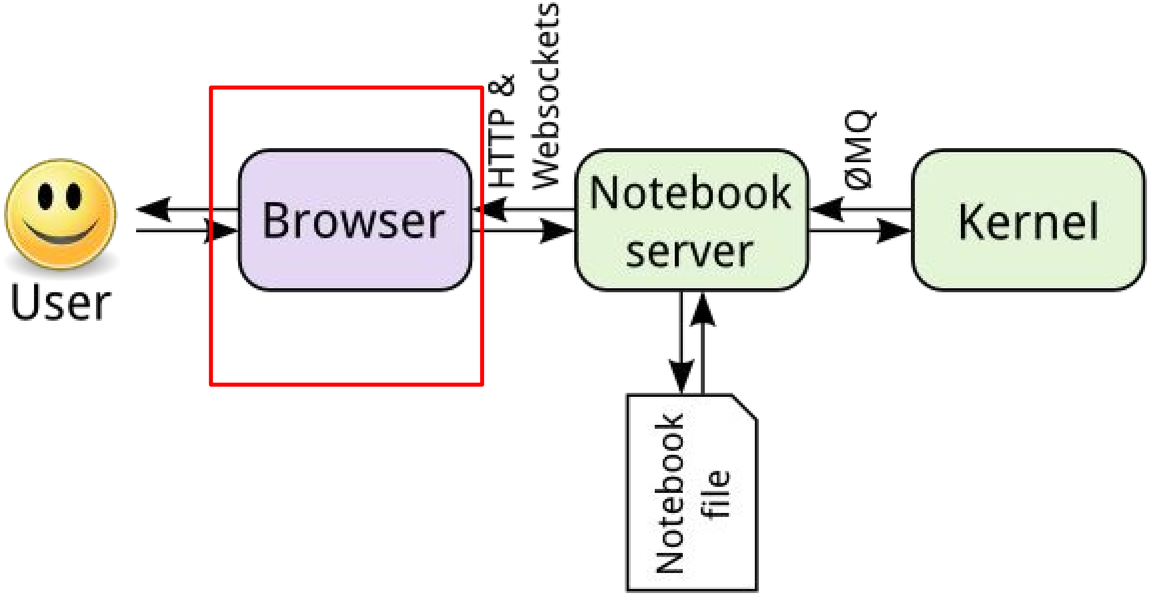
Jupyter Architecture

An overview

What is going on under the hood?
Let’s start with the Browser.



Browser

- You are viewing/editing a document

```
{
 "cells": [...],
 "metadata": {
  "kernelspec": { "display_name": "Python 3", "language": "python", "name": "python3"},

   ...
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}
```

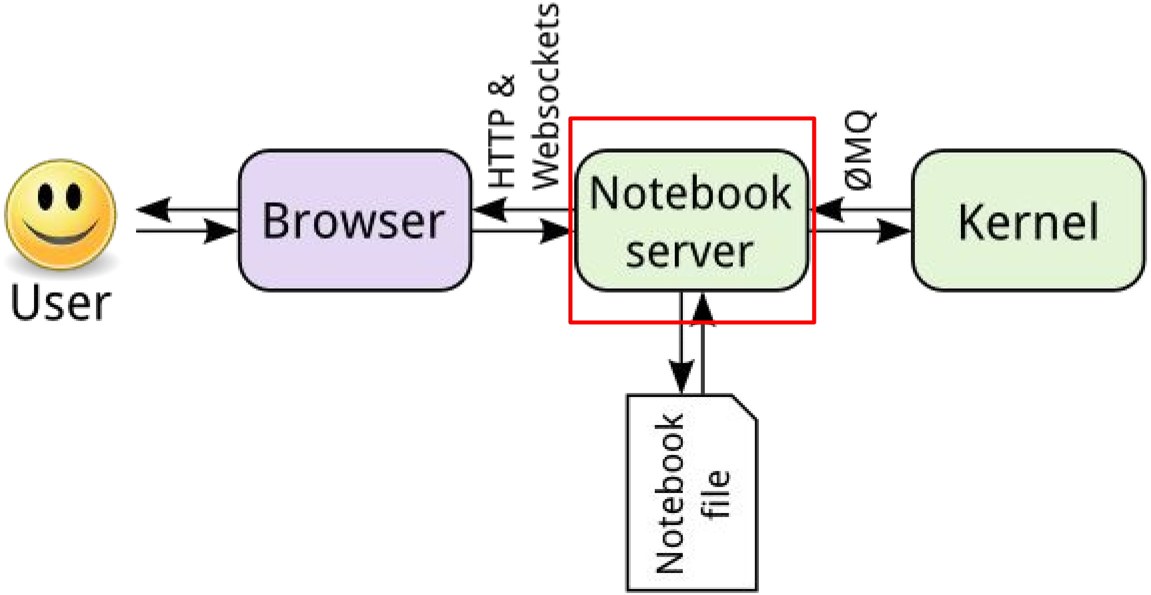
Browser

- Jupyter’s Notebook frontend handles the following
  - User input
  - Rendering (Markdown, HTML, Javascript)
  - Sending changes to the Notebook Server
  - Sending code execute requests to the Notebook Server

