using colcon to build packages - ROS
https://index.ros.org/doc/ros2/Tutorials/Colcon-Tutorial/
Prerequisites
sudo apt install python3-colcon-common-extensions
james@JP-VM:~/ros2_ws$
sudo apt install python3-colcon-common-extensions
james@JP-VM:~/ros2_ws$ **colcon build**
Summary: 0 packages finished [0.79s]
james@JP-VM:~/ros2_ws$ **ls**
build install log src
james@JP-VM:~/ros2_ws$ **cd install/**
james@JP-VM:~/ros2_ws/install$ **ls**
COLCON_IGNORE local_setup.sh local_setup.zsh setup.sh
local_setup.bash _local_setup_util_ps1.py setup.bash setup.zsh
local_setup.ps1 _local_setup_util_sh.py setup.ps1
setup.bash needs to be sourced every time you open a new terminal, or—
edit the .bashrc file:
james@JP-VM:~$ **gedit ~/.bashrc**
add new line 120 --
source ~/ros2_ws/install/setup.bash
add autocomplete to colcon build
This was already done when colcon was installed above
james@JP-VM:~/ros2_ws/install$ **cd /usr/share/colcon_argcomplete/hook/**
james@JP-VM:/usr/share/colcon_argcomplete/hook$ **ls**
colcon-argcomplete.bash colcon-argcomplete.zsh
Need to source colcon-argcomplete.bash:
james@JP-VM:~$ **gedit ~/.bashrc**
add new line 121 --
source /usr/share/colcon_argcomplete/hook/colcon-argcomplete.bash

Lesson 13. Create a Python Package

james@JP-VM:~$ **cd ros2_ws/src/** # NOTE starts from home directory

[CAUTION - - gets changed to – in WordPerfect and doesn't copy/paste correctly]

james@JP-VM:~/ros2_ws/src$ **ros2 pkg create my_py_pkg --build-type ament_python --dependencies rclpy**

```
going to create a new package
package name: my_py_pkg
destination directory: /home/james/ros2_ws/src
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['james <james@todo.todo>']
licenses: ['TODO: License declaration']
build type: ament_python
dependencies: ['rclpy']
creating folder ./my_py_pkg
creating ./my_py_pkg/package.xml
creating source folder
creating folder ./my_py_pkg/my_py_pkg
creating ./my_py_pkg/setup.py
creating ./my_py_pkg/setup.cfg
creating folder ./my_py_pkg/resource
creating ./my_py_pkg/resource/my_py_pkg
creating ./my_py_pkg/my_py_pkg/__init__.py
creating folder ./my_py_pkg/test
creating ./my_py_pkg/test/test_copyright.py
creating ./my_py_pkg/test/test_flake8.py
creating ./my_py_pkg/test/test_pep257.py
```

james@JP-VM:~/ros2_ws/src$ **ls -l my_py_pkg/**

```
total 24
drwxrwxr-x 2 james james 4096 Aug 15 17:41 my_py_pkg
```

same name as parent folder

*all **python** nodes will be placed here*

```
-rw-rw-r-- 1 james james  651 Aug 15 17:41 package.xml
```

see contents below –

```
drwxrwxr-x 2 james james 4096 Aug 15 17:41 resource
-rw-rw-r-- 1 james james   87 Aug 15 17:41 setup.cfg
-rw-rw-r-- 1 james james  615 Aug 15 17:41 setup.py
drwxrwxr-x 2 james james 4096 Aug 15 17:41 test
```

package.xml

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>

<!--The part below if info if sharing / releasing / licensing.  Not used in tutorial.-->
<package format="3">
  <name>my_py_pkg</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="james@todo.todo">james</maintainer>
  <license>TODO: License declaration</license>

<!--The part below is list of dependencies for all the python nodes in the pkg.-->
  <depend>rclpy</depend>

  <test_depend>ament_copyright</test_depend>
  <test_depend>ament_flake8</test_depend>
  <test_depend>ament_pep257</test_depend>
  <test_depend>python3-pytest</test_depend>

<!--Below in the ament build type for the pkg.  NEEDED–>
  <export>
    <build_type>ament_python</build_type>
  </export>
</package>
```

To compile package:
james@JP-VM:~$ **cd ros2_ws/**
james@JP-VM:~/ros2_ws$ **colcon build**
Starting >>> my_py_pkg
Finished <<< my_py_pkg [2.70s]

Summary: 1 package finished [3.16s]

OR, to compile only python and not C++ packages or just the one pkg do:
[autocomplete DOES work but it's slow to respond – give it a couple seconds]
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg**

Lesson 14. Create a C++ Package

james@JP-VM:~/ros2_ws/src $ **ros2 pkg create my_cpp_pkg --build-type ament_cmake --dependencies rclcpp**

```
going to create a new package
package name: my_cpp_pkg
destination directory: /home/james/ros2_ws
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['james <james@todo.todo>']
licenses: ['TODO: License declaration']
build type: ament_cmake
dependencies: ['rclcpp']
creating folder ./my_cpp_pkg
creating ./my_cpp_pkg/package.xml
creating source and include folder
creating folder ./my_cpp_pkg/src
creating folder ./my_cpp_pkg/include/my_cpp_pkg
creating ./my_cpp_pkg/CMakeLists.txt
```

james@JP-VM:~/ros2_ws/src$ **ls**

```
my_cpp_pkg  my_py_pkg
```

my_cpp-pkg/src/

package.xml:

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>

<!--Information section, OPTIONAL-->
<package format="3">
  <name>my_cpp_pkg</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="james@todo.todo">james</maintainer>
  <license>TODO: License declaration</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

<!--List of dependencies-->
  <depend>rclcpp</depend>

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

```
<!--ament_cmake flag-->
  <export>
    <build_type>ament_cmake</build_type>
  </export>
</package>
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5)
project(my_cpp_pkg)

# 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()

# PUT DEPENCENCIES HERE
# find dependencies
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  # the following line skips the linter which checks for copyrights
  # uncomment the line when a copyright and license is not present in all source files
  #set(ament_cmake_copyright_FOUND TRUE)
  # the following line skips cpplint (only works in a git repo)
  # uncomment the line when this package is not in a git repo
  #set(ament_cmake_cpplint_FOUND TRUE)
  ament_lint_auto_find_test_dependencies()
endif()

ament_package()
```

DEPENDENCIES NEED TO BE IN BOTH:
CmakeLists.txt
package.xml

To compile cpp package:
james@JP-VM:~/ros2_ws$ **colcon build**
Starting >>> my_cpp_pkg
Finished <<< my_cpp_pkg [29.5s]
Starting >>> my_py_pkg
Finished <<< my_py_pkg [3.51s]

Summary: 2 packages finished [**34.0s**]

OR, much faster to just do one

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**
Starting >>> my_cpp_pkg
Finished <<< my_cpp_pkg [0.63s]

Summary: 1 package finished [**1.07s**]

Lesson 15. What is a ROS2 Node?
A subprogram in your application, responsible for only one thing
Combined into a graph
Communicate with each other through topics, services, and parameters
Benefits:
- Reduce code complexity
- Fault tolerance
- Can be written in Python, C++, ... any supported language

All node names must be unique

Lesson 16. Write a Python Node - Minimal Code

```
james@JP-VM:~/ros2_ws/src$ ls
my_cpp_pkg  my_py_pkg
james@JP-VM:~/ros2_ws/src$ cd my_py_pkg/          # first layer my_py_pkg
james@JP-VM:~/ros2_ws/src/my_py_pkg$ ls
my_py_pkg  package.xml  resource  setup.cfg  setup.py  test
james@JP-VM:~/ros2_ws/src/my_py_pkg$ cd my_py_pkg/ # second layer my_py_pkg
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ ls
__init__.py
# MAKE NEW NODE FILE HERE
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ touch my_first_node.py
# edit file in Visual Studio Code:

## ros2_ws/src/my_py_pkg/my_py_pkg/my_first_node.py
#!/usr/bin/env python3
import rclpy    # is already in package.xml as depencency
from rclpy.node import Node     # note Case

def main(args=None):
        # must be first line of ROS2 node to initialize ROS2 communication.
        rclpy.init(args=args)
        # THIS defines NODE NAME, not the name of the file
        node = Node("py_test")     node.get_logger().info("Hello ROS2")
        # must be last line of program to clean up communication
        rclpy.shutdown
if __name__ == "__main__":
        main()

james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ chmod +x my_first_node.py
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ ./my_first_node.py
Traceback (most recent call last):
  File "./my_first_node.py", line 3, in <module>
    from rclpy.node import node     # note Case is WRONG, my error, see below!
ImportError: cannot import name 'node' from 'rclpy.node'
(/opt/ros/foxy/lib/python3.8/site-packages/rclpy/node.py)
tried
james@JP-VM:~/ros2_ws$ colcon build --packages-select my_py_pkg
Starting >>> my_py_pkg
Finished <<< my_py_pkg [2.69s]
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ python3 my_first_node.py
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ chmod +x my_first_node.py
Same error after trying each.
```

Change:
   from rclpy.node import **N**ode
and it works fine:
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **./my_first_node.py**
[INFO] [1597614626.597404461] [py_test]: Hello ROS2

Add spin function to keep node running...
```
        rclpy.spin(node)      # keeps node running, kill w/ ^C
        rclpy.shutdown()      # must be last line of program to clean up communication
```

Install file into ROS2 workspace:
edit **setup.py**:

```
from setuptools import setup

package_name = 'my_py_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='james',
   maintainer_email='james@todo.todo',
   description='TODO: Package description',
   license='TODO: License declaration',
   tests_require=['pytest'],
   entry_points={
      'console_scripts': [
         "py_node = my_py_pkg.my_first_node:main"
      ],
   },
)
```

“name_of_executable=name_of_package.name_of_py_file:name_of_start_function

james@JP-VM:~/ros2_ws/install/my_py_pkg/lib$ **cd ~/ros2_ws**
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg**
Starting >>> my_py_pkg
Finished <<< my_py_pkg [2.26s]
Summary: 2 packages finished [4.65s]
james@JP-VM:~/ros2_ws$ **ls**
build **install** log src
james@JP-VM:~/ros2_ws$ **cd install/**
james@JP-VM:~/ros2_ws/install$ **ls**
COLCON_IGNORE _local_setup_util_ps1.py **my_py_pkg** setup.zsh
local_setup.bash _local_setup_util_sh.py setup.bash
local_setup.ps1 local_setup.zsh setup.ps1
local_setup.sh my_cpp_pkg setup.sh
james@JP-VM:~/ros2_ws/install$ **cd my_py_pkg/**
james@JP-VM:~/ros2_ws/install/my_py_pkg$ **ls**
lib share
james@JP-VM:~/ros2_ws/install/my_py_pkg$ **cd lib**
james@JP-VM:~/ros2_ws/install/my_py_pkg/lib$ **ls**
my_py_pkg python3.8
james@JP-VM:~/ros2_ws/install/my_py_pkg/lib$ **cd my_py_pkg/**
james@JP-VM:~/ros2_ws/install/my_py_pkg/lib/my_py_pkg$ **ls**
py_node
james@JP-VM:~/ros2_ws/install/my_py_pkg/lib/my_py_pkg$ **./py_node**
[INFO] [1597859118.994740342] [py_test]: Hello ROS2
^C

Now run from ROS:
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **cd**
james@JP-VM:~$ **source .bashrc** *[refresh .bashrc]*
james@JP-VM:~$ **ros2 run my_py_pkg py_node**
[INFO] [1597857720.643245069] [py_test]: Hello ROS2

Takeaway lesson: name of file, node & executable may all be DIFFERENT.

Lesson 17. Write a Pyton Node - With OOP

my_first_node.py:

```
#!/usr/bin/env python3
# ros2_ws/src/my_py_pkg/my_py_pkg/my_first_node.py WITH OOP
import rclpy
from rclpy.node import Node      # note Case of node Node

class MyNode(Node):

    def __init__(self):
        super().__init__("py_test")
        self.get_logger().info("Hello ROS2")

def main(args=None):
    rclpy.init(args=args)   # must be first line of ROS2 node to initialize ROS2
communication, will fail w/o this!
    node = Node("py_test")  # THIS defines NODE NAME, not the name of the file
    node.MyNode()
    rclpy.spin(node)      # keeps node running, kill w/ ^C
    rclpy.shutdown()      # must be last line of program to clean up communication

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

```
james@JP-VM:~/ros2_ws/install/my_py_pkg/lib/my_py_pkg$ cd ~/ros2_ws/
james@JP-VM:~/ros2_ws$ colcon build --packages-select my_py_pkg
Starting >>> my_py_pkg
Finished <<< my_py_pkg [2.18s]

Summary: 1 package finished [2.54s]

james@JP-VM:~$ ros2 run my_py_pkg py_node
[INFO] [1597863936.300635274] [py_test]: Hello ROS2

^C
```

ADD A TIMER TO THE PROGRAM

```
#!/usr/bin/env python3
# ros2_ws/src/my_py_pkg/my_py_pkg/my_first_node.py
import rclpy
from rclpy.node import Node     # note Case of node Node

class MyNode(Node):

    def __init__(self):
        super().__init__("py_test")
        self.get_logger().info("Hello ROS2")
        self.create_timer(0.5, self.timer_callback)

    def timer_callback(self):
        self.get_logger().info("Hello")

def main(args=None):
    # must be first line of ROS2 node to initialize ROS2 communication, will fail w/o this!
    rclpy.init(args=args)
          node = MyNode()        # THIS defines NODE NAME, not the name of the file
    rclpy.spin(node)      # keeps node running, kill w/ ^C
    rclpy.shutdown()      # must be last line of program to clean up communication

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

```
ames@JP-VM:~/ros2_ws$ colcon build --packages-select my_py_pkg
...
james@JP-VM:~$ ros2 run my_py_pkg py_node
[INFO] [1597865626.180452642] [py_test]: Hello ROS2
[INFO] [1597865626.683175516] [py_test]: Hello
[INFO] [1597865627.185994886] [py_test]: Hello
...
^C
```

ADD A COUNTER

```
        self.counter_ = 0
        self.counter_ += 1
        self.get_logger().info(“Hello “ + str(self.counter_))
```

```
#!/usr/bin/env python3
# ros2_ws/src/my_py_pkg/my_py_pkg/my_first_node.py
import rclpy
from rclpy.node import Node     # note Case of node Node

class MyNode(Node):

    def __init__(self):
        super().__init__("py_test")
        self.counter_ = 0
        self.get_logger().info("Hello ROS2")
        self.create_timer(0.5, self.timer_callback)

    def timer_callback(self):
        self.counter_ += 1
        self.get_logger().info("Hello" + str(self.counter_))

def main(args=None):
    rclpy.init(args=args)       # This must be first line of program
    node = MyNode()             # THIS defines NODE NAME, not the name of the file
    rclpy.spin(node)            # keeps node running, kill w/ ^C
    rclpy.shutdown()            # must be last line of program to clean up communication

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

```
james@JP-VM:~/ros2_ws$ colcon build --packages-select my_py_pkg

james@JP-VM:~$ ros2 run my_py_pkg py_node
[INFO] [1597867021.239430150] [py_test]: Hello ROS2
[INFO] [1597867021.745028736] [py_test]: Hello 1
[INFO] [1597867022.244815368] [py_test]: Hello 2
[INFO] [1597867022.741510160] [py_test]: Hello 3
...
^C
```

Lesson 18. Write a C++ Node - Minimal Code
james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **touch my_first_node.cpp**

#include "rclcpp/rclcpp.hpp" # get warning from editor that path is unknown
do **Ctrl-Shift-P**
enter C/C++ and look for Edit Configurations (JSON)
find james@JP-VM:**/opt/ros/foxy/include**$
edit **c_cpp_properties.json**:

```
        "includePath": [
                "${workspaceFolder}/**",        # NOTE comma here!
                "/opt/ros/foxy/include"         # add include path
         ],
```

Suggestion: learn more about C++ Shared Pointers, they're used EVERYWHERE.

```
#include "rclcpp/rclcpp.hpp"
//~/ros2_ws/src/my_cpp_pkg/src/my_first_node.cpp        // is a C++ comment line

int main(int argc, char **argv)
{
   rclcpp::init(argc, argv);
   auto node = std::make_shared<rclcpp::Node>"cpp_test");
   RCLCPP INFO(node->get_logger(), "Hello Cpp Node");
   rclcpp::spin(node);
   rclcpp::shutdown();
   return 0;
}
```

To compile, edit **CmakeLists.txt**:

```
cmake_minimum_required(VERSION 3.5)
project(my_cpp_pkg)

# 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(rclcpp REQUIRED)

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  # the following line skips the linter which checks for copyrights
  # uncomment the line when a copyright and license is not present in all source files
  #set(ament_cmake_copyright_FOUND TRUE)
  # the following line skips cpplint (only works in a git repo)
  # uncomment the line when this package is not in a git repo
  #set(ament_cmake_cpplint_FOUND TRUE)
  ament_lint_auto_find_test_dependencies()
endif()

ament_package()

BECOMES ...

# 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(rclcpp REQUIRED)

# here we create executable
add_executable(cpp_node src/my_first_node.cpp) // compile to executable
ament_target_dependencies(cpp_node rclcpp)

# here we instal executable
// install to PROJECT_NAME = package name (my_cpp_pkg)
install(TARGETS
        cpp_node
        DESTINATION lib/$(PROJECT_NAME)
)
ament_package()
```

COMPILE:
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**

RUN:
james@JP-VM:~/ros2_ws/install/my_cpp_pkg$ **ls**
share *There should be a "lib" folder here, too*
Skip after* GO BACK TO Lesson 18 above *if this has been redone.

Let's rebuld my_cpp_pkg...
james@JP-VM:~/ros2_ws/install$ **rm -rf my_cpp_pkg**
james@JP-VM:~/ros2_ws/src$ **ros2 pkg create my_cpp_pkg --build-type ament_cmake --dependencies rclcpp**
going to create a new package
package name: my_cpp_pkg
destination directory: /home/james/ros2_ws/src
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['james <james@todo.todo>']
licenses: ['TODO: License declaration']
build type: ament_cmake
dependencies: ['rclcpp']
creating folder ./my_cpp_pkg
creating ./my_cpp_pkg/package.xml
creating source and include folder
creating folder ./my_cpp_pkg/src
creating folder ./my_cpp_pkg/include/my_cpp_pkg
creating ./my_cpp_pkg/CMakeLists.txt
james@JP-VM:~/ros2_ws/src$ **ls**
my_cpp_pkg my_py_pkg
james@JP-VM:~/ros2_ws/src$ **cd my_cpp_pkg/**
james@JP-VM:~/ros2_ws/src/my_cpp_pkg$ **ls**
CMakeLists.txt include package.xml src

GO BACK TO Lesson 18 above

After redoing the cpp tutorial, I STLL fail to get a lib folder and the executable.
I sent an email to the teacher & got this response & my reply (condensed):

Edouard,
Very much appreciate the prompt reply!!
Vesa and I are stuck on the same Lesson 18 cpp node minimal code.
Attached is my CMakeList.txt which I believe is configured as per the lesson.
Also the Package.xml and my_first_node.cpp.
During the my_first_node.py exercise I had an error where I put "node" instead of "Node".
However, I don't see any simple such typos here. Maybe I've overlooked something?
I'm doing the class via Udemy instead of Skillshare as I just want the 1 class, not a whole year. Is it updated there, too?
It seems tutorials demonstrated on the teacher's machine that has been already installed never work the same when done from the student's naive machine.
The teacher always seems to have something configured that wasn't included in the directions.
Can you, following your own tutorial on a completely new machine, get the same results?

Jim

On 8/22/2020 10:14 AM, contact@roboticsbackend.com wrote:
> Hi Jim,
>
> Thanks for reaching out, and thanks for the kind words!
> Actually you heard it right, I said letter "Control + shift + P". And yes, I agree, pronouncing "E" and "I" correctly is quite a challenge for me :) but I'm getting better...
> Also, one quick note: I've just updated the course because some installation instructions changed due to a recent update in the ROS2 Foxy installation process. So the lesson you're talking about (#18) is now lesson #19.
> For your second question: I guess you forgot some lines in the CMakeLists.txt of your my_cpp_pkg.
> First, you have 2 lines to compile the node: add_executable(..) and ament_target_dependencies(..). This will just compile the executable. To install the executable into the lib folder (and thus create the lib folder), you have to also write the install(TARGETS ... DESTINATION lib/...) line. --> Let me know if that doesn't solve the issue. Also, you can download the code archives, and try to compile + run the nodes from the "code_end_section3", so you can be sure your environment is correctly setup.
> One more thing: if you liked the course, I would really appreciate that you give review on Skillshare, that would really help me on the platform, and support me to produce other courses.
>
> And speaking of ROS2 and Raspberry Pi, would you mind telling me what kind of things you'd like to learn about that? This pre-feedback could help me create tutorials (and courses) which really solve a problem for you and other ROS learners!

> Best regards,
> Edouard
>
> On 2020-08-21 01:59, JAMES H PHELAN wrote:
>> From: JAMES H PHELAN <jhphelan@hal-pc.org>
>> Subject: Skillshare: ROS2 for Beginners; Lesson 18 doesn't compile
>>
>> Message:
>> Ed,
>> Really enjoying your course so far! Friend Vesa Pykala recommended it
>> who started ahead of me.
>> Even the French accent! (I took 7 yrs of French in school decades ago.)
>> It did confuse me in Lesson 18 at the #include "rclcpp/rclcpp.hpp"
>> when you said do "Control-P". I wasn't sure if you said "Control-E"
>> or "Control-I (using French pronounciation of "i")" so caused it to
>> offer some bizarre code snippets until I rebooted the VM.
>> Doing well x- for glitch in Lesson 16 where I failed to source
>> ~/.bashrc. But recovered after review.
>> I thought putting source in ~/.bashrc eliminated that need, but....
>>
>> <<Now I'm stuck on Lessoon 18. My files match yours, but after I
>> colcon, there is no lib folder or node. I even went back and deleted
>> ~/ros2_ws/src/my_cpp_pkg and started over, but same failure.
>> Any ideas?>>
>>
>> I see you have many more goodies in roboticsbackend.com. Particularly
>> interested in the article installing ROS2 on Pi4 as that is the next
>> upgrade to the Open Source Rover.
>> Jim Phelan
>> "jhphelan":
>> https://github.com/nasa-jpl/open-source-rover   (our group's (USAi
>> Labs) project lead for rover)
>> https://www.tapatalk.com/groups/jpl_opensource_rover/index.php
>> https://www.tapatalk.com/groups/usailabs/index.php
>> https://www.youtube.com/watch?v=2Wdgss0q63s

In order to include the files in the above e-mail I had to create a shared folder in VirtualBox which I did, but it didn't work as I was denied permission to open. After Googling around I found this:
https://dev.to/rahedmir/virtualbox-cannot-access-shared-folder-items-permission-denied-fixed-59mi
after which I was able to enter the folder in filing cabinet and drop files there to share with my laptop.

2020.08.24 Home

Reply from Eduard, instructor:

Hi Jim,

I've looked at your files and you have a typo in your .cpp file: you wrote "rclcpp::spiin" instead of "rclcpp::spin". Also, if you're still having an error after fixing this, please send me the error log so I can better help you.

And yes, I've just updated the course on Udemy and Skillshare.

Could you tell me for which lessons you had some problems? So I can fix those.

Best regards,
Edouard

To compile:

```
james@JP-VM:~$ cd ros2_ws/
james@JP-VM:~/ros2_ws$ colcon build --packages-select my_cpp_pkg
Starting >>> my_cpp_pkg
[Processing: my_cpp_pkg]
Finished <<< my_cpp_pkg [32.0s]

Summary: 1 package finished [33.3s]
james@JP-VM:~/ros2_ws$ ls
build  install  log  src
james@JP-VM:~/ros2_ws$ cd install
james@JP-VM:~/ros2_ws/install$ ls
COLCON_IGNORE            my_cpp_pkg
local_setup.bash         my_py_pkg
local_setup.ps1          setup.bash
local_setup.sh           setup.ps1
_local_setup_util_ps1.py  setup.sh
_local_setup_util_sh.py   setup.zsh
local_setup.zsh
james@JP-VM:~/ros2_ws/install$ cd my_cpp_pkg/
james@JP-VM:~/ros2_ws/install/my_cpp_pkg$ ls
lib  share
james@JP-VM:~/ros2_ws/install/my_cpp_pkg$ cd lib
james@JP-VM:~/ros2_ws/install/my_cpp_pkg/lib$ ls
my_cpp_pkg
james@JP-VM:~/ros2_ws/install/my_cpp_pkg/lib$ cd my_cpp_pkg/
james@JP-VM:~/ros2_ws/install/my_cpp_pkg/lib/my_cpp_pkg$ ls
cpp_node
james@JP-VM:~/ros2_ws/install/my_cpp_pkg/lib/my_cpp_pkg$ ./cpp_node
[INFO] [1598317119.858566746] [cpp_test]: Hello Cpp Node

^C[INFO] [1598317125.582129836] [rclcpp]: signal_handler(signal_value=2)
[success!]
OR

james@JP-VM:~$ source .bashrc                  [Refresh ros2 enviro.]
james@JP-VM:~$ ros2 run my_cpp_pkg [TAB][TAB]        [show avail nodes in pkg]
cpp_node  --prefix
james@JP-VM:~$ ros2 run my_cpp_pkg cpp_node
[INFO] [1598317380.672977222] [cpp_test]: Hello Cpp Node

^C[INFO] [1598317385.711311789] [rclcpp]: signal_handler(signal_value=2)
[success!]
```

Lesson 20. Write a C++ Node - With OOP
my_first_node.cpp:

```
#include "rclcpp/rclcpp.hpp"

class MyNode: public rclcpp::Node
{
public:
   MyNode(): Node("cpp_test")
   {
      RCLCPP_INFO(this->get_logger(), "Hello Cpp Node");
   }

private:

}

int main(int argc, char **argv)
{
   rclcpp::init(argc, argv);
   auto node = std::make_shared<MyNode>();
   rclcpp::spin(node);
   rclcpp::shutdown();
   return 0;
}
```

Compile:
james@JP-VM:~$ cd ros2_ws/
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**
Starting >>> my_cpp_pkg
Finished <<< my_cpp_pkg [3.21s]

Summary: 1 package finished [4.01s]

Run:
james@JP-VM:~$ **source .bashrc**
james@JP-VM:~$ **ros2 run my_cpp_pkg cpp_node**
[INFO] [1598398673.743272436] [cpp_test]: Hello Cpp Node

^C[INFO] [1598398681.013265144] [rclcpp]: signal_handler(signal_value=2)

Make program more interesting:
add a timer –
WATCH FOR TYPOS!! {} and ; can appear in the strangest places!

my_first_node.cpp:

```
#include "rclcpp/rclcpp.hpp"

class MyNode : public rclcpp::Node
{
public:
   MyNode() : Node("cpp_test")
   {
      RCLCPP_INFO(this->get_logger(), "Hello Cpp Node");

      timer_ = this->create_wall_timer(std::chrono::seconds(1),
                                       std::bind(&MyNode::timerCallback, this));
   }

private:
   void timerCallback()
   {
      RCLCPP_INFO(this->get_logger(), "Hello");
   }

   rclcpp::TimerBase::SharedPtr timer_;
};

int main(int argc, char **argv)
{
   rclcpp::init(argc, argv);
   auto node = std::make_shared<MyNode>();
   rclcpp::spin(node);
   rclcpp::shutdown();
   return 0;
}
```

COMPILE:
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**
RUN:
james@JP-VM:~$ **ros2 run my_cpp_pkg cpp_node**
[INFO] [1598400033.028912482] [cpp_test]: Hello Cpp Node
[INFO] [1598400034.029550353] [cpp_test]: Hello
[INFO] [1598400037.029857251] [cpp_test]: Hello
^C[INFO] [1598400037.944821377] [rclcpp]: signal_handler(signal_value=2)

ADD A COUNTER:

my_first_node.cpp:

```
#include "rclcpp/rclcpp.hpp"

class MyNode : public rclcpp::Node
{
public:
   MyNode() : Node("cpp_test"), counter_(0)
   {
      RCLCPP_INFO(this->get_logger(), "Hello Cpp Node");

      timer_ = this->create_wall_timer(std::chrono::seconds(1),
                                       std::bind(&MyNode::timerCallback, this));
   }

private:
   void timerCallback()
   {
      counter_++;
      RCLCPP_INFO(this->get_logger(), "Hello %d", counter_);
   }

   rclcpp::TimerBase::SharedPtr timer_;
   int counter_;
};

int main(int argc, char **argv)
{
   rclcpp::init(argc, argv);
   auto node = std::make_shared<MyNode>();
   rclcpp::spin(node);
   rclcpp::shutdown();
   return 0;
}
```

Compile:

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**

Run:

james@JP-VM:~$ **ros2 run my_cpp_pkg cpp_node**

```
[INFO] [1598400686.462926693] [cpp_test]: Hello Cpp Node
[INFO] [1598400687.463512617] [cpp_test]: Hello 1
[INFO] [1598400688.463889460] [cpp_test]: Hello 2
[INFO] [1598400689.463580636] [cpp_test]: Hello 3
[INFO] [1598400690.463484330] [cpp_test]: Hello 4
^C[INFO] [1598400690.843565941] [rclcpp]: signal_handler(signal_value=2)
```

Lesson 22. OOP Templates for Your Nodes

OOP Python Code Template for Nodes

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node


class MyCustomNode(Node): # MODIFY NAME
    def __init__(self):
        super().__init__("node_name") # MODIFY NAME


def main(args=None):
    rclpy.init(args=args)
    node = MyCustomNode() # MODIFY NAME
    rclpy.spin(node)
    rclpy.shutdown()


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

OOP C++ Code Template for Nodes

```
#include "rclcpp/rclcpp.hpp"

class MyCustomNode : public rclcpp::Node // MODIFY NAME
{
public:
   MyCustomNode() : Node("node_name") // MODIFY NAME
   {
   }

private:
};

int main(int argc, char **argv)
{
   rclcpp::init(argc, argv);
   auto node = std::make_shared<MyCustomNode>(); // MODIFY NAME
   rclcpp::spin(node);
   rclcpp::shutdown();
   return 0;
}
```

Lesson 23. More about ROS2 Client Libraries for Different Languages

rcl = “ROS Client Library” base client library
You will use super-libraries for certain languages i.e.
rclcpp, rclpy, rcljava, rcl...
dds = Data Distribution Service

Section 4: Introduction to ROS2 Tools
Lesson 25. Intro

[source on ad hoc basis:]
"ros2 command not found" = failed to source ~/.bashrc

james@JP-VM:~$ **source ~/.bashrc**
and
[automatic source for any newly opened terminal
...add following to end of ~/.bashrc:]
james@JP-VM:~$ **cat ~/.bashrc**
source /opt/ros/foxy/setup.bash
source ~/ros2_ws/install/setup.bash
source /usr/share/colcon_argcomplete/hook/colcon-argcomplete.bash

Lesson 26. Debug and Monitor Your Nodes With ros2 cli (command line interface)
[Display ros2 functions]
james@JP-VM:~$ **ros2[space][tab][tab]**

action	extension_points	multicast	security
bag	extensions	**node**	service
component	interface	param	topic
daemon	launch	**pkg**	wtf
doctor	lifecycle	**run**	

james@JP-VM:~$ ros2 run [*pkg*] [*executable*] from ROS2 global installation folder

james@JP-VM:~$ **ros2 run demo_nodes_cpp talker**
[INFO] [1598481010.794593003] [talker]: Publishing: 'Hello World: 1'
[INFO] [1598481011.794765577] [talker]: Publishing: 'Hello World: 2'
^C[INFO] [1598481012.444933363] [rclcpp]: signal_handler(signal_value=2)

[List executables within a package:]
james@JP-VM:~$ **ros2 run demo_nodes_cpp**[space][tab][tab]

add_two_ints_client	parameter_blackboard
add_two_ints_client_async	parameter_events
add_two_ints_server	parameter_events_async
allocator_tutorial	--prefix
even_parameters_node	reuse_timer
listener	set_and_get_parameters
listener_best_effort	set_and_get_parameters_async
listener_serialized_message	talker
list_parameters	talker_loaned_message
list_parameters_async	talker_serialized_message
one_off_timer	

[Get help on ros2 commands. -h = help:]
james@JP-VM:~$ **ros2 run -h**
usage: ros2 run [-h] [--prefix PREFIX] package_name executable_name ...

