5435 AGENDA 11/23/2021 ROS2

Contents

Student Reports if you have not described your project.   Presentation 11/30 or 12/7

Reports and slides due 12/7

TurtleBot 4 - Available in Spring?

https://clearpathrobotics.com/blog/2021/10/clearpath-robotics-announces-turtlebot-4/

ROS2 - Demos

1_1 Let's Try Some Demos Workspace   1_1

https://docs.ros.org/en/foxy/Tutorials/Workspace/Creating-A-Workspace.html#

1. Have ROS2 installed and sourced - $ `source /opt/ros/foxy/setup.bash`
2. Create directory dev_ws and src below

```
mkdir -p ~/dev_ws/src

cd ~/dev_ws/src
```

3. Clone a simple repository of tutorials

```
git clone https://github.com/ros/ros_tutorials.git -b foxy-devel
```

4. Resolve Dependencies (In ws)

```
cd ..
rosdep install -i --from-path src --rosdistro foxy -y
```

5. $ colcon build – Create (build) the files necessary for linking and running programs.
   DIRECTORIES: `build   install   log   src`

6.NEW TERMINAL FOR OVERLAY AND “UNDERLAY” - SEE COMMENTS:

Before sourcing the overlay, it is very important that you open a new terminal, separate from the one where you built the workspace. Sourcing an overlay in the same terminal where you built, or likewise building where an overlay is sourced, may create complex issues.

In the new terminal, source your main ROS 2 environment as the “underlay”, so you can build the overlay “on top of” it:

```
$ source /opt/ros/foxy/setup.bash
$ cd ~/dev_ws
$ . install/local_setup.bash                 (. == source)
```

7. TEST THE CODE

```
$ ros2 run turtlesim turtlesim_node

Alias foxy or noetic
harman@harman-VirtualBox:~$ foxy
harman@harman-VirtualBox:~$ cd dev_ws/
harman@harman-VirtualBox:~/dev_ws$ ls
     build  install  log  src
harman@harman-VirtualBox:~/dev_ws$ cd src
harman@harman-VirtualBox:~/dev_ws/src$ ls
     ros_tutorials
harman@harman-VirtualBox:~/dev_ws/src$ cd ros_tutorials/
harman@harman-VirtualBox:~/dev_ws/src/ros_tutorials$ ls
     roscpp_tutorials  rospy_tutorials  ros_tutorials  turtlesim
harman@harman-VirtualBox:~/dev_ws/src/ros_tutorials$ cd turtlesim
harman@harman-VirtualBox:~/dev_ws/src/ros_tutorials/turtlesim$ ls
     action         CMakeLists.txt  include  msg          src  tutorials
     CHANGELOG.rst  images          launch   package.xml  srv
harman@harman-VirtualBox:~/dev_ws/src/ros_tutorials/turtlesim$ cd src
harman@harman-VirtualBox:~/dev_ws/src/ros_tutorials/turtlesim/src$ ls
     turtle.cpp  turtle_frame.cpp  turtlesim  turtlesim.cpp
harman@harman-VirtualBox:~/dev_ws/src/ros_tutorials/turtlesim/src$ cd turtlesim
```

```
harman@harman-VirtualBox:~/dev_ws/src/ros_tutorials/turtlesim/src/turtlesim$ ls
     __init__.py
```

LOOK AT THE FILES – LET'S GO TO DEMOS.
Creating a workspace – ROS 2 Documentation_ Foxy.pdf (1_1)
Build_workspace_dev_ws_11_22_2021a.txt (1_2)
CMakeLists_Tsim_tutorials.txt (1_3)
packageXML_Tsim_tutorials.txt (1_4)

2_0. LETS'S DO A PACKAGE

https://docs.ros.org/en/foxy/Tutorials/Creating-Your-First-ROS2-Package.html

A package can be considered a container for your ROS 2 code. If you want to be able to install your code or share it with others, then you'll need it organized in a package. With packages, you can release your ROS 2 work and allow others to build and use it easily.

Package creation in ROS 2 uses ament as its build system and colcon as its build tool. You can create a package using either CMake or Python, which are officially supported, though other build types do exist.

Python Package

- `package.xml` file containing meta information about the package
- `setup.py` containing instructions for how to install the package
- `setup.cfg` is required when a package has executables, so `ros2 run` can find them
- `/<package_name>` - a directory with the same name as your package, used by ROS 2 tools to find your package, contains `__init__.py`

Let's use the workspace you created in the previous tutorial, `dev_ws`, for your new package.

```
        $ cd ~/dev_ws/src   (Go to /src in Workspace)
        $ ros2 pkg create --build-type ament_python --node-name my_node my_package
```

CreateMyPackage_11_23_2021 **(2_1)**

```
Alias foxy or noetic
harman@harman-VirtualBox:~$ foxy
harman@harman-VirtualBox:~$ cd dev_ws/
harman@harman-VirtualBox:~/dev_ws$ ls
        build  install  log  src
harman@harman-VirtualBox:~/dev_ws$ cd ~
harman@harman-VirtualBox:~$ cd ~/dev_ws/src

harman@harman-VirtualBox:~/dev_ws/src$ ros2 pkg create --build-type ament_python --node-name my_node my_package
      going to create a new package
      package name: my_package
      destination directory: /home/harman/dev_ws/src
      package format: 3
      version: 0.0.0
      description: TODO: Package description
      maintainer: ['harman <harman@todo.todo>']
      licenses: ['TODO: License declaration']
      build type: ament_python
      dependencies: []
      node_name: my_node
      creating folder ./my_package
      creating ./my_package/package.xml
      creating source folder
      creating folder ./my_package/my_package
      creating ./my_package/setup.py
      creating ./my_package/setup.cfg
      creating folder ./my_package/resource
      creating ./my_package/resource/my_package
      creating ./my_package/my_package/__init__.py

      creating folder ./my_package/test
      creating ./my_package/test/test_copyright.py
      creating ./my_package/test/test_flake8.py
      creating ./my_package/test/test_pep257.py
      creating ./my_package/my_package/my_node.py
harman@harman-VirtualBox:~/dev_ws/src$
```

```
       my_package  package.xml  resource  setup.cfg  setup.py  test
```

```
def main():
    print('Hi from my_package.')    # my_node.py

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

1. BUILD THE PACKAGE

```
$ cd ~/dev_ws
$ colcon build
```

- To use your new package and executable, first open a new terminal and **source your main ROS 2 installation**. Then, from inside the `dev_ws` directory, run the following command to source your workspace:

```
$ . install/setup.bash   (Source your workspace in dev_ws)
```

BUILD AND RUN MY_NODE

