xeus

Dec 19, 2018

INSTALLATION

C++ implementation of the Jupyter Kernel protocol

`xeus` is a library meant to facilitate the implementation of kernels for Project Jupyter. It takes the burden of implementing the Jupyter Kernel protocol so developers can focus on implementing the interpreter part of the kernel.

CHAPTER 1

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` has been packaged on all platforms for the conda package manager.

```
conda install xeus -c QuantStack -c conda-forge
```

1.1.2 From Source

`xeus` depends on the following libraries:

- libzmq, cppzmq, cryptopp and xtl

On linux platforms, `xeus` also requires libuuid, which is available in all linux distributions.

We have packaged all these dependencies for the conda package manager. The simplest way to install them with conda is to run:

```
conda install cmake zeromq cppzmq cryptopp xtl -c QuantStack -c conda-forge .
```

On Linux platform, you will also need:

```
conda install libuuid -c conda-forge
```

Once you have installed the dependencies, you can build and install *xeus*:

```
cmake -D BUILD_EXAMPLES=ON -D CMAKE_BUILD_TYPE=Release .
make
make install
```

If you need the `xeus` library only, you can omit the `BUILD_EXAMPLES` settings.

1.1.3 Installing the Dependencies from Source

The dependencies can also be installed from source. Simply clone the directories and run the following cmake and make instructions.

libzmq

```
cmake -D WITH_PERF_TOOL=OFF -D ZMQ_BUILD_TESTS=OFF -D ENABLE_CPACK=OFF -D CMAKE_BUILD_
↪TYPE=Release .
make
make install
```

cppzmq

cppzmq is a header only library:

```
cmake -D CMAKE_BUILD_TYPE=Release .
make install
```

cryptopp

cryptopp must be built as a static library, building cryptopp as a shared library is not supported on Windows.

```
cmake -D BUILD_SHARED=OFF -D BUILD_TESTING=OFF -D CMAKE_BUILD_TYPE=Release .
make
make install
```

xtl

xtl is a header only library:

```
cmake -DCMAKE_BUILD_TYPE=Release .
make install
```

1.2 Usage

`xeus` enables custom kernel authors to implement Jupyter kernels more easily. It takes the burden of implementing the Jupyter Kernel protocol so developers can focus on implementing the interpreter part of the Kernel.

The easiest way to get started with a new kernel is to inherit from the base interpreter class `xeus::xinterpreter` and implement the private virtual methods

- `execute_request_impl`

- complete_request_impl
- inspect_request_impl
- is_complete_request_impl

as seen in the echo kernel provided as an example.

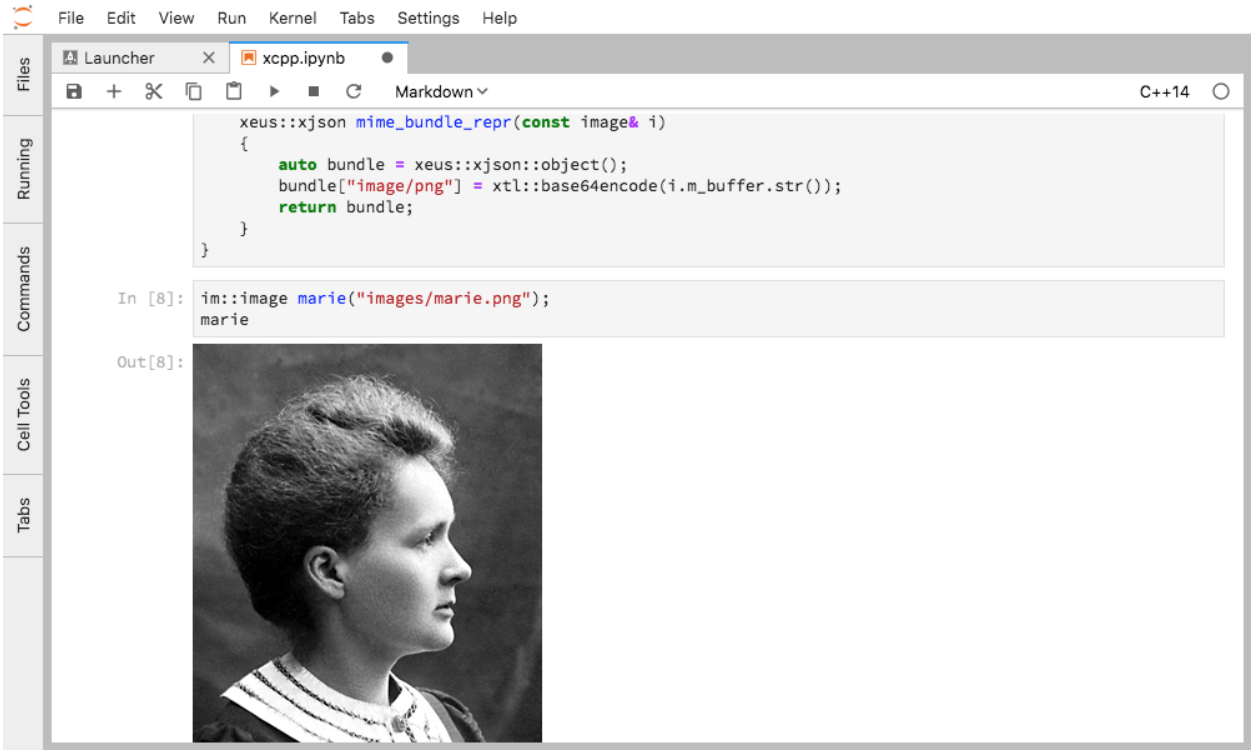
```
#include "xeus/xinterpreter.hpp"

using xeus::xinterpreter;
using xeus::xjson;

namespace echo_kernel
{
    class echo_interpreter : public xinterpreter
    {

    public:

        echo_interpreter() = default;
        virtual ~echo_interpreter() = default;

    private:

        void configure_impl() override;

        xjson execute_request_impl(int execution_counter,
                                   const std::string& code,
                                   bool silent,
                                   bool store_history,
                                   const xeus::xjson_node* user_expressions,
                                   bool allow_stdin) override;

        xjson complete_request_impl(const std::string& code,
                                    int cursor_pos) override;

        xjson inspect_request_impl(const std::string& code,
                                   int cursor_pos,
                                   int detail_level) override;

        xjson is_complete_request_impl(const std::string& code) override;

        xjson kernel_info_request_impl() override;
    };
}
```

Kernel authors can then rebind to the native APIs of the interpreter that is being interfaced, providing richer information than with the classical approach of a wrapper kernel capturing textual output.

1.3 Related projects

1.3.1 xeus-cling

The xeus-cling project is a Jupyter kernel for the C++ programming language based on the Cling C++ interpreter from CERN and Xeus, the native implementation of the Jupyter protocol.



1.3.2 xeus-python

The xeus-python project is a Jupyter kernel for the Python programming language based on the Xeus implementation of the protocol.

1.3.3 xwidgets

The xwidgets project is a C++ implementation of the Jupyter interactive widgets protocol. The Python reference implementation is available in the ipywidgets ipywidgets project.

xwidgets enables the use of the Jupyter interactive widgets in the C++ notebook, powered by the xeus-cling kernel and the cling C++ interpreter from CERN. xwidgets can also be used to create applications making use of the Jupyter interactive widgets without the C++ kernel per se.

1.3.4 JuniperKernel

The JuniperKernel project is a Jupyter kernel for the R programming language built with Xeus and Rcpp.