Run a package specific executable

positional arguments:
  package_name   Name of the ROS package
  executable_name Name of the executable
  argv   Pass arbitrary arguments to the executable

optional arguments:
  -h, --help   show this help message and exit
  --prefix PREFIX Prefix command, which should go before the executable. Command must be wrapped in quotes if it contains spaces (e.g. --prefix 'gdb -ex run --args').

james@JP-VM:~/ros2_ws$ **ros2 run my_py_pkg py_node**
[INFO] [1598481715.285515173] [**py_test**]: Hello ROS2
[INFO] [1598481715.789116793] [py_test]: Hello 1
[INFO] [1598481716.288605195] [py_test]: Hello 2
[INFO] [1598481716.787994366] [py_test]: Hello 3
...

In SEPARATE terminal window:
[Display commands avail for ros2 node:]
james@JP-VM:~$ ros2 node[space][tab][tab]
info list
[Run list to see what nodes are running:]
james@JP-VM:~$ ros2 node list
/py_test   [note / before executable name, will need for -info command]
james@JP-VM:~$ **ros2 run[space][tab][tab]**
Display all 265 possibilities? (y or **n**)

james@JP-VM:~/ros2_ws$ **ros2 node list**
/py_node_one
/py_node_two
james@JP-VM:~/ros2_ws$ **ros2 node info /py_node_one**   [note /, see above]
/py_node_one
  Subscribers: ...
  Publishers: ...
  Service Servers: ...
  Service Clients: ...
  Action Servers: ...
  Action Clients: ...

Lesson 27. Rename a Node at Runtime
running 2 nodes with same name will NOT cause error message,
BUT will cause interference between nodes and "unintended side effects."
e.g. if have multiple sensors, need to launch node for each sensor under different name
e.g. "temp_node" > temp_node1, temp_node2, ...

Run in 2 separate terminal windows:
james@JP-VM:~/ros2_ws$ **ros2 run my_py_pkg py_node**
james@JP-VM:~/ros2_ws$ **ros2 run my_py_pkg py_node**
In 3d window run:
james@JP-VM:~$ **ros2 node list**
WARNING: Be aware that are nodes in the graph that share an exact name, this can have unintended side effects.
/py_test
/py_test

james@JP-VM:~/ros2_ws$ **ros2 run my_py_pkg py_node**
--ros-args
--remap __node:py_node_one [or -r]

Lesson 28. Colcon
be sure this line at end of ~/.bashrc: [$ gedit ~/.bashrc]
source /usr/share/colcon_argcomplete/hook/colcon-argcomplete.bash

colcon build
build all packages (ineffecient if multiple packages)
colcon build --packages-select *package_name*
only builds named package (more effecient)
colcon build --packages-select *package_name* --symlink-install
only useful for python packages as C++ requires recompilation each time
creates link to source file to recompile at run time w/o rebuilding
requires *python_code*.py to executable so must do
$ chmod +x *python_code*.py

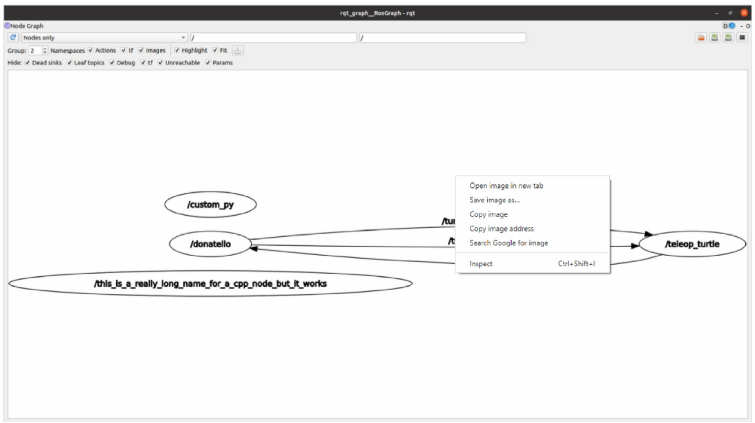
Lesson 29. Rtq and rqt_graph [should be Rqt]
james@JP-VM:~$ **rqt**
rqt executable w/in ROS2. Brings up separate window.
Open: Plugins / Introspection / Node Graph
shows only active nodes
james@JP-VM:~$ **rqt_graph**
rqt_graph skips the above dropdowns
shows active nodes & their relationships (better demo'd w/ turtlesim below)

Lesson 30. Discover Turtlesim
If not already installed:
$ sudo apt install ros-*foxy*-turtlesim [or your ros2 distro]
james@JP-VM:~$ **source ~/.bashrc** [or open new terminal]
james@JP-VM:~$ **ros2 run turtlesim[space][tab][tab]**
draw_square --prefix turtle_teleop_key
mimic turtlesim_node
james@JP-VM:~$ **ros2 run turtlesim turtlesim_node**
opens turtlesim window w/ turtle in the center
james@JP-VM:~$ **ros2 node list**
/turtlesim
james@JP-VM:~$ **ros2 run turtlesim turtle_teleop_key**

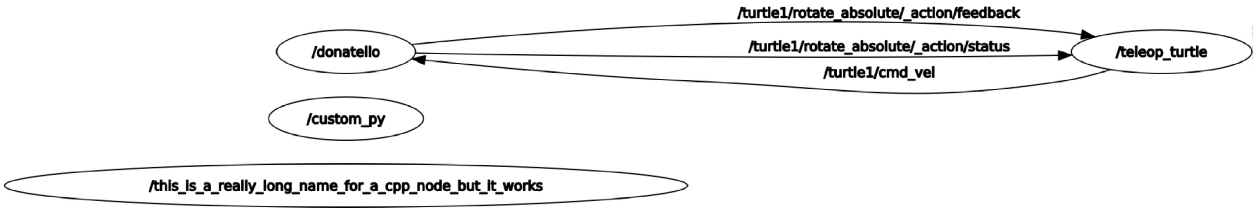
rqt_graph:
Nodes Only
Nodes/Topics (active)
Nodes/Topics (all)

Lesson 31. Activity 001 ROS2 Nodes and Tools
"(right click > "view image" to increase the size of the image in your browser)"
[This doesn't work. You can Ctrl-+ to increase the browser window then Ctrl-0 to resize. Or just view the Activity001.pdf]

(right click > "view image" to increase the size of the image in your browser)



james@JP-VM:~$ **ros2 run turtlesim turtlesim_node --ros-args --remap __node:=donatello**
james@JP-VM:~$ **ros2 run turtlesim turtle_teleop_key**
james@JP-VM:~$ **ros2 run my_py_pkg py_node --ros-args -r __node:=custom_py**
james@JP-VM:~$ **ros2 run my_cpp_pkg cpp_node --ros-args -r __node:=this_is_a_really_long_name_for_a_cpp_node_but_it_works**



Lesson 35. What is a ROS Topic?
A topic is a named bus over which nodes exchange messages
A topic name must begin with a letter followed by
letters, numbers, underscores _, tilde ~, forward slashes /
[tilde ~ pronounced “tyled” in the lesson but should be “til-dee”]
A topic is a unidirectional data stream publisher > subscriber
publishers & subscribers are anonymous
A topic has a message type
publishers/subscribers can be created
directly inside ROS nodes
using any supported language
using ROS2 libraries
A node can have many publishers/subscribers for many different topics

Lesson 36. Writing a Python Publisher
james@JP-VM:~$ **cd ros2_ws/src/my_py_pkg/my_py_pkg/**
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **ls**
build __init__.py install log my_first_node.py
[create program:]
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **touch robot_news_station.py**
[make executable:]
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$
chmod +x robot_news_station.py
Copy from template_python_node.py
[template_python_node.py is NOT in folder.
Copy from Shared Folder from course downloads, OR build it yourself.]

Convention is to have .py file, node name and executable name all the SAME.

~/ros2_ws/src/my_py_pkg/my_py_pkg**/robot_news_station.py** (template):

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node

class RobotNewsStationNode(Node):
    def __init__(self):
        super().__init__("RobotNewsStation")

def main(args=None):
    rclpy.init(args=args)
    node = RobotNewsStationNode()
    rclpy.spin(node)
    rclpy.shutdown()

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

Add node to console_scripts in setup.py:
[“*executable_name* = *package_name*.*program_name*:*function_name*”]

```
...
    entry_points={
        'console_scripts': [
            "py_node = my_py_pkg.my_first_node:main",          [note comma!]
            "robot_news_station = my_py_pkg.robot_news_station:main"
        ],
    },
```

Add publisher:
from Node class create publisher:
self.publisher_ = self.create_publisher() with 3 arguments:
        message type - find example from:
james@JP-VM:~$ **ros2 interface show example_interfaces/msg/String**
This is an example message of using a primitive datatype, string.
If you want to test with this that's fine, but if you are deploying
it into a system you should create a semantically meaningful message type.
If you want to embed it in another message, use the primitive data type instead.
string data

interface
show [print to screen]
example_interfaces [package name]
msg [folder for messages (topics)]
String [type of message, note Caps]
displays...
string data [data_type, data_field]

from example_interfaces.msg import String

Add example_interfaces dependency to my_py_pkg/**package.xml**:
<depend>**example_interfaces**</depend>

Add publisher to robot_news_station.py:
self.publisher_ = self.create_publisher(String, "robot_news", 10)
String = data type
“robot_news” = topic
10 = queue size for data buffer in case of asynchrony. 10 is good.
Create function to publish news:
def publish_news(self): [create publisher]
msg = String [note message data type]
msg.data = "Hello" [fill data field as “Hello”]
Create timer with right to publish on a topic
self.timer_ = self.create_timer(0.5, self.publish_news)
0.5 = period of timer (2 Hz)
self.publish_news = publisher function from above
Create log function for debugging at end of node constructor
self.get_logger().info("Robot New Station has been started")

What does it do?:
initialize ROS communication
rclpy.init(args=args)
create the node
node = RobotNewsStationNode()
calls the constructor from the Class
super().__init__("robot_news_station")
inside node create publisher with data type String and a name
self.publisher_ = self.create_publisher(String, "robot_news", 10)
create timer to call publish function at 2Hz
self.timer_ = self.create_timer(0.5, self.publish_news)
output debugging message
self.get_logger().info("Robot News Station has been started")
spin the node publishing news until ^C
rclpy.spin(node)
shutdown and cleanup
rclpy.shutdown()

~/ros2_ws/src/my_py_pkg/my_py_pkg**/robot_news_station.py** (final):

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node

from example_interfaces.msg import String

class RobotNewsStationNode(Node):
    def __init__(self):
        super().__init__("robot_news_station")

        self.publisher_ = self.create_publisher(String, "robot_news", 10)
        self.timer_ = self.create_timer(0.5, self.publish_news)
        self.get_logger().info("Robot News Station has been started")

    def publish_news(self):
        msg = String()
        msg.data = "Hello"
        self.publisher_.publish(msg)

def main(args=None):
    rclpy.init(args=args)
    node = RobotNewsStationNode()
    rclpy.spin(node)
    rclpy.shutdown()

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

Compile package to install node:
be sure program is executable:
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **ls -l**
...
-rwxrwxr-**x** 1 james james 703 Aug 27 17:38 **robot_news_station.py**
-rwxrwx--- 1 james james 353 Jun 10 03:11 template_python_node.py
Compile:
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg**
--symlink-install
Run (in another terminal)
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ ros2 **run my_py_pkg[space][tab][tab]**
--prefix py_node **robot_news_station**
james@JP-VM:~$ **ros2 run my_py_pkg robot_news_station**
[INFO] [1598662536.255517794] [robot_news_station]: Robot News Station has been started

james@JP-VM:~$ **ros2 node list**
/robot_news_station
james@JP-VM:~$ **ros2 topic list**
/parameter_events
/robot_news
/rosout
james@JP-VM:~$ **ros2 topic echo /robot_news**
data: Hello
---
data: Hello

Add name for robot:
self.robot_name_ = "C3PO"
Elaborate message:
msg.data = "Hi, this is " + str(self.robot_name_) + " from the robot news station."
[Because of --symlink-install, package does NOT have to be recompiled
BUT the .py file DOES have to be SAVED from the editor for this to work!!]
^Cjames@JP-VM:~$ **ros2 topic echo /robot_news**
data: Hi, this is C3PO from the robot news station.

Lesson 37. Write a Python Subscriber
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **ls**
build install my_first_node.py robot_news_station.py
__init__.py log __pycache__ template_python_node.py
Create smartphone.py subscriber to robot_news topic:
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **touch smartphone.py**
Make file executable for --symlink-install:
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **chmod +x smartphone.py**
Edit file & cc from python template:
class **SmartphoneNode**(Node): ~~# MODIFY NAME~~
node = **SmartphoneNode**() ~~# MODIFY NAME~~
super().__init__("**smartphone**") ~~# MODIFY NAME~~
Can use find/replace ~~# MODIFY NAME~~ >> [blank]
Add smartphone to setup.py to create executable:
"smartphone = my_py_pkg.smartphone:main"
[executable_name] = [package_name].[file_name]:[start_function]
Add logger:
self.get_logger().info("Smartphone has been started.")
Create subscriber:
Import String function:
from example_interfaces.msg import String
Create subscriber:
self.subscriber_ = self.create_subscription(
String, "robot_news", self.callback_robot_news, 10)

Create callback function to use the subscription data:

```
    def callback_robot_news(self, msg):
        self.get_logger().info(msg.data)
```

~/ros2_ws/src/my_py_pkg/my_py_pkg/**smartphone.py**:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node

from example_interfaces.msg import String

class SmartphoneNode(Node):
    def __init__(self):
        super().__init__("smartphone")
        self.subscriber_ = self.create_subscription(
            String, "robot_news", self.callback_robot_news, 10)
        self.get_logger().info("Smartphone has been started.")

    def callback_robot_news(self, msg):
        self.get_logger().info(msg.data)

def main(args=None):
    rclpy.init(args=args)
    node = SmartphoneNode()
    rclpy.spin(node)
    rclpy.shutdown()

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

PROBLEM - Could not get smartphone to compile. “Executable not found.” Also couldn’t get robot_news_station to execute.

SOLUTION:

```
    entry_points={
        'console_scripts': [
            "py_node = my_py_pkg.my_first_node:main",
```

[Missing comma here. All but last line needs comma after!:]

```
            "robot_news_station = my_py_pkg.robot_news_station:main",
"smartphone = my_py_pkg.smartphone:main"
        ],
    },
```

Compile, Run:
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg**
[new terminal window:]
james@JP-VM:~/ros2_ws$ **ros2 run my_py_pkg robot_news_station**
[INFO] [1598807527.741581062] [robot_news_station]: Robot News Station has been started
[new terminal window:]
james@JP-VM:~$ **ros2 topic echo /robot_news**
data: Hi, this is C3PO from the robot news station.
...
[new terminal window:]
james@JP-VM:~$ **ros2 run my_py_pkg smartphone**
[INFO] [1598807646.802981341] [smartphone]: Smartphone has been started.
[INFO] [1598807648.200931197] [smartphone]: Hi, this is C3PO from the robot news station.
...
[INFO] [1598807660.700438240] [smartphone]: Hi, this is C3PO from the robot news station.
^C
data: Hi, this is C3PO from the robot news station.
^C
Success!

Lesson 38. Write a C++ Publisher:
james@JP-VM:~/ros2_ws$ **cd src/my_cpp_pkg/src**
james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **ls**
my_first_node.cpp template_cpp_node.cpp
[NOTE robot_news_station & robot_news SAME as .py examples.
Will need to REMAP at run time to prevent conflict if run at same time]
james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **touch robot_news_station.cpp**
copy from template_cpp_node.cpp
rename MyCustomNode > RobotNewsStationNode
rename node_name > robot_news_station
Include example_interfaces:
```
#include "example_interfaces/msg/string.hpp"
```
Add to dependencies in package.xml:
```
  <depend>example_interfaces</depend>
```
Add to dependencies in CmakeLists.txt:
```
find_package(example_interfaces REQUIRED)
```
Declare publisher
```
rclcpp::Publisher<example_interfaces::msg::String>::SharedPtr publisher_;
```
Create publisher:
```
publisher_ =
this->create_publisher<example_interfaces::msg::String>("robot_news", 10);
```

Create function to publish news:

```
{
        auto msg = example_interfaces::msg::String();
        msg.data = std::string("Hi, this is ") + robot_name_ + std::string(" from the Robot
                News Station");
        publisher_->publish(msg);
}
```

Create timer to call the publisher:

```
        rclcpp::TimerBase::SharedPtr timer_;
        timer_ = this->create_wall_timer(
                std::chrono::milliseconds(500),
                std::bind(&RobotNewsStationNode::publishNews, this));
```

Create a debug message logger:

```
        RCLCPP_INFO(this->get_logger(), "Robot New Station has been started.");
```

Add executable to CmakeLists.txt:

```
        add_executable(robot_news_station src/robot_news_station.cpp)
        ament_target_dependencies(robot_news_station rclcpp example_interfaces)
        install(TARGETS
                cpp_node
                robot_news_station
                DESTINATION lib/${PROJECT_NAME}
        )
```

Compile; Run:

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**
[new terminal window:]
james@JP-VM:~/ros2_ws$ ros2 run my_cpp_pkg**[space][tab][tab]**
cpp_node --prefix **robot_news_station**
james@JP-VM:~/ros2_ws$ **ros2 run my_cpp_pkg robot_news_station**
[INFO] [1598818224.344787873] [robot_news_station]: Robot New Station has been started.
[new terminal window:]
james@JP-VM:~$ **ros2 node list**
/robot_news_station
james@JP-VM:~$ **ros2 node info /robot_news_station**
...
Publishers:
...
  /robot_news: example_interfaces/msg/String
...
james@JP-VM:~$ **ros2 topic list**
/parameter_events
/robot_news
/rosout

james@JP-VM:~$ **ros2 topic echo /robot_news**
data: Hi, this is R2D2 from the Robot News Station

[new terminal window:]
Run smartphone subscriber [py node can subscribe to a cpp node publisher]:
james@JP-VM:~$ **ros2 run my_py_pkg smartphone**
[INFO] [1598819003.738782892] [smartphone]: Smartphone has been started.
[INFO] [1598819005.349060962] [smartphone]: Hi, this is R2D2 from the Robot News Station

Lesson 39. Write a C++ Subscriber:
james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **ls**
my_first_node.cpp robot_news_station.cpp template_cpp_node.cpp
james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **touch smartphone.cpp**
Copy template_cpp_node.cpp
Rename MyCustomNode > SmartphoneNode
Rename node_name > smartphone
Include example_interfaces definition:

```
#include "example_interfaces/msg/string.hpp"
```

Is already in CmakeLists.txt from last
Is already in package.xml from last
Create subscriber and callback to handle message:

```
void callbackRobotNews(const example_interfaces::msg::String::SharedPtr msg)
{
    RCLCPP_INFO(this->get_logger(), "%s", msg->data.c_str());
}
```

Declare subscriber:

```
subscriber_ = this->create_subscription<example_interfaces::msg::String>(
    "robot_news", 10,
    std::bind(&SmartphoneNode::callbackRobotNews, this,
    std::placeholders::_1));
```

Create a debug message logger:

```
RCLCPP_INFO(this->get_logger(), "Smartphone has been started.");
```

~/ros2_ws/src/my_cpp_pkg/src/**smartphone.cpp**:

```
#include "rclcpp/rclcpp.hpp"
#include "example_interfaces/msg/string.hpp"
class SmartphoneNode : public rclcpp::Node
{
public:
   SmartphoneNode() : Node("smartphone")
   {
      subscriber_ = this->create_subscription<example_interfaces::msg::String>(
         "robot_news", 10,
         std::bind(&SmartphoneNode::callbackRobotNews, this, std::placeholders::_1));
      RCLCPP_INFO(this->get_logger(), "Smartphone has been started.");
   }

private:
   void callbackRobotNews(const example_interfaces::msg::String::SharedPtr msg)
   {
      RCLCPP_INFO(this->get_logger(), "%s", msg->data.c_str());
   }

   rclcpp::Subscription<example_interfaces::msg::String>::SharedPtr subscriber_;

};

int main(int argc, char **argv)
{
   rclcpp::init(argc, argv);
   auto node = std::make_shared<SmartphoneNode>();
   rclcpp::spin(node);
   rclcpp::shutdown();
   return 0;
}
```

Add executable to CmakeLists.txt:

```
        add_executable(smartphone src/smartphone.cpp)
        ament_target_dependencies(smartphone rclcpp example_interfaces)
install(TARGETS
        cpp_node
        robot_news_station
        smartphone
        DESTINATION lib/${PROJECT_NAME}
)
```

Compile; Run:
james@JP-VM:~/ros2_ws$ **source ~/.bashrc**
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**
[new terminal window:]
james@JP-VM:~$ **ros2 run my_cpp_pkg smartphone**
No executable found
Review and re-save all work files
james@JP-VM:~/ros2_ws$ **source ~/.bashrc**
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**
[new terminal window:]
james@JP-VM:~$ **ros2 run my_cpp_pkg smartphone**
[INFO] [1598823290.029121810] [smartphone]: Smartphone has been started.
[new terminal window:]
james@JP-VM:~/ros2_ws$ **source ~/.bashrc**
james@JP-VM:~/ros2_ws$ **ros2 run my_cpp_pkg robot_news_station**
[INFO] [1598823515.851804934] [robot_news_station]: Robot New Station has been started.
[Now in smartphone terminal window...]
[INFO] [1598823516.353357289] [smartphone]: Hi, this is **R2D2** from the Robot News Station
...
[new terminal window:]
james@JP-VM:~$ **ros2 node list**
/robot_news_station
/smartphone
[In my_cpp_pkg robot_news_station terminal:]
^C
james@JP-VM:~/ros2_ws$ **ros2 run my_py_pkg robot_news_station**
[INFO] [1598823889.343345041] [robot_news_station]: Robot News Station has been started
[From smartphone.cpp terminal now subscribing to python publisher:]
[INFO] [1598823942.244049672] [smartphone]: Hi, this is **C3PO** from the robot news station.

Lesson 40. Debug ROS2 Topics with Command Line Tools:

james@JP-VM:~$ **ros2 topic[space][tab][tab]**

bw	**hz**	pub
delay	--include-hidden-topics	**info**
type	**echo**	
find	**list**	

[new terminal window:]
Start a publisher for demo:]
james@JP-VM:~$ **ros2 run my_py_pkg robot_news_station**
[INFO] [1598825919.810228503] [robot_news_station]: Robot News Station has been started

list
[original terminal window:]
james@JP-VM:~$ **ros2 topic list**
/parameter_events
/robot_news [disappears if publisher node is killed ^C]
/rosout

info
james@JP-VM:~$ **ros2 topic info /robot_news**
Type: **example_interfaces/msg/String**
Publisher count: **1**
Subscription count: **0**
[new terminal window:]
james@JP-VM:~$ **ros2 topic echo /robot_news**
data: Hi, this is C3PO from the robot news station.

[original terminal window:]
james@JP-VM:~$ **ros2 topic info /robot_news**
Type: example_interfaces/msg/String
Publisher count: 1
Subscription count: **1** [now shows echo subscriber]

hz
james@JP-VM:~$ **ros2 topic hz /robot_news** [frequency of topic publication]
WARNING: topic [/robot_news] does not appear to be published yet
average rate: 1.999
min: 0.499s max: 0.501s std dev: 0.00069s window: 3
average rate: 2.000
min: 0.499s max: 0.501s std dev: 0.00090s window: 6
^C

bw
james@JP-VM:~$ **ros2 topic bw /robot_news** [size of message in Bytes]
WARNING: topic [/robot_news] does not appear to be published yet
Subscribed to [/robot_news]
162 B/s from 2 messages
Message size mean: 56 B min: 56 B max: 56 B
133 B/s from 4 messages
Message size mean: 56 B min: 56 B max: 56 B

pub
james@JP-VM:~$ **ros2 topic pub -r 10 /robot_news example_interfaces/msg/String "{data: 'hello from terminal'}"**
publisher: beginning loop
publishing #1: example_interfaces.msg.String(data='hello from terminal')

publishing #2: example_interfaces.msg.String(data='hello from terminal')
[NOTE: the ‘ ’ around the message are not necessary
(data= 57) and (data= hello from terminal) work fine.]

[new terminal window:]
james@JP-VM:~$ **ros2 topic echo /robot_news**
data: hello from terminal

[If BOTH publishers are active:]
james@JP-VM:~$ **ros2 topic echo /robot_news**
data: hello from terminal
---
data: hello from terminal
---
data: hello from terminal
---
data: hello from terminal
---
data: Hi, this is C3PO from the robot news station.
...
^C

Lesson 41. Remap a Topic at Runtime:
[Rename node, as prev:]
james@JP-VM:~$ **ros2 run my_py_pkg robot_news_station --ros-args -r __node:=my_station**
[INFO] [1598830735.740653813] [my_station]: Robot News Station has been started

[new terminal window:]
james@JP-VM:~$ **ros2 node list**
/my_station
james@JP-VM:~$ **ros2 topic list**
/parameter_events
/robot_news
/rosout

[Rename topic:]
james@JP-VM:~$ ros2 run my_py_pkg robot_news_station --ros-args -r __node:=my_station **-r robot_news:=my_news**
[INFO] [1598830985.091001548] [my_station]: Robot News Station has been started

[new terminal window:]
james@JP-VM:~$ ros2 node list
/my_station
james@JP-VM:~$ ros2 topic list
/my_news
/parameter_events
/rosout
james@JP-VM:~$

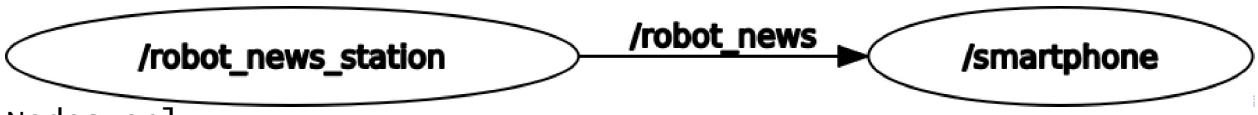
[launch normal subscriber:]
james@JP-VM:~$ **ros2 run my_py_pkg smartphone**
[INFO] [1598831317.152913687] [smartphone]: Smartphone has been started.

[no messages]
^C

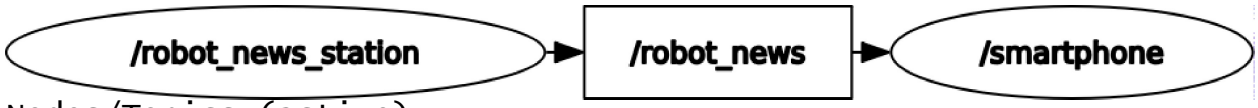
[launch remapped subscriber:]
james@JP-VM:~$ ros2 run my_py_pkg smartphone
--ros-args -r robot_news:=my_news
[INFO] [1598831458.953169650] [smartphone]: Smartphone has been started.
[INFO] [1598831459.501489370] [smartphone]: Hi, this is C3PO from the robot news station.

Lesson 42. Monitor Topics With rqt and rqt_graph
[Launch a publisher:]
james@JP-VM:~$ **ros2 run my_cpp_pkg robot_news_station**
[INFO] [1598832391.607556290] [robot_news_station]: Robot New Station has been started

[new terminal window:]
[Launch a subscriber:]
james@JP-VM:~$ **ros2 run my_py_pkg smartphone**
[INFO] [1598832440.116793978] [smartphone]: Smartphone has been started.
[INFO] [1598832440.217019972] [smartphone]: Hi, this is R2D2 from the Robot News Station
...



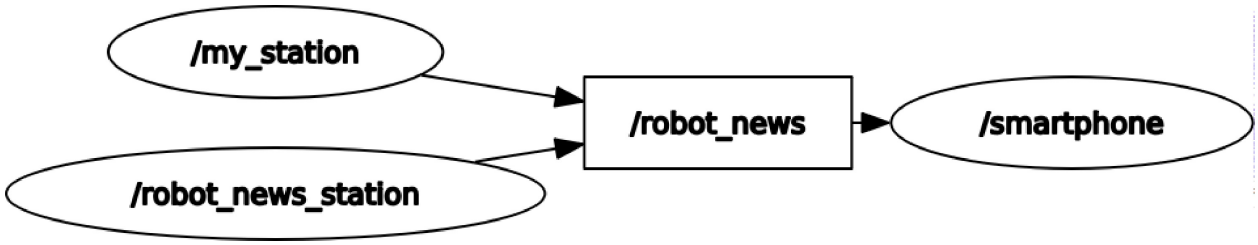
Nodes only



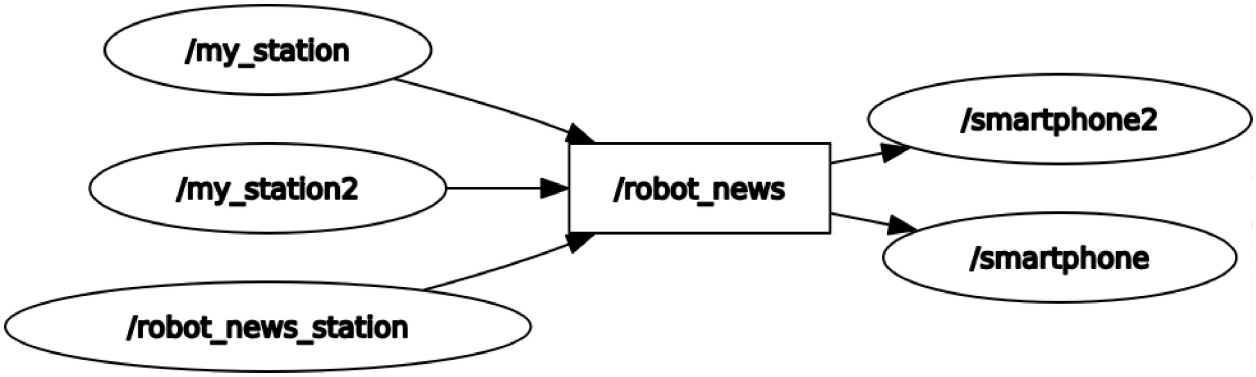
Nodes/Topics (active)

[Start a second publisher w/ remapped node:]
[new terminal window:]
james@JP-VM:~$ **ros2 run my_py_pkg robot_news_station --ros-args -r __node:=my_station**
[INFO] [1598833292.077705815] [my_station]: Robot News Station has been started

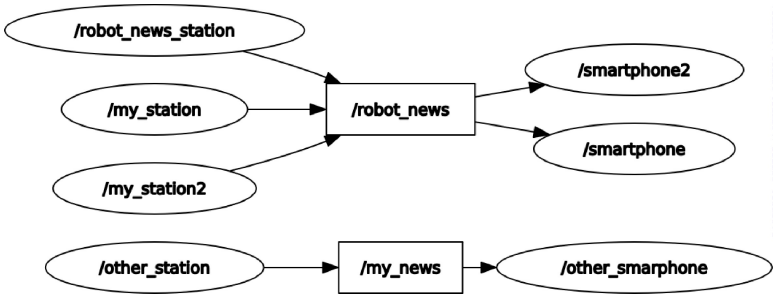
[observe subscriber terminal window - receiving BOTH publishers (same topic):]
[INFO] [1598833314.079237249] [smartphone]: Hi, this is **R2D2** from the Robot News Station
[INFO] [1598833314.080685724] [smartphone]: Hi, this is **C3PO** from the robot news station.