```
Alias foxy or noetic

harman@harman-VirtualBox:~$ foxy

harman@harman-VirtualBox:~$ cd ~/dev_ws/

harman@harman-VirtualBox:~/dev_ws$ colcon build --packages-select my_package

Starting >>> my_package

Finished <<< my_package [0.46s]


Summary: 1 package finished [0.55s]

harman@harman-VirtualBox:~/dev_ws$ . install/setup.bash

harman@harman-VirtualBox:~/dev_ws$ ros2 run my_package my_node
```

Hi from my_package 😇 .

YOU WANT DETAILS OF THE FILES AND MEANINGS? (3_0)

https://roboticsbackend.com/create-a-ros2-python-package/

```
my_python_pkg/
├── my_python_pkg
│ └── __init__.py
├── package.xml
├── resource
│ └── my_python_pkg
├── setup.cfg
├── setup.py
└── test
├── test_copyright.py
├── test_flake8.py
└── test_pep257.py
```

LET’S WATCH A BOTBUILDER VIDEO TO SUMMARIZE

ROS2 Basics #3 - Understanding ROS2 Packages and Workspace

5,764 views Jun 12, 2020 9:09

ROS2 Basics #3 - Understanding ROS2 Packages and ...

3_0 LET'S DO A PUBLISHER AND SUBSCRIBER – DO THIS CAREFULLY IN PROPER DIRECTORIES!!

https://docs.ros.org/en/foxy/Tutorials/Writing-A-Simple-Py-Publisher-And-Subscriber.html

Navigate into the `dev_ws` directory created in a previous tutorial. Recall that packages should be created in the `src` directory, not the root of the workspace. So, navigate into `dev_ws/src`, and run the package creation command:

1. **dev_ws: source foxy; cd to /src**

```
$ ros2 pkg create --build-type ament_python py_pubsub
```

2. Write the publisher node

Navigate (cd) into ~/**dev_ws/src/py_pubsub/py_pubsub**. Recall that this directory is a Python package

with the same name as the ROS 2 package it's nested in.

```
harman@harman-VirtualBox:~/dev_ws/src$ cd ~/dev_ws/src/py_pubsub/py_pubsub

harman@harman-VirtualBox:~/dev_ws/src/py_pubsub/py_pubsub$ ls

    __init__.py
```

Download the example talker code

Enter the following command:

```
$ wget
https://raw.githubusercontent.com/ros2/examples/foxy/rclpy/topics/minimal_publisher/examples_rclpy_minimal_publisher/publisher_member_function.py
```

The wget command allows you to download files from the Internet using a Linux operating system such as Ubuntu. Use this command to download either a single Web page or a complete copy of your company's website. It also includes an option for downloading any external links included on the site. The command **recreates the complete directory structure of the site downloaded** on your computer's hard drive, and you can store the local copy as a backup or use it for testing purposes.

```
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub/py_pubsub     (Be Careful of Directories)
$ wget
https://raw.githubusercontent.com/ros2/examples/foxy/rclpy/topics/minimal_publisher/examples_rclpy_minimal_publisher/publisher_member_function.py
        --2021-11-23 14:45:16--
        https://raw.githubusercontent.com/ros2/examples/foxy/rclpy/topics/minimal_publisher/examples_rclpy_minimal_publisher/publisher_member_function.py
        Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
        Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
        HTTP request sent, awaiting response... 200 OK
        Length: 1576 (1.5K) [text/plain]
        Saving to: ‘publisher_member_function.py’

        publisher_member_fu 100%[===================>]   1.54K  --.-KB/s    in 0s

        2021-11-23 14:45:16 (42.4 MB/s) - ‘publisher_member_function.py’ saved [1576/1576]

harman@harman-VirtualBox:~/dev_ws/src/py_pubsub/py_pubsub$ ls -la
total 12
        drwxrwxr-x 2 harman harman 4096 Nov 23 14:45 .
        drwxrwxr-x 5 harman harman 4096 Nov 23 14:36 ..
        -rw-rw-r-- 1 harman harman    0 Nov 23 14:36 __init__.py
        -rw-rw-r-- 1 harman harman 1576 Nov 23 14:45 publisher_member_function.py

$ gedit publisher_member_function.py
```

FOR A WALK THROUGH:

https://docs.ros.org/en/foxy/Tutorials/Writing-A-Simple-Py-Publisher-And-Subscriber.html

The first lines of code after the comments import `rclpy` so its `Node` class can be used.   ETC.

publisher_member_function.py

```
# Copyright 2016 Open Source Robotics Foundation, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

import rclpy
from rclpy.node import Node

from std_msgs.msg import String


class MinimalPublisher(Node):

    def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(String, 'topic', 10)
        timer_period = 0.5  # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.i = 0

    def timer_callback(self):
        msg = String()
        msg.data = 'Hello World: %d' % self.i
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        self.i += 1
```

```
def main(args=None):
    rclpy.init(args=args)

    minimal_publisher = MinimalPublisher()

    rclpy.spin(minimal_publisher)

    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    minimal_publisher.destroy_node()
    rclpy.shutdown()


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

2.2 Add dependencies

1- Navigate one level back to the **dev_ws/src/py_pubsub** directory, where the setup.py, setup.cfg, and package.xml files have been created for you.

```
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub/py_pubsub$ cd ..
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub$ ls
        package.xml  py_pubsub  resource  setup.cfg  setup.py  test

harman@harman-VirtualBox:~/dev_ws/src/py_pubsub$ gedit package.xml
```

2- As mentioned in the previous tutorial, make sure to fill in the <description>, <maintainer> and <license> tags: (Optional)

3- After the lines above, add the following dependencies corresponding to your node’s **import statements**:

```
<exec_depend>rclpy</exec_depend>
<exec_depend>std_msgs</exec_depend>
```

This declares the package needs rclpy and std_msgs when its code is executed.

```
<?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>py_pubsub</name>
  <version>0.0.0</version>
  <description>Examples of minimal publisher/subscriber using rclpy</description>
  <maintainer email="you@email.com">Your Name</maintainer>
  <license>Apache License 2.0</license>

  <exec_depend>rclpy</exec_depend>
  <exec_depend>std_msgs</exec_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>

  <export>
    <build_type>ament_python</build_type>
  </export>
</package>
```

2.3 Add an entry point in setup.py

```
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub$ ls
package.xml  py_pubsub  resource  setup.cfg  setup.py  test
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub$ gedit setup.py
```

------------ setup.py

```
from setuptools import setup

package_name = 'py_pubsub'

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='YourName',
    maintainer_email='you@email.com',
    description='Examples of minimal publisher/subscriber using rclpy',
    license='Apache License 2.0',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
                'talker = py_pubsub.publisher_member_function:main',
        ],
},
)
```

```
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub$ ls
          package.xml  py_pubsub  resource  setup.cfg  setup.py  test
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub$ gedit setup.cfg
```

```
[develop]
script-dir=$base/lib/py_pubsub
[install]
install-scripts=$base/lib/py_pubsub                    (OK)
```

This is simply telling setuptools to put your executables in lib, because ros2 run will look for them there.
You could build your package now, source the local setup files, and run it, but let's create the subscriber node first so you can see the full system at work.

