QuantStack

Jupyter: Humans in the Read-Eval-Print-Loop

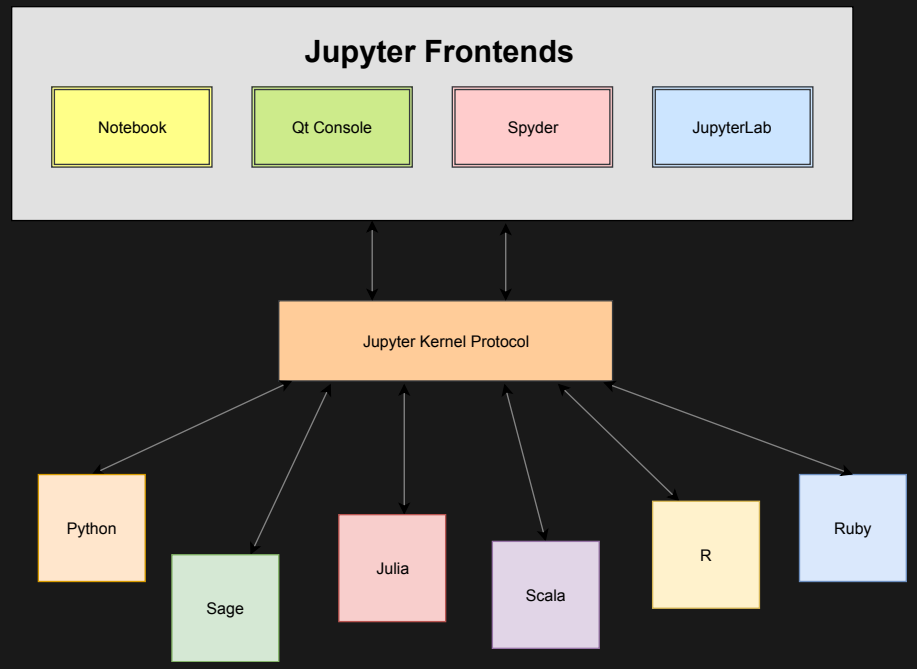
@JohanMabille @QuantStack

What is Jupyter?

A consistent set of tools (protocols, standards, libraries) meant to improve the workflow of engineers, scientists

- from the *exploratory* phase of their work
- to the *communication* of their result
- including all the intermediary steps

THE JUPYTER ARCHITECTURE



- A well-specified protocol built upon web standards
- Implemented for more than 40 languages

The Jupyter ecosystem: main apps

- Jupyter Lab
- Notebook
- Voilà
- Qt Console
- Jupyter Hub

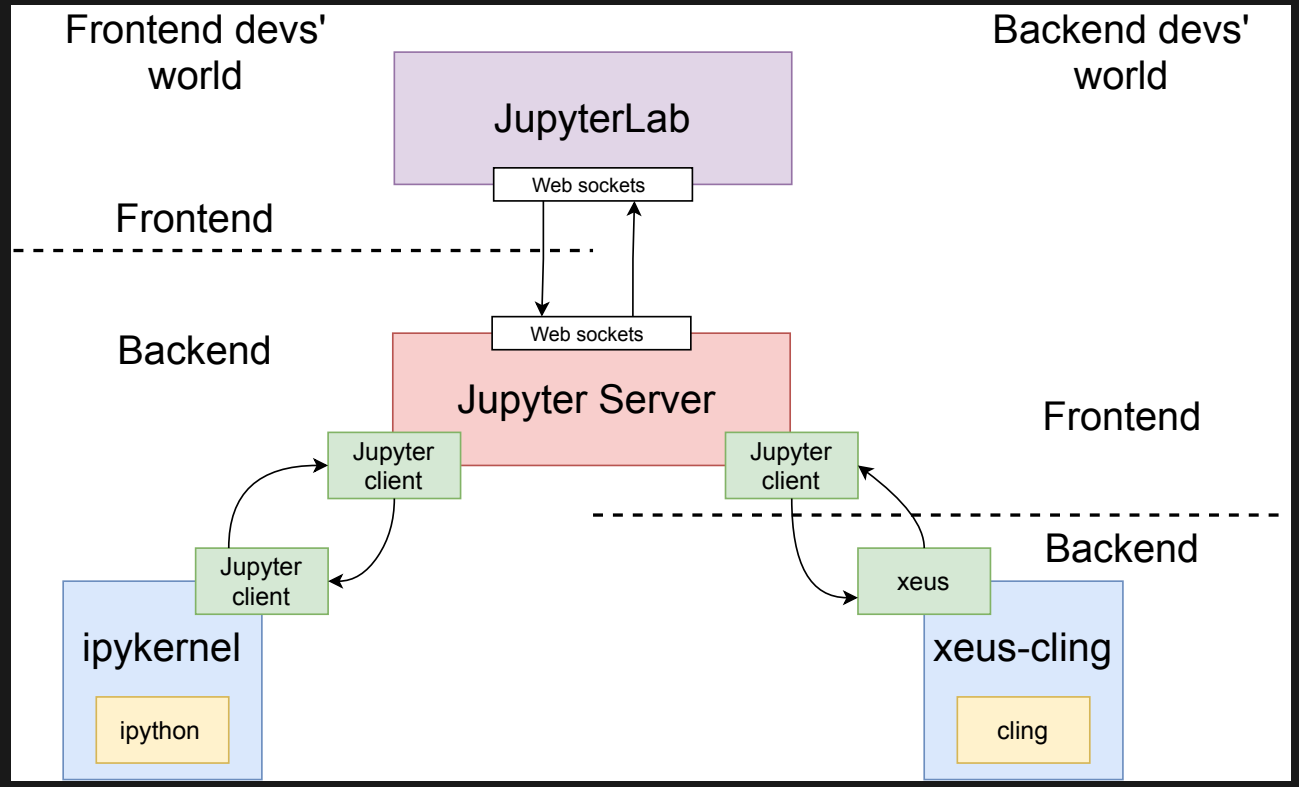
The Jupyter ecosystem: core libraries (Python)