[Launch 3d publisher, remapped:]
james@JP-VM:~$ **ros2 run my_py_pkg robot_news_station --ros-args -r __node:=my_station2**
[INFO] [1598834918.309131931] [my_station2]: Robot News Station has been started
[Launch 2nd subscriber, remapped:]
james@JP-VM:~$ **ros2 run my_cpp_pkg smartphone --ros-args -r __node:=smartphone2**
[INFO] [1598835235.725673369] [smartphone2]: Smartphone has been started.
[INFO] [1598835237.268700553] [smartphone2]: Hi, this is **C3PO** from the robot news station.
[INFO] [1598835237.530852769] [smartphone2]: Hi, this is **C3PO** from the robot news station.
[INFO] [1598835237.621733747] [smartphone2]: Hi, this is **R2D2** from the Robot News Station



[Create remapped publisher on remapped topic:]
james@JP-VM:~$ **ros2 run my_py_pkg robot_news_station --ros-args -r __node:=other_station -r robot_news:=my_news**
[INFO] [1598835923.198886984] [other_station]: Robot News Station has been started
[Create remapped subscriber on (same) remapped topic:]
james@JP-VM:~$ **ros2 run my_py_pkg smartphone --ros-args -r __node:=other_smarphone -r robot_news:=my_news**
[INFO] [1598836116.410754487] [other_smarphone]: Smartphone has been started.
[INFO] [1598836118.641331967] [other_smarphone]: Hi, this is C3PO from the robot news station.
[INFO] [1598836118.641331967] [other_smarphone]: Hi, this is C3PO from the robot news station.



Lesson 43. Experiment on Topics With Turtlesim

[Launch turtlesim_node:]

james@JP-VM:~$ **ros2 run turtlesim turtljames@JP-VM:~$ ros2 run turtlesim turtlesim_node**

[INFO] [1598836716.771285911] [turtlesim]: Starting turtlesim with node name /turtlesim

[INFO] [1598836716.818951984] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

[Launch turtlesim_teleop_key to control turtle:]

[new terminal window:]

james@JP-VM:~$ **ros2 run turtlesim turtle_teleop_key**

Reading from keyboard

Use arrow keys to move the turtle.

Use G|B|V|C|D|E|R|T keys to rotate to absolute orientations. 'F' to cancel a rotation.

'Q' to quit.

[new terminal window:]

james@JP-VM:~$ **ros2 node info /turtlesim**

/turtlesim

  Subscribers:

    /parameter_events: rcl_interfaces/msg/ParameterEvent

    /turtle1/cmd_vel: geometry_msgs/msg/Twist

  Publishers:

    /parameter_events: rcl_interfaces/msg/ParameterEvent

    /rosout: rcl_interfaces/msg/Log

    /turtle1/color_sensor: turtlesim/msg/Color

    /turtle1/pose: turtlesim/msg/Pose

  Service Servers:

    /clear: std_srvs/srv/Empty

    /kill: turtlesim/srv/Kill

    /reset: std_srvs/srv/Empty

    /spawn: turtlesim/srv/Spawn

    /turtle1/set_pen: turtlesim/srv/SetPen

    /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute

    /turtle1/teleport_relative: turtlesim/srv/TeleportRelative

    /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters

    /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes

    /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters

    /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters

    /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters

    /turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically

  Service Clients:

  Action Servers:

    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute

  Action Clients:

james@JP-VM:~$ **ros2 node info /teleop_turtle**
/teleop_turtle
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Service Servers:
    /teleop_turtle/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /teleop_turtle/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /teleop_turtle/get_parameters: rcl_interfaces/srv/GetParameters
    /teleop_turtle/list_parameters: rcl_interfaces/srv/ListParameters
    /teleop_turtle/set_parameters: rcl_interfaces/srv/SetParameters
    /teleop_turtle/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:

  Action Clients:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute

james@JP-VM:~$ **ros2 interface show geometry_msgs/msg/Twist**
This expresses velocity in free space broken into its linear and angular parts.

Vector3 linear
Vector3 angular

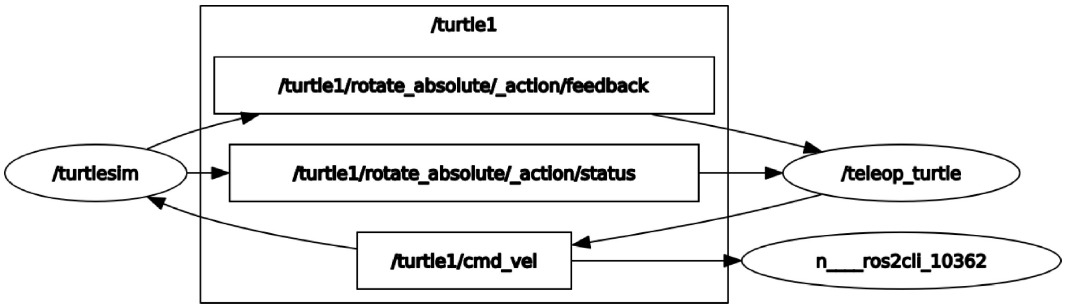
james@JP-VM:~$ **ros2 interface show geometry_msgs/msg/Vector3**
This represents a vector in free space.

This is semantically different than a point.
A vector is always anchored at the origin.
When a transform is applied to a vector, only the rotational component is applied.

float64 x
float64 y
float64 z

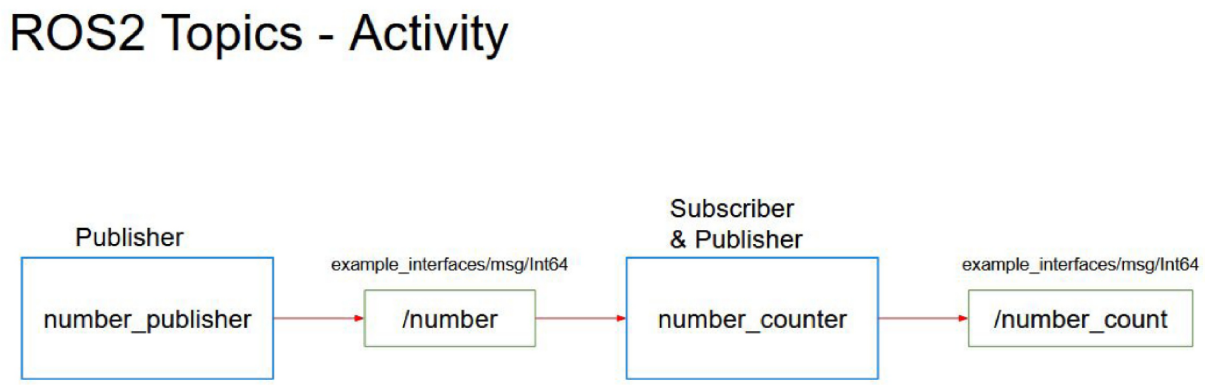
james@JP-VM:~$ **ros2 topic echo /turtle1/cmd_vel**
[nothing happens until you press a turtle_teleop_key key:]
[forward (up arrow) key:]
linear:
  x: 2.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0

[T (Northwest) key:]
nothing x-
turtlesim_node responds with:
[INFO] [1598838162.424688694] [turtlesim]: Rotation goal completed successfully
[D (West) key:]
nothing x-
[INFO] [1598838167.719935297] [turtlesim]: Rotation goal completed successfully
[rotate left/counter-clockwise (left-arrow) key:]
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: **2.0**
[rotate right/clockwise (right-arrow) key:]
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -2.0



[My graph has extra n__ros2ctl_10362 node.
Because instructor killed node from
james@JP-VM:~$ ros2 topic echo /turtle1/cmd_vel
while I did not, so echo served as add'l node]

Activity 002 - ROS2 Topics



You'l create 2 nodes from scratch. In the first one you'l have 1 publisher, and in the second
one, 1 publisher & 1 subscriber.
●The number_publisher node publishes a number (always the same) on the "number" opic, with the existing type example_interfaces/msg/Int64.
●The number_counter node subscribes to the "number"topic. It keeps a counter variable.
Every time a new number is received, it' added to the counter. The node also has a publisher on the "number_count"topic. When the counter is updated, the publisher directly publishes the new value on the topic.

A few hints:
●Check what to put into the example_interfaces/msg/Int64 with the "ros2 interface show" command line tool.
●It may be easier to do the activity in this order: first create the number_publisher node, check that the publisher is working with "os2 topic" Then create the number_counter, focus on the subscriber. And finally create the last publisher.
●In the number_counter node, the publisher will publish messages directly from the subscriber callback.

ros2 topic echo /number_count
data: 38
---
data: 40

PLAN: Start with a previous publisher & adapt.

james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **touch number_publisher.py**
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **chmod +x**
number_publisher.py
Cc from robot_news_station.py to number_publisher.py
james@JP-VM:~$ **ros2 interface show example_interfaces/msg/Int64**
int64 data

james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **python3 number_publisher.py**
Traceback (most recent call last):
  File "number_publisher.py", line 29, in <module>
    main()
  File "number_publisher.py", line 23, in main
    node = NumberPublisherNode()
  File "number_publisher.py", line 12, in __init__
    self.publisher_ = self.create_publisher(Int64, number, 10)
NameError: name 'number' is not defined

james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **python3 number_publisher.py**
[INFO] [1598999608.274137181] [number_publisher]: 57
Traceback (most recent call last):
  File "number_publisher.py", line 31, in <module>
    main()
  File "number_publisher.py", line 25, in main
    node = NumberPublisherNode()
  File "number_publisher.py", line 14, in __init__
    self.publisher_ = self.create_publisher(Int64, self.number, 10)
  File "/opt/ros/foxy/lib/python3.8/site-packages/rclpy/node.py", line 1144, in
create_publisher
    publisher_capsule = _rclpy.rclpy_create_publisher(
TypeError: bad argument type for built-in operation
 FIX: self.publisher_ = self.create_publisher(Int64, **"number"**, 10)

james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **python3 number_publisher.py**
[INFO] [1598999752.233460173] [number_publisher]: 57
[INFO] [1598999752.351038421] [number_publisher]: Number Publisher has been
started
^C

Success! See final code below:

~/ros2_ws/src/my_py_pkg/my_py_pkg/**number_publisher.py**:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node

from example_interfaces.msg import Int64

class NumberPublisherNode(Node):
    def __init__(self):
        super().__init__("number_publisher")

        self.number = 57
        self.get_logger().info(str(self.number))

        self.publisher_ = self.create_publisher(Int64, "number", 10)
        self.timer_ = self.create_timer(0.5, self.publish_number)
        self.get_logger().info("Number Publisher has been started")

    def publish_number(self):
        msg = Int64()
        msg.data = self.number
        self.publisher_.publish(msg)

def main(args=None):
    rclpy.init(args=args)
    node = NumberPublisherNode()
    rclpy.spin(node)
    rclpy.shutdown()


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

Compile, Run:
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg**
[new terminal window:]
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 run my_py_pkg number_publisher**
[INFO] [1598999861.907609027] [number_publisher]: 57
[INFO] [1598999861.912294114] [number_publisher]: Number Publisher has been started

james@JP-VM:~$ **ros2 node list**
/number_publisher
james@JP-VM:~$ **ros2 topic list**
/number
/parameter_events
/rosout
james@JP-VM:~$ **ros2 topic echo /number**
data: 57
---
data: 57

Now to create number_counter subscriber/publisher:
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **touch number_counter.py**
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **chmod +x number_counter.py**
cc from smartphone.py
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **python3 number_counter.py**
TypeError: argument 3 must be str, not int
    self.get_logger().info(**str(**msg.data**)**)

james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **python3 number_counter.py**
[INFO] [1599003365.817426624] [number_counter]: Number Counter has been started.
[INFO] [1599003366.942768281] [number_counter]: 57
[INFO] [1599003367.420016344] [number_counter]: 57

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg**
  "number_counter - my_py_pkg.number_counter:main"
  ^
SyntaxError: invalid syntax
FIX: node = NumberCounter**Node**()

  "number_counter **-** my_py_pkg.number_counter:main" [- instead of =]
SyntaxError: invalid syntax
FIX (setup.py): "number_counter **=** my_py_pkg.number_counter:main"

class NumberCounter(Node):
class NumberCounter**Node**(Node):

setup.py:
"number_publisher = my_py_pkg.number_publisher:main". [period]
"number_counter = my_py_pkg.number_counter:main"
FIX:
"number_publisher = my_py_pkg.number_publisher:main", [comma!]
"number_counter = my_py_pkg.number_counter:main"

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg**
[success]

james@JP-VM:~$ **ros2 run my_py_pkg number_publisher**
[INFO] [1598999861.907609027] [number_publisher]: 57
[INFO] [1598999861.912294114] [number_publisher]: Number Publisher has been started

james@JP-VM:~/ros2_ws$ **ros2 run my_py_pkg number_counter**
[INFO] [1599009345.421640302] [number_counter]: Number Counter has been started.
[INFO] [1599009346.917684524] [number_counter]: 1
[INFO] [1599009347.416675824] [number_counter]: 2
[INFO] [1599009347.914742577] [number_counter]: 3

^Cjames@JP-VM:~$ **ros2 topic echo /count**
data: 29
---
data: 30
---
data: 31
---
data: 32

james@JP-VM:~$ **rqt_graph**

Source code follows:

~/ros2_ws/src/my_py_pkg/my_py_pkg/**number_counter.py**

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node

from example_interfaces.msg import Int64

class NumberCounterNode(Node):
    def __init__(self):
        super().__init__("number_counter")
        self.subscriber_ = self.create_subscription(
            Int64, "number", self.callback_number, 10)

        self.count = 0

        self.publisher_ = self.create_publisher(Int64, "number_count", 10)
        self.get_logger().info("Number Counter has been started.")

    def callback_number(self, msg):
        self.count += 1
        outmsg = Int64()
        outmsg.data = self.count
        self.publisher_.publish(outmsg)
        self.get_logger().info(str(outmsg.data))

def main(args=None):
    rclpy.init(args=args)
    node = NumberCounterNode()
    rclpy.spin(node)
    rclpy.shutdown()


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

~/ros2_ws/src/my_py_pkg/**setup.py**

```
from setuptools import setup

package_name = 'my_py_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='james',
    maintainer_email='james@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            "py_node = my_py_pkg.my_first_node:main",
            "robot_news_station = my_py_pkg.robot_news_station:main",
            "smartphone = my_py_pkg.smartphone:main",
            "number_publisher = my_py_pkg.number_publisher:main",
            "number_counter = my_py_pkg.number_counter:main"
        ],
    },
)
```

Now for C++ number publisher & subscriber/counter/publisher:
james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **touch number_publisher.cpp**
james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **ls**
my_first_node.cpp robot_news_station.cpp template_cpp_node.cpp
number_publisher.cpp smartphone.cpp

THE LESSONS GIVE EXAMPLES OF STRING FUNCTIONS, BUT NOT OF NUMERIC FUNCTIONS like how to define the variable number = 57 esp in OOP.
I was able to stumble on the answer in python. C++ is much more difficult.

/home/james/ros2_ws/src/my_cpp_pkg/src/number_publisher.cpp:7:55: error: class 'NumberPublisherNode' does not have any field named 'number'
    7 | NumberPublisherNode() : Node("number_publisher"), number = 49
      | ^~~~~~

[Insight: variables ending in "_" like "counter_" are predefined "fields" within the Node class! You can't just make them up. How do you find out what predefined fields are available to use?]

Change "number" or "number_" to "counter_" and just don't increment it!

/home/james/ros2_ws/src/my_cpp_pkg/src/number_publisher.cpp: In constructor 'NumberPublisherNode::NumberPublisherNode()':
/home/james/ros2_ws/src/my_cpp_pkg/src/number_publisher.cpp:7:55: error: class 'NumberPublisherNode' does not have any field named 'counter_'
    7 | NumberPublisherNode() : Node("number_publisher"), counter_(49)
      | ^~~~~~~~

[Wait! my_first_node.cpp had one!!]

Ok, do w/o counter_ and just make msg.data = 49
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**
Success! At least it compiled.
james@JP-VM:~$ **ros2 run my_cpp_pkg number_publisher**
[INFO] [1599099653.851433697] [number_publisher]: Number Publisher has been started.
james@JP-VM:~$ **ros2 topic echo /number**
data: 49
---
data: 49
Success!!

Now to make number_counter.cpp
james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **touch number_counter.cpp**
cc from template_cpp_node.cpp
borrow from number_publisher.cpp
        #include "example_interfaces/msg/int64.hpp"
change MyCustomNode to NumberCounterNode
change "node_name" to "number_counter"
borrow from smartphone.cpp
compile FAIL. Sleep on it.

2020.09.06 Home
Time to admit ignorance of C++ and just look at the solution.

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**
Starting >>> my_cpp_pkg
[Processing: my_cpp_pkg]
Finished <<< my_cpp_pkg [54.0s]
Summary: 1 package finished [54.8s]

james@JP-VM:~/ros2_ws$ **ros2 run my_cpp_pkg number_publisher**
[INFO] [1599412735.436036439] [number_publisher]: Number Publisher has been started

james@JP-VM:~/ros2_ws$ **ros2 run my_cpp_pkg number_counter**
[INFO] [1599412798.271127025] [number_counter]: Number Counter has been started.

james@JP-VM:~/ros2_ws$ **ros2 topic list**
/number
/number_count

james@JP-VM:~/ros2_ws$ **ros2 topic echo /number**
data: 2
---
data: 2

james@JP-VM:~/ros2_ws$ **ros2 topic echo /number_count**
data: 176
---
data: 178

CmakeLists.txt:

```
...
add_executable(number_publisher src/number_publisher.cpp)
ament_target_dependencies(number_publisher rclcpp example_interfaces)

add_executable(number_counter src/number_counter.cpp)
ament_target_dependencies(number_counter rclcpp example_interfaces)

install(TARGETS
...
  number_publisher
  number_counter
  DESTINATION lib/${PROJECT_NAME}
```

number_publisher.cpp:

```
#include "rclcpp/rclcpp.hpp"
#include "example_interfaces/msg/int64.hpp"

class NumberPublisherNode : public rclcpp::Node
{
public:
    NumberPublisherNode() : Node("number_publisher"), number_(2)
    {
        number_publisher_ =
this->create_publisher<example_interfaces::msg::Int64>("number", 10);
        number_timer_ = this->create_wall_timer(std::chrono::seconds(1),
                                std::bind(&NumberPublisherNode::publishNumber, this));
        RCLCPP_INFO(this->get_logger(), "Number Publisher has been started.");
    }

private:
    void publishNumber()
    {
        auto msg = example_interfaces::msg::Int64();
        msg.data = number_;
        number_publisher_->publish(msg);
    }

    int number_;
    rclcpp::Publisher<example_interfaces::msg::Int64>::SharedPtr number_publisher_;
    rclcpp::TimerBase::SharedPtr number_timer_;
};

int main(int argc, char **argv)
{
    rclcpp::init(argc, argv);
    auto node = std::make_shared<NumberPublisherNode>();
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

number_counter.cpp:

```
#include "rclcpp/rclcpp.hpp"
#include "example_interfaces/msg/int64.hpp"

class NumberCounterNode : public rclcpp::Node
{
public:
   NumberCounterNode() : Node("number_counter"), counter_(0)
   {
      counter_publisher_ =
this->create_publisher<example_interfaces::msg::Int64>("number_count", 10);
      number_subscriber_ = this->create_subscription<example_interfaces::msg::Int64>(
         "number", 10,
         std::bind(&NumberCounterNode::callbackNumber, this, std::placeholders::_1));

      RCLCPP_INFO(this->get_logger(), "Number Counter has been started.");
   }

private:
   void callbackNumber(const example_interfaces::msg::Int64::SharedPtr msg)
   {
      counter_ += msg->data;
      auto newMsg = example_interfaces::msg::Int64();
      newMsg.data = counter_;
      counter_publisher_->publish(newMsg);
   }

   int counter_;
   rclcpp::Publisher<example_interfaces::msg::Int64>::SharedPtr counter_publisher_;
   rclcpp::Subscription<example_interfaces::msg::Int64>::SharedPtr
number_subscriber_;
};

int main(int argc, char **argv)
{
   rclcpp::init(argc, argv);
   auto node = std::make_shared<NumberCounterNode>();
   rclcpp::spin(node);
   rclcpp::shutdown();
   return 0;
}
```

Section 6: ROS2 Services - Client/Server Communication Between Nodes
Intro: Topics are used for data streams, and Services for a client/server interaction.
Lesson 49. What is a ROS2 Service?
Weather service example.
Single Server, multiple clients
> location
< forecase
Battery / LED example
Service = Set LED
Client = Battery node request set LED state ON/OFF
Server = LED panel response, state set success TRUE
synchronous vs asynchronous

Lesson 50. Write a Python Service Server
james@JP-VM:~$ **ros2 interface show example_interfaces/srv/[space][tab][tab]**
example_interfaces/srv/AddTwoInts
example_interfaces/srv/Trigger
example_interfaces/srv/SetBool

james@JP-VM:~$ **ros2 interface show example_interfaces/srv/AddTwoInts**
int64 a
int64 b
---
int64 sum

james@JP-VM:~$ **cd ros2_ws/src/my_py_pkg/my_py_pkg/**
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **ls**
build log number_publisher.py smartphone.py
__init__.py my_first_node.py __pycache__ template_python_node.py
install number_counter.py robot_news_station.py
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **touch add_two_ints_server.py**
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **chmod +x add_two_ints_server.py**
cc from template_python_node.py
Chang name: AddTwoIntsServerNode, “add_two_ints_server”
#import service type
from example_interfaces.srv import AddTwoInts
(already in dependencies in setup.py)

Construct server

self.server_ = self.create_service(

AddTwoInts, "add_two_ints", self.callback_add_two_ints)

[*ServiceType*, *“service_name”, callback_function*]

recommend starting *“service_name”* with a VERB+OBJECT to indicate a service

Create callback function

def callback_add_two_ints(self, request, response):

response.sum = request.a + request.b

return response

[*response_field_name* = function {*request_field_name*(s)}]

Create debug logger

self.get_logger().info(str(request.a) + " + " + str(request.b) + " = " str(response.sum))

Add to setup.py

"add_two_ints_server = my_py_pkg.add_two_ints_server:main"

Compile

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg** OR

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg --symlink-install**

james@JP-VM:~$ **source ~/.bashrc**

james@JP-VM:~$ **ros2 run my_py_pkg[space][tab][tab]**

add_two_ints_server

number_counter

number_publisher

...

Run

james@JP-VM:~$ **ros2 run my_py_pkg add_two_ints_server**

[INFO] [1599433029.330816578] [add_two_ints_server]: Add Two Ints Server has been started

Test server from terminal:

james@JP-VM:~$ **ros2 node list**
/add_two_ints_server
james@JP-VM:~$ **ros2 service list**
/add_two_ints
/add_two_ints_server/describe_parameters
/add_two_ints_server/get_parameter_types
/add_two_ints_server/get_parameters
/add_two_ints_server/list_parameters
/add_two_ints_server/set_parameters
/add_two_ints_server/set_parameters_atomically

james@JP-VM:~$ **ros2 node info /add_two_ints_server**
/add_two_ints_server
  Subscribers:
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
  Service Servers:
    /add_two_ints: example_interfaces/srv/AddTwoInts
    /add_two_ints_server/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /add_two_ints_server/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /add_two_ints_server/get_parameters: rcl_interfaces/srv/GetParameters
    /add_two_ints_server/list_parameters: rcl_interfaces/srv/ListParameters
    /add_two_ints_server/set_parameters: rcl_interfaces/srv/SetParameters
    /add_two_ints_server/set_parameters_atomically:
rcl_interfaces/srv/SetParametersAtomically
  Service Clients:
  Action Servers:
  Action Clients:

james@JP-VM:~$ **ros2 service call /add_two_ints**
example_interfaces/srv/AddTwoInts "{a: 7, b: 3}"
waiting for service to become available...
requester: making request: example_interfaces.srv.AddTwoInts_Request(a=7, b=3)
response:
example_interfaces.srv.AddTwoInts_Response(sum=10)

[meanwhile, in other terminal window:]
james@JP-VM:~$ **ros2 run my_py_pkg add_two_ints_server**
[INFO] [1599433029.330816578] [add_two_ints_server]: Add Two Ints Server has been started
[INFO] [1599434158.055448606] [add_two_ints_server]: 7 + 3 = 10

add_two_int_server.py":

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from example_interfaces.srv import AddTwoInts

class AddTwoIntsServerNode(Node):
    def __init__(self):
        super().__init__("add_two_ints_server")
        self.server_ = self.create_service(
            AddTwoInts, "add_two_ints", self.callback_add_two_ints)
        self.get_logger().info("Add Two Ints Server has been started")
    def callback_add_two_ints(self, request, response):
        response.sum = request.a +  request.b
        self.get_logger().info(str(request.a) + " + " + str(request.b) + " = " +
str(response.sum))
        return response

def main(args=None):
    rclpy.init(args=args)
    node = AddTwoIntsServerNode()
    rclpy.spin(node)
    rclpy.shutdown()

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

Lesson 51. Write a Python Service Client - no OOP
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **touch add_two_ints_client_no_oop.py**
[create the oop file while we're at it:]
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **touch add_two_ints_client.py**
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **chmod +x add_two_ints_client***
cc template
Rename
see code
Add to setup.py
Compile
Run
james@JP-VM:~$ **ros2 run my_py_pkg add_two_ints_client_no_oop**
[WARN] [1599436768.744584030] [add_two_ints_client_no_oop]: Waiting for Server Add Two Ints...
[WARN] [1599436769.750968710] [add_two_ints_client_no_oop]: Waiting for Server Add Two Ints...
[meanwhile run Server]
james@JP-VM:~$ **ros2 run my_py_pkg add_two_ints_server**
[INFO] [1599436826.683316936] [add_two_ints_server]: Add Two Ints Server has been started
[and (bare bones) client sees server has started so closes]

[resume Server:]
james@JP-VM:~$ ros2 run my_py_pkg add_two_ints_server
[INFO] [1599436826.683316936] [add_two_ints_server]: Add Two Ints Server has been started
[Server waits for client request, below, then responds:]
[INFO] [1599438135.456751032] [add_two_ints_server]: 3 + 8 = 11

james@JP-VM:~$ **ros2 run my_py_pkg add_two_ints_client_no_oop**
[WARN] [1599438134.699850633] [add_two_ints_client_no_oop]: Waiting for Server Add Two Ints...
[INFO] [1599438135.458941402] [add_two_ints_client_no_oop]: 3 + 8 = 11

add_two_ints_client_no_oop.py:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node

from example_interfaces.srv import AddTwoInts


def main(args=None):
    rclpy.init(args=args)
    node = Node("add_two_ints_client_no_oop")

    client = node.create_client(AddTwoInts, "add_two_ints")
    while not client.wait_for_service(1.0):
        node.get_logger().warn("Waiting for Server Add Two Ints...")

    request = AddTwoInts.Request()
    request.a = 3
    request.b = 8

    future = client.call_async(request)
    rclpy.spin_until_future_complete(node, future)

    try:
        response = future.result()
        node.get_logger().info(
            str(request.a) + " + " + str(request.b) + " = " +
            str(response.sum))

    except Exception as e:
        node.get_logger().error("Service call failed %r" % (e, ))

    rclpy.shutdown()

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

Lesson 52. Write a Python Service Client - OOP

template for future python oop clients:

```
class AddTwoIntsClientNode(Node):
   def __init__(self):
      super().__init__("add_two_ints_client")
      self.call_add_two_ints_server(6, 7)
   def call_add_two_ints_server(self, a, b):
      client = self.create_client(AddTwoInts, "add_two_ints")
      while not client.wait_for_service(1.0):
         self.get_logger().warn("Waiting for Server Add Two Ints...")

      request = AddTwoInts.Request()
      request.a = a
      request.b = b

      future = client.call_async(request)
      future.add_done_callback(partial(self.callback_call_add_two_ints, a=a, b=b))

   def callback_call_add_two_ints(self, future, a, b):
      try:
         response = future.result()
         self.get_logger().info(str(a) + " + " + str(b) + " = " +
                        str(response.sum))
      except Exception as e:
         self.get_logger().error("Service call failed %r" % (e, ))
```

Compile
Start server:
james@JP-VM:~$ **ros2 run my_py_pkg add_two_ints_server**
[INFO] [1599436826.683316936] [add_two_ints_server]: Add Two Ints Server has been started
Start client:
james@JP-VM:~$ **ros2 run my_py_pkg add_two_ints_client**
[INFO] [1599443440.513248210] [add_two_ints_client]: 6 + 7 = 13
Server also responds:
[INFO] [1599443440.429686332] [add_two_ints_server]: 6 + 7 = 13

May call service multiple times:

```
class AddTwoIntsClientNode(Node):
   def __init__(self):
      super().__init__("add_two_ints_client")
      self.call_add_two_ints_server(6, 7)
      self.call_add_two_ints_server(3, 1)
      self.call_add_two_ints_server(5, 2)
```

Lesson 53. Write a C++ Service Server
as before:
james@JP-VM:~$ **ros2 interface show example_interfaces/srv/AddTwoInts**
int64 a
int64 b
---
int64 sum

james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **touch AddTwoIntsServer.cpp**
[or can make new file from Visual Studio Code]
Rename AddTwoIntsServerNode, add_two_ints_server
Add
    #include "example_interfaces/srv/add_two_ints.hpp"
[Dependency already in CmakeLists.txt & package.xml]

james@JP-VM:~$ **ros2 run my_cpp_pkg add_two_ints_server**
[INFO] [1599497448.438357931] [add_two_ints_server]: Service server has been started.
[INFO] [1599497470.556811282] [add_two_ints_server]: 7 + 3 = 10
[INFO] [1599497497.060445161] [add_two_ints_server]: 42 + 73 = 115
[INFO] [1599497584.171814469] [add_two_ints_server]: 6 + 7 = 13
[INFO] [1599497584.179805523] [add_two_ints_server]: 3 + 1 = 4
[INFO] [1599497584.188013074] [add_two_ints_server]: 5 + 2 = 7

james@JP-VM:~$ **ros2 service call /add_two_ints example_interfaces/srv/AddTwoInts "{a: 7, b: 3}"**
requester: making request: example_interfaces.srv.AddTwoInts_Request(a=7, b=3)

response:
example_interfaces.srv.AddTwoInts_Response(sum=10)
james@JP-VM:~$ **ros2 service call /add_two_ints example_interfaces/srv/AddTwoInts "{a: 42, b: 73}"**
waiting for service to become available...
requester: making request: example_interfaces.srv.AddTwoInts_Request(a=42, b=73)
response:
example_interfaces.srv.AddTwoInts_Response(sum=115)

james@JP-VM:~$ **ros2 run my_py_pkg add_two_ints_client**
[WARN] [1599497583.167796268] [add_two_ints_client]: Waiting for Server Add Two Ints...
[INFO] [1599497584.191379158] [add_two_ints_client]: 5 + 2 = 7
[INFO] [1599497584.193397945] [add_two_ints_client]: 3 + 1 = 4
[INFO] [1599497584.195098325] [add_two_ints_client]: 6 + 7 = 13
[curious that client responses are in opposite order from server's]

james@JP-VM:~$ **ros2 service list** [must run from home directory?]
/add_two_ints
/add_two_ints_client/describe_parameters
/add_two_ints_client/get_parameter_types
/add_two_ints_client/get_parameters
/add_two_ints_client/list_parameters
/add_two_ints_client/set_parameters
/add_two_ints_client/set_parameters_atomically
/add_two_ints_server/describe_parameters
/add_two_ints_server/get_parameter_types
/add_two_ints_server/get_parameters
/add_two_ints_server/list_parameters
/add_two_ints_server/set_parameters
/add_two_ints_server/set_parameters_atomically

add_two_ints_server.cpp:

```
#include "rclcpp/rclcpp.hpp"
#include "example_interfaces/srv/add_two_ints.hpp"

using std::placeholders::_1;
using std::placeholders::_2;

class AddTwoIntsServerNode : public rclcpp::Node
{
public:
  AddTwoIntsServerNode() : Node("add_two_ints_server")
  {
    server_ = this->create_service<example_interfaces::srv::AddTwoInts>(
      "add_two_ints",
      std::bind(&AddTwoIntsServerNode::callbackAddTwoInts, this, _1, _2));
    RCLCPP_INFO(this->get_logger(), "Service server has been started.");
  }

private:
  void callbackAddTwoInts(const
example_interfaces::srv::AddTwoInts::Request::SharedPtr request,
                const example_interfaces::srv::AddTwoInts::Response::SharedPtr
response)
  {
    response->sum = request->a + request->b;
    RCLCPP_INFO(this->get_logger(), "%d + %d = %d", request->a, request->b,
response->sum);
  }

  rclcpp::Service<example_interfaces::srv::AddTwoInts>::SharedPtr server_;
};

int main(int argc, char **argv)
{
  rclcpp::init(argc, argv);
  auto node = std::make_shared<AddTwoIntsServerNode>();
  rclcpp::spin(node);
  rclcpp::shutdown();
  return 0;
}
```

Lesson 54. Write a C++ Service Client - no OOP

james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **touch add_two_ints_client_no_oop.cpp**

[and, while we're at it, the with oop file:]

james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **touch add_two_ints_client.cpp**

cc template_cpp_node.cpp

keep only the main function

rename

change node creation:

```
auto node = std::make_shared<rclcpp::Node>("add_two_ints_client_no_oop");
```

remove spin

this is a one-shot client

include interface

```
#include "example_interfaces/srv/add_two_ints.hpp"
```

create client

```
auto client = node->create_client<
    example_interfaces::srv::AddTwoInts>(add_two_ints);
    interface_name                    (service_name)
```

wait for service

```
while (!client->wait_for_service(std::chrono::seconds(1)))
{
   RCLCPP_WARN(node->get_logger(). "Waiting for the server to be up...");
}
```

create request

```
auto request = std::make_shared<example_interfaces::srv::AddTwoInts::Request>();
request->a = 3;
request->b = 8;
```

call client

```
auto future = client->async_send_request(request);
 if (rclcpp::spin_until_future_complete(node, future) ==
rclcpp::executor::FutureReturnCode::SUCCESS)
 {
    RCLCPP_INFO(node_logger(), "%d + % d = %d", request->a, request->b,
future.get()->sum);
 }
 else
 {
    RCLCPP_ERROR(node->get_logger(), "Error while calling service");
 }
```

add executable to CmakeLists.txt:

```
add_executable(add_two_ints_client_no_oop src/add_two_ints_client_no_oop.cpp)
ament_target_dependencies(add_two_ints_client_no_oop rclcpp example_interfaces)

add_two_ints_client_no_oop
```

Compile:
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**
[build from ros2_ws or may appear to build but no executable!]
Start client:
james@JP-VM:~$ **ros2 run my_cpp_pkg add_two_ints_client_no_oop**
[WARN] [1599530067.749214753] [add_two_ints_client_no_oop]: Waiting for the server to be up...
Start server:
james@JP-VM:~$ **ros2 run my_cpp_pkg add_two_ints_server**
[INFO] [1599530077.558152936] [add_two_ints_server]: Service server has been started.
[INFO] [1599530077.692567812] [add_two_ints_server]: 3 + 8 = 11
Client response:
[INFO] [1599530077.693169004] [add_two_ints_client_no_oop]: 3 + 8 = 11

Lesson 55. Write a C++ Service Client - OOP

Can't spin a node while it's already spinning. This won't work:

```
if (rclcpp::spin_until_future_complete(node, future) ==
    rclcpp::executor::FutureReturnCode::SUCCESS)
```

cc template:

```
template_cpp_node.cpp
```

rename:

```
AddTwoIntsNode, add_two_ints
```

#include:

```
#include "example_interfaces/srv/add_two_ints.hpp"
```

add function to call service:

create client (same as no OOP):

```
auto client =
node->create_client<example_interfaces::srv::AddTwoInts>("add_two_ints");
```

change "node" to "this->"

```
auto client =
this->create_client<example_interfaces::srv::AddTwoInts>("add_two_ints");
```

wait for service:

```
while (!client->wait_for_service(std::chrono::seconds(1)))
{
  RCLCPP_WARN(node->get_logger(), "Waiting for the server to be up...");
}
```

change "node" to "this->":

```
RCLCPP_WARN(this->get_logger(), "Waiting for the server to be up...");
```

create request (same as no OOP):

```
auto request = std::make_shared<example_interfaces::srv::AddTwoInts::Request>();
request->a = 3;
request->b = 8;
```

change hard values to variables from function definition:

```
request->a = a;
request->b = b;
```

call request as future object:

```
auto future = client->async_send_request(request);
```

Need different wait function as wait will block execution.

Need to start a different thread:

```
AddTwoIntsClientNode() : Node("add_two_ints_client")
{
  thread1_ = std::thread(std::bind(&AddTwoIntsClientNode::callAddTwoIntsService,
this, 1, 4));
}
```

add executable to CmakeList.txt:

```
add_executable(add_two_ints_client src/add_two_ints_client.cpp)
ament_target_dependencies(add_two_ints_client rclcpp example_interfaces)
```

Compile:

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**

james@JP-VM:~$ **ros2 run my_cpp_pkg add_two_ints_server**
[INFO] [1599534628.961577029] [add_two_ints_server]: Service server has been started.
[INFO] [1599534641.617421210] [add_two_ints_server]: 1 + 4 = 5

james@JP-VM:~$ **ros2 run my_cpp_pkg add_two_ints_client**
[INFO] [1599534641.617954000] [add_two_ints_client]: 1 + 4 = 5

to call multiple threads:
change
```
   std::thread thread1_;
```
to
```
   std::vector<std::thread> threads_;
```
and
```
        thread1_ = std::thread
                (std::bind(&AddTwoIntsClientNode::callAddTwoIntsService, this, 1, 4));
```
to
```
        threads_.push_back(std::thread
        (std::bind(&AddTwoIntsClientNode::callAddTwoIntsService, this, 1, 4)));
```
which may be repeated w/ different values:
```
        threads_.push_back(std::thread
        (std::bind(&AddTwoIntsClientNode::callAddTwoIntsService, this, 5, 7)));
```

james@JP-VM:~$ ros2 run my_cpp_pkg add_two_ints_server
[INFO] [1599535479.906283459] [add_two_ints_server]: Service server has been started.
[INFO] [1599535486.651960217] [add_two_ints_server]: 5 + 6 = 11
[INFO] [1599535486.652810090] [add_two_ints_server]: 3 + 8 = 11
[INFO] [1599535486.654204635] [add_two_ints_server]: 1 + 4 = 5

james@JP-VM:~$ **ros2 run my_cpp_pkg add_two_ints_client**
[INFO] [1599535486.653344462] [add_two_ints_client]: 3 + 8 = 11
[INFO] [1599535486.653796832] [add_two_ints_client]: 5 + 6 = 11
[INFO] [1599535486.656878146] [add_two_ints_client]: 1 + 4 = 5

Lesson 56. Debug Services with ROS2 Tools

james@JP-VM:~$ **ros2 service[tab][tab]**
call
find
list
type
--include-hidden-services
james@JP-VM:~$ **ros2 service list**
[blank - nothing is running]

james@JP-VM:~$ **ros2 run my_cpp_pkg add_two_ints_server**
[INFO] [1599700321.326887048] [add_two_ints_server]: Service server has been started.
james@JP-VM:~$ **ros2 service list**
/add_two_ints
/add_two_ints_server/describe_parameters
/add_two_ints_server/get_parameter_types
/add_two_ints_server/get_parameters
/add_two_ints_server/list_parameters
/add_two_ints_server/set_parameters
/add_two_ints_server/set_parameters_atomically

james@JP-VM:~$ **ros2 node list**
/add_two_ints_server

james@JP-VM:~$ **ros2 node info /add_two_ints_server**
/add_two_ints_server
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
  Service Servers:
    /add_two_ints: example_interfaces/srv/AddTwoInts
    /add_two_ints_server/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /add_two_ints_server/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /add_two_ints_server/get_parameters: rcl_interfaces/srv/GetParameters
    /add_two_ints_server/list_parameters: rcl_interfaces/srv/ListParameters
    /add_two_ints_server/set_parameters: rcl_interfaces/srv/SetParameters
    /add_two_ints_server/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:
  Action Servers:
  Action Clients:

james@JP-VM:~$ **ros2 service type /add_two_ints**
example_interfaces/srv/AddTwoInts

james@JP-VM:~$ **ros2 interface show example_interfaces/srv/AddTwoInts**
int64 a
int64 b
---
int64 sum

james@JP-VM:~$ **rqt**
in rqt:
Plugins / Services / Service Caller

Service /add_two_ints

/add_two_ints	example_interfaces/srv/AddTwoInts	Expression
a	int64	0
b	int64	0

Change values of Expression:

/add_two_ints	example_interfaces/srv/AddTwoInts	Expression
a	int64	5
b	int64	6

then click Call button upper right:

Field	Type	Value
/	example_interfaces/srv/AddTwoInts.Response	
sum	int64	11

(also reflected in terminal running Service)
Node graph only shows server node

Lesson 57. Remap a Service at Runtime

james@JP-VM:~$ **ros2 run my_py_pkg add_two_ints_server**
[INFO] [1599768210.802374042] [add_two_ints_server]: Add Two Ints Server has been started

james@JP-VM:~$ **ros2 service list**
/add_two_ints
...

Remap the service name:
james@JP-VM:~$ **ros2 run my_py_pkg add_two_ints_server --ros-args -r add_two_ints:=new_name**
[INFO] [1599768433.156924584] [add_two_ints_server]: Add Two Ints Server has been started

james@JP-VM:~$ **ros2 service list**
...
/new_name

james@JP-VM:~/ros2_ws/src$ **ros2 run my_py_pkg add_two_ints_client**
[WARN] [1599769344.318255085] [add_two_ints_client]: Waiting for Server Add Two Ints...
[waits forever for service to start as service has NEW name!
Need to rename service for client, too.]

james@JP-VM:~/ros2_ws/src$ **ros2 run my_py_pkg add_two_ints_client --ros-args -r add_two_ints:=new_name**
[WARN] [1599769576.471504041] [add_two_ints_client]: Waiting for Server Add Two Ints...
[INFO] [1599769576.993577410] [add_two_ints_client]: 6 + 7 = 13
[INFO] [1599769577.074766057] [add_two_ints_client]: 5 + 2 = 7
[INFO] [1599769577.081597872] [add_two_ints_client]: 3 + 1 = 4

Lesson 58. Experiment on Services with Turtlesim
james@JP-VM:~$ **ros2 run turtlesim turtlesim_node**
james@JP-VM:~$ **ros2 run turtlesim turtle_teleop_key**

james@JP-VM:~$ **ros2 service list**
/clear
/kill
/reset
/spawn
/teleop_turtle/describe_parameters
/teleop_turtle/get_parameter_types
/teleop_turtle/get_parameters
/teleop_turtle/list_parameters
/teleop_turtle/set_parameters
/teleop_turtle/set_parameters_atomically
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/describe_parameters
/turtlesim/get_parameter_types
/turtlesim/get_parameters
/turtlesim/list_parameters
/turtlesim/set_parameters
/turtlesim/set_parameters_atomically
[*italics* represent standard parameters for each node, ignore.
bold represent relevant services.]

clear:
james@JP-VM:~$ **ros2 service type /clear**
std_srvs/srv/Empty
[std_srvs/srv is old, deprecated standard services.
It and std_srvs/msg Replaced by example_interfaces.]

james@JP-VM:~$ **ros2 interface show std_srvs/srv/Empty**

[as expected, interface is empty]

james@JP-VM:~$ **ros2 service call /clear std_srvs/srv/Empty**
waiting for service to become available...
requester: making request: std_srvs.srv.Empty_Request()
response:
std_srvs.srv.Empty_Response()
[Makes turtle trace disappear. Note request & response.]
turtlesim_node reports:
[INFO] [1599788646.960811084] [turtlesim]: Clearing turtlesim.
[instead of computation, this service is an action (clear trace)]

spawn:
james@JP-VM:~$ **ros2 service type /spawn**
turtlesim/srv/Spawn

james@JP-VM:~$ **ros2 interface show turtlesim/srv/Spawn**
float32 x
float32 y
float32 theta
string name # Optional. A unique name will be created and returned if this is empty
---
string name

james@JP-VM:~$ **ros2 service call /spawn turtlesim/srv/Spawn "{x: 5.0, y: 5.0, theta: 0.0, name: 'my_turtle'}"**
requester: making request: turtlesim.srv.Spawn_Request(x=5.0, y=5.0, theta=0.0, name='my_turtle')
response:
turtlesim.srv.Spawn_Response(name='my_turtle')
[new turtle placed in middle of window facing East]

james@JP-VM:~$ **ros2 service call /spawn turtlesim/srv/Spawn "{x: 10.0, y: 10.0, theta: 0.0, name: 'turtle3'}"**
[new turtle placed in upper right facing East but not all the way in the NE corner.]

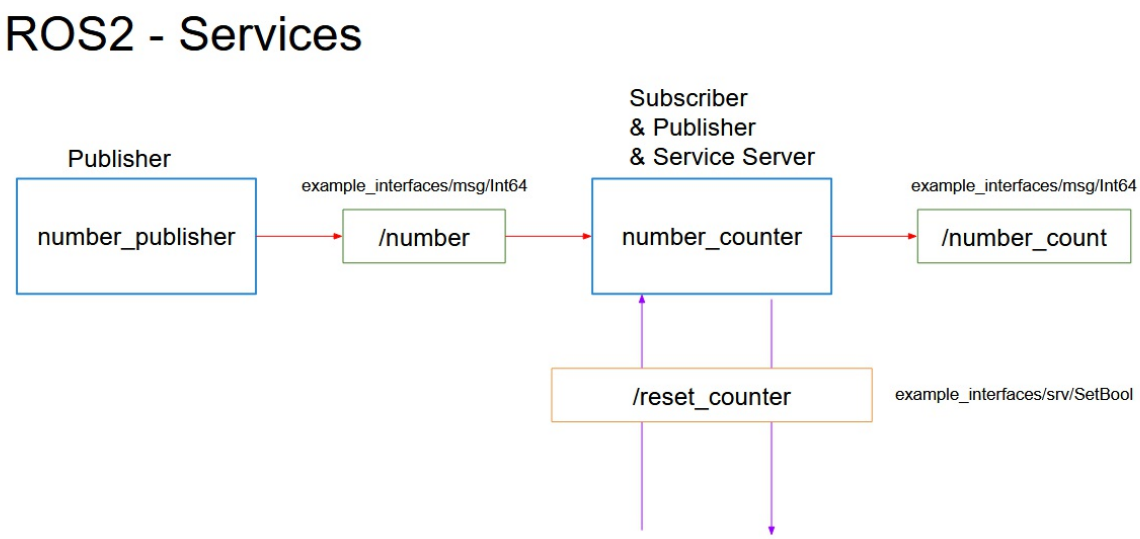
kill:
james@JP-VM:~$ **ros2 service type /kill**
turtlesim/srv/Kill
james@JP-VM:~$ **ros2 interface show turtlesim/srv/Kill**
string name
---
james@JP-VM:~$ **ros2 service call /kill turtlesim/srv/Kill "{name: 'turtle4'}"**
waiting for service to become available...
requester: making request: turtlesim.srv.Kill_Request(name='turtle4')
response:
turtlesim.srv.Kill_Response()
[removes turtle4 from screen]

reset:

james@JP-VM:~$ **ros2 service type /reset**
std_srvs/srv/Empty

james@JP-VM:~$ **ros2 service call /reset std_srvs/srv/Empty**
waiting for service to become available...
requester: making request: std_srvs.srv.Empty_Request()
response:
std_srvs.srv.Empty_Response()
[resets window to turtle1 in center facing East]

Lesson 59. Activity 003 - ROS2 Services



Add a functionality to reset the counter to zero:
Create a service server inside the “number_counter” node.
Service name: “/reset_counter”
Service type: example_interfaces/srv/SetBool. Use “ros2 interface show” to discover what’s inside!
When the server is called, you check the boolean data from the request. If true, you set the counter variable to 0.
We will then call the service directly from the command line. You can also decide - for more practice - to create your own custom node to call this “/reset_counter” service.
And of course, as always, you can do the activity for both the Python and Cpp versions.

Service type: example_interfaces/srv/SetBool. Use “ros2 interface show” to discover what’s inside!

james@JP-VM:~$ **ros2 interface show example_interfaces/srv/SetBool**
This is an example of a service to set a boolean value.
This can be used for testing but a semantically meaningful
one should be created to be built upon.

bool data # e.g. for hardware enabling / disabling
---
bool success   # indicate successful run of triggered service
string message # informational, e.g. for error messages

Create file number_server.py:
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **touch number_server.py**
Made executable:
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **chmod +x number_server.py**
cc code from **number_counter.py**
Change name to **number_server**
Import interface type:
    from example_interfaces.srv import SetBool
cc service setup code from **add_two_ints_server.py**
Change service name to **reset_counter**
Change callback to **callback_reset_counter**
Change callback function
    self.count = 0
Change response to
    response.success = True
Report action:
    self.get_logger().info("Counter has been reset.")
Update setup.py:
    "number_server = my_py_pkg.number_server:main"

ros2_ws/src/my_py_pkg/my_py_pkg/**number_server.py**:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from example_interfaces.msg import Int64
from example_interfaces.srv import SetBool

class NumberServerNode(Node):
    def __init__(self):
        super().__init__("number_server")
        self.subscriber_ = self.create_subscription(
            Int64, "number", self.callback_number, 10)

        self.count = 0

        self.publisher_ = self.create_publisher(Int64, "number_count", 10)

        self.server_ = self.create_service(
            SetBool, "reset_counter", self.callback_reset_counter)

        self.get_logger().info("Number Server has been started.")

    def callback_number(self, msg):
        self.count += 1
        outmsg = Int64()
        outmsg.data = self.count
        self.publisher_.publish(outmsg)
        self.get_logger().info(str(outmsg.data))

    def callback_reset_counter(self, request, response):
        self.count = 0
        response.success = True
        self.get_logger().info("Counter has been reset.")
        return response

def main(args=None):
    rclpy.init(args=args)
    node = NumberServerNode()
    rclpy.spin(node)
    rclpy.shutdown()

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

Compile:
save file
[ignore --symlink-install for now]
james@JP-VM:~/ros2_ws/src$ **colcon build --packages-select my_py_pkg**
Starting >>> my_py_pkg
Finished <<< my_py_pkg [5.30s]
Summary: 1 package finished [6.50s]

james@JP-VM:~$ **ros2 run my_py_pkg number_publisher**
[INFO] [1599939378.867710224] [number_publisher]: 57

james@JP-VM:~$ **ros2 run my_py_pkg number_server**
No executable found

james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **python3 number_server.py**
[INFO] [1599939416.967796101] [number_server]: Number Server has been started.
[INFO] [1599939416.972447894] [number_server]: 1
[INFO] [1599939417.446267515] [number_server]: 2
[INFO] [1599939417.952986412] [number_server]: 3

[So clearly the program itself works... Why can't it find the executable?]
[Try w/ symlink-install:]
james@JP-VM:~/ros2_ws/src$ **colcon build --packages-select my_py_pkg**
--symlink-install
james@JP-VM:~$ **ros2 run my_py_pkg number_server**
No executable found

james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **ls -l**
-rwxrwxr-**x** 1 james james 1195 Sep 12 14:35 number_server.py
[So it IS executable...]
Kill all running progs.
[close / reopen Terminator]

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg --symlink-install**

james@JP-VM:~$ **ros2 run my_py_pkg number_server**
[INFO] [1599942016.797013091] [number_server]: Number Server has been started.
[INFO] [1599942017.113552266] [number_server]: 1
[INFO] [1599942017.606240528] [number_server]: 2

[For some reason it wants to build from ~/ros2_ws instead of ~/ros_ws/src as it has done before???]
[No this is an error. Has always built from ~/ros2_ws. If built from ~/ros_ws/src it appears to work but fails.]

james@JP-VM:~$ **ros2 run my_py_pkg number_publisher**
[INFO] [1599945927.145720383] [number_publisher]: 2
[INFO] [1599945927.151023360] [number_publisher]: Number Publisher has been started

james@JP-VM:~$ **ros2 run my_py_pkg number_server**
[INFO] [1599946124.494643574] [number_server]: Number Server has been started.
[INFO] [1599946125.659023557] [number_server]: 2
[INFO] [1599946126.159643496] [number_server]: 4

james@JP-VM:~$ **ros2 service call /reset_counter example_interfaces/srv/SetBool "{True}"**
Failed to populate field: getattr(): attribute name must be string
james@JP-VM:~$ **ros2 service call /reset_counter example_interfaces/srv/SetBool "{'True'}"**
Failed to populate field: 'SetBool_Request' object has no attribute 'True'
james@JP-VM:~$ **ros2 service call /reset_counter example_interfaces/srv/SetBool "{data:'True'}"**
Failed to populate field: 'SetBool_Request' object has no attribute 'data:'True"
james@JP-VM:~$ **ros2 service call /reset_counter example_interfaces/srv/SetBool "{data: 1}"**
waiting for service to become available...
requester: making request: example_interfaces.srv.SetBool_Request(data=True)
response:
example_interfaces.srv.SetBool_Response(success=True, message='Counter has been reset')
[Finally! Why is it saying it wants a string when the correct value is 1?]

james@JP-VM:~$ **ros2 service call /reset_counter example_interfaces/srv/SetBool "{data: True}"** [...works also]
[...so 'data' is the attribute and 'True' or '1' is the value]

[INFO] [1599949270.715465051] [number_server]: 718
[INFO] [1599949271.216601157] [number_server]: 720 **[/reset_counter sent]**
[INFO] [1599949271.716067496] [number_server]: 2
[INFO] [1599949272.215822882] [number_server]: 4
[INFO] [1599949272.716251919] [number_server]: 6
[Had to cheat and look at the .py code and insert the "if / else" code although now I think it would have run w/o it as the problem was in the command line syntax.]
Final version of **number_server.py**: (some blank lines removed to save space)

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from example_interfaces.msg import Int64
from example_interfaces.srv import SetBool
class NumberServerNode(Node):
    def __init__(self):
        super().__init__("number_server")
        self.count = 0
        self.subscriber_ = self.create_subscription(
            Int64, "number", self.callback_number, 10)
        self.publisher_ = self.create_publisher(Int64, "number_count", 10)
        self.server_ = self.create_service(
            SetBool, "reset_counter", self.callback_reset_counter)
        self.get_logger().info("Number Server has been started.")
    def callback_number(self, msg):
        self.count += msg.data
        outmsg = Int64()
        outmsg.data = self.count
        self.publisher_.publish(outmsg)
        self.get_logger().info(str(outmsg.data))
    def callback_reset_counter(self, request, response):
        if request.data:
            self.count = 0
            response.success = True
            response.message = "Counter has been reset"
        else:
            response.success = False
            response.message = "Counter has NOT been reset"
        return response
def main(args=None):
    rclpy.init(args=args)
    node = NumberServerNode()
    rclpy.spin(node)
    rclpy.shutdown()
```

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

Now to try in C++:
james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **touch number_server.cpp**
cc from number_counter.cpp
add dependencies:
        #include "example_interfaces/srv/set_bool.hpp"
change name > NumberServerNode
cc server structure from add_two_ints_server.cpp
create callbackResetCounter

CmakeLists.txt:
...
add_executable(number_server src/number_server.cpp)
ament_target_dependencies(number_server rclcpp example_interfaces)

install(TARGETS
...
number_server
  DESTINATION lib/${PROJECT_NAME}
)

save files:
Compile:
Initial compile crashed system. Rebooted. Code matched solution. Recompiled ok.
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**
Starting >>> my_cpp_pkg
[Processing: my_cpp_pkg]
[Processing: my_cpp_pkg]
Finished <<< my_cpp_pkg [1min 4s]
Summary: 1 package finished [1min 6s]

james@JP-VM:~/ros2_ws$ **ros2 run my_cpp_pkg number_publisher**
[INFO] [1600015998.376513088] [number_publisher]: Number Publisher has been started.

james@JP-VM:~$ **ros2 run my_cpp_pkg number_server**
[INFO] [1600016220.450988014] [number_server]: Number Server has been started.

james@JP-VM:~/ros2_ws$ **ros2 service call /reset_counter example_interfaces/srv/SetBool "{data: 1}"**
waiting for service to become available...
requester: making request: example_interfaces.srv.SetBool_Request(data=True)
response:
example_interfaces.srv.SetBool_Response(success=True, message='Counter reset success')

james@JP-VM:~$ **ros2 topic echo /number_count**
data: 340
---
data: 342
---
data: 2
---
data: 4

james@JP-VM:~$ **rqt**

[How do you get the graph at the beginning of this section that shows the /reset_counter service in rqt, or is it custom?]

Lesson 63. What is/are (a) ROS2 Interface(s)?

Topic:
- Name (ex: /number_count)
- Msg definition (ex: example_interfaces/msg/Int64)

Service:
- Name (ex: /reset_number_count)
- Srv definition (ex: example_interfaces/srv/SetBool)
  - Request : msg
  - ---
  - Response : msg

Build system (eg Colcon) takes Msg definition from interface package and generates source code in appropriate language to create interface.

https://index.ros.org/doc/ros2/Concepts/About-ROS-Interfaces/
- see 2.1.1 Field Types for supported field types

https://github.com/ros2/common_interfaces
For multiple packages with predefined interfaces
For example sensor_msg:
https://github.com/ros2/common_interfaces/tree/master/sensor_msgs
Which includes both msg and svc
For example JointState.msg:
https://github.com/ros2/common_interfaces/blob/master/sensor_msgs/msg/JointState.msg
std_msgs/Header header [*another_pkg*]/[Caps indicate NOT a primitive data type but yet another message in the prev pkg]:
https://github.com/ros2/common_interfaces/blob/master/std_msgs/msg/Header.msg
[clearly msg types can be nexted QUITE deeply!]

string[] name
float64[] position
float64[] velocity
float64[] effort

See Twist.msg from Turtlesim:
https://github.com/ros2/common_interfaces/blob/master/geometry_msgs/msg/Twist.msg
Vector3 linear
Vector3 angular

Note caps in Vectyor3 indicating non-primitive data type w/in the same pkg:
https://github.com/ros2/common_interfaces/blob/master/geometry_msgs/msg/Vector3.msg
float64 x
float64 y
float64 z

Look at services:
sensor_msg/srv/:
https://github.com/ros2/common_interfaces/tree/master/sensor_msgs/srv
SetCameraInfo.srv
This service requests that a camera stores the given CameraInfo as that
camera's calibration information.
#
The width and height in the camera_info field should match what the
camera is currently outputting on its camera_info topic, and the camera
will assume that the region of the imager that is being referred to is
the region that the camera is currently capturing.

sensor_msgs/CameraInfo camera_info # The camera_info to store
---
bool success # True if the call succeeded
string status_message # Used to give details about success

You can include other msg in a svc, but you can't include other svc in a svc.
Look at CameraInfo msg:
https://github.com/ros2/common_interfaces/blob/master/sensor_msgs/msg/CameraInfo.msg
which is a very complex message consisting of primitive data types & other msg types, ie Header.

ROS2 - Msg and Srv

- **Use msg primitive types to create a message definition**
- **You can create a message definition using other message definitions**

Lesson 64. Create and Build Your First Custom Msg

Where to put the pkg? You *can* put them anywhere, BUT it's better to create a folder just for custom interfaces where you only have to create *one* dependency. There is no code so no build type nor dependencies are needed:

```
james@JP-VM:~/ros2_ws/src$ ros2 pkg create my_robot_interfaces
going to create a new package
package name: my_robot_interfaces
destination directory: /home/james/ros2_ws/src
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['james <james@todo.todo>']
licenses: ['TODO: License declaration']
build type: ament_cmake
dependencies: []
creating folder ./my_robot_interfaces
creating ./my_robot_interfaces/package.xml
creating source and include folder
creating folder ./my_robot_interfaces/src
creating folder ./my_robot_interfaces/include/my_robot_interfaces
creating ./my_robot_interfaces/CMakeLists.txt

james@JP-VM:~/ros2_ws/src$ ls
build  install  log  my_cpp_pkg  my_py_pkg  my_robot_interfaces
james@JP-VM:~/ros2_ws/src$ cd my_robot_interfaces/
james@JP-VM:~/ros2_ws/src/my_robot_interfaces$ ls
CMakeLists.txt  include  package.xml  src
```

builder assume could be C++ pkg so creates include & src folders. Delete - no needed:

```
james@JP-VM:~/ros2_ws/src/my_robot_interfaces$ rm -rf include/
james@JP-VM:~/ros2_ws/src/my_robot_interfaces$ rm -rf src/
james@JP-VM:~/ros2_ws/src/my_robot_interfaces$ ls
CMakeLists.txt  package.xml
```

Create msg folder:

```
james@JP-VM:~/ros2_ws/src/my_robot_interfaces$ mkdir msg
james@JP-VM:~/ros2_ws/src/my_robot_interfaces$ ls
CMakeLists.txt  msg  package.xml
```

Configure msg package:
Edit package.xml (ignore italics unless publishing pkg):

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>my_robot_interfaces</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="james@todo.todo">james</maintainer>
  <license>TODO: License declaration</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

  <build_depend>rosidl_default_generators</build_depend>
  <exec_depend>rosidl_default_runtime</exec_depend>
  <member_of_group>rosidl_interface_packages</member_of_group>

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

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

Configure CMakeList.txt:

```
cmake_minimum_required(VERSION 3.5)
project(my_robot_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)
# uncomment the following section in order to fill in
# further dependencies manually.
# find_package(<dependency> REQUIRED)
find_package(rosidl_default_generators REQUIRED)

rosidl_generate_interfaces(${PROJECT_NAME}
        "msg/HardwareStatus.msg"                    [insert file name(s) here]
)

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  # the following line skips the linter which checks for copyrights
  # uncomment the line when a copyright and license is not present in all source files
  #set(ament_cmake_copyright_FOUND TRUE)
  # the following line skips cpplint (only works in a git repo)
  # uncomment the line when this package is not in a git repo
  #set(ament_cmake_cpplint_FOUND TRUE)
  ament_lint_auto_find_test_dependencies()
endif()

ament_package()
```

Create first msg:
Go to msg directory:
james@JP-VM:~/ros2_ws/src/my_robot_interfaces$ **cd msg**
james@JP-VM:~/ros2_ws/src/my_robot_interfaces/msg$ **touch HardwareStatus.msg**
conventions:
Start message name with UPPER case: **H**ardware**S**tatus.**msg**
Start new words with UPPER case (NOT "-" or "_", etc.).
Don't put msg in the name, leave it to the .msg extension.

james@JP-VM:~/ros2_ws/src/my_robot_interfaces/msg/**HardwareStatus.msg**:

int64 temperature
bool are_motors_ready
string debug_message

Compile:
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_interfaces**
Starting >>> my_robot_interfaces
[Processing: my_robot_interfaces]
[Processing: my_robot_interfaces]
Finished <<< my_robot_interfaces [1min 19s]
Summary: 1 package finished [**1min 20s**] **[takes long time first time...]**

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_interfaces**
Starting >>> my_robot_interfaces
Finished <<< my_robot_interfaces [2.52s]
Summary: 1 package finished [**2.89s**] **[second time not so long]**

james@JP-VM:~/ros2_ws$ **ls**
build **install** log src
james@JP-VM:~/ros2_ws$ **cd install/**
james@JP-VM:~/ros2_ws/install$ **ls**
COLCON_IGNORE
_local_setup_util_ps1.py
_local_setup_util_sh.py
local_setup.bash
local_setup.ps1
local_setup.sh
local_setup.zsh
my_cpp_pkg
my_py_pkg
my_robot_interfaces
setup.bash
setup.sh
setup.zsh

james@JP-VM:~/ros2_ws/install$ **cd my_robot_interfaces/**
james@JP-VM:~/ros2_ws/install/my_robot_interfaces$ **ls**
include **lib** share
james@JP-VM:~/ros2_ws/install/my_robot_interfaces$ **cd lib**
james@JP-VM:~/ros2_ws/install/my_robot_interfaces/lib$ **ls**
libmy_robot_interfaces__python.so
libmy_robot_interfaces__rosidl_generator_c.so
libmy_robot_interfaces__rosidl_typesupport_connext_cpp.so
libmy_robot_interfaces__rosidl_typesupport_connext_c.so
libmy_robot_interfaces__rosidl_typesupport_cpp.so
libmy_robot_interfaces__rosidl_typesupport_c.so
libmy_robot_interfaces__rosidl_typesupport_fastrtps_cpp.so
libmy_robot_interfaces__rosidl_typesupport_fastrtps_c.so
libmy_robot_interfaces__rosidl_typesupport_introspection_cpp.so
libmy_robot_interfaces__rosidl_typesupport_introspection_c.so
python3.8
james@JP-VM:~/ros2_ws/install/my_robot_interfaces/lib$ **cd python3.8/**
james@JP-VM:~/ros2_ws/install/my_robot_interfaces/lib/python3.8$ **ls**
site-packages
james@JP-VM:~/ros2_ws/install/my_robot_interfaces/lib/python3.8$ **cd site-packages/**
james@JP-VM:~/ros2_ws/install/my_robot_interfaces/lib/python3.8/site-packages$ **ls**
my_robot_interfaces
james@JP-VM:~/ros2_ws/install/my_robot_interfaces/lib/python3.8/site-packages$ **cd my_robot_interfaces/**
james@JP-VM:~/ros2_ws/install/my_robot_interfaces/lib/python3.8/site-packages/my_robot_interfaces$ **ls**
__init__.py
msg
my_robot_interfaces_s__rosidl_typesupport_c.cpython-38-x86_64-linux-gnu.so
my_robot_interfaces_s__rosidl_typesupport_connext_c.cpython-38-x86_64-linux-gnu.so
my_robot_interfaces_s__rosidl_typesupport_fastrtps_c.cpython-38-x86_64-linux-gnu.so
my_robot_interfaces_s__rosidl_typesupport_introspection_c.cpython-38-x86_64-linux-gnu.so
__pycache__
james@JP-VM:~/ros2_ws/install/my_robot_interfaces/lib/python3.8/site-packages/my_robot_interfaces$ **cd msg**
james@JP-VM:~/ros2_ws/install/my_robot_interfaces/lib/python3.8/site-packages/my_robot_interfaces/msg$ **ls**
_hardware_status.py _hardware_status_s.c __init__.py
james@JP-VM:~/ros2_ws/install/my_robot_interfaces/lib/python3.8/site-packages/my_robot_interfaces/msg$ **gedit _hardware_status.py**
See following page...