3 Write the subscriber node

```
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub/py_pubsub$ ls
          __init__.py  publisher_member_function.py
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub/py_pubsub$
```

```
wget
https://raw.githubusercontent.com/ros2/examples/foxy/rclpy/topics/minimal_subscriber/examples_rclpy_minimal_subscriber/subscriber_member_functi
on.py
    --2021-11-23 15:52:45--
    https://raw.githubusercontent.com/ros2/examples/foxy/rclpy/topics/minimal_subscriber/examples_rclpy_minimal_subscriber/subscriber_membe
    r_function.py
    Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
    Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 1469 (1.4K) [text/plain]
    Saving to: ‘subscriber_member_function.py’

    subscriber_member_f 100%[===================>]   1.43K  --.-KB/s    in 0s

    2021-11-23 15:52:46 (13.6 MB/s) - ‘subscriber_member_function.py’ saved [1469/1469]

harman@harman-VirtualBox:~/dev_ws/src/py_pubsub/py_pubsub$ ls
    __init__.py  publisher_member_function.py  subscriber_member_function.py
```

subscriber_member_function.py

```
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub/py_pubsub$ ls
    __init__.py  publisher_member_function.py  subscriber_member_function.py
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub/py_pubsub$ gedit subscriber_member_function.py
Copyright 2016 Open Source Robotics Foundation, Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

import rclpy
```

```
from rclpy.node import Node

from std_msgs.msg import String


class MinimalSubscriber(Node):

    def __init__(self):
        super().__init__('minimal_subscriber')
        self.subscription = self.create_subscription(
            String,
            'topic',
            self.listener_callback,
            10)
        self.subscription  # prevent unused variable warning

    def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)


def main(args=None):
    rclpy.init(args=args)

    minimal_subscriber = MinimalSubscriber()

    rclpy.spin(minimal_subscriber)

    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    minimal_subscriber.destroy_node()
    rclpy.shutdown()


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

```
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub/py_pubsub$ cd ..
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub$ ls
package.xml  py_pubsub  resource  setup.cfg  setup.py  test
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub$ gedit setup.py

from setuptools import setup

package_name = 'py_pubsub'

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='YourName',
    maintainer_email='you@email.com',
    description='Examples of minimal publisher/subscriber using rclpy',
    license='Apache License 2.0',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
                'talker = py_pubsub.publisher_member_function:main',
                'listener = py_pubsub.subscriber_member_function:main',
        ],
},
)
```

CHECK DEPENDS AND BUILD AND SOURCE

It’s good practice to run **rosdep** in the root of your workspace (dev_ws) to check for missing dependencies before building:

```
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub$ gedit setup.py
harman@harman-VirtualBox:~/dev_ws/src/py_pubsub$ cd ../..
harman@harman-VirtualBox:~/dev_ws$ rosdep install -i --from-path src --rosdistro foxy -y
        #All required rosdeps installed successfully


harman@harman-VirtualBox:~/dev_ws$ colcon build --packages-select py_pubsub
Starting >>> py_pubsub
Finished <<< py_pubsub [0.52s]

Summary: 1 package finished [0.69s]



harman@harman-VirtualBox:~/dev_ws$ . install/setup.bash          (SOURCE WS)
harman@harman-VirtualBox:~/dev_ws$ ros2 run py_pubsub talker
        [INFO] [1637705319.685421058] [minimal_publisher]: Publishing: "Hello World: 0"
        [INFO] [1637705320.182406979] [minimal_publisher]: Publishing: "Hello World: 1"
        [INFO] [1637705320.663758873] [minimal_publisher]: Publishing: "Hello World: 2"
        [INFO] [1637705321.167959189] [minimal_publisher]: Publishing: "Hello World: 3"
        [INFO] [1637705321.663720993] [minimal_publisher]: Publishing: "Hello World: 4"
        [INFO] [1637705322.1636068
```

SO FAR SO GOOD – LET’S DO LISTENER

NEW TERMINAL

Alias foxy or noetic

```
harman@harman-VirtualBox:~$ foxy
harman@harman-VirtualBox:~$ ros2 run py_pubsub listener
Package 'py_pubsub' not found          NOT SOURCED IN WS
```

```
harman@harman-VirtualBox:~$ cd dev_ws/
harman@harman-VirtualBox:~/dev_ws$ ls
        build  install  log  src
harman@harman-VirtualBox:~/dev_ws$ . install/setup.bash
harman@harman-VirtualBox:~/dev_ws$ ros2 run py_pubsub listener          (WAITS FOR TALKER)
[INFO] [1637705599.176930115] [minimal_subscriber]: I heard: "Hello World: 0"
[INFO] [1637705599.668214407] [minimal_subscriber]: I heard: "Hello World: 1"
[INFO] [1637705600.166659450] [minimal_subscriber]: I heard: "Hello World: 2"
[INFO] [1637705600.664777796] [minimal_subscriber]: I heard: "Hello World: 3"
[INFO] [1637705601.166307250] [minimal_subscriber]: I heard: "Hello World: 4"
[INFO] [1637705601.665638267] [minimal_subscriber]: I heard: "Hello World: 5"
[INFO] [1637705602.163834846] [minimal_subscriber]: I heard: "Hello World: 6"
[INFO] [1637705602.665828716] [minimal_subscriber]: I heard: "Hello World: 7"
[INFO] [1637705603.164060688] [minimal_subscriber]: I heard: "Hello World: 8"
[INFO] [1637705603.664722926] [minimal_subscriber]: I heard: "Hello World: 9"
[INFO] [1637705604.169637847] [minimal_subscriber]: I heard: "Hello World: 10"

