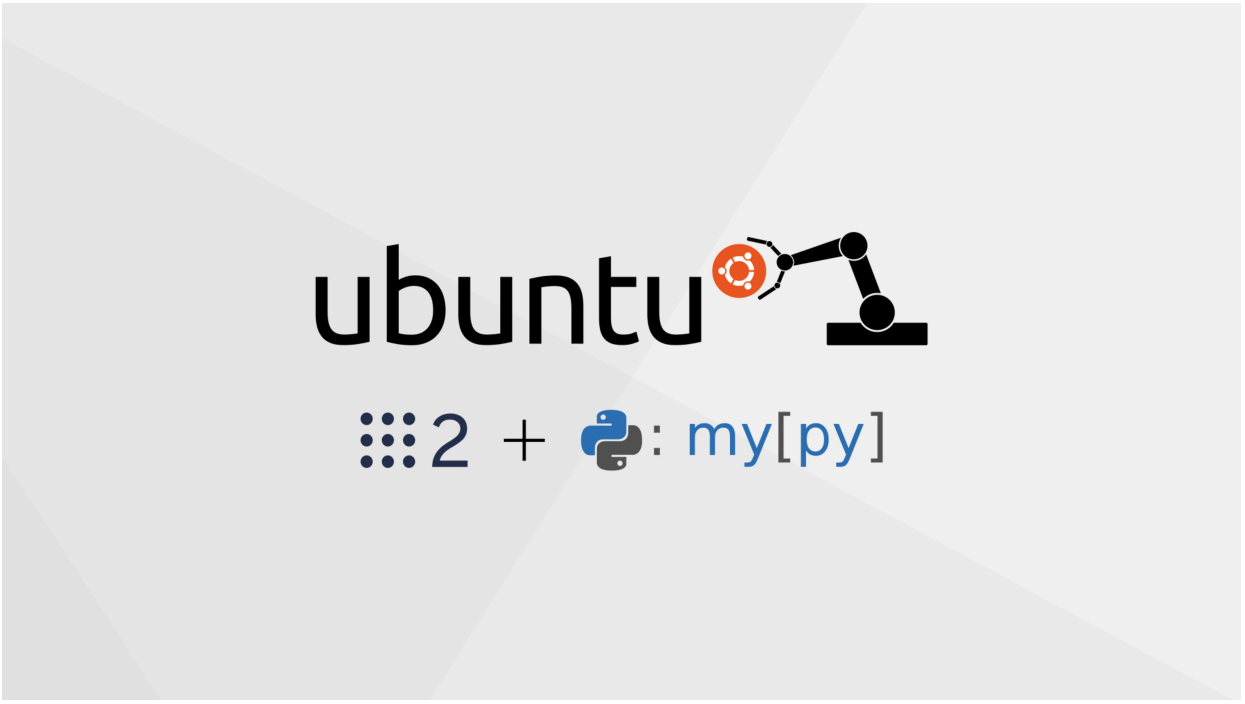
This article was last updated **1 year ago**.

Please note that this blog post has old information that may no longer be correct. We invite you to read the content as a starting point but please search for more updated information in the ROS documentation <https://docs.ros.org/>.



One of the most common complaints from developers moving into large Python codebases is the difficulty in figuring out type information, and the ease by which type mismatch errors can appear at runtime.

Python 3.5 added support for a type annotation system, described in PEP 484 <https://www.python.org/dev/peps/pep-0484/>. Python 3.6+ expands this with individual variable annotations (PEP 526 <https://www.python.org/dev/peps/pep-0526/>). While purely decorative and optional, a tool like mypy can use it to perform static type analysis and catch errors, just like compilers and linters for statically typed languages.

There are limitations to mypy, however. It only knows what it's explicitly told. Functions and classes without annotations are by default not checked, though they can be configured to default to `Any` or raise mypy errors.

The ROS 2 build farm is essentially only set up to run `colcon test`. As a result, any contributor wishing to use mypy currently needs to do so manually and hope that no other changes were made by someone not using annotations, or incorrectly annotating their code. This leads to many packages that are partially annotated, or with incorrect annotations ignored when by falling back to `Any`.

Seeking a fix that 1) helped us remember to check our contributions and 2) maintains a guarantee that packages that are annotated correctly stay so, we created a mypy linter for ament <https://github.com/ament/ament_lint/tree/master/ament_mypy> that can be integrated with the rest of the package test suite, allowing for mypy to be run automatically in the ROS 2 build farm and as part of the CI process. Now we can guarantee type correctness in our python code, and avoid the dreaded type mismatch errors!

ament_lint in action

The `ament_lint` metapackage defines many common linters that can integrate into the build/test pipeline for ROS 2. The package `ament_mypy` within handles mypy integration.