159 lines of Python code including our msg definition:

```
...
61      _fields_and_field_types = {
62              'temperature': 'int64',
63              'are_motors_ready': 'boolean',
64              'debug_message': 'string',
65      }
...
```

james@JP-VM:~/ros2_ws/install/my_robot_interfaces/include/my_robot_interfaces/msg$
gedit hardware_status.hpp

```
// generated from rosidl_generator_cpp/resource/idl.hpp.em
// generated code does not contain a copyright notice

#ifndef MY_ROBOT_INTERFACES__MSG__HARDWARE_STATUS_HPP_
#define MY_ROBOT_INTERFACES__MSG__HARDWARE_STATUS_HPP_

#include "my_robot_interfaces/msg/detail/hardware_status__struct.hpp"
#include "my_robot_interfaces/msg/detail/hardware_status__builder.hpp"
#include "my_robot_interfaces/msg/detail/hardware_status__traits.hpp"

#endif  // MY_ROBOT_INTERFACES__MSG__HARDWARE_STATUS_HPP_
```

james@JP-VM:~/ros2_ws/install/my_robot_interfaces/include/my_robot_interfaces/msg$
cd detail
james@JP-VM:~/ros2_ws/install/my_robot_interfaces/include/my_robot_interfaces/msg/detail$ **ls**

```
dds_fastrtps
hardware_status__builder.hpp
hardware_status__functions.c
hardware_status__functions.h
hardware_status__rosidl_typesupport_fastrtps_c.h
hardware_status__rosidl_typesupport_fastrtps_cpp.hpp
hardware_status__rosidl_typesupport_introspection_c.h
hardware_status__rosidl_typesupport_introspection_cpp.hpp
hardware_status__struct.h
hardware_status__struct.hpp
hardware_status__traits.hpp
hardware_status__type_support.c
hardware_status__type_support.cpp
hardware_status__type_support.h
```

[C++ header & support files are detailed here]

```
james@JP-VM:~$ source ~/.bashrc
james@JP-VM:~$ ros2 interface show my_robot_interfaces/msg/HardwareStatus
int64 temperature                                    [^ can TAB from here]
bool are_motors_ready
string debug_message
```

Lesson 65. Use Your Custom Msg in a Python Node

Create Hardware Status Publisher to publish on hardware_status topic:

james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **touch hw_status_publisher.py**

james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **chmod +x hw_status_publisher.py**

cc from template

change names

import msg type:

from my_robot_interfaces.msg import HardwareStatus

[May need to specify path of my_robot_interfaces.msg]:

Visual Studio Code

File

Preferences

Settings[Ctrl+,]

Search: python path

Choose: Python>Auto Complete: Extra Paths

Choose: Edit in settings json

```
{
   "python.autoComplete.extraPaths": [
        "~/ros2_ws/install/my_robot_interfaces/lib/python3.8/site-packages/...
my_robot_interfaces"
   ]
}
```

Add dependency to package.xml:

<depend>my_robot_interfaces</depend>

Create publisher:

self.hardware_status_publisher_ = self.create_publisher(HardwareStatus, "hardware_status", 10)

Create function to publish status:

```
   def publish_hw_status(self):
      msg = HardwareStatus()
      msg.temperature = 45
      msg.are_motors_ready = True
      msg.debug_message = "Nothing special here"
      self.hardware_status_publisher_.publish(msg)
```

Create timer:

self.timer_ = self.create_timer(1.0, self.publish_hw_status)

Create logger:

self.get_logger().info("Harware status publisher has been started.")

Add install to setup.py:

"hw_status_publisher = my_py_pkg.hw_status_publisher:main"

Compile:
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg --symlink-install**
Starting >>> my_py_pkg
Finished <<< my_py_pkg [3.82s]
Summary: 1 package finished [4.33s]
Run:
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 run my_py_pkg hw_status_publisher**
james@JP-VM:~$ **ros2 node list**
/hardware_status_publisher
james@JP-VM:~$ **ros2 node info /hardware_status_publisher**
/hardware_status_publisher
  Subscribers:
  Publishers:
    /hardware_status: my_robot_interfaces/msg/HardwareStatus
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
  Service Servers:
    /hardware_status_publisher/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /hardware_status_publisher/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /hardware_status_publisher/get_parameters: rcl_interfaces/srv/GetParameters
    /hardware_status_publisher/list_parameters: rcl_interfaces/srv/ListParameters
    /hardware_status_publisher/set_parameters: rcl_interfaces/srv/SetParameters
    /hardware_status_publisher/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:
  Action Servers:
  Action Clients:
james@JP-VM:~$ **ros2 topic list**
/hardware_status
/parameter_events
/rosout
james@JP-VM:~$ **ros2 topic echo /hardware_status**
...
ModuleNotFoundError: No module named 'my_robot_interfaces'
forgot to:
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 topic echo /hardware_status**
temperature: 45
are_motors_ready: true
debug_message: Nothing special here

...

Leson 66. Use Your Custom Msg in a Cpp Node
james@JP-VM:~/ros2_ws/src/my_cpp_pkg/src$ **touch hw_status_publisher.cpp**
cc from **number_publisher.cpp**
~~#include "example_interfaces/msg/int64.hpp"~~
#include "my_robot_interfaces/msg/hardware_status.hpp"

change names
change interface

Edit package.xml:
    <depend>my_robot_interfaces</depend>

Edit CmakeLists.txt:
find_package(my_robot_interfaces REQUIRED)

add_executable(hardware_status_publisher src/hw_status_publisher.cpp)
ament_target_dependencies(hardware_status_publisher rclcpp my_robot_interfaces)

hardware_status_publisher

Edit:
    .vscode/c_cpp_properties.json
        "~/ros2_ws/install/my_robot_interfaces/include"

~/ros2_ws/src/my_cpp_pkg/src/**hw_status_publisher.cpp**:

```
#include "rclcpp/rclcpp.hpp"
#include "my_robot_interfaces/msg/hardware_status.hpp"

class HardwareStatusPublisher : public rclcpp::Node
{
public:
   HardwareStatusPublisher() : Node("hardware_status_publisher")
   {
        publisher_ = this->
                create_publisher<my_robot_interfaces::msg::HardwareStatus>(
                "hardware_status", 10);
        timer_ = this->create_wall_timer(
                std::chrono::seconds(1),
                std::bind(&HardwareStatusPublisher::publishHardwareStatus, this));
      RCLCPP_INFO(this->get_logger(), "Hardware Status Publisher has been
started.");
   }

private:
   void publishHardwareStatus()
   {
        auto msg = my_robot_interfaces::msg::HardwareStatus();
        msg.temperature = 57;
        msg.are_motors_ready = false;
        msg.debug_message = "Motors are too hot!";
        publisher_->publish(msg);
   }

        rclcpp::Publisher<my_robot_interfaces::msg::HardwareStatus>::
                SharedPtr publisher_;
        rclcpp::TimerBase::SharedPtr timer_;
};

int main(int argc, char **argv)
{
   rclcpp::init(argc, argv);
   auto node = std::make_shared<HardwareStatusPublisher>();
   rclcpp::spin(node);
   rclcpp::shutdown();
   return 0;
}
```

Compile:
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg my_robot_interfaces** [need my_robot_interfaces to compile or won't work]

Run:
james@JP-VM:~$ **ros2 run my_cpp_pkg hardware_status_publisher**
[INFO] [1600044142.960240123] [hardware_status_publisher]: Hardware Status Publisher has been started.

james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 node list**
/hardware_status_publisher
james@JP-VM:~$ **ros2 topic list**
/hardware_status
/parameter_events
/rosout
james@JP-VM:~$ **ros2 topic echo /hardware_status**
temperature: 57
are_motors_ready: false
debug_message: Motors are too hot!

Lesson 67. Create and Build Your First Custom Server

james@JP-VM:~$ **cd ros2_ws/src/my_robot_interfaces/**
james@JP-VM:~/ros2_ws/src/my_robot_interfaces$ **ls**
CMakeLists.txt msg package.xml
james@JP-VM:~/ros2_ws/src/my_robot_interfaces$ **mkdir srv**
james@JP-VM:~/ros2_ws/src/my_robot_interfaces$ **cd srv**
james@JP-VM:~/ros2_ws/src/my_robot_interfaces/srv$ **touch ComputeRectangleArea.srv**

```
float64 length
float64 width
---
float64 area
```

package.xml already has –

```
  <build_depend>rosidl_default_generators</build_depend>
  <exec_depend>rosidl_default_runtime</exec_depend>
  <member_of_group>rosidl_interface_packages</member_of_group>
```

CmakeLists.txt already has -

```
find_package(rosidl_default_generators REQUIRED)
add -
rosidl_generate_interfaces(${PROJECT_NAME}
  "msg/HardwareStatus.msg"
  "srv/ComputeRectangleArea.srv"
```

james@JP-VM:~/ros2_ws/src/my_robot_interfaces/srv$ **cd ~/ros2_ws/**
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_interfaces**

james@JP-VM:~$ **source .bashrc**
james@JP-VM:~$ **ros2 interface show my_robot_interfaces/**

```
Traceback (most recent call last):
  File "/opt/ros/foxy/bin/ros2", line 11, in <module>
    load_entry_point('ros2cli==0.9.7', 'console_scripts', 'ros2')()
  File "/opt/ros/foxy/lib/python3.8/site-packages/ros2cli/cli.py", line 67, in main
    rc = extension.main(parser=parser, args=args)
  File "/opt/ros/foxy/lib/python3.8/site-packages/ros2interface/command/interface.py",
line 35, in main
    return extension.main(args=args)
  File "/opt/ros/foxy/lib/python3.8/site-packages/ros2interface/verb/show.py", line 50, in
main
    file_path = get_interface_path(args.type)
  File "/opt/ros/foxy/lib/python3.8/site-packages/rosidl_runtime_py/get_interfaces.py", line
173, in get_interface_path
    raise ValueError(f"Invalid name '{interface_name}'. Must not contain empty parts")
ValueError: Invalid name 'my_robot_interfaces/'. Must not contain empty parts
```

Google suggests this means there are no interfaces in the package. (But there are!)

james@JP-VM:~/ros2_ws/install/my_robot_interfaces/lib/python3.8/site-packages/my_robot_interfaces/srv$ **ls**
_compute_rectangle_area.py _compute_rectangle_area_s.c __init__.py

2020.09.21
Email from Ed Renard:

Re: ROS2 for Beginners: Activity 003 compile issue
Hi Jim,
Sorry for the late reply!
This error is normal: you should only use colcon build from the root of your ROS2 workspace (so: ~/ros2_ws), and not from the src/ folder of that workspace. (if you spot somewhere in the videos where I do this please let me know so I can update it!)
One additional thing: after you've create a new file + compile with colcon, it's a good practice to source again your workspace in your already opened terminals (source ~/ros2/install/setup.bash) so you're sure that everything (executables, messages) is found.
Best regards,
Edouard

Re: ROS2 For Beginners Lesson 67 Error message
Hi Jim,
You get this error simply because you didn't provide a full name for the interface :)
If you do "ros2 interface list | grep my_robot", you'll get the full names.
And here in that case you'll have to run ros2 interface show "my_robot_interfaces/srv/ComputeRectangleArea.srv" for example.
--> "my_robot_interfaces/srv/ComputeRectangleArea.srv" is the full name of the interface.
--> "my_robot_interfaces/" is just a "namespace" for the interface, and it's part of the name. But by itself it represents nothing.
Hope this helps!
Best regards,
Edouard

My reply:

Edouard,

Ah, I didn't catch the "tab twice" to do the search!

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_interfaces**
[new]
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 interface show my_robot_interfaces/[tab][tab]**
my_robot_interfaces/msg/HardwareStatus
my_robot_interfaces/srv/ComputeRectangleArea

james@JP-VM:~$ **ros2 interface show my_robot_interfaces/srv/[tab][tab]**
ComputeRectangleArea
float64 length
float64 width
---
float64 area

Much appreciate your responses! I hope my mistakes will help you make your lessons more "idiot proof"!

Jim [Jacques ;-)]

Lesson 68. Debug Msg and Srv with ROS2 Tools
james@JP-VM:~$ source ~/.bashrc
james@JP-VM:~$ **ros2 interface [tab][tab]**
list package packages proto show

```
james@JP-VM:~$ ros2 interface package sensor_msgs
sensor_msgs/msg/JoyFeedback
sensor_msgs/srv/SetCameraInfo
...
```

james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 run my_py_pkg hw_status_publisher**
[INFO] [1600737025.077255687] [hardware_status_publisher]: Harware status publisher has been started.
[new]
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 node list**
/hardware_status_publisher
james@JP-VM:~$ **ros2 node info /hardware_status_publisher**
/hardware_status_publisher
  Subscribers:
  Publishers:
    /hardware_status: my_robot_interfaces/msg/HardwareStatus *[topic]: [interface]*
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
...
  Service Servers:
...
  Service Clients:
  Action Servers:
  Action Clients:

james@JP-VM:~$ **ros2 topic list**
/hardware_status
/parameter_events
/rosout

james@JP-VM:~$ **ros2 topic info /har[tab]**dware_status
Type: my_robot_interfaces/msg/HardwareStatus
Publisher count: 1
Subscription count: 0

james@JP-VM:~$ **ros2 interface show**
my_ro[tab]bot_interfaces/**msg/Hard[tab]**wareStatus
int64 temperature
bool are_motors_ready
string debug_message

[kill hw_status_publisher ^C]

---

[2]
james@JP-VM:~$ **ros2 run my_py_pkg add_two_ints_server**
[INFO] [1600738235.724657652] [add_two_ints_server]: Add Two Ints Server has been started
[3]
james@JP-VM:~$ **ros2 node list**
/add_two_ints_server
james@JP-VM:~$ **ros2 node info /add_two_ints_server**
/add_two_ints_server
  Subscribers:
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
  Service Servers:
    /add_two_ints: example_interfaces/srv/AddTwoInts *[service]: [interface]*
...
  Service Clients:
  Action Servers:
  Action Clients:
james@JP-VM:~$ **ros2 service list**
/add_two_ints
...
james@JP-VM:~$ **ros2 service type /add_two_ints**
example_interfaces/srv/AddTwoInts
james@JP-VM:~$ **ros2 interface show example_interfaces/srv/AddTwoInts**
int64 a
int64 b
---
int64 sum

Activity 004 - ROS2 Custom Interfaces
see instruction .pdf

BatteryNode
batteryState = true
request service /set_led Led_number: 3, state: false
wait 4 sec
batteryState = false
request service /set_led Led_number: 3, state: true
wait 6 sec
repeat

LedPanelNode
create publisher: /led_states
led = [0,0,0]; led[0,1,2]
create service: /set_led
    Led_number; Int64 (1, 2 or 3)
    state Boole (false/off, true/on)
if request: Led_number, state: true
    led[Led_number - 1] = true
    publish /led_states: led [0,0,0] response: success = true
if request: Led_number, state: off
    led[Led_number - 1] = false
    publish /led_states: [0,0,0]
    response: success = true

/LedStates.msg
Int64 led This is tricky, not sure how ROS2 handles python lists..?

/SetLed.svc
Int64 Led_number
Bool state

Bool success

james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg/**led_panel_server.py**:

ImportError: cannot import name 'LedStates' from 'my_robot_interfaces.msg' (/home/james/ros2_ws/install/my_robot_interfaces/lib/python3.8/site-packages/my_robot_interfaces/msg/__init__.py)

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from example_interfaces.msg import Int64
from my_robot_interfaces.msg import LedStates
from my_robot_interfaces.srv import SetLed


class LedPanelServerNode(Node):
    def __init__(self):
        super().__init__("led_panel_server")
        self.led = [0,0,0]

        self.publisher_ = self.create_publisher(Int64, "led_states", 10)

        self.server_ = self.create_service(
            SetLed, "set_led", self.callback_set_led)
        self.get_logger().info("Set LED Server has been started.")

    def callback_led_states(self, msg):
        outmsg = Int64()
        outmsg.led = self.led
        self.publisher_.publish(outmsg)
        self.get_logger().info(str(outmsg.led))

    def callback_set_led(self, request, response):
        if request.state:
            self.led[request.Led_number - 1] = 0
            response.success = True
        else:
            self.led[request.Led_number - 1] = 1
            response.success = True
        return response

def main(args=None):
    rclpy.init(args=args)
    node = LedPanelServerNode()
    rclpy.spin(node)
    rclpy.shutdown()
```

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

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg**
[ ]
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 run my_py_pkg led_panel_server**
...
ImportError: cannot import name 'LedStates' from 'my_robot_interfaces.msg' (/home/james/ros2_ws/install/my_robot_interfaces/lib/python3.8/site-packages/my_robot_interfaces/msg/__init__.py)

Ah, forgot to:
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **chmod +x led_panel_server.py**

james@JP-VM:~$ **ros2 run my_py_pkg led_panel_server**
same error

Ah, forgot to:
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_interfaces**
[ ]
james@JP-VM:~$ ros2 **interface show my_robot_interfaces/[tab][tab]**
my_robot_interfaces/msg/HardwareStatus
my_robot_interfaces/srv/ComputeRectangleArea
my_robot_interfaces/msg/LedStates is missing
my_robot_interfaces/srv/SetLed is missing
Try:
james@JP-VM:~/ros2_ws/src/my_robot_interfaces/msg$ **chmod +x LedStates.msg**
Shouldn't matter as the other .srv and .msg files are not executable.
Somehow .msg and .srv are not being compiled properly.

CHEAT: Go to Solution Part I:
Instead of
    LedStates.msg as Int64 led
create
    LedStateArray.msg as Int64[] led_states

Ah, forgot to update CmakeLists.txt
  "msg/LedStatesArray.msg"

Error on colcon build. Start over.

Create **LedStateArray.msg**
james@JP-VM:~/ros2_ws/src/my_robot_interfaces/msg$ **ls**
HardwareStatus.msg
james@JP-VM:~/ros2_ws/src/my_robot_interfaces/msg$ **touch LedStateArray.msg**
james@JP-VM:~/ros2_ws/src/my_robot_interfaces/msg$ **ls**
HardwareStatus.msg LedStateArray.msg

LedStateArray.msg:
int64[] led_states

CmakeLists.txt
add:
  "msg/LedStateArray.msg"

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_interfaces**
success

james@JP-VM:~$ **ros2 interface list**
.....
    my_robot_interfaces/msg/LedStateArray
.....

james@JP-VM:~$ **ros2 interface show my_robot_interfaces/msg/LedStateArray**
int64[] led_states

Create Node:
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **touch led_panel.py**
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **chmod +x led_panel.py**

cc template_python_node.py
change name:
LedPanelNode

```
from my_robot_interfaces.msg import LedStateArray
self.led_states_publisher_ = self.create_publisher(LedStateArray, "led_states", 10)
```

create variable:
```
self.led_states_ = [0, 0, 0]
```

create publisher:
```
        def publish_led_states(self):
                msg = LedStateArray()
                msg.led_states = self.led_states_
                self.led_states_publisher_.publish(msg)
```
create timer:
```
        self.led_states_timer_ = self.create_timer(4, self.publish_led_states)
```

led_panel.py:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node

from my_robot_interfaces.msg import LedStateArray


class LedPanelNode(Node):
    def __init__(self):
        super().__init__("led_panel")
        self.led_states_ = [0, 0, 0]
        self.led_states_publisher_ = self.create_publisher(LedStateArray, "led_states", 10)
        self.led_states_timer_ = self.create_timer(4, self.publish_led_states)
        self.get_logger().info("LED panel node has been started.")

    def publish_led_states(self):
        msg = LedStateArray()
        msg.led_states = self.led_states_
        self.led_states_publisher_.publish(msg)

def main(args=None):
    rclpy.init(args=args)
    node = LedPanelNode()
    rclpy.spin(node)
    rclpy.shutdown()

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

Setup.py:

```
        "led_panel = my_py_pkg.led_panel:main"
```

[new]

james@JP-VM:~$ **source ~/.bashrc**

james@JP-VM:~$ **ros2 run my_py_pkg led_panel**

nothing happens - no log output. Add in (above):

```
        self.get_logger().info("LED panel node has been started.")
```

[new]

james@JP-VM:~$ **ros2 run my_py_pkg led_panel**

[INFO] [1601171071.664970984] [led_panel]: LED panel node has been started.

[new]
james@JP-VM:~$ **ros2 node list**
/led_panel
james@JP-VM:~$ **ros2 node info /led_panel**
/led_panel
  Subscribers:
  Publishers:
    /led_states: my_robot_interfaces/msg/LedStateArray
.....
james@JP-VM:~$ **ros2 interface show my_robot_interfaces/msg/LedStateArray**
int64[] led_states

james@JP-VM:~$ **ros2 topic list**
/led_states
/parameter_events
/rosout

james@JP-VM:~$ **ros2 topic echo /led_states**
led_states:
- 0
- 0
- 0

...

Now try adding Server...
Colcon didn't like --

SetLed.srv:
Int64 led_number
Bool state

Bool success

but was ok with –

SetLed.srv
int64 led_number changed to lednumber when failed to compile, didn't help
bool state
---
bool success

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_interfaces**
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg --symlink-install**
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 run my_py_pkg led_panel**
ModuleNotFoundError: No module named 'my_robot_interfaces.svc'

see failed led_panel.py next page.

Resort to cheat Solution #2:

led_panel.py: FAILED

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node

from my_robot_interfaces.msg import LedStateArray
from my_robot_interfaces.svc import SetLed

class LedPanelNode(Node):
    def __init__(self):
        super().__init__("led_panel")
        self.led_states_ = [0, 0, 0]
        self.led_states_publisher_ = self.create_publisher(LedStateArray, "led_states", 10)
        self.server_ = self.create_service(
            SetLed, "set_led", self.callback_set_led)
        self.get_logger().info("LED panel node has been started.")

    def publish_led_states(self):
        msg = LedStateArray()
        msg.led_states = self.led_states_
        self.led_states_publisher_.publish(msg)

    def callback_set_led(self, request, response):
        if request.state:
            self.led_states_[lednumber - 1] = False
            response.success = True
            self.get_logger().info("LED " + self.led_states_.lednumber + "has been turned
OFF.")
        else:
            self.led_states_[lednumber - 1] = True
            response.success = True
            self.get_logger().info("LED " + self.led_states_.lednumber + "has been turned
ON.")
        return response

def main(args=None):
    rclpy.init(args=args)
    node = LedPanelNode()
    rclpy.spin(node)
    rclpy.shutdown()

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

Solution #2:
james@JP-VM:~/ros2_ws/src/my_robot_interfaces/srv$ **touch SetLed.srv**
SetLed.srv:
int64 led_number
int64 state
---
bool success

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_interfaces**

[new]
james@JP-VM:~$ source ~/.bashrc
james@JP-VM:~$ **ros2 interface show my_rob[tab]**ot_interfaces/**[tab][tab]**
my_robot_interfaces/msg/HardwareStatus
my_robot_interfaces/msg/LedStateArray
my_robot_interfaces/srv/ComputeRectangleArea
my_robot_interfaces/srv/SetLed
james@JP-VM:~$ **ros2 interface show my_robot_interfaces/srv/SetLed**
int64 led_number
int64 state
---
bool success

Edit led_panel.py:
from my_robot_interfaces.svc import SetLed
Create Service Server:

```
        self.set_led_service_ = self.create_service(
                SetLed, "set_led", self.callback_set_led)
```

Create Callback Function:

```
   def callback_set_led(self, request, response):
      led_number = request.led_number
      state = request.state

      if led_number > len(self.led_stated) or led_number <= 0:
         response.success = False
         return response

      if state not in [0, 1]:
         response.success = False
         return response

      self.led_states_[led_number - 1] = state
      response.success = True
      self.publish_led_states()
      return response
```

led_panel.py: From Solution #2 (blank lines removed to save space)

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node

from my_robot_interfaces.msg import LedStateArray
from my_robot_interfaces.svc import SetLed

class LedPanelNode(Node):
    def __init__(self):
        super().__init__("led_panel")
        self.led_states_ = [0, 0, 0]
        self.led_states_publisher_ = self.create_publisher(
            LedStateArray, "led_states", 10)
        self.led_states_timer_ = self.create_timer(4, self.publish_led_states)
        self.set_led_service_ = self.create_service(
            SetLed, "set_led", self.callback_set_led)
        self.get_logger().info("LED panel node has been started.")

    def publish_led_states(self):
        msg = LedStateArray()
        msg.led_states = self.led_states_
        self.led_states_publisher_.publish(msg)

    def callback_set_led(self, request, response):
        led_number = request.led_number
        state = request.state
        if led_number > len(self.led_states_) or led_number <= 0:
            response.success = False
            return response
        if state not in [0, 1]:
            response.success = False
            return response
        self.led_states_[led_number - 1] = state
        response.success = True
        self.publish_led_states()
        return response

def main(args=None):
    rclpy.init(args=args)
    node = LedPanelNode()
    rclpy.spin(node)
    rclpy.shutdown()

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

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_interfaces**
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg --symlink-install**

[new]
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 interface show my_robot_interfaces/srv/SetLed**
int64 led_number
int64 state
---
bool success

[new]
james@JP-VM:~$ **source .bashrc**
james@JP-VM:~$ **ros2 run my_py_pkg led_panel**
Traceback (most recent call last):
  File "/home/james/ros2_ws/install/my_py_pkg/lib/my_py_pkg/led_panel", line 11, in <module>
    load_entry_point('my-py-pkg', 'console_scripts', 'led_panel')()
  File "/usr/lib/python3/dist-packages/pkg_resources/__init__.py", line 490, in load_entry_point
    return get_distribution(dist).load_entry_point(group, name)
  File "/usr/lib/python3/dist-packages/pkg_resources/__init__.py", line 2854, in load_entry_point
    return ep.load()
  File "/usr/lib/python3/dist-packages/pkg_resources/__init__.py", line 2445, in load
    return self.resolve()
  File "/usr/lib/python3/dist-packages/pkg_resources/__init__.py", line 2451, in resolve
    module = __import__(self.module_name, fromlist=['__name__'], level=0)
  File "/home/james/ros2_ws/build/my_py_pkg/my_py_pkg/led_panel.py", line 6, in <module>
    from my_robot_interfaces.svc import SetLed
ModuleNotFoundError: No module named 'my_robot_interfaces.svc'

james@JP-VM:~/ros2_ws/src/my_robot_interfaces/srv$ **ls**
ComputeRectangleArea.srv SetLed.srv
[ComputeRectangleArea.srv is WHITE while SetLed.srv is GREEN]

Reply from instructor:
Hi Jim,
It seems you have a typo in your code.
from my_robot_interfaces.**svc** import SetLed --> should be from my_robot_interfaces.**srv** import SetLed.
Hope this solves the issue.
Best regards,
Edouard

setup.py:

```
from setuptools import setup

package_name = 'my_py_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='james',
    maintainer_email='james@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            "py_node = my_py_pkg.my_first_node:main",
            "robot_news_station = my_py_pkg.robot_news_station:main",
            "smartphone = my_py_pkg.smartphone:main",
            "number_publisher = my_py_pkg.number_publisher:main",
            "number_counter = my_py_pkg.number_counter:main",
            "add_two_ints_server = my_py_pkg.add_two_ints_server:main",
            "add_two_ints_client_no_oop = my_py_pkg.add_two_ints_client_no_oop:main",
            "add_two_ints_client = my_py_pkg.add_two_ints_client:main",
            "number_server = my_py_pkg.number_server:main",
            "hw_status_publisher = my_py_pkg.hw_status_publisher:main",
            "led_panel = my_py_pkg.led_panel:main"
        ],
    },
)
```

CMakeLists.txt:

```
cmake_minimum_required(VERSION 3.5)
project(my_robot_interfaces)

# 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)

rosidl_generate_interfaces(${PROJECT_NAME}
  "msg/HardwareStatus.msg"
  "msg/LedStateArray.msg"
  "srv/ComputeRectangleArea.srv"
  "srv/SetLed.srv"
)

ament_package()
```

2020.09.28
Corrected error:
from my_robot_interfaces.**svc** import SetLed --> should be from
my_robot_interfaces.**srv** import SetLed.
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_interfaces**
...
Summary: 1 package finished [9.68s]
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg**
--symlink-install
...
Summary: 1 package finished [3.10s]
[new term]
james@JP-VM:~$ **source .bashrc**
james@JP-VM:~$ **ros2 run my_py_pkg led_panel**
[INFO] [1601338093.587636631] [led_panel]: LED panel node has been started.
[new term]
james@JP-VM:~$ **ros2 node list**
/led_panel
james@JP-VM:~$ **ros2 topic list**
/led_states
/parameter_events
/rosout
james@JP-VM:~$ **ros2 topic echo /led_states**
led_states:
- 0
- 0
- 0

[new term]james@JP-VM:~$ **ros2 service call /set_led**
my_robot_interfaces/srv/SetLed "{led_number: 3, state: 1}"
requester: making request: my_robot_interfaces.srv.SetLed_Request(led_number=3,
state=1)
response:
my_robot_interfaces.srv.SetLed_Response(success=True)
led_states:
- 0
- 0
- **0**
---
led_states:
- 0
- 0
- **1**

Now, on to part 3, create a client for the SetLed Server:

james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **touch battery.py**
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **chmod +x battery.py**

Cc from add_two_ints_client??
Don't know how to handle 2 time frames - 4 sec discharge & 6 sec recharge.

Solution 004 3/3:
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **touch battery.py**
james@JP-VM:~/ros2_ws/src/my_py_pkg/my_py_pkg$ **chmod +x battery.py**
Initial node w/o client/server, just timer --
battery.py:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
class BatteryNode(Node):
    def __init__(self):
        super().__init__("battery")
        self.battery_state_ = "full"
        self.last_time_battery_state_changed_ = self.get_current_time_seconds()
        self.battery_timer_ = self.create_timer(0.1, self.check_battery_state)
        self.get_logger().info("Battery node has been started")
    def get_current_time_seconds(self):
        secs, nsecs = self.get_clock().now().seconds_nanoseconds()
        return secs + nsecs / 1000000000.0
    def check_battery_state(self):
        time_now = self.get_current_time_seconds()
        if self.battery_state_ == "full":
            if time_now - self.last_time_battery_state_changed_ > 4.0:
                self.battery_state_ = "empty"
                self.get_logger().info("Battery is empty! Charging battery...")
                self.last_time_battery_state_changed_ = time_now
        else:
            if time_now - self.last_time_battery_state_changed_ > 6.0:
                self.battery_state_ = "full"
                self.get_logger().info("Battery is now full again.")
                self.last_time_battery_state_changed_ = time_now
def main(args=None):
    rclpy.init(args=args)
    node = BatteryNode()
    rclpy.spin(node)
    rclpy.shutdown()
if __name__ == "__main__":
    main()
```

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_py_pkg --symlink-install**
[new term]james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 run my_py_pkg battery**
[INFO] [1601513134.366814553] [battery]: Battery node has been started
[INFO] [1601513138.353385155] [battery]: Battery is empty! Charging battery...
[INFO] [1601513144.354101517] [battery]: Battery is now full again.
...