harman@harman-VirtualBox:~/dev_ws$ ros2 run py_pubsub talker
[INFO] [1637705599.176400902] [minimal_publisher]: Publishing: "Hello World: 0"
[INFO] [1637705599.667565695] [minimal_publisher]: Publishing: "Hello World: 1"
[INFO] [1637705600.166341248] [minimal_publisher]: Publishing: "Hello World: 2"
[INFO] [1637705600.664293723] [minimal_publisher]: Publishing: "Hello World: 3"
[INFO] [1637705601.165902978] [minimal_publisher]: Publishing: "Hello World: 4"
[INFO] [1637705601.665338898] [minimal_publisher]: Publishing: "Hello World: 5"
[INFO] [1637705602.163464268] [minimal_publisher]: Publishing: "Hello World: 6"
[INFO] [1637705602.665452640] [minimal_publisher]: Publishing: "Hello World: 7"
[INFO] [1637705603.163785664] [minimal_publisher]: Publishing: "Hello World: 8"
[INFO] [1637705603.664236603] [minimal_publisher]: Publishing: "Hello World: 9"
[INFO] [1637705604.169356819] [minimal_publisher]: Publishing: "Hello World: 10"
[INFO] [1637705604.664805502] [minimal_publisher]: Publishing: "Hello World: 11"
[INFO] [1637705605.191205825] [minimal_publisher]: Publishing: "Hello World: 12"
[INFO] [1637705605.664472927] [minimal_publisher]: Publishing: "Hello World: 13"
^CTrace
```

BotBuilder Video #11 Pub and Sub

ROS2 Basics #11 - Writing a Simple Publisher and Subscriber (Python)

4,055 views Jun 15, 2020 12:09

https://www.youtube.com/watch?v=eqfoy2ctixE&t=191s

In this video you will learn how to create a ROS2 Publisher and Subscriber in Python. We will also step by step explain the code involve in it. Github: https://github.com/ros2torial/ros2_ba...

3_GAZEBO, PYTHON, C++ (ALWAYS CHECK THE LATEST UPDATES!)

Gazebo: Some goals of the refactoring were:

- Take advantage of new ROS 2 features, such as masterless discovery.
- Remove code which duplicates functionality already present in Gazebo.
- Reduce duplication by standardizing common functionality, such as how to set ROS namespaces, parameters and topic remapping.
- Modernize the codebase, making use of the latest SDFormat, Gazebo and Ignition APIs, as well as ROS 2's style guidelines and linters.
- Add tests and demos for all ported functionality.

The official Gazebo version supported with Foxy is 11. This is defined on REP-2000.

HOME/THOUGHTS

Ignition vs Gazebo

Since Gazebo 11 will be the last major version I thought I'd test the replacement Ignition. There's a handy comparison chart with feature comparisons between the two programs.

https://www.allisonthackston.com/articles/ignition-vs-gazebo.html

Simple Summary

ATTRIBUTE	IGNITION	GAZEBO
Inertia	x	x
Friction		
Bounce		x
Dynamics	x	x

More Details of Ignition vs Gaxebo

A Review of Gazebo Ignition Citadel

2,825 views Apr 9, 2020 18:22

https://www.youtube.com/watch?v=nAHM2LYShsc

ROS 2 + DDS Interoperation 1:03:00 371 views •Oct 8, 2020 1:03:00

https://www.youtube.com/watch?v=GGqcrccWfeE&t=8s

- TIME: Starts with Vehicles
- Uses of DDS
- 10:00 What is DDS?
- 43:00 ROS2 and DDS

12_Learn all about Robot Operating System 2, including how it's used and how it differs from the original ROS. (ROS Maker)

https://maker.pro/ros/tutorial/robot-operating-system-2-ros-2-introduction-and-getting-started

ROS 1 vs. ROS 2 Table Summary

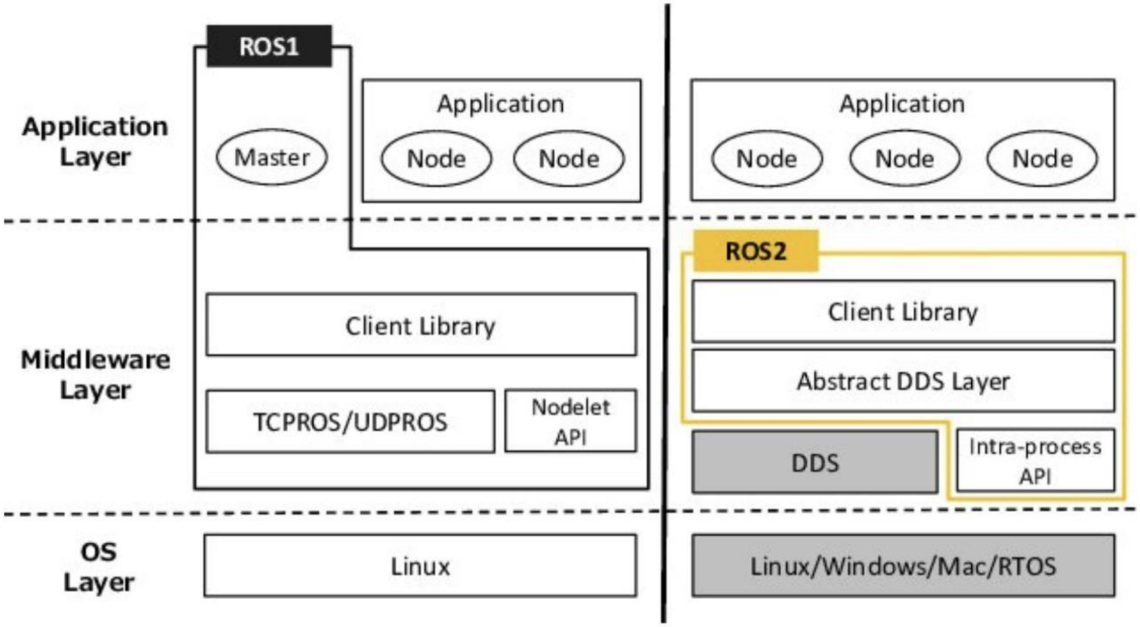
ROS	**ROS 2**
Uses TCPROS (custom version of TCP/IP) communication protocol	Uses DDS (Data Distribution System) for communication