To add it as a test within your test suite, you'll need to make a few changes to your package:

- Add `ament_mypy` as a test dependency in your `package.xml`
- Add `pytest` as a test requirement in `setup.py`
- Write a test case that invokes `ament_mypy` and fails accordingly
- Add `ament_mypy` as a testing requirement to `CMakeLists.txt`, if using CMake

package.xml

For the first, find the section of your `package.xml` after the name/author/license information, where the dependencies are declared. Alongside the other depend blocks, add an entry

```
<test_depend>ament_mypy</test_depend>
```

setup.py

For `setup.py`, add the keyword argument

```
tests_require=['pytest']
```

if its not already present.

Test Case

Finally, we add a file `test/test_mypy.py`, that contains a call to `ament_mypy.main()`

```
from ament_mypy.main import main

import pytest


@pytest.mark.mypy
@pytest.mark.linter
def test_mypy():
    rc = main()
    assert rc == 0, 'Found code style errors / warnings'
```

If `ament_mypy.main()` returns non-zero, our test will fail and the error messages will display.

CMake

For configuring CMake, there are two options: manually list out each individual linter and run them, or use the `ament_lint_auto` convenience package to run all `ament_lint` dependencies.

In either case, `package.xml` needs to be configured as above, with an additional dependency of

```
<buildtool_depend>ament_cmake</buildtool_depend
```

To manually add `ament_mypy`, add the following code to your `CMakeLists.txt` file:

```
find_package(ament_cmake REQUIRED)
if(BUILD_TESTING)
  find_package(ament_cmake_mypy REQUIRED)
  ament_cmake_mypy()
endif()
```

To use `ament_lint_auto`, add it as a test dependency to `package.xml`

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

And add the following to `CMakeLists.txt`, before the `ament_package()` call

```
# this must happen before the invocation of ament_package()
if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  ament_lint_auto_find_test_dependencies()
endif()
```

(Optional) Configuring mypy

To pass custom configurations to mypy, you can specify a '.ini' configuration file (documented here <https://mypy.readthedocs.io/en/stable/config_file.html>) in the arguments to main.

setup.py

Create a `config` directory under `test`, and a `mypy.ini` file within. Fill the file with your custom configuration, e.g.:

```
# Global options:

[mypy]
python_version = 3.5
warn_return_any = True
warn_unused_configs = True

# Per-module options:

[mypy-mycode.foo.*]
disallow_untyped_defs = True

[mypy-mycode.bar]
warn_return_any = False

[mypy-somelibrary]
ignore_missing_imports = True
```

In setup.py, pass in the `--config` option with the path to your desired file.

```
from pathlib import Path

from ament_mypy.main import main

import pytest


@pytest.mark.mypy
@pytest.mark.linter
def test_mypy():
    config_path = Path(__file__).parent / 'config' / 'mypy.ini'
    rc = main(argv=['--exclude', 'test', '--config', str(config_path.resolve())])
    assert rc == 0, 'Found code style errors / warnings'
```

CMake

When using CMake, you'll need to pass the `CONFIG_FILE` arg. In the manual invocation example, that means changing the `BUILD_TESTING` block as follows (assuming your `mypy.ini` file is in the same directory as above):

```
find_package(ament_cmake REQUIRED)
if(BUILD_TESTING)
  find_package(ament_cmake_mypy REQUIRED)
  ament_cmake_mypy(CONFIG_FILE "${CMAKE_CURRENT_LIST_DIR}/test/config/mypy.ini")
endif()
```

The additional argument means `ament_cmake_mypy` cannot be auto invoked by `ament_lint_auto`. If you're already using `ament_lint_auto` for other packages, you'll need to exclude `ament_mypy`.

To exclude `ament_cmake_mypy`, set the `AMENT_LINT_AUTO_EXCLUDE` variable and then manually find and invoke it:

```
# this must happen before the invocation of ament_package()
if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  list(APPEND AMENT_LINT_AUTO_EXCLUDE
    ament_cmake_mypy
  )
  ament_lint_auto_find_test_dependencies()

  find_package(ament_cmake_mypy REQUIRED)
  ament_cmake_mypy(CONFIG_FILE "${CMAKE_CURRENT_LIST_DIR}/test/config/mypy.ini")
endif()
```

Running the Test

To run the test and get output to the console, run the following in your workspace:

```
colcon test -event-handlers console_direct+
```

To test only your package:

```
colcon test --packages-select <YOUR_PACKAGE> --event-handlers console_direct+
```