Now add client to battery.py:
battery.py:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from functools import partial

from my_robot_interfaces.srv import SetLed

class BatteryNode(Node):
    def __init__(self):
        super().__init__("battery")
        self.battery_state_ = "full"
        self.last_time_battery_state_changed_ = self.get_current_time_seconds()
        self.battery_timer_ = self.create_timer(0.1, self.check_battery_state)
        self.get_logger().info("Battery node has been started")

    def get_current_time_seconds(self):
        secs, nsecs = self.get_clock().now().seconds_nanoseconds()
        return secs + nsecs / 1000000000.0

    def check_battery_state(self):
        time_now = self.get_current_time_seconds()
        if self.battery_state_ == "full":
            if time_now - self.last_time_battery_state_changed_ > 4.0:
                self.battery_state_ = "empty"
                self.get_logger().info("Battery is empty! Charging battery...")
                self.last_time_battery_state_changed_ = time_now
                self.call_set_led_server(3, 1)
        else:
            if time_now - self.last_time_battery_state_changed_ > 6.0:
                self.battery_state_ = "full"
                self.get_logger().info("Battery is now full again.")
                self.last_time_battery_state_changed_ = time_now
                self.call_set_led_server(3, 0)
```

```
    def call_set_led_server(self, led_number, state):
        client = self.create_client(SetLed, "set_led")
        while not client.wait_for_service(1.0):
            self.get_logger().warn("Waiting for Server to set LED...")
        request = SetLed.Request()
        request.led_number = led_number
        request.state = state
        future = client.call_async(request)
        future.add_done_callback(partial(self.callback_call_set_led, led_number =
led_number, state = state))
    def callback_call_set_led(self, future, led_number, state):
        try:
            response = future.result()
            self.get_logger().info(str(response.success))
        except Exception as e:
            self.get_logger().error("Service call failed %r" % (e, ))
def main(args=None):
    rclpy.init(args=args)
    node = BatteryNode()
    rclpy.spin(node)
    rclpy.shutdown()
if __name__ == "__main__":
    main()

james@JP-VM:~/ros2_ws$ colcon build --packages-select my_py_pkg
--symlink-install
[new term]james@JP-VM:~$ source ~/.bashrc
james@JP-VM:~$ ros2 run my_py_pkg led_panel
[INFO] [1601514939.323122563] [led_panel]: LED panel node has been started.
[new term]james@JP-VM:~$ ros2 run my_py_pkg battery
[INFO] [1601514735.544075968] [battery]: Battery node has been started
[INFO] [1601514739.529531054] [battery]: Battery is empty! Charging battery...
[INFO] [1601514739.553020191] [battery]: True
[INFO] [1601514745.629986731] [battery]: Battery is now full again.
[INFO] [1601514745.649511545] [battery]: True
[new term]james@JP-VM:~$ ros2 topic echo /led_states
led_states:
- 0
- 0
- 0
---
led_states:
- 0
- 0
- 1
---
```

Section Conclusion
In this section you have seen how to create your own ROS2 interfaces, for Topics and Services.
Topics and Services are the communication layer. Interfaces are the actual content that you send.

To recap, here's how to create a custom interface:
Create a new package only for your msg and srv definitions.
Setup the package (CMakeLists.txt and package.xml)
Create a msg/ and srv/ folders, place your custom msg definitions and srv definitions here.

Once you've setup your package, adding a new interface is really simple:
Add a new file in the right folder: msg/ or srv/
Add one line into CMakeLists.txt
Compile with "colcon build"
And don't forget to source your ROS2 workspace when you want to use those messages!

Here's what you can use inside a msg or srv definition:
Any primitive type defined by ROS2 (most common ones: int64, float64, bool, string, and array of those)
Any message you've already created in this package.
Any message from another package. In this case don't forget to add a dependency for the other package in both package.xml and CMakeLists.txt.

And now, when you compile the definitions, new interfaces will be created, along with headers/modules ready to be included in your C++ or Python nodes.
You're almost ready to start your own ROS2 application! In the next section you'll see how to add settings to your nodes at run time with ROS2 parameters.

Download the complete code for this section (this is the code from all previous sections + the current one).
Resources for this lecture
code_end_section_7_interfaces.zip

Lesson 75. What is a ROS2 Parameter?
Settings for your nodes, value set at run time
A parameter is specific to a node, only valid at run time
Parameter has a name & a data type (Boolean, Int, Double, String, Lists, ...)

Lesson 76. Declare Your Parameters
Parameters must be declared from within the node.
[Terminator window 2]
james@JP-VM:~$ **ros2 run my_py_pkg number_publisher**
[INFO] [1602433616.382869736] [number_publisher]: 2
[INFO] [1602433616.442467935] [number_publisher]: Number Publisher has been started
[4]
james@JP-VM:~$ ros2 run my_py_pkg number_counter
[INFO] [1602433704.527456557] [number_counter]: Number Counter has been started.
[INFO] [1602433706.949768703] [number_counter]: 1
[INFO] [1602433707.443950551] [number_counter]: 2
[INFO] [1602433707.944530148] [number_counter]: 3
[3]
james@JP-VM:~$ **ros2 param[tab][tab]**
delete describe dump get list set
james@JP-VM:~$ **ros2 param list**
/number_counter:
use_sim_time [This is a built-in parameter in ROS]
/number_publisher:
use_sim_time instructor also shows _publisher but mine doesn't
james@JP-VM:~$ **ros2 node list**
/number_counter
/number_publisher
james@JP-VM:~$ **ros2 param get /number_publisher use_sim_time**
Boolean value is: False
[mine lists param for publisher even tho not listed previously]
james@JP-VM:~$ **ros2 param get /number_counter use_sim_time**
Boolean value is: False

Even tho they have the same name, the use_sim_time parameters are unique to each node. One could be True and the other False.

Now when I do param list, I get a different output:
james@JP-VM:~$ **ros2 param list [tab][tab]**
--include-hidden-nodes /number_counter --param-prefixes
--no-daemon /number_publisher --spin-time
james@JP-VM:~$ **ros2 param list[CR]**
[no output?!]

[Kill all jobs. Clear all windows. Source each window. Now it behaves better.]
[2]
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 run my_py_pkg number_publisher**
[INFO] [1602435337.251904436] [number_publisher]: 2
[INFO] [1602435337.256822252] [number_publisher]: Number Publisher has been started
[4]
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 run my_py_pkg number_counter**
[INFO] [1602435716.720212782] [number_counter]: Number Counter has been started.
[INFO] [1602435718.786973458] [number_counter]: 1
[INFO] [1602435719.265449567] [number_counter]: 2
[INFO] [1602435719.765552557] [number_counter]: 3
[3]
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 node list**
/number_counter
/number_publisher
james@JP-VM:~$ **ros2 param list**
/number_publisher:
  use_sim_time
james@JP-VM:~$ **ros2 param get /number_counter**
usage: ros2 param get [-h] [--spin-time SPIN_TIME] [--no-daemon]
                      [--include-hidden-nodes] [--hide-type]
                      node_name parameter_name
ros2 param get: error: the following arguments are required: parameter_name
james@JP-VM:~$ **ros2 param get /number_counter use_sim_time**
Boolean value is: False

Modify parameter in **number_publisher.py**:
```
class NumberPublisherNode(Node):
    def __init__(self):
        super().__init__("number_publisher")
        self.declare_parameter("test123")
```

james@JP-VM:~$ **ros2 param list**
/number_publisher:
  test123
  use_sim_time

james@JP-VM:~$ **ros2 param get /number_publisher test123**
Parameter not set.

May use ros2 “- -ros-args -p” to set parameter value at runtime:
[2]
james@JP-VM:~$ **ros2 run my_py_pkg number_publisher --ros-args -p test123:=3**
[INFO] [1602437080.954576321] [number_publisher]: 2
[INFO] [1602437080.959502367] [number_publisher]: Number Publisher has been started
[3]
james@JP-VM:~$ **ros2 param get /number_publisher test123**
Integer value is: 3

the value is set dynamically by ROS
[2]
james@JP-VM:~$ **ros2 run my_py_pkg number_publisher --ros-args -p test123:=3.14**
[INFO] [1602437249.888049071] [number_publisher]: 2
[INFO] [1602437249.895483289] [number_publisher]: Number Publisher has been started
[3]
james@JP-VM:~$ **ros2 param get /number_publisher test123**
Double value is: 3.14

james@JP-VM:~$ **ros2 run my_py_pkg number_publisher --ros-args -p test123:="3.14"**
james@JP-VM:~$ **ros2 param get /number_publisher test123**
Double value is: 3.14

james@JP-VM:~$ **ros2 run my_py_pkg number_publisher --ros-args -p test123:="hello"**
james@JP-VM:~$ **ros2 param get /number_publisher test123**
String value is: hello

Setting a parameter on the command line WITHOUT declaring in the code will NOT give an error. [But if declared & used in the code and it’s NOT set in the command line, you WILL get an error as it will be undefined.]

More than one parameter may be set:

```
class NumberPublisherNode(Node):
    def __init__(self):
        super().__init__("number_publisher")
        self.declare_parameter("test123")
        self.declare_parameter("another_param")
```

james@JP-VM:~$ **ros2 run my_py_pkg number_publisher --ros-args -p test123:="hello" -p another_param:="Hi"**

james@JP-VM:~$ **ros2 param list** [THIS FUNCTION **INCONSISTENT**!!]
/number_publisher:
  another_param
  test123
  use_sim_time
james@JP-VM:~$ **ros2 param get /number_publisher another_param**
String value is: Hi

modify parameter in **number_publisher.cpp**:

```
class NumberPublisherNode : public rclcpp::Node
{
public:
    NumberPublisherNode() : Node("number_publisher"), number_(2)
    {
        this->declare_parameter("parameter_name");
```

Lesson 77. Get Parameters from a Python Node
How to use parameters once declared and set.

number_publisher.py:

```
class NumberPublisherNode(Node):
    def __init__(self):
        super().__init__("number_publisher")
        self.declare_parameter("number_to_publish")

            self.number = 2
becomes
            self.number = self.get_parameter("number_to_publish").value
```

james@JP-VM:~$ **ros2 run my_py_pkg number_publisher --ros-args -p number_to_publish:=4**
[INFO] [1602441256.962790116] [number_publisher]: 4
[INFO] [1602441257.033768159] [number_publisher]: Number Publisher has been started

james@JP-VM:~$ **ros2 param get /number_publisher number_to_publish**
Integer value is: 4

[THIS FUNCTION **INCONSISTENT**!!]
All terminals clear, resourced. Colcon rebuilt.
number_publisher & number_counter running
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 param list**
/number_counter:
  use_sim_time
james@JP-VM:~$ **ros2 param get /number_publisher number_to_publish**
Integer value is: 5
james@JP-VM:~$ **ros2 param list**
/number_publisher:
  number_to_publish
  use_sim_time
james@JP-VM:~$ **ros2 param list**
/number_publisher:
  number_to_publish
  use_sim_time
james@JP-VM:~$ **ros2 param list**
james@JP-VM:~$

But if a parameter is declared & used in the code and it's NOT set in the command line, you WILL get an error as it will be undefined. UNLESS you add a default value in the declaration:

```
        self.declare_parameter("number_to_publish", 2)
```

Add parameter for publish frequency [period = 1 / frequency]:

```
        self.declare_parameter("publish_frequency", 1.0)
        self.publish_frequency_ = self.get_parameter("publish_frequency").value

        self.timer_ = self.create_timer(
         1.0 / self.publish_frequency_, self.publish_number)
```

In CPP:
remove variable default from node definition

```
        NumberPublisherNode() : Node("number_publisher"), number_(2)
```

becomes

```
        NumberPublisherNode() : Node("number_publisher")

        this->declare_parameter("number_to_publish", 1);
```

[default = 1]

```
        number_ = this->get_parameter("number_to_publish").as_int;
```

[as opposed to python dynamic variable typing, cpp requires specification]

add parameter for publish frequency:

```
        this->declare_parameter("publish_frequency", 1.0);
        double publish_frequency = this->get_parameter("publish_frequency").as_double;
```

seconds has to be int. Change to milliseconds to divide by publish_frequency:

```
        number_timer_ = this->create_wall_timer(std::chrono::seconds(1),
```

becomes

```
        number_timer_ = this->create_wall_timer(std::chrono::milliseconds((int) (1000.0 /
publish_frequency)),
```

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_cpp_pkg**

james@JP-VM:~$ **ros2 run my_cpp_pkg number_publisher**
[INFO] [1602451875.670596915] [number_publisher]: Number Publisher has been started.

james@JP-VM:~$ **ros2 param list**
/number_publisher:
  number_to_publish
  publish_frequency
  use_sim_time

james@JP-VM:~$ **ros2 run my_py_pkg number_counter**
[INFO] [1602452007.093880472] [number_counter]: Number Counter has been started.
[INFO] [1602452008.678796876] [number_counter]: 1
[INFO] [1602452009.677505600] [number_counter]: 2

Activity 005 - ROS2 Parameters

1.

Do you remember one of the first node we created in the Topic section, with the robot news radio? This node publishes a string on a topic, similar to this "Hi, this is R2D2 from the Robot News Station!".

Now, it would be better if we could set the robot's name at run time, so we can launch the node multiple times with different robot names.

Add a "robot_name" parameter, and use the value to publish the string on the "robot_news" topic. Your string template ("Hi, this is <robot_name> from the Robot News Station!") will now use the name you set at runtime.

robot_news_station.py:

```
        self.declare_parameter("robot_name", "C3PO")
        self.robot_name_ = self.get_parameter("robot_name").value
```

james@JP-VM:~$ **ros2 run my_py_pkg robot_news_station**
[INFO] [1602453990.921911358] [robot_news_station]: Robot News Station has been started

james@JP-VM:~$ **ros2 topic echo /robot_news**
data: Hi, this is C3PO from the robot news station.
---

james@JP-VM:~$ **ros2 run my_py_pkg robot_news_station --ros-args -p robot_name:="R2D2"**
[INFO] [1602455650.318884817] [robot_news_station]: Robot News Station has been started

james@JP-VM:~$ **ros2 topic echo /robot_news**
data: Hi, this is R2D2 from the robot news station.
---

robot_news_station.cpp:

```
   RobotNewsStationNode() : Node("robot_news_station")
   {
      this->declare_parameter("robot_name", "C3PO");
      robot_name_ = this->get_parameter("robot_name").as_string();
```

2.
Go back to the “led_panel_node”. Here you have an int array representing the states of your LEDs (0 for powered off, 1 for powered on). Set this array with a parameter named “led_states”.

```
        self.declare_parameter("led_states", [0, 0, 0])
        self.led_states_ = self.get_parameter("led_states").value
```

```
james@JP-VM:~/ros2_ws$ colcon build --packages-select my_py_pkg
--symlink-install

james@JP-VM:~$ ros2 run my_py_pkg led_panel
[INFO] [1602456712.681667863] [led_panel]: LED panel node has been started.

james@JP-VM:~$ ros2 topic list
/led_states
/parameter_events
/rosout
james@JP-VM:~$ ros2 param list
/led_panel:
  led_states
  use_sim_time
james@JP-VM:~$ ros2 param get /led_panel led_states
Integer values are: array('q', [0, 0, 0])

james@JP-VM:~$ ros2 topic echo /led_states
led_states:
- 0
- 0
- 0
---

james@JP-VM:~$ ros2 run my_py_pkg led_panel --ros-args -p led_states:=[1,1,1]
[INFO] [1602457000.494116870] [led_panel]: LED panel node has been started.

james@JP-VM:~$ ros2 param get /led_panel led_states
Integer values are: array('q', [1, 1, 1])

^Cjames@JP-VM:~$ ros2 topic echo /led_states
led_states:
- 1
- 1
- 1
---
```

Section 9: Scale Your Application With ROS2 Launch Files
Lesson 84. ROS2 Launch Files
Launch multiple nodes, multiple renames, multiple parameters
Lesson 85. Create and Install a Launch File
number app: number_publisher + number_counter nodes
Where to put the launch files?:
james@JP-VM:~/ros2_ws/src$ **ls**
build install
log my_cpp_pkg
my_py_pkg my_robot_interfaces

Why another pkg?
keep files together, easy access to other pkg, reduce number of dependencies
convention: name_of_robot_bringup
james@JP-VM:~/ros2_ws/src$ **ros2 pkg create my_robot_bringup**
going to create a new package
package name: my_robot_bringup
destination directory: /home/james/ros2_ws/src
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['james <james@todo.todo>']
licenses: ['TODO: License declaration']
build type: ament_cmake
dependencies: []
creating folder ./my_robot_bringup
creating ./my_robot_bringup/package.xml
creating source and include folder
creating folder ./my_robot_bringup/src
creating folder ./my_robot_bringup/include/my_robot_bringup
creating ./my_robot_bringup/CMakeLists.txt

james@JP-VM:~/ros2_ws/src$ **ls**
build install log my_cpp_pkg my_py_pkg my_robot_bringup my_robot_interfaces
james@JP-VM:~/ros2_ws/src$ **cd my_robot_bringup/**
james@JP-VM:~/ros2_ws/src/my_robot_bringup$ **ls**
CMakeLists.txt include package.xml src

Assumes ament cpp type package. Not all parts needed:
james@JP-VM:~/ros2_ws/src/my_robot_bringup$ **rm -rf include/**
james@JP-VM:~/ros2_ws/src/my_robot_bringup$ **rm -rf src/**
james@JP-VM:~/ros2_ws/src/my_robot_bringup$ **mkdir launch**
james@JP-VM:~/ros2_ws/src/my_robot_bringup$ **ls**
CmakeLists.txt launch package.xml

configure pkg files:
package.xml: [no changes until launch files created]

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>my_robot_bringup</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="james@todo.todo">james</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>

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

CmakeList.txt:

```
cmake_minimum_required(VERSION 3.5)
project(my_robot_bringup)

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

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  # the following line skips the linter which checks for copyrights
  # uncomment the line when a copyright and license is not present in all source files
  #set(ament_cmake_copyright_FOUND TRUE)
  # the following line skips cpplint (only works in a git repo)
  # uncomment the line when this package is not in a git repo
  #set(ament_cmake_cpplint_FOUND TRUE)
  ament_lint_auto_find_test_dependencies()
endif()

install(DIRECTORY
  launch
  DESTINATION share/${PROJECT_NAME}
)

ament_package()
```

becomes...

CMakeList.txt:

```
cmake_minimum_required(VERSION 3.5)
project(my_robot_bringup)

# 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)

install(DIRECTORY
  launch
  DESTINATION share/${PROJECT_NAME}
)

ament_package()
```

Create first launch file:
james@JP-VM:~/ros2_ws/src/my_robot_bringup$ **cd launch/**
james@JP-VM:~/ros2_ws/src/my_robot_bringup/launch$ **touch number_app.launch.py**
james@JP-VM:~/ros2_ws/src/my_robot_bringup/launch$ **chmod +x number_app.launch.py**

create minimal template launch file:

number_app.launch.py:

```
from launch import LaunchDescription

def generate_launch_description():
    ld = LaunchDescription()

return ld
```

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_bringup --symlink-install**

[2]
james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~$ **ros2 launch my_robot_bringup number_app.launch.py**
[INFO] [launch]: All log files can be found below /home/james/.ros/log/2020-10-11-19-36-28-857869-JP-VM-9571
[INFO] [launch]: Default logging verbosity is set to INFO

Add node(s) to launch file:
number_app.launch.py:

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    ld = LaunchDescription()

    number_publisher_node = Node(
        package="my_py_pkg",
        executable="number_publisher"
    )

    number_counter_node = Node(
        package="my_cpp_pkg",
        executable="number_counter"
    )

    ld.add_action(number_publisher_node)
    ld.add_action(number_counter_node)
    return ld
```

Add dependencies to **package.xml**:

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>my_robot_bringup</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="james@todo.todo">james</maintainer>
  <license>TODO: License declaration</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

  <exec_depend>my_py_pkg</exec_depend>
  <exec_depend>my_cpp_pkg</exec_depend>


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

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

james@JP-VM:~$ **ros2 launch my_robot_bringup number_app.launch.py**
[INFO] [launch]: All log files can be found below
/home/james/.ros/log/2020-10-11-19-53-07-647411-JP-VM-9663
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [number_publisher-1]: process started with pid [9665]
[INFO] [number_counter-2]: process started with pid [9667]
[number_counter-2] [INFO] [1602463988.722947570] [number_counter]: Number Counter has been started.
[number_publisher-1] [INFO] [1602463990.301241157] [number_publisher]: 1
[number_publisher-1] [INFO] [1602463990.346402144] [number_publisher]: Number Publisher has been started
^C [killing launch file kills all listed nodes]
[restart]

james@JP-VM:~$ **ros2 node list**
/number_counter
/number_publisher

Lesson 86. Configure Your Nodes in a Launch File
A launch file can also add parameters, rename nodes, remap topics, etc.

RENAME NODES:
number_app.launch.py:
add to each node description --
```
        name="my_number_publisher"
        name="my_number_counter"
```

james@JP-VM:~$ **ros2 launch my_robot_bringup number_app.launch.py**
[INFO] [launch]: All log files can be found below
/home/james/.ros/log/2020-10-12-20-45-43-961561-JP-VM-2516
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [number_publisher-1]: process started with pid [2518]
[INFO] [number_counter-2]: process started with pid [2520]
[number_counter-2] [INFO] [1602553545.518885127] **[my_number_counter]**: Number Counter has been started.
[number_publisher-1] [INFO] [1602553546.687505807] [my_number_publisher]: 1
[number_publisher-1] [INFO] [1602553546.741567148] **[my_number_publisher]**: Number Publisher has been started

james@JP-VM:~$ **ros2 node list**
/my_number_publisher [sometimes doesn't work fully, try again]
james@JP-VM:~$ **ros2 node list**
/my_number_counter
/my_number_publisher

REMAP TOPICS:
james@JP-VM:~$ **ros2 topic list**
/number
/number_count
/parameter_events
/rosout

```
                ,                [end line above with a comma]
        remappings=[
                ("number", "my_number")
        ]
                ,
        remappings=[
                ("number", "my_number"),
                ("number_count", "my_number_count")
        ]
```

or global remap with variable tuple:

```
        remap_number_topic = ("number", "my_number")
        remappings=[
                remap_number_topic
        ]
        remappings=[
                remap_number_topic,
                ("number_count", "my_number_count")
        ]
```

```
james@JP-VM:~$ ros2 topic list
/my_number
/my_number_count
/parameter_events
/rosout
james@JP-VM:~$ ros2 topic echo /my_number_count
data: 109
---
data: 110
```

RENAME SERVICE NAMES

same as rename topic names

ADD PARAMETERS

```
        parameters=[
                {"number_to_publish": 4},
                {"publish_frequency": 5.0}
        ]
```

```
james@JP-VM:~$ ros2 launch my_robot_bringup number_app.launch.py
james@JP-VM:~$ ros2 topic echo /my_number_count
data: 424
---
data: 428
---
james@JP-VM:~$ ros2 topic hz /my_number_count
WARNING: topic [/my_number_count] does not appear to be published yet
average rate: 4.995
        min: 0.196s max: 0.203s std dev: 0.00217s window: 6
```

Since launch file is written in python -

full use of python logic, if's, loops, etc may be used

number_app.launch.py:

```
from launch import LaunchDescription
from launch_ros.actions import Node
def generate_launch_description():
    ld = LaunchDescription()
    remap_number_topic = ("number", "my_number")
    number_publisher_node = Node(
        package="my_py_pkg",
        executable="number_publisher",
        name="my_number_publisher",
        remappings=[
            remap_number_topic
        ],
        parameters=[
            {"number_to_publish": 4},
            {"publish_frequency": 5.0}
        ]
    )
    number_counter_node = Node(
        package="my_cpp_pkg",
        executable="number_counter",
        name="my_number_counter",
        remappings=[
            remap_number_topic,
            ("number_count", "my_number_count")
        ]
    )
    ld.add_action(number_publisher_node)
    ld.add_action(number_counter_node)
    return ld
```

CMakeLists.txt:

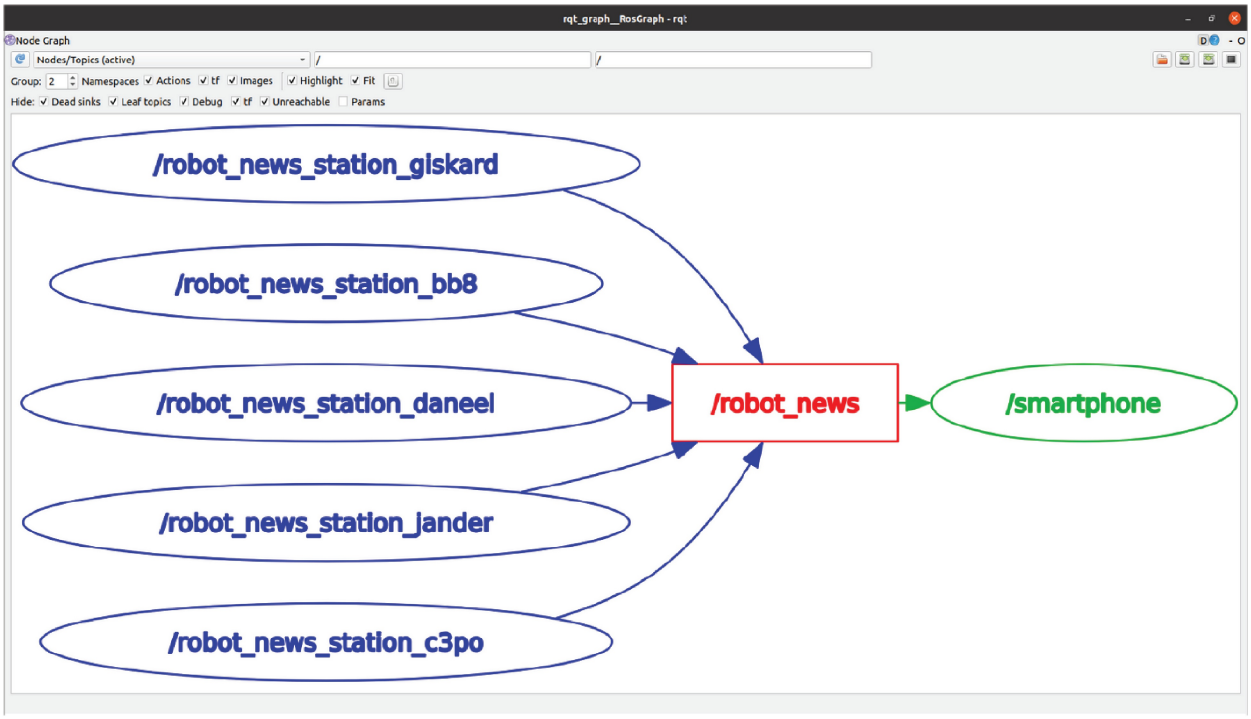
```
cmake_minimum_required(VERSION 3.5)
project(my_robot_bringup)
# Default to C++14
if(NOT CMAKE_CXX_STANDARD)
  set(CMAKE_CXX_STANDARD 14)
endif()
if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES
"Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()
# find dependencies
find_package(ament_cmake REQUIRED)
install(DIRECTORY
  launch
  DESTINATION share/${PROJECT_NAME}
)
ament_package()
```

package.xml:

```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>my_robot_bringup</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="james@todo.todo">james</maintainer>
  <license>TODO: License declaration</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

  <exec_depend>my_py_pkg</exec_depend>
  <exec_depend>my_cpp_pkg</exec_depend>

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

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

Activity 006 - ROS2 Launch Files

In this activity you will practice by creating a new launch file, with the nodes that you’e already
created during this course.
Goal:
●Start 5 “obot_news_station”nodes and 1 smartphone node.
●Each “obot_news_station”will need a different name, and will publish "Hi, this is <robot_name> from the Robot News Station!"
●The “martphone”node gets all the messages from all other nodes.
Here’ the graph you should get:



So, no need to create or modify any node here. You just need to create a single launch file to
start your (renamed) nodes with parameters.
Bonus point if you find the book/movie reference for all robot names! (shouldn’ be too hard)

2020.10.24 Home

/james@JP-VM:~/ros2_ws/src/my_robot_bringup/launch$ **cd /home/james/ros2_ws/src/my_robot_bringup/launch**
james@JP-VM:~/ros2_ws/src/my_robot_bringup/launch$ **ls**
number_app.launch.py
james@JP-VM:~/ros2_ws/src/my_robot_bringup/launch$ **touch robot_news_app.launch.py**
james@JP-VM:~/ros2_ws/src/my_robot_bringup/launch$ **chmod -x robot_news_app.launch.py**

robot_news_app.launch.py:

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    ld = LaunchDescription()

    robot_news_station_giskard_node = Node(
        package="my_py_pkg",
        executable="robot_news_station",
        name="robot_news_station_giskard",
        parameters=[
            {"robot_name": "giskard"},
            {"period": 1.0}
        ]
    )
```

in repeat blocks of same code, change “giskard” to “bb8", “daneel”, “jander”, “c3po”. Change “period”: to 2.1, 3.2, 4.3, 5.4 to avoid interference. [not sure if this worked]

```
    smartphone_node = Node(
        package="my_py_pkg",
        executable="smartphone",
        name="smartphone"
    )

    ld.add_action(robot_news_station_giskard_node)
    ld.add_action(robot_news_station_bb8_node)
    ld.add_action(robot_news_station_daneel_node)
    ld.add_action(robot_news_station_jander_node)
    ld.add_action(robot_news_station_c3po_node)
    ld.add_action(smartphone_node)
    return ld
```

robot_news_station.py (parameter added):

```
self.declare_parameter("period", 1.0)
self.period_ = self.get_parameter("period").value
self.timer_ = self.create_timer(self.period_, self.publish_news)
```

All 5 nodes are active:

```
james@JP-VM:~$ ros2 node list
/robot_news_station_bb8
/robot_news_station_c3po
/robot_news_station_daneel
/robot_news_station_giskard
/robot_news_station_jander
/rqt_gui_py_node_4982
/smartphone
```

smartphone is receiving all 5 news stations:

```
james@JP-VM:~/ros2_ws$ ros2 launch my_robot_bringup
robot_news_app.launch.py
...
[smartphone-6] [INFO] [1603591780.871657582] [smartphone]: Hi, this is jander from
the robot news station.
[smartphone-6] [INFO] [1603591781.331681723] [smartphone]: Hi, this is giskard from
the robot news station.
[smartphone-6] [INFO] [1603591782.337478342] [smartphone]: Hi, this is giskard from
the robot news station.
[smartphone-6] [INFO] [1603591782.824964776] [smartphone]: Hi, this is bb8 from the
robot news station.
[smartphone-6] [INFO] [1603591782.948328924] [smartphone]: Hi, this is c3po from the
robot news station.
[smartphone-6] [INFO] [1603591783.332280540] [smartphone]: Hi, this is giskard from
the robot news station.
[smartphone-6] [INFO] [1603591783.937481160] [smartphone]: Hi, this is daneel from
the robot news station.
[smartphone-6] [INFO] [1603591784.348304077] [smartphone]: Hi, this is giskard from
the robot news station.
```

However, only 1 shows up on echo: [sometimes two, never all 5]

```
james@JP-VM:~$ ros2 topic echo /robot_news
data: Hi, this is bb8 from the robot news station.
data: Hi, this is bb8 from the robot news station.
data: Hi, this is bb8 from the robot news station.
data: Hi, this is bb8 from the robot news station.
```

Only 2 show up on Node Graph: [this or some other combination in another run]
..._giskard ..._jander The smartphone node doesn't show up on the graph.