Uses ROS Master for centralized discovery and registration. Complete communication pipeline is prone to failure if the master fails	Uses DDS distributed discovery mechanism. ROS 2 provides a custom API to get all the information about nodes and topics

ROS	**ROS 2**
ROS is only functional on Ubuntu OS	ROS 2 is compatible with Ubuntu, Windows 10 and OS X
Uses C++ 03 and Python2	Uses C++ 11 (potentially upgradeable) and Python3
ROS only uses CMake build system	ROS 2 provides options to use other build systems
Has a combined build for multiple packages invoked using a single CMakeLists.txt	Supports isolated independent builds for packages to better handle inter-package dependencies
Data Types in message files do not support default values	Data types in message files can now have default values upon initialization
roslaunch files are written in XML with limited capabilities	roslaunch files are written in Python to support more configurable and conditioned execution
Cannot support real-time behavior deterministically even with real-time OS	Supports real-time response with apt RTOS like RTPREEMPT

13_ROS2 ARCHITECTURE AND DDS

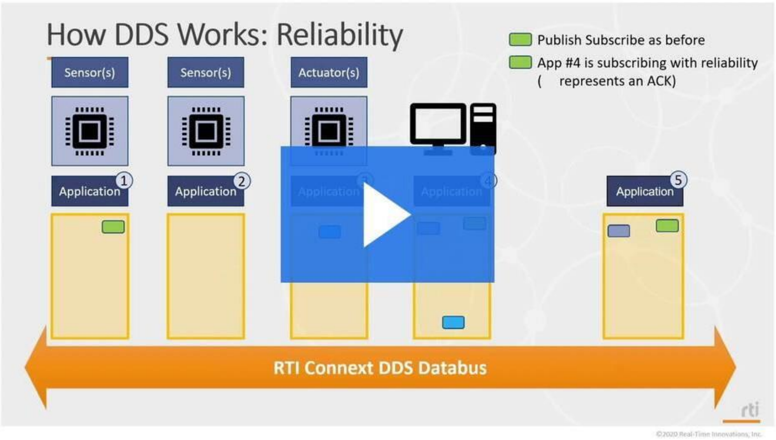
This is the big Claim to Fame for ROS 2 – No Master, Special Communications

13_a_Architecture



https://www.researchgate.net/publication/309128426_Exploring_the_performance_of_ROS2/figures?lo=1

13_b_Short_Video-of_DDS


How DDS Works: Reliability
🟩 Publish Subscribe as before
🟩 App #4 is subscribing with reliability (represents an ACK)
Sensor(s)
Sensor(s)
Actuator(s)
Application 1
Application 2
Application 3
Application 4
Application 5
RTI Connext DDS Databus
rti
©2020 Real-Time Innovations, Inc.

[Data Distribution Service (DDS) for Complex Systems | RTI]

13_c_DDS-Security overview

[https://design.ros2.org/articles/ros2_dds_security.html]

The [DDS-Security specification] expands upon the [DDS specification], adding security enhancements by defining a Service Plugin Interface (SPI) architecture, a set of builtin implementations of the SPIs, and the security model enforced by the SPIs. Specifically, there are five SPIs defined:

- **Authentication**: Verify the identity of a given domain participant.
- **Access control**: Enforce restrictions on the DDS-related operations that can be performed by an authenticated domain participant.
- **Cryptographic**: Handle all required encryption, signing, and hashing operations.

- **Logging**: Provide the ability to audit DDS-Security-related events.
- **Data tagging**: Provide the ability to add tags to data samples.

ROS 2's security features currently utilize only the first three. This is due to the fact that neither **Logging** nor **Data Tagging** are required in order to be compliant with the DDS-Security spec (see section 2.3), and thus not all DDS implementations support them. Let's delve a little further into those first three plugins.

13_d ROS on DDS

https://design.ros2.org/articles/ros_on_dds.html

14_INTRODUCTION TO REAL-TIME SYSTEMS

This article is a brief survey of real-time computing requirements and methods to achieve real-time performance.

Original Author: Jackie Kay

https://design.ros2.org/articles/realtime_background.html

Some examples of real-time environments:

- The `RT_PREEMPT` Linux kernel patch, which modifies the Linux scheduler to be fully preemptible (3).