- jupyter_client
- jupyter_server
- ipython
- ipykernel
- nbconvert
- ipywidgets

The Jupyter ecosystem: the Xeus stack (C++)

- xeus
- xeus-cling
- xeus-python
- xeus-sqlite
- xeus-sql
- xwidgets
- xena (coming soon...)

THE JUPYTER ARCHITECTURE



THE JUPYTER PROTOCOL

Clients and kernels communicate through 5 channels

- Shell: code execution, code completion
- Control: stop and restart, kernel info, debugging
- stdin: input request
- IOPub: broadcast channel to publish results and kernel state
- Heartbeat: to check the kernel is still alive

MESSAGE FORMAT

- Header: identifiers, message type, protocol version
- Parent header: when the message is the "result" of another one
- Metadata: additional information
- Content: body of the message
- Buffers: binary buffers for perfomance considerations

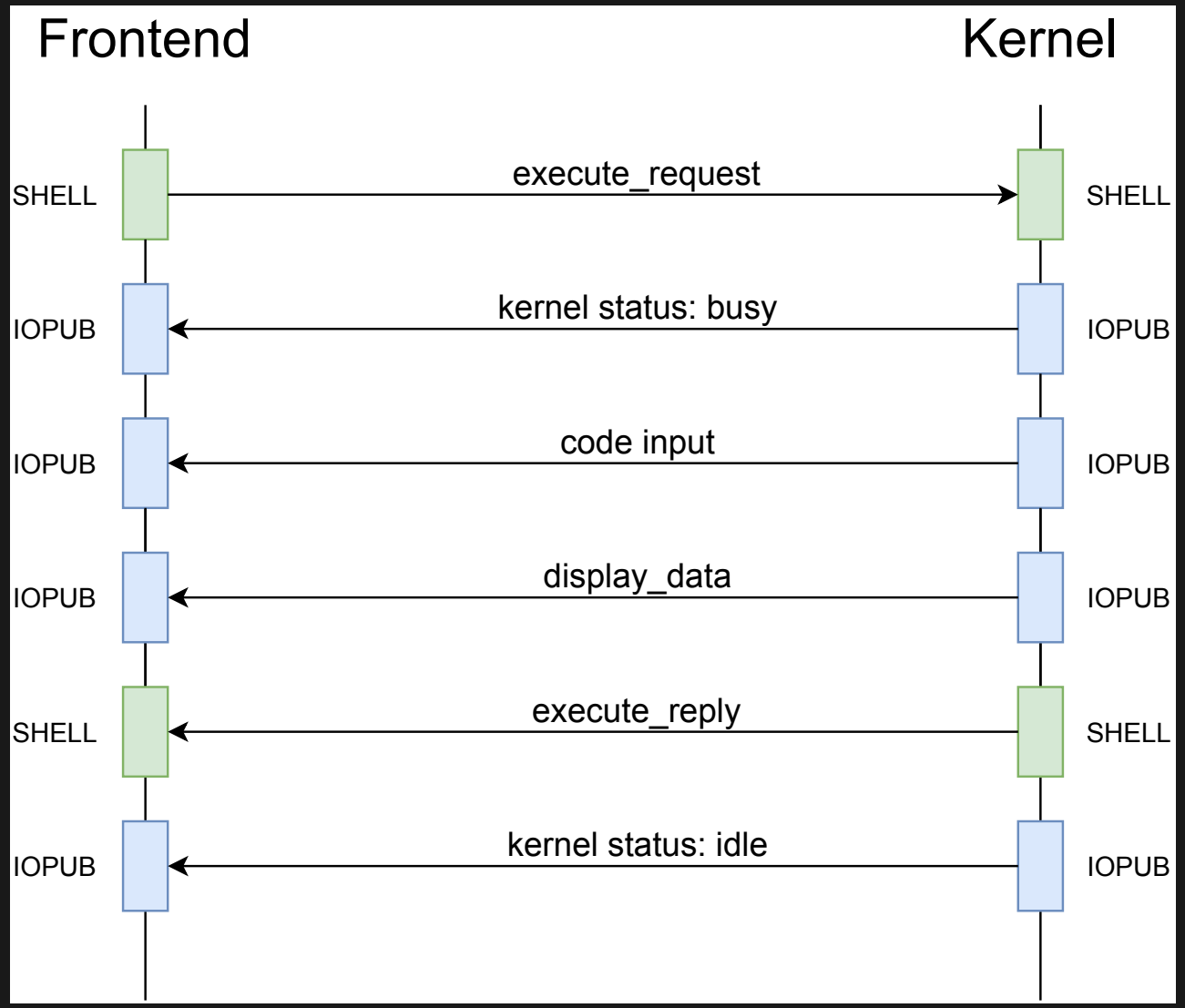
MESSAGE FORMAT

```
{
    "header" : {
        "msg_id": "...",
        "msg_type": "...",
        ...
    },
    "parent_header": {},
    "metadata": {},
    "content": {},
    "buffers": [],
}
```