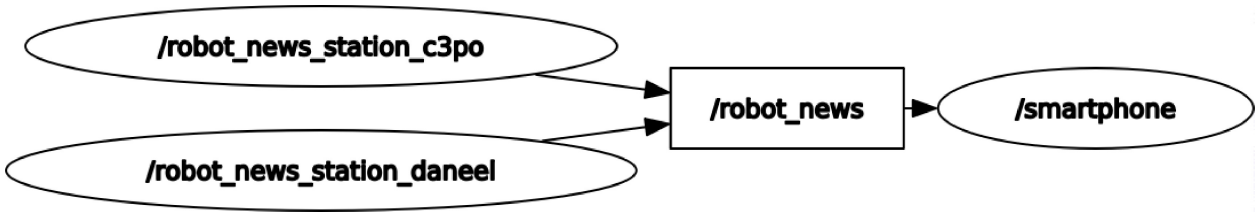
2020.10.25
All terminal windows: james@JP-VM:~$ **source ~/.bashrc**
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_bringup --symlink-install**

james@JP-VM:~$ **ros2 launch my_robot_bringup robot_news_app.launch.py**
[INFO] [launch]: All log files can be found below /home/james/.ros/log/2020-10-25-16-26-34-754545-JP-VM-2468
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [robot_news_station-1]: process started with pid [2470]
[INFO] [robot_news_station-2]: process started with pid [2472]
[INFO] [robot_news_station-3]: process started with pid [2474]
[INFO] [robot_news_station-4]: process started with pid [2476]
[INFO] [robot_news_station-5]: process started with pid [2479]
[INFO] [smartphone-6]: process started with pid [2481]
[robot_news_station-5] [INFO] [1603661204.808124652] [robot_news_station_c3po]: Robot News Station has been started
[robot_news_station-3] [INFO] [1603661204.809685703] [robot_news_station_daneel]: Robot News Station has been started
[robot_news_station-4] [INFO] [1603661204.816925333] [robot_news_station_jander]: Robot News Station has been started
[robot_news_station-1] [INFO] [1603661204.822692692] [robot_news_station_giskard]: Robot News Station has been started
[robot_news_station-2] [INFO] [1603661204.840933023] [robot_news_station_bb8]: Robot News Station has been started
[smartphone-6] [INFO] [1603661204.930741503] [smartphone]: Smartphone has been started.
[smartphone-6] [INFO] [1603661205.333147067] [smartphone]: Hi, this is giskard from the robot news station.
[smartphone-6] [INFO] [1603661206.332123531] [smartphone]: Hi, this is giskard from the robot news station.
[smartphone-6] [INFO] [1603661206.467526119] [smartphone]: Hi, this is bb8 from the robot news station.
[smartphone-6] [INFO] [1603661207.347433760] [smartphone]: Hi, this is giskard from the robot news station.
[smartphone-6] [INFO] [1603661207.541579866] [smartphone]: Hi, this is daneel from the robot news station.
[smartphone-6] [INFO] [1603661208.340108023] [smartphone]: Hi, this is giskard from the robot news station.
[smartphone-6] [INFO] [1603661208.561857866] [smartphone]: Hi, this is bb8 from the robot news station.
[smartphone-6] [INFO] [1603661208.638215665] [smartphone]: Hi, this is jander from the robot news station.
[smartphone-6] [INFO] [1603661209.338771415] [smartphone]: Hi, this is giskard from the robot news station.
[smartphone-6] [INFO] [1603661209.747885229] [smartphone]: Hi, this is c3po from the robot news station.

james@JP-VM:~$ **ros2 node list**
/robot_news_station_c3po
/robot_news_station_daneel
/smartphone

james@JP-VM:~$ **ros2 topic list**
/parameter_events
/robot_news
/rosout

james@JP-VM:~$ **ros2 topic echo /robot_news**
data: Hi, this is daneel from the robot news station.
---
data: Hi, this is daneel from the robot news station.
---
data: Hi, this is c3po from the robot news station.
---
data: Hi, this is daneel from the robot news station.
---
data: Hi, this is daneel from the robot news station.
---
data: Hi, this is c3po from the robot news station.



Activity 006 Solution:
Changed code to create nodes from list

robot_news_app.launch.py:

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    ld = LaunchDescription()

    robot_names = ["Giskard", "BB8", "Daneel", "Jander", "C3PO"]

    robot_news_station_nodes = []

    for name in robot_names:
        robot_news_station_nodes.append(Node(
            package="my_py_pkg",
            executable="robot_news_station",
            name="robot_news_station_" + name.lower(),
            parameters=[{"robot_name": name}]
        ))

    smartphone_node = Node(
        package="my_py_pkg",
        executable="smartphone",
        name="smartphone"
    )

    for node in robot_news_station_nodes:
        ld.add_action(node)
    ld.add_action(smartphone_node)
    return ld
```

Still only shows partial lists of nodes.

2020.11.02

Hi Jim,

I haven't seen your complete launch file, but I guess maybe the error comes from the line name="robot_news_station_giskard",

In the explanation you say that you change the name of the node (i.e the name of the object you create when you do "node_name = Node..."), but maybe you didn't change the name inside the node --> which in fact will create new nodes with the same name. With this situation, your program may "work", but all of the introspection tools (rqt, cmd line, etc) will display wrong info.

Let me know if it wasn't this, and in this case having the complete launch file would help me find the error.

Best regards,
Edouard

james@JP-VM:~$ **colcon build --packages-select my_robot_bringup --symlink-install**

james@JP-VM:~$ **ros2 topic list**
/parameter_events
/robot_news
/rosout

james@JP-VM:~$ **ros2 node list**
/robot_news_station_giskard
/robot_news_station_jander
/smartphone
james@JP-VM:~$ **ros2 node list**
/robot_news_station_bb8
/robot_news_station_daneel
/robot_news_station_jander
james@JP-VM:~$ **ros2 node list**
/robot_news_station_c3po
/robot_news_station_giskard
/robot_news_station_jander

2020.11.04

Turtlesim “Catch Them All” project
Create a project to:
launch the turtle environment
spawn the main “predator” turtle
spawn a “prey” turtle
direct the “predator” turtle to the “prey” turtle and eat it.
swawn additional turtles at reasonably fair random intervals and at random positions and orientations.
Continue to hunt and eat the new “prey” turtles
The example shows the “predator” turtle heading for the closest, not the oldest “prey” turtle but not sure this is important.
It is important to stay focused on the target “prey” turtle and not get distracted by newer nearby turtles.

2020.11.05

see: **ROS2+For+Beginners+-+Turtlesim+Project.pdf** for more details and tips.

james@JP-VM:~$ **ros2 run turtlesim turtlesim_node**
james@JP-VM:~$ **ros2 run turtlesim turtle_teleop_key**
james@JP-VM:~$ **ros2 node list**
/turtlesim
james@JP-VM:~$ **ros2 topic list**
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose

james@JP-VM:~$ **ros2 topic info /turtle1/cmd_vel**
Type: geometry_msgs/msg/Twist
Publisher count: 0
Subscription count: 1

james@JP-VM:~$ **ros2 node info /turtlesim**
/turtlesim
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/color_sensor: turtlesim/msg/Color
    /turtle1/pose: turtlesim/msg/Pose
  Service Servers:
    /clear: std_srvs/srv/Empty
    /kill: turtlesim/srv/Kill
    /reset: std_srvs/srv/Empty
    /spawn: turtlesim/srv/Spawn
    /turtle1/set_pen: turtlesim/srv/SetPen
    /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
    /turtle1/teleport_relative: turtlesim/srv/TeleportRelative
    /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
    /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
    /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
    /turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
  Action Clients:

Couldn't figure out proper format for ros2 topic pub ...
Found help here:
https://index.ros.org/doc/ros2/Tutorials/Topics/Understanding-ROS2-Topics/#ros2-topic-pub
...which successfully got me here:

james@JP-VM:~$ **ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 1.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"**
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))

...resulting in the turtle moving 1 unit forward.

Other variants have turtle moving in an arc:

james@JP-VM:~$ **ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 1.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.0}}"**
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=1.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=1.0))

change **- - once** to **r- <number>** and the turtle will turn in a circle

^Cjames@JP-VM:~$ **ros2 topic pub -r 10 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 1.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.5}}"**

2020.11.08

Lesson 93: Project Solution 1/6

Launch the Master turtle node

james@JP-VM:~$ **ros2 run turtlesim turtlesim_node**
[INFO] [1604874124.239745888] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1604874124.318822585] [turtlesim]: Spawning turtle [**turtle1**] at x=[5.544445], y=[5.544445], theta=[0.000000]

james@JP-VM:~$ **ros2 topic list**
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose

james@JP-VM:~$ **ros2 topic echo /turtle1/pose**
x: 5.544444561004639
y: 5.544444561004639
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0

Create a new package for Turtlesim Catch Them All:
james@JP-VM:~$ **cd ros2_ws/src/**
james@JP-VM:~/ros2_ws/src$ **ls**
build my_cpp_pkg my_robot_interfaces
install my_py_pkg
log my_robot_bringup
james@JP-VM:~/ros2_ws/src$ **ros2 pkg create turtlesim_catch_them_all --build-type ament_python**

Defer dependency declarations to package itself.
Enter the package & the source directory within:
james@JP-VM:~/ros2_ws/src$ **cd turtlesim_catch_them_all/turtlesim_catch_them_all/**

Create the python program & make it executable:
james@JP-VM:~/ros2_ws/src/turtlesim_catch_them_all/turtlesim_catch_them_all$ **touch turtle_controller.py**
james@JP-VM:~/ros2_ws/src/turtlesim_catch_them_all/turtlesim_catch_them_all$ **chmod +x turtle_controller.py**

Copy python template.
Change Node name to **TurtleControllerNode**
Change name to **turtle_controller**
Add to package.xml **<depend>rclpy</depend>**

Obtain published topic info:
james@JP-VM:~$ **ros2 topic info /turtle1/pose**
Type: **turtlesim/msg/Pose**
Publisher count: 1
Subscription count: 0

james@JP-VM:~$ **ros2 interface show turtlesim/msg/Pose**
float32 x
float32 y
float32 theta

float32 linear_velocity
float32 angular_velocity

Create subscriber:
from turtlesim.msg import Pose

Add turtlesim pkg to package.xml:
<depend>turtlesim</depend>

Create subscriber and Callback function for **turtle1/Pose**:

```
class TurtleControllerNode(Node):
    def __init__(self):
        super().__init__("turtle_controller")
        self.pose_ = None
        self.pose_subscriber_ = self.create_subscription(
            Pose, "turtle1/pose", self.callback_turtle_pose 10)

    def callback_turtle_pose(self, msg):
        self.pose_ = msg
```

Create publisher for **turtle1/cmd_vel**:

```
james@JP-VM:~$ ros2 topic info /turtle1/cmd_vel
Type: geometry_msgs/msg/Twist
Publisher count: 0
Subscription count: 1

james@JP-VM:~$ ros2 interface show geometry_msgs/msg/Twist
# This expresses velocity in free space broken into its linear and angular parts.

Vector3  linear
Vector3  angular
```

To further define Vector3 (in same pkg since no other listed):

```
james@JP-VM:~$ ros2 interface show geometry_msgs/msg/Vector3
# This represents a vector in free space.

# This is semantically different than a point.
# A vector is always anchored at the origin.
# When a transform is applied to a vector, only the rotational component is applied.

float64 x
float64 y
float64 z
```

Import msg format:
from geometry_msgs.msg import Twist

Add dependency to package.xml:

```
<depend>geometry_msgs</depend>
```

```
self.cmd_vel_publisher_ = self.create_publisher(
   Twist, "turtle/cmd_vel", 10)
```

Add timer and control loop to control turtle:

```
self.control_loop_timer_ = self.create_timer(0.01, self.control_loop)
```

Create practice target (0.0, 0.0 is LLQ. Upper Right is 11.0, 11.0):

```
self.target_x = 8.0
self.target_y = 4.0
```

Compute distance from turtle1 to Target:

```
import math
dist_x =self.target_x - self.target_x
dist_y =self.target_y - self.target_y
distance = math.sqrt(dist_ * dist_x + dist_y * dist_y)
```

Create Twist msg with velocities:
linear velocity only needs to be x – the forward direction of the turtle
angular velocity only need to by z – the angle in the place of the turtle

```
msg = Twist()
# some stuff
self.cmd_vel_publisher_.publish(msg)
```

some stuff:
[distance < 0.5 is close enough to “catch” the target turtles]

```
msg = Twist()
if distance > 0.5:          [keep going]
    pass
else:                       [stop]
    msg.linear.x = 0.0
    msg.angular.z = 0.0
```

pass becomes...

 # position
Set forward movement to the distance to the target
msg.linear.x = distance
 # orientation
angles are in RADIANS
tan A = opposite / adjacent = dist_y / dist_x
angle A = arctan(opposite / adjacent) = arctan(dist_y / dist_x)

```
goal_theta = math.atan2(dist_y, dist_x)
diff = goal_theta - self.pose_.theta
```

normalize angle if > pi or < -pi by subtracting or adding 2pi
if diff > math.pi:
    diff -= 2*math.pi
elif diff < -math.pi:
    diff += 2*math.pi
Change turtle angle by the difference between current and goal angle
msg.angular.z = diff

Add constants for Proportional control [not explained but derived experimentally]
Tutorial uses linear constant of 2*distance and angular constant of 6*diff
I'm going to make these variables initially set to 1.0 to see how it changes the behavior of the system.

Add to setup.py / console scripts:
"turtlesim_controller = turtlesim_catch_them_all.turtle_controller:main"

Compile:
james@JP-VM:~/ros2_ws$ **colcon build --packages-select turtlesim_catch_them_all --symlink-install**
Starting >>> turtlesim_catch_them_all
Finished <<< turtlesim_catch_them_all [4.96s]

Summary: 1 package finished [5.65s]

james@JP-VM:~$ **ros2 run turtlesim turtlesim_node**
[INFO] [1604892963.036685165] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1604892963.054697557] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

james@JP-VM:~$ **ros2 run turtlesim_catch_them_all turtlesim_controller**
    does nothing
james@JP-VM:~/ros2_ws/src/turtlesim_catch_them_all/turtlesim_catch_them_all$ **python3 turtle_controller.py**
    does nothing

...hang it up for the night.

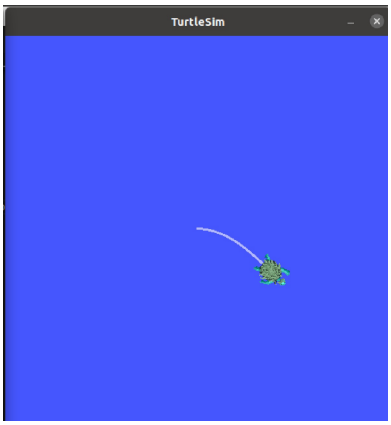
2020.11.09
Issue solved:
    Twist, "turtle/cmd_vel", 10)
should have been
    Twist, "turtle**1**/cmd_vel", 10)

turtlesim_catch_them_all/turtlesim_controller:

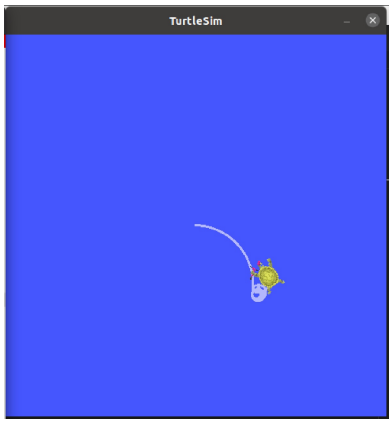
```
#!/usr/bin/env python3
import math
import rclpy
from rclpy.node import Node

from turtlesim.msg import Pose
from geometry_msgs.msg import Twist

class TurtleControllerNode(Node):
    def __init__(self):
        super().__init__("turtle_controller")
        self.target_x = 8.0
        self.target_y = 4.0
        self.linear_constant = 2.0
        self.angular_constant = 6.0

        self.pose_ = None
        self.cmd_vel_publisher_ = self.create_publisher(
            Twist, "turtle1/cmd_vel", 10)
        self.pose_subscriber_ = self.create_subscription(
            Pose, "turtle1/pose", self.callback_turtle_pose, 10)
        self.control_loop_timer_ = self.create_timer(0.01, self.control_loop)

    def callback_turtle_pose(self, msg):
        self.pose_ = msg
    def control_loop(self):
        if self.pose_ == None:
            return

        dist_x = self.target_x - self.pose_.x
        dist_y = self.target_y - self.pose_.y
        distance = math.sqrt(dist_x * dist_x + dist_y * dist_y)

        msg = Twist()
        if distance > 0.5:
            # position
            msg.linear.x = distance * self.linear_constant
            # orientation
            goal_theta = math.atan2(dist_y, dist_x)
            diff = goal_theta - self.pose_.theta
            if diff > math.pi:
                diff -= 2*math.pi
            elif diff < -math.pi:
                diff += 2*math.pi
            msg.angular.z = diff * self.angular_constant
        else:
```

```
            msg.linear.x = 0.0
            msg.angular.z = 0.0

        self.cmd_vel_publisher_.publish(msg)

def main(args=None):
    rclpy.init(args=args)
    node = TurtleControllerNode()
    rclpy.spin(node)
    rclpy.shutdown()


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

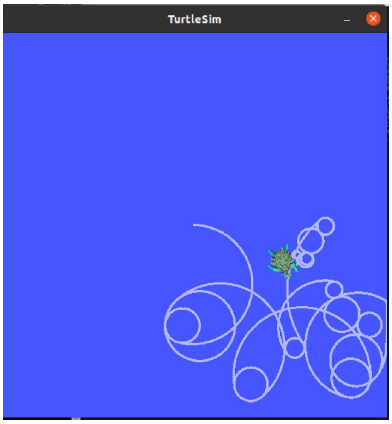
Experiments:
decrease the publish interval from 0.01 to 1.0 to observe step-by-step
(decr to 0.1 didn't change it much)



Pub 0.01

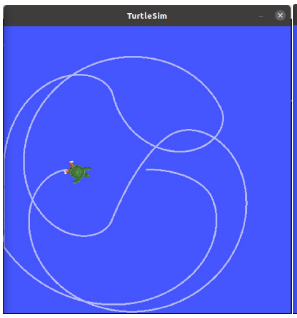


Pub 0.5

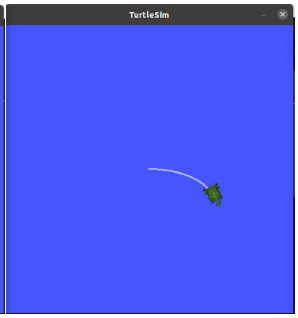


Pub 1.0

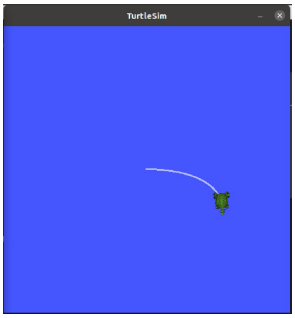
As the pub interval increases, the turtle takes longer to update its trajectory
Change the linear_constant (Kl) and angular_constant (Ka) to 1.0 & titrate:



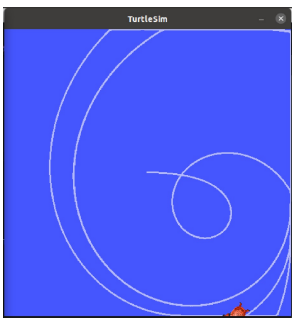
Kl 1.0 Ka 1.0
Multiple targets



Kl 1.0 Ka 1.0



Kl 1.5 Ka 1.0



Kl 2.0 Ka 1.0

2020.11.12

james@JP-VM:~$ **ros2 node list**
/turtlesim
james@JP-VM:~$ **ros2 node info /turtlesim**
/turtlesim
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Publishers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/color_sensor: turtlesim/msg/Color
    /turtle1/pose: turtlesim/msg/Pose
  Service Servers:
    /clear: std_srvs/srv/Empty
    /kill: turtlesim/srv/Kill
    /reset: std_srvs/srv/Empty
    /spawn: turtlesim/srv/Spawn
    /turtle1/set_pen: turtlesim/srv/SetPen
    /turtle1/teleport_absolute: turtlesim/srv/TeleportAbsolute
    /turtle1/teleport_relative: turtlesim/srv/TeleportRelative
    /turtlesim/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /turtlesim/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /turtlesim/get_parameters: rcl_interfaces/srv/GetParameters
    /turtlesim/list_parameters: rcl_interfaces/srv/ListParameters
    /turtlesim/set_parameters: rcl_interfaces/srv/SetParameters
    /turtlesim/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:
  Action Servers:
    /turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
  Action Clients:

james@JP-VM:~$ **ros2 service list**
/clear
/kill
/reset
/spawn
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/describe_parameters
/turtlesim/get_parameter_types
/turtlesim/get_parameters
/turtlesim/list_parameters
/turtlesim/set_parameters
/turtlesim/set_parameters_atomically

james@JP-VM:~$ **ros2 service type /kill**
turtlesim/srv/Kill
james@JP-VM:~$ **ros2 service type /spawn**
turtlesim/srv/Spawn

james@JP-VM:~$ **ros2 interface show turtlesim/srv/Kill**
string name
---

james@JP-VM:~$ **ros2 interface show turtlesim/srv/Spawn**
float32 x
float32 y
float32 theta
string name # Optional. A unique name will be created and returned if this is empty
---
string name

james@JP-VM:~$ **ros2 service call /kill turtlesim/srv/Kill "{name: turtle1}"**
requester: making request: turtlesim.srv.Kill_Request(name='turtle1')

response:
turtlesim.srv.Kill_Response()

turtle1 disappears

james@JP-VM:~$ **ros2 service call /spawn turtlesim/srv/Spawn "{x: 5.5, y: 5.5, theta: 0.0}"**
waiting for service to become available...
requester: making request: turtlesim.srv.Spawn_Request(x=5.5, y=5.5, theta=0.0, name='')
response:
turtlesim.srv.Spawn_Response(name='turtle2')

turtle2 appears mid screen (5.5, 5.5) facing right (theta = 0.0)

90.0o = 1.57 radians
james@JP-VM:~$ **ros2 service call /spawn turtlesim/srv/Spawn "{x: 3.0, y: 3.0, theta: 1.57}"**
waiting for service to become available...
requester: making request: turtlesim.srv.Spawn_Request(x=3.0, y=3.0, theta=1.57, name='')
response:
turtlesim.srv.Spawn_Response(name='turtle3')

turtle3 appears lower right (3.0, 3.0) facing up (theta = 1.57)

2020.11.14

turtle_spawner:
Define parameter of number of turtles to spawn and frequency of spawing (possibly random w/in parameters)
Call /spawn service to spawn a new turtle at a random location [0-11.0, 0-11.0, 0.0].
Accept default of theta = 0.0 or provide random theta 0 - 2pi. [2 * pi * degrees/360]
Provide no name for turtle but accept name in response of /spawn service.
Get response of new turtle's name (turtleN).
Add turtleN's name and pose.x and pose.y to the /alive_turtles list
(only 1 target turtle to start).
Publish /alive_turtles topic with name & pose of living turtle(s).

turtle_controller:
Subscribe to active_turtles topic
Make the first turtle on the active_turtles list the target_turtle noting name and pose
Plot & follow course to target_turtle
When within tolerance distance, "catch" target_turtle by calling catch_service with turtleN name

turtle_spawner:
Provide "catch" service.
Request is turtleN name.
Service is
    1) remove turtleN from active_turtle list
    2) remove turtleN from board by requesting /kill service with turtleN name
    3) republish active_turtle list
Response is True

Create turtle_spawner node:
In Visual Code Studio right click tab for turtle_controller and copy path.
In Terminator $ cd and paste the path. Delete turtle_controller.py. Enter.

```
james@JP-VM:~$ cd
/home/james/ros2_ws/src/turtlesim_catch_them_all/turtlesim_catch_them_all/
james@JP-VM:~/ros2_ws/src/turtlesim_catch_them_all/turtlesim_catch_them_all$ ls
__init__.py  __pycache__  turtle_controller.py
james@JP-VM:~/ros2_ws/src/turtlesim_catch_them_all/turtlesim_catch_them_all$
touch turtle_spawner.py
james@JP-VM:~/ros2_ws/src/turtlesim_catch_them_all/turtlesim_catch_them_all$
chmod +x turtle_spawner.py
james@JP-VM:~/ros2_ws/src/turtlesim_catch_them_all/turtlesim_catch_them_all$ ls -l
total 8
-rw-rw-r-- 1 james james    0 Nov  8 16:35 __init__.py
drwxrwxr-x 2 james james 4096 Nov 12 19:47 __pycache__
-rwxrwxr-x 1 james james 1749 Nov 12 19:48 turtle_controller.py
-rwxrwxr-x 1 james james    0 Nov 14 15:47 turtle_spawner.py
```

first attempt –
turtle_spawner.py:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from functools import partial
import math
from turtlesim.msg import Pose
from turtlesim.srv import Spawn
from turtlesim.srv import Kill

class TurtleSpawnerNode(Node):
    def __init__(self):
        super().__init__("turtle_spawner")

    def call_turtle_spawn_server(self, x, y, theta):
        client = self.create_client(Spawn, "spawn")
        while not client.wait_for_service(1.0):
            self.get_logger().warn("Waiting for Server Spawn...")

        request = Spawn.Request()
        request.x = 7.5
        request.y = 2.5
        request.theta = 0.0

        future = client.call_async(request)
        future.add_done_callback(partial(self.callback_call_turtle_spawn, x=x, y=y,
theta=theta))

    def callback_call_turtle_spawn(self, future, x, y, theta):
        try:
            response = future.result()
            self.get_logger().info(str(request.x) + ", " + str(request.y) + ", ", str(request.theta)
+ ", " + response.name)
        except Exception as e:
            self.get_logger().error("Service call failed %r" % (e, ))

def main(args=None):
    rclpy.init(args=args)
    node = TurtleSpawnerNode()
    rclpy.spin(node)
    rclpy.shutdown()


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

james@JP-VM:~/ros2_ws/src$ **colcon build --packages-select turtlesim_catch_them_all --symlink-install**

james@JP-VM:~/ros2_ws/src$ **ros2 run turtlesim turtlesim_node**
[INFO] [1605404561.593174421] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1605404561.741428582] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

turtle appears center screen facing right/East

james@JP-VM:~$ **ros2 run turtlesim_catch_them_all turtle_spawner**
No executable found

james@JP-VM:~/ros2_ws/src/turtlesim_catch_them_all/turtlesim_catch_them_all$ **python3 turtle_spawner.py**

No error message but no response on board nor output

james@JP-VM:~/ros2_ws/src/install/turtlesim_catch_them_all/lib/turtlesim_catch_them_all$ **ls**
turtlesim_controller

No executable for turtle_spawner

setup.py:

```
...
'console_scripts': [
        "turtlesim_controller = turtlesim_catch_them_all.turtle_controller:main",
        "turtle_spawner = turtlesim_catch_them_all.turtle_spawner:main"
    ],
```

Enough for today. Will have to cheat and see Solution 2 tomorrow.

2020.11.15
Project Solution 2/6:
Create a turtle spawner that will spawn turtles at a given rate in random poses
turtle_spawner.py:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from functools import partial
import random
import math
from turtlesim.srv import Spawn

class TurtleSpawner(Node):
    def __init__(self):
        super().__init__("turtle_spawner")
        self.turtle_name_prefix_ = "turtle"
        self.turtle_counter_ = 0
        self.spawn_turtle_timer_ = self.create_timer(2.0, self.spawn_new_turtle)

    def spawn_new_turtle(self):
        self.turtle_counter_ += 1
        name = self.turtle_name_prefix_ + str(self.turtle_counter_)
        x = random.uniform(0.0, 11.0)
        y = random.uniform(0.0, 11.0)
        theta = random.uniform(0.0, 2*math.pi)
        self.call_spawn_server(name, x, y, theta)

    def call_spawn_server(self, turtle_name, x, y, theta):
        client = self.create_client(Spawn, "spawn")
        while not client.wait_for_service(1.0):
            self.get_logger().warn("Waiting for Server Spawn...")

        request = Spawn.Request()
        request.x = x
        request.y = y
        request.theta = theta
        request.name = turtle_name

        future = client.call_async(request)
        future.add_done_callback(
            partial(self.callback_call_spawn, turtle_name=turtle_name, x=x, y=y,
theta=theta))

    def callback_call_spawn(self, future, turtle_name, x, y, theta):
        try:
            response = future.result()
            if response.name != "":
```

```
            self.get_logger().info("Turtle " + response.name + " is now alive")
        except Exception as e:
            self.get_logger().error("Service call failed %r" % (e, ))

def main(args=None):
    rclpy.init(args=args)
    node = TurtleSpawner()
    rclpy.spin(node)
    rclpy.shutdown()

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

Project Solution 3/6:
Improve turtle_spawner to Publish a list of active turtles as /alive_turtles
Improve turtle_controller to Subscribe to /alive_turtles, seek and Kill active turtles.