- Xenomai, a POSIX-compliant co-kernel (or hypervisor) that provides a real-time kernel cooperating with the Linux kernel. The Linux kernel is treated as the idle task of the real-time kernel's scheduler (the lowest priority task).
- RTAI, an alternative co-kernel solution.
- QNX Neutrino, a POSIX-compliant real-time operating system for mission-critical systems.

14_b Mike Moore's Slides for Bobble Bot

Slides_Real-Time
Control of Balancing

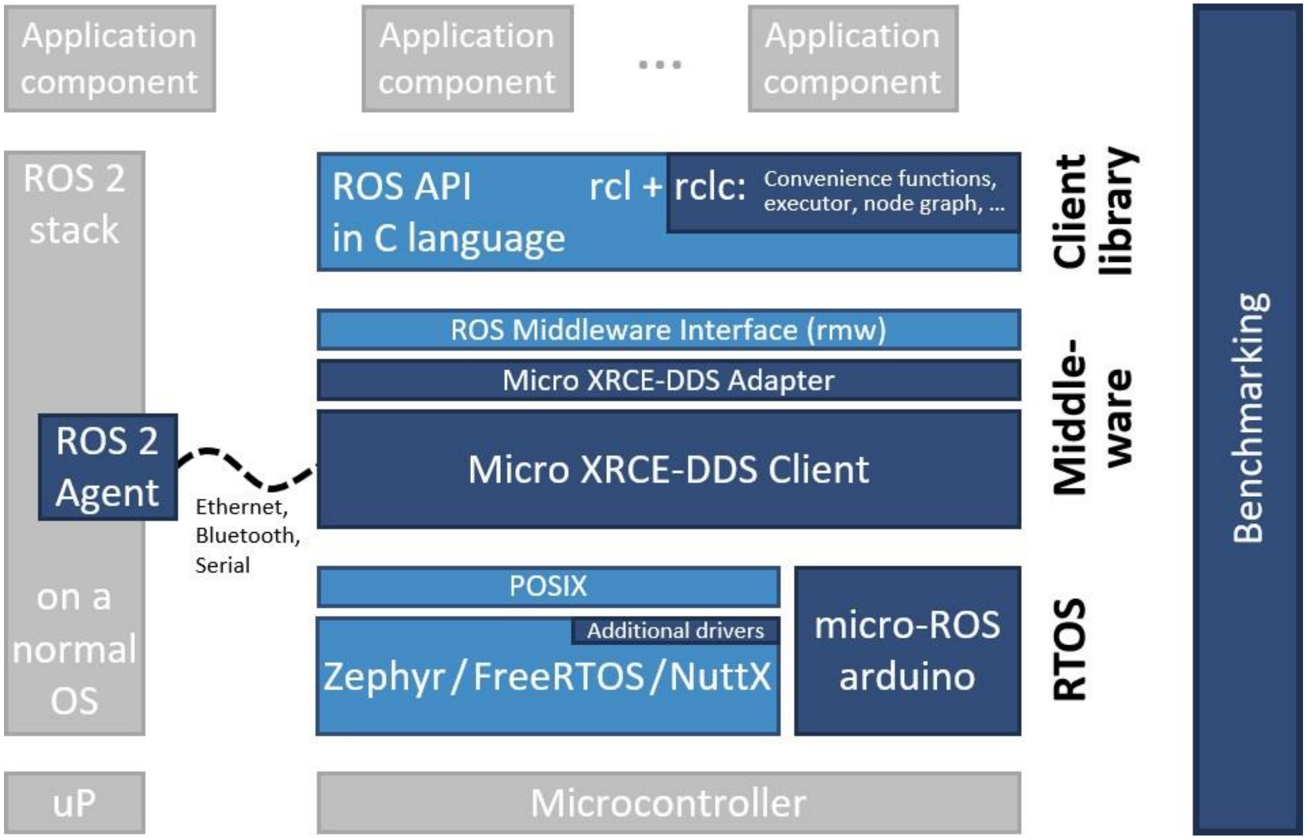
GET SLIDES 14_C FROM 5435 LECTURE REFERENCES 12/1/2020

14_c_Bobble_Bot_ super-owesome/bobble_controllers_ PREEMPT_RT

https://github.com/super-owesome/bobble_controllers

15_MICRO-ROS | ROS 2 FOR MICROCONTROLLERS

https://micro-ros.github.io/



micro-ROS- Short Video

micro-ROS puts the Robot Operating System on Microcontrollers _SHORT VIDEO
5,219 views •Nov 5, 2019 4:36

https://youtu.be/slMhPRnBVwM

ROS2 Videos

I have listed a few videos here that help us understand the changes in ROS 2. Any information presented before the official release of a distribution and anything before 2020 should be taken as “possibilities” rather than a formal description of the software. Check the date on any document or video.

ROS2 for Beginners from BotBuilder - 14 short videos.

14 videos7,729 viewsLast updated on Jun 16, 2020

A series of short videos averaging about 10 to 12 minutes about ROS2 including Python and C++ examples.

ROS2 Basics #1 - Installing and Configuring Your ROS2 Environment

ROS2 Basics #2 - Introducing Turtlesim, Command Line

ROS2 Basics #3 - Understanding ROS2 Packages and ...

ROS2 Basics #4 - Understanding ROS2 Executables and Nodes

ROS2 Basics #5 - Understanding ROS2 Topics - YouTube

ROS2 Basics #6 - Understanding ROS2 Services

ROS2 Basics #7 - Understanding ROS2 Actions

ROS2 Basics #8 - Understanding ROS2 Parameters

ROS2 Basics #9 - Understanding ROS2 Launch File

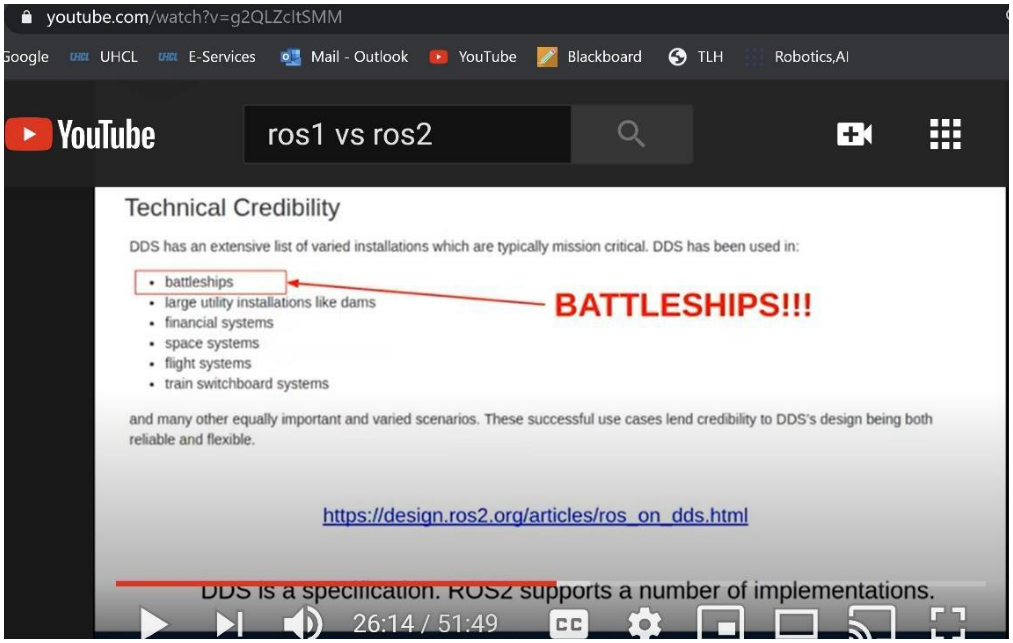
ROS2 Basics #10 - Writing a Simple Publisher and Subscriber

ROS2 Basics #11 - Writing a Simple Publisher and Subscriber

ROS2 Basics #12 - Writing a Simple Service and Client (C++)

ROS2 Basics #13 - Writing a Simple Service and Client ...

ROS2 Basics #14 - ROS2 tools - RQt and Ros2bag



Technical Credibility

DDS has an extensive list of varied installations which are typically mission critical. DDS has been used in:

- battleships
- large utility installations like dams
- financial systems
- space systems
- flight systems
- train switchboard systems

BATTLESHIPS!!!

and many other equally important and varied scenarios. These successful use cases lend credibility to DDS's design being both reliable and flexible.

https://design.ros2.org/articles/ros_on_dds.html

DDS is a specification. ROS2 supports a number of implementations.

His examples of Talker and Listener describe the changes in Python programs. Very detailed.

Do you really want to understand DDS?

ROS 2 + DDS Interoperation 1:03:00

371 views •Oct 8, 2020   1:03:00

https://www.youtube.com/watch?v=GGqcrccWfeE&t=8s

Starts with Vehicles

Uses of DDS

10:00 What is DDS?

43:00 ROS2 and DDS

Appendix ROS2 Cheat Sheet

ROS 2 Cheats Sheet

Command Line Interface

AU llOS 2 CLI t-061:; :;to.rt. wjLh th<:))rcfox 'roo2' roUowed by Q (:t)uttufUtd, fi , rh Aud (plh•:;ihly) pc,...:;:i1,iom11joptiounl tirguweutts.

Por n.uy t.ool, tJu",!or:1m1Pnl:1Jion iR :u:,:ft'l ihlP wi1l1.

