Mini Guide: Maintainer and License in ROS 2 Humble Python Packages

This guide explains why in ROS 2 Humble, when developing Python packages with `ament`, it is necessary to indicate maintainer and license in both `package.xml` and `setup.py`.

---

1. Role of `package.xml`

The `package.xml` file is the **standard ROS 2 manifest**. It is used by ROS and `ament` build tools to know:

- `<name>`: package name
- `<version>`: package version
- `<maintainer>`: who maintains the package
- `<license>`: package license
- `<depend>`: ROS dependencies

These data are used by ROS tools (`ros2 pkg`, `rosdep`, `ros2 launch`) and to generate `.deb` or `.zip` packages for distribution.

Minimal `package.xml` example:

```
<package format="3">
  <name>mypackage</name>
  <version>0.1.0</version>
  <description>Example ROS 2 Python package</description>
  <maintainer email="mario.rossi@example.com">Mario Rossi</maintainer>
  <license>Apache-2.0</license>
  <buildtool_depend>ament_cmake</buildtool_depend>
  <exec_depend>rclpy</exec_depend>
</package>
```

---

2. Role of `setup.py`

When the package is **Python**, `setup.py` is used by `setuptools`/`pip` to install the package. Here, metadata like `maintainer` and `license` are used by **Python** and tools like `pip show`.

Minimal `setup.py` example:

```
from setuptools import setup, find_packages

setup(
    name='mypackage',
```

```
    version='0.1.0',
    packages=find_packages(),
    maintainer='Mario Rossi',
    maintainer_email='mario.rossi@example.com',
    license='Apache-2.0',
    install_requires=['rclpy'],
)
```

> Without these data, users installing the Python package would not know the maintainer or the license.

---

3. Why specify data in both files

- **package.xml** → necessary for the **ROS system**
- **setup.py** → necessary for the **Python/pip system**

Situation	Consequence
Only `package.xml`	ROS knows maintainer/license, but `pip show` shows nothing
Only `setup.py`	Python knows, but ROS does not know the license/maintainer and ROS tools show warnings

---

4. Best Practices for ROS 2 Humble Python

1. In `package.xml` include at least: **name, version, license, maintainer, description, dependencies**
2. In `setup.py` include at least: **name, version, maintainer, license**
3. Ensure data are consistent between the two files to avoid warnings or inconsistencies between ROS and Python.

---

5. Practical Reference

A well-formed ROS2 Python package will have:

```
package.xml
setup.py
src/mypackage/__init__.py
```

With aligned metadata in both files to ensure compatibility with ROS 2, `pip`, and potential distributions.