xeus-python

Martin Renou, Johan Mabille, Sylvain Corlay and Wolf Vollprecht

Jul 22, 2020

INSTALLATION

`xeus-python` is a Jupyter kernel for Python based on the native implementation of the Jupyter protocol xeus.

CHAPTER

ONE

LICENSING

We use a shared copyright model that enables all contributors to maintain the copyright on their contributions.

This software is licensed under the BSD-3-Clause license. See the LICENSE file for details.

1.1 Installation

1.1.1 With Conda

xeus-python has been packaged for the conda package manager.

To ensure that the installation works, it is preferable to install *xeus-python* in a fresh conda environment. It is also needed to use a miniconda installation because with the full anaconda you may have a conflict with the *zeromq* library which is already installed in the anaconda distribution.

The safest usage is to create an environment named *xeus-python* with your miniconda installation

```
conda create -n xeus-python
conda activate xeus-python # Or `source activate xeus-python` for conda < 4.6
```

Then you can install in this freshly created environment *xeus-python* and its dependencies

```
conda install xeus-python notebook -c conda-forge
```

or, if you prefer to use JupyterLab

```
conda install xeus-python jupyterlab -c conda-forge
```

1.1.2 From PyPI

Depending on the platform, PyPI wheels may be available for xeus-python.

```
pip install xeus-python notebook
```

However, the wheels uploaded on PyPI are **experimental**.

In general we strongly recommend using a package manager instead. We maintain the conda-forge package, and nothing prevents you from creating a package your favorite Linux distribution or FreeBSD.

The ongoing effort to package xeus-python for pip takes place in the xeus-python-wheel repository.

1.1.3 From Source

You can install `xeus-python` from source with cmake. This requires that you have all the dependencies installed in the same prefix.

```
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/path/to/prefix ..
make install
```

On Windows platforms, from the source directory:

```
mkdir build
cd build
cmake -G "NMake Makefiles" -DCMAKE_INSTALL_PREFIX=/path/to/prefix ..
nmake
nmake install
```

1.2 Installing the Kernel Spec

When installing xeus-python in a given installation prefix, the corresponding Jupyter kernelspecs are installed in the same environment and are automatically picked up by Jupyter if it is installed in the same prefix.

However, if Jupyter is installed in a different location, it will not pick up the new kernel. The xeus-python can be registered with the following command:

```
jupyter kernelspec install PREFIX/share/jupyter/xpython --sys-prefix
```

For more information on the `jupyter kernelspec` command, please consult the `jupyter_client` documentation.

1.3 Usage

Launch the Jupyter notebook with *jupyter notebook* or Jupyter lab with *jupyter lab* and launch a new Python notebook by selecting the **xpython** kernel.

1.3.1 Code execution and variable display

1.3.2 Output streams

1.3.3 Input streams

1.3.4 Error handling

1.3.5 Inspect

1.3.6 Code completion

1.3.7 Rich display

1.3.8 And of course widgets

1.4 Build and configuration

1.4.1 General Build Options

Building the xeus-python library

`xeus-python` build supports the following options:

- `XPYT_BUILD_SHARED`: Build the xeus-python shared library. **Enabled by default**.
- `XPYT_BUILD_STATIC`: Build the xeus-python static library. **Enabled by default**.

Xeus-python must link with xeus dynamically or statically.

- `XPYT_USE_SHARED_XEUS`: Link with the xeus shared library (instead of the static library). **Enabled by default**.

Building the kernel

The package includes two options for producing a kernel: an executable `xpython` and a Python extension module, which is used to launch a kernel from Python.

- `XPYT_BUILD_XPYTHON_EXECUTABLE`: Build the xpython executable. **Enabled by default**.
- `XPYT_BUILD_XPYTHON_EXTENSION`: Build the Python extension. **Disabled by default**.

The build of the Python extension is used in the context of the PyPI package to support virtual environments and user installations.

Both target make use of the two following options:

- `XPYT_USE_SHARED_XEUS_PYTHON`: Link xpython with the xeus-python shared library. **Enabled by default**.

If `XPYT_USE_SHARED_XEUS_PYTHON` is disabled, xpython will be linked statically with xeus-python.

Building the Tests

- `XPYT_BUILD_TESTS`: enables the `xtest` and `xbenchmark` targets (see below). **Disabled by default**.
- `XPYT_DOWNLOAD_GTEST`: downloads `gtest` and builds it locally instead of using a binary installation. **Disabled by default**.
- `XPYT_GTEST_SRC_DIR`: indicates where to find the `gtest` sources instead of downloading them. **Unset by default**.

Enabling `XPYT_DOWNLOAD_GTEST` or setting `XPYT_GTEST_SRC_DIR` enables `XPYT_BUILD_TESTS`. If the `XPYT_BUILD_TESTS` option is enabled, the *xtest* target is made available, which builds and runs the test suite.

Other options

- `XPYT_ENABLE_PYPI_WARNING`: We enable this option when building PyPI wheel to show a warning discouraging the use of PyPI. **Disabled by default**.
- `XEUS_PYTHONHOME_RELPATH`: indicates the relative path of the PYTHONHOME with respect to the installation prefix of the `xpython` target. This variable is unset by default.

By default, `XEUS_PYTHONHOME_RELPATH` is unset and the PYTHONHOME is set to the installation prefix, which is the expected behavior for most cases. A situation in which we may need to specify a different value for `XEUS_PYTHONHOME_RELPATH` is when using a Python installation from a different prefix. This occurs for example when building the conda package for xeus-python windows, since Python is installed in the general `PREFIX` while xeus-python is installed in the `LIBRARY_PREFIX`.