```
$ ros2 command   -help
```

:md similarly for verb docun:ien1at.ion,

```
$ ros2 command verb   -h
```

Simjlit.rly o.uto-oowpletiou is onUJabl•·· for Qi.I c0tu· m.wcls/verbs and most JX>Sition.:i.l/optioual arguments. E.g.,

```
$ ros2 command    (tabl[tab(
```

Somt:,u(die t:xnmples he)o.,., rely un:

ROS 2 d,-wu. paclu1g<>.

action   AJJow;; to llliwm,Uy ::;cud ıgoo..Ln.:.ul dj:oplOy$ do-bugging information about t\Ctious.

Verbis:

info	Outp:11.iufrmuatiou >1h(mt. fut a(:ti<m.
list	Ouq,aL" list.of u,)tiou uruu0:s.
send.goal	Semi tlu act ion c:»LI.
show	Outp:1t J.110 ;u:tiou dE>Jiuition.

Examples:

```
$ r0s2 action info /libon.i«i
$ ros2 action list

$ r0s2 action send   gool /fibon.acci \

  actjon.tutorials/actjon/Fibonacci "ord4:!!r: 5"
$ n:.,s2.'tction show actio,uu10,iils/.lc.lioo/Fiboi,a«.i
```

bag AUows to record/play toJ>ics t•>/[rom a rosb..1g,

Verba:

info	Out.µut. jufoiumt.ivu <Jf a IJ&g.
pl:.y	Plf,y A h,'l_g.
record	He<:ord a bag.

list	Output a list, of rwminJ? containers and oomponcuta.
load	Load a comp,:ment into a container node.
standalone	Run t'l component into it..s own st.a.n-dalout <:outait1er uode.
	Output A lil'II ,,f ('..Ompont>lll}I l'P.l:,':il'llP. (I iu tLe mucnt iii<lex.
un!03d	uu1.,.,u.1 a ('Owpoueut from a oout.lliu r node.

8xamplc!i:

```
i  ros2 comp0nent list
! ros2 component lo.1d /ComponeotM;:in;:i,ger \
  composition composition::Talker
! ros2 component types
!  ros2 component un·0.-d /Componen;Manager l
```

daemon   VMir.,,s1IA1>11ıl>11 n•l>,l I wrh:-.

Vto'rbs:

start	St.art the daemon if i; isu·1 running.
status	Omput the st l.tu:.of tbe dac:mou.
stop	top thed..'U'moo if it is numiug

doctor   A t0()l to ch«!k ROS setup 1'l.lld oLhi?r JJOWuLiuJ i uessuch M ue1,work, pad,A,b'e ver.-it)LL", nuw midc.Ut WMQ ev.

AUt\.'>: wLr (wht:ni.s tl,e fira).

Arg,uucu :;;

--reportj-r	Ouqml, n pc)rt.of a!.I i::ltecks.
rep<>rt foil/ rr	OutJHtt. rtJJ0rt of!u.ik<lchtc:Lru only.
-·indu:;le-w;)rniog/-iw	Jucludt waruitlg-:1 tt.:S fo.iltl clt S.

E. aiupk:.'!:

```
i  ros2 doctOf

!  1os2 doctOf  --iepol't
i  ros2 docto..- --report-fail
!  ros2 doctOf --·nclude-waming
```

in terrace   Vt,riOlujHOS inLtl'fac:t:g{acli<Ju:j/lo1>h. /t1t:rvic:c:j)• reh:e.te<l \'<-rbs. h1U-rfaoo ty1,e <:au be iiJwre<.I wit.,h•ithcr of 1be rollowiug option. •--only-actions','••only-msgs·, •- only•

):l'\' •.

V€rl>.;;:

list	List i:111 iuterfr1ce l,yJJes avi'Lilable.
padtage	Outpul, n ljst of n,·nih,hle iule1f11<."8 types ,vithiu onP pM: P..
pac,kag	Outpul, >t list orp>l(::J:aie. that. i,nwide in-terf: .
proto	PriuL I.ht 1m:>1.-0ty1.>e(hody) of au iuter-foCO!l.
show	Output the iutert'.:ice defmition.

Exutupl<.'$:

```
S ros2 interface list

S ros2 interfoee p:.ck.lg'C std msg.-s

S ros2 interface packages --only-msgs
S ros2 interface pro:o example.intErfaces/srv/AddTwolots
S r(')!.2 interr <: shoo gom try ms.gs/mr:,g/Pose
```

l::wnch Allows t.o ruu ıı lllw1cl1 ti16 iu Wt ,,rl>it.rw:y p I) without to'al' lhere Jirat..

UMgt::

```
S ros2 launch <pjckJge> <launch•file>
```

Ex&uple:

```
S ros2 launch demo.nodes..cpp add_tm.intsJaunch.py
```

lifocyc,le   V U'ious lifocycle :rehd,ed verh.<i.

Ve,rbs:

gP.t	C:P.t. lifP"VdP sull.<!for on" or mrn•p flOflr .
list	Outpm; Ji:jt 1Jf av'._iilabl\: Lrtuu1.it.iuu .
nod«	Output a Jist 1.11' uoc.l.:,s with lifo<:yde.
set	Trigger Efecycle state t.rausition.

msg   (dE>prut:bte<l} Di.;;pl!\yS ddmg,giug iufonun.li<m abc)ut mes.-.nscs.

V€rbs·

Ex m1ple :