WIRE PROTOCOL

```
[
    b'u-u-i-d',          # zmq identy(ies)
    b'<IDS|MSG>',        # delimiter
    b'baddad42',         # HMAC signature
    b'{header}',         # serialized header dict
    b'{parent_header}', # serialized parent header dict
    b'{metadata}',       # serialized metadata dict
    b'{content}',        # serialized content dict
    b'\xf0\x9f\x90\xb1' # extra raw data buffer(s)
    ...
]
```

MESSAGE ACTIVITY



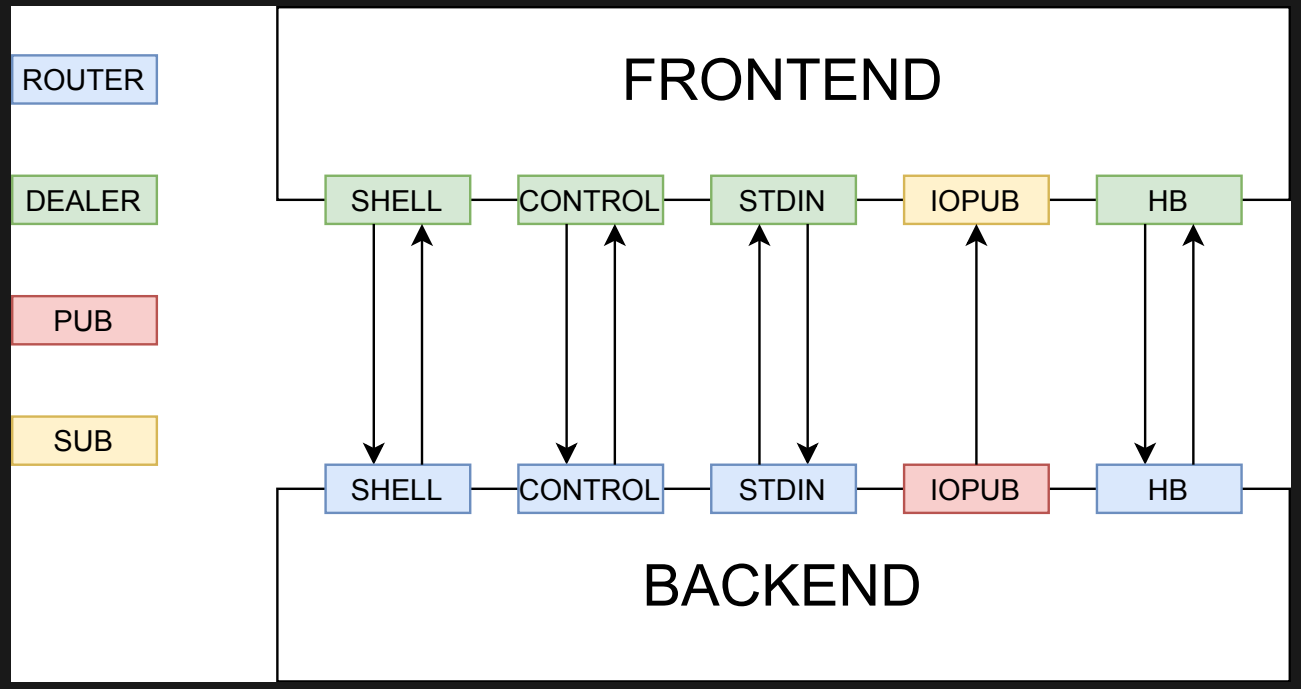
ZEROMQ SOCKETS

- REQ: sends a request, blocks until it receives a reply
- REP: waits for a request and sends a reply
- ROUTER: can handle multi requests from many sockets (REQ or DEALER)
- DEALER: can send multi request to many sockets (REP or ROUTER)
- PUB: publisher socket, sends messages to all its suscribers
- SUB: subscriber, connects to a PUB sockets and can filter messages
- XPUB: publisher tracking new subscribers
- XSUB: subscriber tracking new publishers

ZEROMQ COMMON PATTERNS

- REQ - REP : 1 - 1 (blocking - blocking)
- REQ - ROUTER : N - 1 (blocking - async)
- DEALER - REP : 1 - N (async - blocking)
- DEALER - ROUTER : N - N (async - async)
- PUB - SUB
- PUB - XSUB
- XPUB - SUB

ZEROMQ SOCKETS



END