Create Publisher in turtle_spawner.py:
Create Interface for /alive_turtles topic in existing my_robot_interfaces package/msg:
Create 2 new message types:
        Turtle.msg
        TurtleArray.msg   [array of Turtle.msg]

```
james@JP-VM:~/ros2_ws/src/my_robot_interfaces/msg$ touch Turtle.msg
james@JP-VM:~/ros2_ws/src/my_robot_interfaces/msg$ touch TurtleArray.msg
james@JP-VM:~/ros2_ws/src/my_robot_interfaces/msg$ ls
HardwareStatus.msg  TurtleArray.msg
LedStateArray.msg   Turtle.msg
```

Edit **CmakeLists.txt**:

```
...
rosidl_generate_interfaces(${PROJECT_NAME}
  "msg/HardwareStatus.msg"
  "msg/LedStateArray.msg"
  "msg/Turtle.msg"
  "msg/TurtleArray.msg"
  "srv/ComputeRectangleArea.srv"
  "srv/SetLed.srv"
```

```
james@JP-VM:~/ros2_ws$ colcon build --packages-select my_robot_interfaces
james@JP-VM:~$ source .bashrc        [because new interface compiled]
james@JP-VM:~$ ros2 interface show my_robot_interfaces/msg/Turtle
string name
float64 x
float64 y
float64 theta
```

james@JP-VM:~$ **ros2 interface show my_robot_interfaces/msg/TurtleArray**
Turtle[] turtles

Edit **packages.xml**:

```
...
  <depend>rclpy</depend>
  <depend>turtlesim</depend>
  <depend>geometry_msgs</depend>
  <depend>my_robot_interfaces</depend>
```

Edit **turtle_spawner.py**:

import message types:

```
from my_robot_interfaces import Turtle
from my_robot_interfaces import TurtleArray
```

create publisher:

```
        self.alive_turtle_publisher_ = self.create_publisher(
            TurtleArray, "alive_turtles", 10)
```

create alive_turtles array:

```
        self.alive_turtles_ = []

    def publish_alive_turtles(self):
        msg = TurtleArray()
        msg.turtles = self.alive_turtles_
        self.alive_turtles_publisher_.publish(msg)
```

add spawned turtles to array:

```
    def callback_call_spawn(self, future, turtle_name, x, y, theta):
        try:
            response = future.result()
            if response.name != "":
                self.get_logger().info("Turtle " + response.name + " is now alive")
                new_turtle = Turtle()
                new_turtle.name = response.name
                new_turtle.x = x
                new_turtle.y = y
                new_turtle.theta = theta
                self.alive_turtles_publisher_.append(new_turtle)
                self.publish_alive_turtles()
```

turtle_spawner.py:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from functools import partial
import random
import math
from turtlesim.srv import Spawn
from my_robot_interfaces import Turtle
from my_robot_interfaces import TurtleArray

class TurtleSpawner(Node):
    def __init__(self):
        super().__init__("turtle_spawner")
        self.turtle_name_prefix_ = "turtle"
        self.turtle_counter_ = 0
        self.alive_turtles_ = []
        self.alive_turtles_publisher_ = self.create_publisher(
            TurtleArray, "alive_turtles", 10)
        self.spawn_turtle_timer_ = self.create_timer(
            2.0, self.spawn_new_turtle)

    def publish_alive_turtles(self):
        msg = TurtleArray()
        msg.turtles = self.alive_turtles_
        self.alive_turtles_publisher_.publish(msg)

    def spawn_new_turtle(self):
        self.turtle_counter_ += 1
        name = self.turtle_name_prefix_ + str(self.turtle_counter_)
        x = random.uniform(0.0, 11.0)
        y = random.uniform(0.0, 11.0)
        theta = random.uniform(0.0, 2*math.pi)
        self.call_spawn_server(name, x, y, theta)

    def call_spawn_server(self, turtle_name, x, y, theta):
        client = self.create_client(Spawn, "spawn")
        while not client.wait_for_service(1.0):
            self.get_logger().warn("Waiting for Server Spawn...")

        request = Spawn.Request()
        request.x = x
        request.y = y
        request.theta = theta
        request.name = turtle_name

        future = client.call_async(request)
```

```
        future.add_done_callback(
            partial(self.callback_call_spawn, turtle_name=turtle_name, x=x, y=y,
theta=theta))

    def callback_call_spawn(self, future, turtle_name, x, y, theta):
        try:
            response = future.result()
            if response.name != "":
                self.get_logger().info("Turtle " + response.name + " is now alive")
                new_turtle = Turtle()
                new_turtle.name = response.name
                new_turtle.x = x
                new_turtle.y = y
                new_turtle.theta = theta
                self.alive_turtles_publisher_.append(new_turtle)
                self.publish_alive_turtles()
        except Exception as e:
            self.get_logger().error("Service call failed %r" % (e, ))

def main(args=None):
    rclpy.init(args=args)
    node = TurtleSpawner()
    rclpy.spin(node)
    rclpy.shutdown()

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

james@JP-VM:~$ **source .bashrc**
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_interfaces**
Starting >>> my_robot_interfaces
Finished <<< my_robot_interfaces [8.51s]

Summary: 1 package finished [9.65s]

james@JP-VM:~/ros2_ws$ **colcon build --packages-select turtlesim_catch_them_all --symlink-install**
Starting >>> turtlesim_catch_them_all
Finished <<< turtlesim_catch_them_all [5.96s]

Summary: 1 package finished [6.37s]

james@JP-VM:~$ **ros2 run turtlesim turtlesim_node**
[INFO] [1605580732.528245651] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1605580732.593695067] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

field and turtle1 appear

james@JP-VM:~$ **ros2 run turtlesim_catch_them_all turtle_spawner**
Traceback (most recent call last):
  File "/home/james/ros2_ws/install/turtlesim_catch_them_all/lib/turtlesim_catch_them_all/turtle_spawner", line 11, in <module>
    load_entry_point('turtlesim-catch-them-all', 'console_scripts', 'turtle_spawner')()
  File "/usr/lib/python3/dist-packages/pkg_resources/__init__.py", line 490, in load_entry_point
    return get_distribution(dist).load_entry_point(group, name)
  File "/usr/lib/python3/dist-packages/pkg_resources/__init__.py", line 2854, in load_entry_point
    return ep.load()
  File "/usr/lib/python3/dist-packages/pkg_resources/__init__.py", line 2445, in load
    return self.resolve()
  File "/usr/lib/python3/dist-packages/pkg_resources/__init__.py", line 2451, in resolve
    module = __import__(self.module_name, fromlist=['__name__'], level=0)
  File "/home/james/ros2_ws/build/turtlesim_catch_them_all/turtlesim_catch_them_all/turtle_spawner.py", line 8, in <module>
    from my_robot_interfaces import Turtle
ImportError: **cannot import name 'Turtle' from 'my_robot_interfaces'** (/home/james/ros2_ws/install/my_robot_interfaces/lib/python3.8/site-packages/my_robot_interfaces/__init__.py)

2020.11.19
james@JP-VM:~/ros2_ws/src/turtlesim_catch_them_all/turtlesim_catch_them_all$
python3 turtle_spawner.py
Traceback (most recent call last):
  File "turtle_spawner.py", line 8, in <module>
    from my_robot_interfaces import Turtle
ImportError: cannot import name 'Turtle' from 'my_robot_interfaces'
(/home/james/ros2_ws/install/my_robot_interfaces/lib/python3.8/site-packages/my_robot_interfaces/__init__.py)

from my_robot_interfaces import Turtle
from my_robot_interfaces import TurtleArray
should be:
from my_robot_interfaces.**msg** import Turtle
from my_robot_interfaces.**msg** import TurtleArray
Now it works!

2020.11.20
james@JP-VM:~$ **ros2 run turtlesim turtlesim_node**
[INFO] [1605923114.420632891] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1605923114.477569955] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
[ERROR] [1605923224.437344241] [turtlesim]: A turtle named [turtle1] already exists
[INFO] [1605923226.408761772] [turtlesim]: Spawning turtle [turtle2] at x=[8.021623], y=[1.110910], theta=[3.401379] *Expected as master turtle = turtle1*

james@JP-VM:~$ **ros2 run turtlesim_catch_them_all turtle_spawner**
[INFO] [1605923226.514175108] [turtle_spawner]: Turtle turtle2 is now alive
[ERROR] [1605923226.515866186] [turtle_spawner]: Service call failed
AttributeError("'Publisher' object has no attribute 'append'")
[INFO] [1605923228.385330479] [turtle_spawner]: Turtle turtle3 is now alive

self.alive_turtles_publisher_.append(new_turtle)
should be
        self.alive_turtles_.append(new_turtle)

Demonstrate that the interface msg have been successfully compiled:
james@JP-VM:~$ **ros2 interface show my_robot_interfaces/msg/Turtle**
string name
float64 x
float64 y
float64 theta
james@JP-VM:~$ **ros2 interface show my_robot_interfaces/msg/TurtleArray**
Turtle[] turtles

james@JP-VM:~$ **ros2 topic list**
/alive_turtles
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
/turtle10/cmd_vel
/turtle10/color_sensor
/turtle10/pose
/turtle11/cmd_vel
/turtle11/color_sensor
/turtle11/pose
...

```
james@JP-VM:~$ source .bashrc
james@JP-VM:~$ ros2 topic echo /alive_turtles
turtles:
- name: turtle2
  x: 8.139294600009933
  y: 0.2018707104080657
  theta: 5.88779205211263
- name: turtle3
  x: 2.10317020822809
  y: 4.783238955194649
  theta: 5.962230371437574
- name: turtle4
  x: 8.266514877821866
  y: 2.1149857932045264
  theta: 0.2575161455079035
...
```

turtle_controller.py:

```
from my_robot_interfaces.msg import Turtle
from my_robot_interfaces.msg import TurtleArray

    def callback_turtle_pose(self, msg):
        self.pose_ = msg

        self.alive_turtles_subscriber_ = self.create_subscription(
            TurtleArray, "alive_turtles", self.callback_alive_turtles, 10)

        self.target_x = 8.0
        self.target_y = 4.0

        self.turtle_to_catch_ = None

    def callback_alive_turtles(self, msg):
        if len(msg.turtles) > 0:
            self.turtle_to_catch_ = msg.turtles[0]

        dist_x = self.target_x - self.pose_.x
        dist_y = self.target_y - self.pose_.y
```

Becomes:

```
        dist_x = self.turtle_to_catch_.x - self.pose_.x
        dist_y = self.turtle_to_catch_.y - self.pose_.y
```

turtle_controller.py:

```
#!/usr/bin/env python3
import math
import rclpy
from rclpy.node import Node

from turtlesim.msg import Pose
from geometry_msgs.msg import Twist
from my_robot_interfaces.msg import Turtle
from my_robot_interfaces.msg import TurtleArray

class TurtleControllerNode(Node):
    def __init__(self):
        super().__init__("turtle_controller")
        self.turtle_to_catch_ = None

        self.linear_constant = 2.0
        self.angular_constant = 6.0

        self.pose_ = None
        self.cmd_vel_publisher_ = self.create_publisher(
            Twist, "turtle1/cmd_vel", 10)
        self.pose_subscriber_ = self.create_subscription(
            Pose, "turtle1/pose", self.callback_turtle_pose, 10)
        self.alive_turtles_subscriber_ = self.create_subscription(
            TurtleArray, "alive_turtles", self.callback_alive_turtles, 10)

        self.control_loop_timer_ = self.create_timer(0.01, self.control_loop)

    def callback_turtle_pose(self, msg):
        self.pose_ = msg

    def callback_alive_turtles(self, msg):
        if len(msg.turtles) > 0:
            self.turtle_to_catch_ = msg.turtles[0]


    def control_loop(self):
        if self.pose_ == None or self.turtle_to_catch_ == None:
            return

        dist_x = self.turtle_to_catch_.x - self.pose_.x
        dist_y = self.turtle_to_catch_.y - self.pose_.y
        distance = math.sqrt(dist_x * dist_x + dist_y * dist_y)

        msg = Twist()
        if distance > 0.5:
```

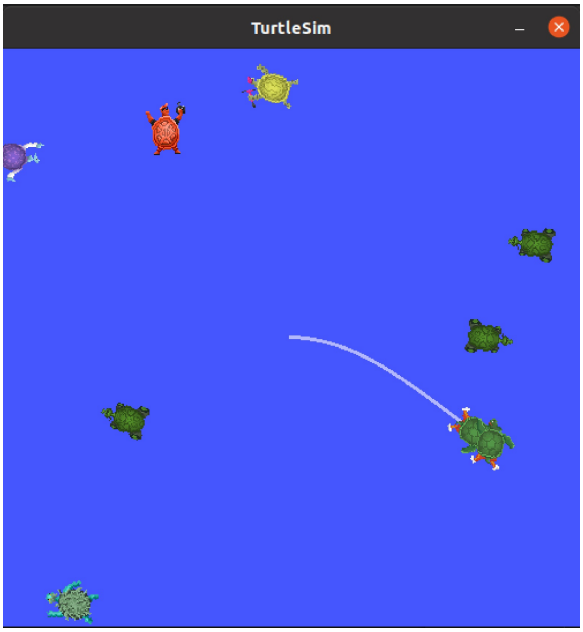
```
            # position
            msg.linear.x = distance * self.linear_constant
            # orientation
            goal_theta = math.atan2(dist_y, dist_x)
            diff = goal_theta - self.pose_.theta
            if diff > math.pi:
                diff -= 2*math.pi
            elif diff < -math.pi:
                diff += 2*math.pi
            msg.angular.z = diff * self.angular_constant
        else:
            msg.linear.x = 0.0
            msg.angular.z = 0.0

        self.cmd_vel_publisher_.publish(msg)

def main(args=None):
    rclpy.init(args=args)
    node = TurtleControllerNode()
    rclpy.spin(node)
    rclpy.shutdown()


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

kill ^C all and start over:
james@JP-VM:~$ **colcon build --packages-select turtlesim_catch_them_all --symlink-install**

james@JP-VM:~$ **ros2 run turtlesim turtlesim_node**
[INFO] [1605929948.764550620] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1605929948.778593122] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

james@JP-VM:~$ **ros2 run turtlesim_catch_them_all turtlesim_controller**
awaits turtle_spawner else has nothing to do

james@JP-VM:~$ **ros2 run turtlesim_catch_them_all turtle_spawner**
[INFO] [1605930022.843297989] [turtle_spawner]: Turtle turtle2 is now alive
[INFO] [1605930024.757438594] [turtle_spawner]: Turtle turtle3 is now alive



turtle1 has found turtle2. Now it needs to “catch” it and call the Kill service to remove it. Then move on to turtle3, etc.

Good place to stop for the night!

2020.11.22
Project Solution 4/6. Catch the turtle.
Create catch turtle service
james@JP-VM:~$ **cd ros2_ws/src/my_robot_interfaces/srv**
james@JP-VM:~/ros2_ws/src/my_robot_interfaces/srv$ **touch CatchTurtle.srv**

CatchTurtle.srv:
string name
---
bool success

CmakeLists.txt:
...
  "srv/CatchTurtle.srv"

compile the new interface
james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_interfaces**
if properly compiled, it will appear in the autofill in the next step...

turtle_spawner.py:
...
from my_robot_interfaces.srv import CatchTurtle

```
    def callback_catch_turtle(self, request, response):
        pass

        self.catch_turtle_service_ = self.create_service(
            CatchTurtle, "catch_turtle", self.callback_catch_turtle)
```

ros2 service type *only works if the turtle is running:*
james@JP-VM:~/ros2_ws$ **ros2 run turtlesim turtlesim_node**
james@JP-VM:~$ **ros2 service type /kill**
turtlesim/srv/Kill
james@JP-VM:~$ **ros2 interface show turtlesim/srv/Kill**
string name

This means to kill a turtle you just supply its name. Expect no response.

from turtlesim.srv import Kill

Use spawn_server as a template & cc below it:

```
    def call_spawnkill_server(self, turtle_name, x, y, theta):
        client = self.create_client(SpawnKill, "spawn" “kill”)
        while not client.wait_for_service(1.0):
            self.get_logger().warn("Waiting for Server Spawn...")

        request = SpawnKill.Request()
        request.x = x
        request.y = y
        request.theta = theta
        request.name = turtle_name

        future = client.call_async(request)
        future.add_done_callback(
            partial(self.callback_call_spawnkill, turtle_name=turtle_name, x=x, y=y,
theta=theta))

    def callback_call_spawnkill(self, future, turtle_name, x, y, theta):
        try:
            response = future.result()          there is no response but you want to catch err.
           for (i, turtle) in enumerate(self.alive_turtles_):    run through list of turtles
                    if turtle.name == turtle_name:               find the named turtle
                            del self.alive_turtles_[I]           and delete it
                            self.publish_alive_turtles()         since array has changed
                            break
            if response.name != "":
                self.get_logger().info("Turtle " + response.name + " is now alive")
                new_turtle = Turtle()
                new_turtle.name = response.name
                new_turtle.x = x
                new_turtle.y = y
                new_turtle.theta = theta
                self.alive_turtles_.append(new_turtle)
                self.publish_alive_turtles()
        except Exception as e:
            self.get_logger().error("Service call failed %r" % (e, ))
```

Now to respond to the catch turtle request...

```
    def callback_catch_turtle(self, request, response):
        self.call_kill_server(request.name)
        response.success = True
        return response
```

Add CatchTurtle service to turtle_controller.py

```
from my_robot_interfaces.srv import CatchTurtle
```

borrow same call_kill_server template from turtle_spawner and paste at end:

```
    def call_kill_server(self, turtle_name):
    def call_catch_turtle_server(self, turtle_name):
        client = self.create_client(Kill, "kill")
        client = self.create_client(CatchTurtle, "catch_turtle")
        while not client.wait_for_service(1.0):
            self.get_logger().warn("Waiting for Server")
        request = Kill.Request()
        request = CatchTurtle.Request()
        request.name = turtle_name
        future = client.call_async(request)
        future.add_done_callback(
            partial(self.callback_call_kill, turtle_name=turtle_name))
            partial(self.callback_call_catch_turtle, turtle_name=turtle_name))
    def callback_call_kill(self, future, turtle_name):
    def callback_call_catch_turtle(self, future, turtle_name):
        try:
            response = future.result()
            if not response.success:
                self.get_logger().error("Turtle " + str(turtle_name) + " could not be
caught")
            for (i, turtle) in enumerate(self.alive_turtles_):
                if turtle_name == turtle.name:
                    del self.alive_turtles_[i]
                    self.publish_alive_turtles()
                    break
        except Exception as e:
            self.get_logger().error("Service call failed %r" % (e, ))
```

call catch_turtle_server from control_loop:

```
        else:
            # target reached!
            msg.linear.x = 0.0
            msg.angular.z = 0.0
            self.call_catch_turtle_server(self.turtle_to_catch_.name)       request =
CatchTurtle.Request()
        request.name = turtle_name
```

```
from functools import partial
```

Try...

turtle_controller.py:

```
#!/usr/bin/env python3
import math
import rclpy
from functools import partial
from rclpy.node import Node

from turtlesim.msg import Pose
from geometry_msgs.msg import Twist
from my_robot_interfaces.msg import Turtle
from my_robot_interfaces.msg import TurtleArray
from my_robot_interfaces.srv import CatchTurtle

class TurtleControllerNode(Node):
    def __init__(self):
        super().__init__("turtle_controller")
        self.turtle_to_catch_ = None

        self.linear_constant = 2.0
        self.angular_constant = 6.0

        self.pose_ = None
        self.cmd_vel_publisher_ = self.create_publisher(
            Twist, "turtle1/cmd_vel", 10)
        self.pose_subscriber_ = self.create_subscription(
            Pose, "turtle1/pose", self.callback_turtle_pose, 10)
        self.alive_turtles_subscriber_ = self.create_subscription(
            TurtleArray, "alive_turtles", self.callback_alive_turtles, 10)

        self.control_loop_timer_ = self.create_timer(0.01, self.control_loop)

    def callback_turtle_pose(self, msg):
        self.pose_ = msg

    def callback_alive_turtles(self, msg):
        if len(msg.turtles) > 0:
            self.turtle_to_catch_ = msg.turtles[0]

    def control_loop(self):
        if self.pose_ == None or self.turtle_to_catch_ == None:
            return

        dist_x = self.turtle_to_catch_.x - self.pose_.x
        dist_y = self.turtle_to_catch_.y - self.pose_.y
        distance = math.sqrt(dist_x * dist_x + dist_y * dist_y)

        msg = Twist()
```

```
        if distance > 0.5:
            # position
            msg.linear.x = distance * self.linear_constant
            # orientation
            goal_theta = math.atan2(dist_y, dist_x)
            diff = goal_theta - self.pose_.theta
            if diff > math.pi:
                diff -= 2*math.pi
            elif diff < -math.pi:
                diff += 2*math.pi
            msg.angular.z = diff * self.angular_constant
        else:
            # target reached!
            msg.linear.x = 0.0
            msg.angular.z = 0.0
            self.call_catch_turtle_server(self.turtle_to_catch_.name)

        self.cmd_vel_publisher_.publish(msg)

    def call_catch_turtle_server(self, turtle_name):
        client = self.create_client(CatchTurtle, "catch_turtle")
        while not client.wait_for_service(1.0):
            self.get_logger().warn("Waiting for Server")
        request = CatchTurtle.Request()
        request.name = turtle_name

        future = client.call_async(request)
        future.add_done_callback(
            partial(self.callback_call_catch_turtle, turtle_name=turtle_name))

    def callback_call_catch_turtle(self, future, turtle_name):
        try:
            response = future.result()
            if not response.success:
                self.get_logger().error("Turtle " + str(turtle_name) + " could not be caught")

        except Exception as e:
            self.get_logger().error("Service call failed %r" % (e, ))

def main(args=None):
    rclpy.init(args=args)
    node = TurtleControllerNode()
    rclpy.spin(node)
    rclpy.shutdown()

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

turtle_spawner.py:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from functools import partial
import random
import math
from turtlesim.srv import Spawn
from turtlesim.srv import Kill
from my_robot_interfaces.msg import Turtle
from my_robot_interfaces.msg import TurtleArray
from my_robot_interfaces.srv import CatchTurtle

class TurtleSpawner(Node):
    def __init__(self):
        super().__init__("turtle_spawner")
        self.turtle_name_prefix_ = "turtle"
        self.turtle_counter_ = 0
        self.alive_turtles_ = []
        self.alive_turtles_publisher_ = self.create_publisher(
            TurtleArray, "alive_turtles", 10)
        self.spawn_turtle_timer_ = self.create_timer(
            2.0, self.spawn_new_turtle)
        self.catch_turtle_service_ = self.create_service(
            CatchTurtle, "catch_turtle", self.callback_catch_turtle)

    def callback_catch_turtle(self, request, response):
        self.call_kill_server(request.name)
        response.success = True
        return response

    def publish_alive_turtles(self):
        msg = TurtleArray()
        msg.turtles = self.alive_turtles_
        self.alive_turtles_publisher_.publish(msg)

    def spawn_new_turtle(self):
        self.turtle_counter_ += 1
        name = self.turtle_name_prefix_ + str(self.turtle_counter_)
        x = random.uniform(0.0, 11.0)
        y = random.uniform(0.0, 11.0)
        theta = random.uniform(0.0, 2*math.pi)
        self.call_spawn_server(name, x, y, theta)

    def call_spawn_server(self, turtle_name, x, y, theta):
        client = self.create_client(Spawn, "spawn")
        while not client.wait_for_service(1.0):
```

```
            self.get_logger().warn("Waiting for Server Spawn...")

        request = Spawn.Request()
        request.x = x
        request.y = y
        request.theta = theta
        request.name = turtle_name

        future = client.call_async(request)
        future.add_done_callback(
            partial(self.callback_call_spawn, turtle_name=turtle_name, x=x, y=y,
theta=theta))

    def callback_call_spawn(self, future, turtle_name, x, y, theta):
        try:
            response = future.result()
            if response.name != "":
                self.get_logger().info("Turtle " + response.name + " is now alive")
                new_turtle = Turtle()
                new_turtle.name = response.name
                new_turtle.x = x
                new_turtle.y = y
                new_turtle.theta = theta
                self.alive_turtles_.append(new_turtle)
                self.publish_alive_turtles()
        except Exception as e:
            self.get_logger().error("Service call failed %r" % (e, ))

    def call_kill_server(self, turtle_name):
        client = self.create_client(Kill, "kill")
        while not client.wait_for_service(1.0):
            self.get_logger().warn("Waiting for Server Kill...")
        request = Kill.Request()
        request.name = turtle_name
        future = client.call_async(request)
        future.add_done_callback(
            partial(self.callback_call_kill, turtle_name=turtle_name))

    def callback_call_kill(self, future, turtle_name):
        try:
            future.result()
            for (i, turtle) in enumerate(self.alive_turtles_):
                if turtle_name == turtle.name:
                    del self.alive_turtles_[i]
                    self.publish_alive_turtles()
                    break
        except Exception as e:
```

```
            self.get_logger().error("Service call failed %r" % (e, ))

def main(args=None):
    rclpy.init(args=args)
    node = TurtleSpawner()
    rclpy.spin(node)
    rclpy.shutdown()

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

james@JP-VM:~$ **cd ros2_ws/**
james@JP-VM:~/ros2_ws$ **colcon build --packages-select turtlesim_catch_them_all --symlink-install**

james@JP-VM:~/ros2_ws$ **ros2 run turtlesim turtlesim_node**
[INFO] [1606091933.060155021] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1606091933.106770900] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

james@JP-VM:~$ **ros2 run turtlesim_catch_them_all turtlesim_controller**

james@JP-VM:~$ **ros2 run turtlesim_catch_them_all turtle_spawner**

[ERROR] [1606092168.445178710] [turtlesim]: A turtle named [turtle1] already exists
this is expected error
[INFO] [1606092170.575936557] [turtlesim]: Spawning turtle [turtle2] at x=[1.987371], y=[0.286160], theta=[2.328916]
[ERROR] [1606092172.100745162] [turtlesim]: Tried to kill turtle [turtle2], which does not exist
[ERROR] [1606092172.131969693] [turtlesim]: Tried to kill turtle [turtle2], which does not exist

until new alive_turtles message arrives, will keep trying to catch & kill the same turtle.
Add to control_loop:

```
            self.turtle_to_catch_ = None
```

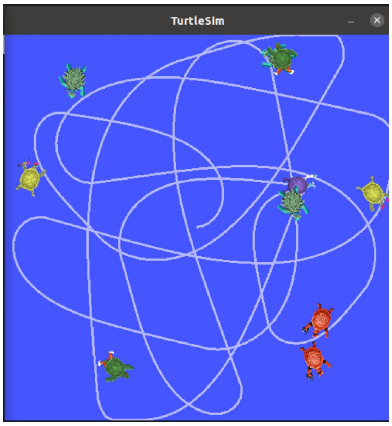
Try again:
james@JP-VM:~$ **ros2 run turtlesim turtlesim_node**
[INFO] [1606092678.898264445] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1606092678.930889691] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]

james@JP-VM:~$ **ros2 run turtlesim_catch_them_all turtlesim_controller**

james@JP-VM:~$ **ros2 run turtlesim_catch_them_all turtle_spawner**

SUCCESS!!

Try decreasing the period of turtle_spawner 2.0 > 1.0
It can't keep up –



Project Solution 5/6:
Improve system to seek the closest, not the first target turtle.
Create flag to allow catching first turtle first (set to False):

```
        self.catch_closest_turtle_first = True

    def callback_alive_turtles(self, msg):
        if len(msg.turtles) > 0:
            if self.catch_closest_turtle_first_:
                pass
            else:
                self.turtle_to_catch_ = msg.turtles[0]

    def callback_alive_turtles(self, msg):
        if len(msg.turtles) > 0:
            if self.catch_closest_turtle_first_:
                closest_turtle = None
                closest_turtle_distance = None

                for turtle in msg.turtles:
                    dist_x = turtle.x - self.pose_.x
                    dist_y = turtle.y - self.pose_.y
                    distance = math.sqrt(dist_x * dist_x + dist_y * dist_y)
                    if closest_turtle == None or distance < closest_turtle_distance:
                        closest_turtle = turtle
                        closest_turtle_distance = distance
                self.turtle_to_catch_ = closest_turtle
            else:
                self.turtle_to_catch_ = msg.turtles[0]
```

turtle_spawner.py:

```
        self.spawn_turtle_timer_ = self.create_timer(
            1.0, self.spawn_new_turtle)                    Decrease from 2.0
```

```
james@JP-VM:~$ ros2 run turtlesim turtlesim_node

james@JP-VM:~$ ros2 run turtlesim_catch_them_all turtlesim_controller

james@JP-VM:~$ ros2 run turtlesim_catch_them_all turtle_spawner
```

```
        self.spawn_turtle_timer_ = self.create_timer(
            1.5, self.spawn_new_turtle)
        self.spawn_turtle_timer_ = self.create_timer(
            1.1, self.spawn_new_turtle)
```

As more turtles spawn and master turtle falls behind, it starts giving "turtle doesn't exist" errors and the path to the next turtle becomes serpentine & less efficient. It seems it can't get through all the turtles in the current alive_turtles array before it get a new one so changes course.

Try speeding up the turtle:
self.linear_constant = 3.0 *2.0 > 3.0*
self.angular_constant = 6.0

Doesn't work well. Turtle gets even loopier if self.linear_constant is too high.

Check on turtlesim_node servises – is there a way to mark each spawned turtle position even after they've been killed?

^Cjames@JP-VM:~$ **rosservice list**
/clear
/kill
/reset
/spawn
/turtle1/set_pen
/turtle1/teleport_absolute
/turtle1/teleport_relative
/turtlesim/describe_parameters
/turtlesim/get_parameter_types
/turtlesim/get_parameters
/turtlesim/list_parameters
/turtlesim/set_parameters
/turtlesim/set_parameters_atomically

james@JP-VM:~$ **ros2 interface show turtlesim/srv/SetPen**
uint8 r
uint8 g
uint8 b
uint8 width
uint8 off
---

james@JP-VM:~$ **ros2 topic echo /turtle1/color_sensor**
r: 255
g: 255
b: 255
---

turtle_spawner period of 1.2 keeps the spawner just ahead of the controller

2020.11.24
Project Solution 6/6
Scale program with Parameters and Launch file

```
        self.catch_closest_turtle_first_ = True
```

define at beginning of Class
turtle_controller.py:

```
        self.declare_parameter("catch_closest_turtle_first", True)
        self.catch_closest_turtle_first_ =
          self.get_parameter("catch_closest_turtle_first").value
```

turtle_spawner.py:

```
        self.declare_parameter("spawn_frequency", 1.0)
        self.declare_parameter("turtle_name_prefix", "turtle")

        self.spawn_frequency_ = self.get_parameter("spawn_frequency").value
        self.turtle_name_prefix_ = self.get_parameter("turtle_name_prefix").value

        self.spawn_turtle_timer_ = self.create_timer(
           1.0/self.spawn_frequency_, self.spawn_new_turtle)
```

Create launch file in my_robot_bringup:
turtlesim_catch_them_all.launch.py
make executable
chmod +x turtlesim_catch_them_all.launch.py

package.xml:

```
  <exec_depend>turtlesim</exec_depend>
  <exec_depend>turtlesim_catch_them_all</exec_depend>
```

james@JP-VM:~/ros2_ws$ **colcon build --packages-select my_robot_bringup**

james@JP-VM:~$ **source .bashrc**
james@JP-VM:~$ **ros2 launch my_robot_bringup**
turtlesim_catch_them_all.launch.py

turtlesim_catch_them_all.launch.py:

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    id = LaunchDescription()

    turtlesim_node = Node(
        package="turtlesim",
        executable="turtlesim_node"
    )

    turtle_spawner_node = Node(
        package="turtlesim_catch_them_all",
        executable="turtle_spawner",
        parameters=[
            {"spawn_frequency": 1.0},
            {"turtle_name_prefix": "my_turtle"}
        ]
    )

    turtle_controller_node = Node(
        package="turtlesim_catch_them_all",
        executable="turtlesim_controller",
        parameters=[
            {"catch_closest_turtle_first": True}
        ]
    )

    id.add_action(turtlesim_node)
    id.add_action(turtle_spawner_node)
    id.add_action(turtle_controller_node)
    return id
```

Add parameters linear_constant, angular_constant:

Challenge: *Change the pen color each time the “master turtle” has caught another turtle.*