```
    $ ros2 info <.bag-name>
    $  ros2 play  <bag-n.lme>

    $ ros2 record •a
```

(OlllpOlletll	Vi:u·io•u.-. OOUll)OU(·?UI relAled \'('!rl.x,;.

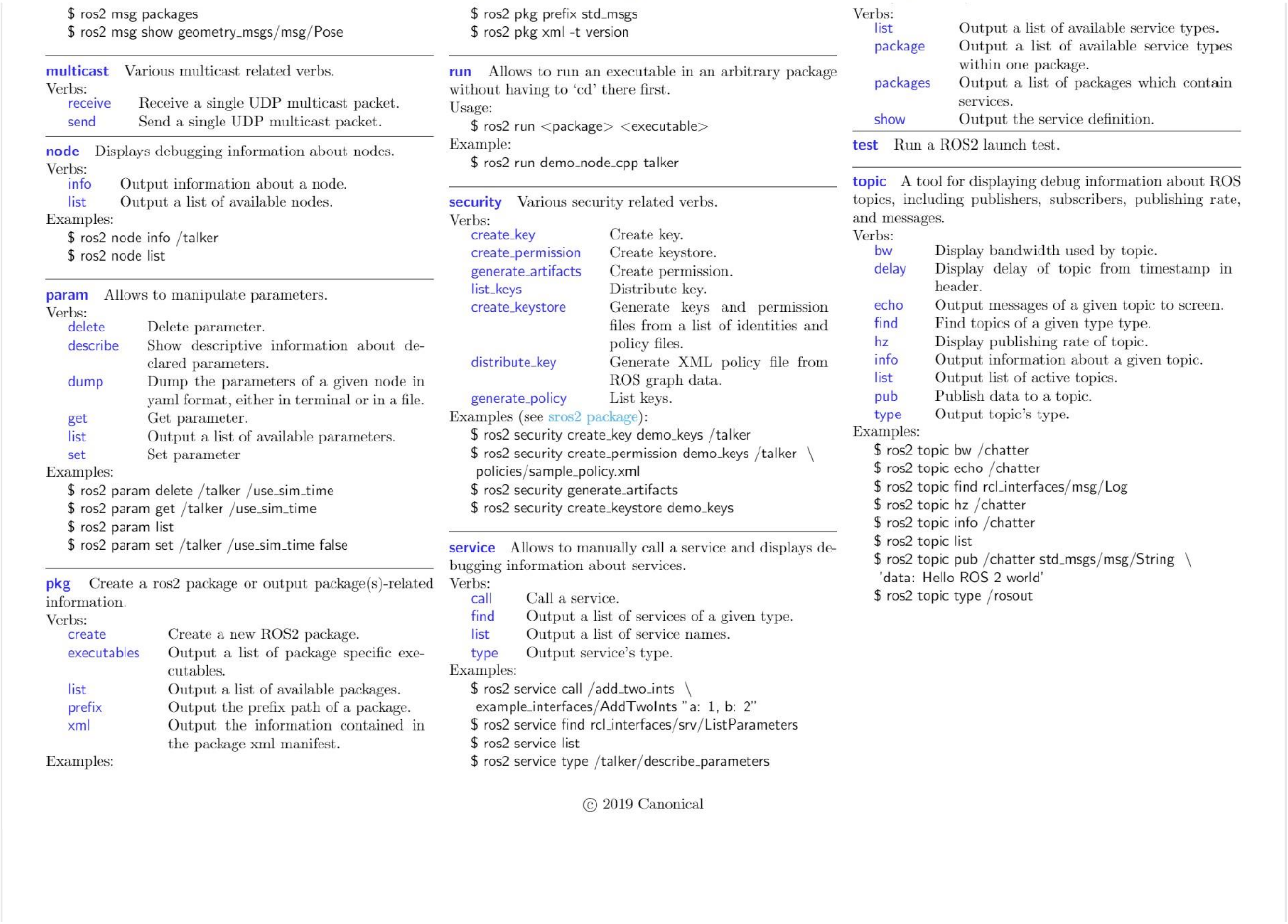
Vt rl,:.:

```
     S  ros2 docto, -·-ncludo-warning  --r8pOrt-f.lil
or  similaJ'Ly)
     S ros2 wtf
```

e.xtension_points	LL-.t e:d,e.r iou point:-,.
extensions	Li.-,1 e..\.letL ious.

list package	Outpul. A lfl1, of m o iypoo. Output a Jist. of message t}'l>es \Vitllin t\ 0-i,·eu packnge.
packages	Output a Jis1,of paclcages which couta.iu ute&s."lg(lS,
shO'w	Ontpul. tlu" 111 .:;.:.l.J.':" ,i<-finition.

l-l,ruu1,les:



```
$ ros2 msg packages
$ ros2 msg show geometry_msgs/msg/Pose
```

multicast Various multicast related verbs.
Verbs:

receive	Receive a single UDP multicast packet.
send	Send a single UDP multicast packet.

node Displays debugging information about nodes.
Verbs:

info	Output information about a node.
list	Output a list of available nodes.

Examples:

```
$ ros2 node info /talker
$ ros2 node list
```

param Allows to manipulate parameters.
Verbs:

delete	Delete parameter.
describe	Show descriptive information about declared parameters.
dump	Dump the parameters of a given node in yaml format, either in terminal or in a file.
get	Get parameter.
list	Output a list of available parameters.
set	Set parameter

Examples:

```
$ ros2 param delete /talker /use_sim_time
$ ros2 param get /talker /use_sim_time
$ ros2 param list
$ ros2 param set /talker /use_sim_time false
```

pkg Create a ros2 package or output package(s)-related information.
Verbs:

create	Create a new ROS2 package.
executables	Output a list of package specific executables.
list	Output a list of available packages.
prefix	Output the prefix path of a package.
xml	Output the information contained in the package xml manifest.

Examples:

```
$ ros2 pkg prefix std_msgs
$ ros2 pkg xml -t version
```

run Allows to run an executable in an arbitrary package without having to 'cd' there first.
Usage:

```
$ ros2 run <package> <executable>
```

Example:

```
$ ros2 run demo_node_cpp talker
```

security Various security related verbs.
Verbs:

create_key	Create key.
create_permission	Create keystore.
generate_artifacts	Create permission.
list_keys	Distribute key.
create_keystore	Generate keys and permission files from a list of identities and policy files.
distribute_key	Generate XML policy file from ROS graph data.
generate_policy	List keys.

Examples (see sros2 package):

```
$ ros2 security create_key demo_keys /talker
$ ros2 security create_permission demo_keys /talker \
  policies/sample_policy.xml
$ ros2 security generate_artifacts
$ ros2 security create_keystore demo_keys
```

service Allows to manually call a service and displays debugging information about services.
Verbs:

call	Call a service.
find	Output a list of services of a given type.
list	Output a list of service names.
type	Output service's type.

Examples:

```
$ ros2 service call /add_two_ints \
  example_interfaces/AddTwoInts "a: 1, b: 2"
$ ros2 service find rcl_interfaces/srv/ListParameters
$ ros2 service list
$ ros2 service type /talker/describe_parameters
```

Verbs:

list	Output a list of available service types.
package	Output a list of available service types within one package.
packages	Output a list of packages which contain services.
show	Output the service definition.

test Run a ROS2 launch test.

topic A tool for displaying debug information about ROS topics, including publishers, subscribers, publishing rate, and messages.
Verbs:

bw	Display bandwidth used by topic.
delay	Display delay of topic from timestamp in header.
echo	Output messages of a given topic to screen.
find	Find topics of a given type type.
hz	Display publishing rate of topic.
info	Output information about a given topic.
list	Output list of active topics.
pub	Publish data to a topic.
type	Output topic's type.

Examples:

```
$ ros2 topic bw /chatter
$ ros2 topic echo /chatter
$ ros2 topic find rcl_interfaces/msg/Log
$ ros2 topic hz /chatter
$ ros2 topic info /chatter
$ ros2 topic list
$ ros2 topic pub /chatter std_msgs/msg/String \
  'data: Hello ROS 2 world'
$ ros2 topic type /rosout
```