Browser

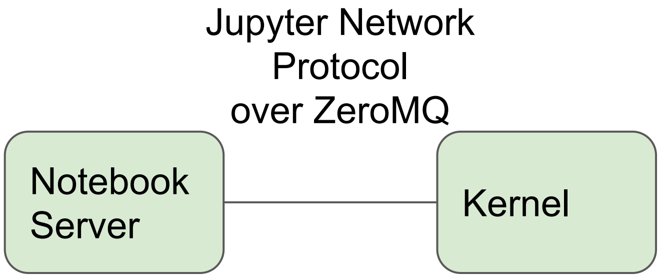
- Communicates using HTTP and Websockets
  - Websockets allow for a 2-way link between your browser and the server
  - Network client

Notebook Server



Notebook Server

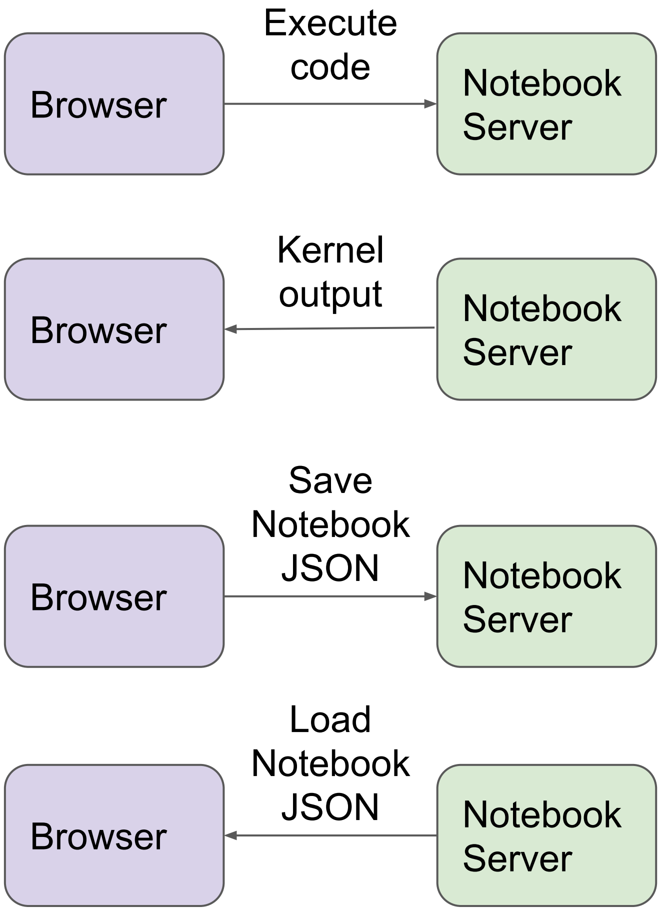
- Responsibilities
  - Saves/Loads notebook files
  - Spawns a kernel for you
  - Shuts down kernels
  - Communicates with the kernel
    - uses the ZeroMQ (Message Queue) protocol



Jupyter Network Protocol over ZeroMQ

Notebook Server

Kernel



Execute code

Browser

Notebook Server

Kernel output

Browser

Notebook Server

Save Notebook JSON

Browser

Notebook Server

Load Notebook JSON

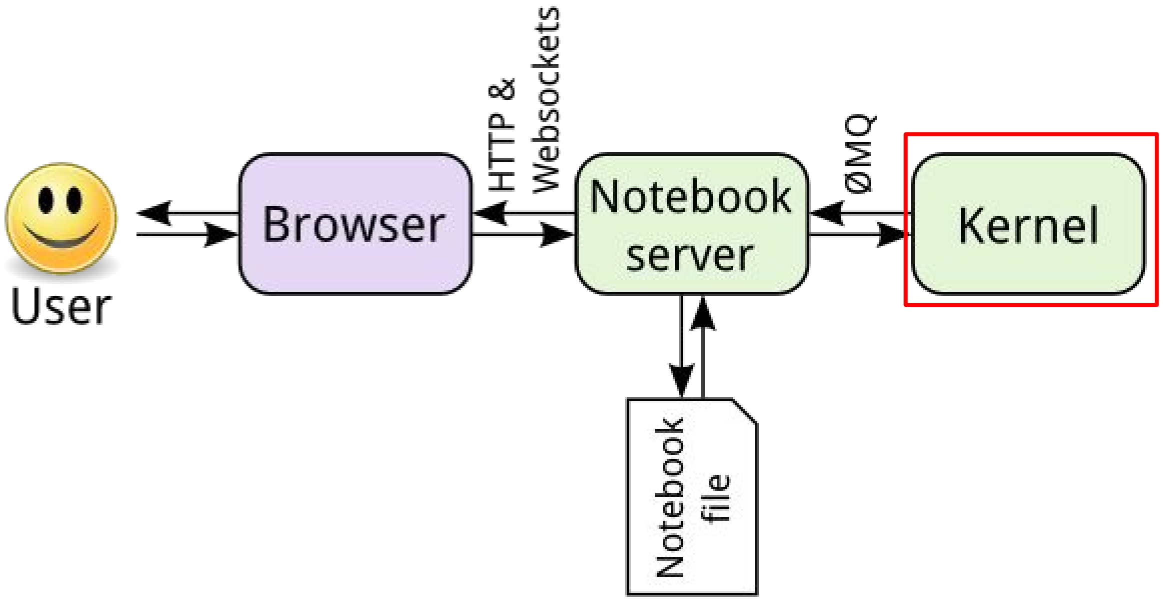
Browser

Notebook Server

Notebook Server

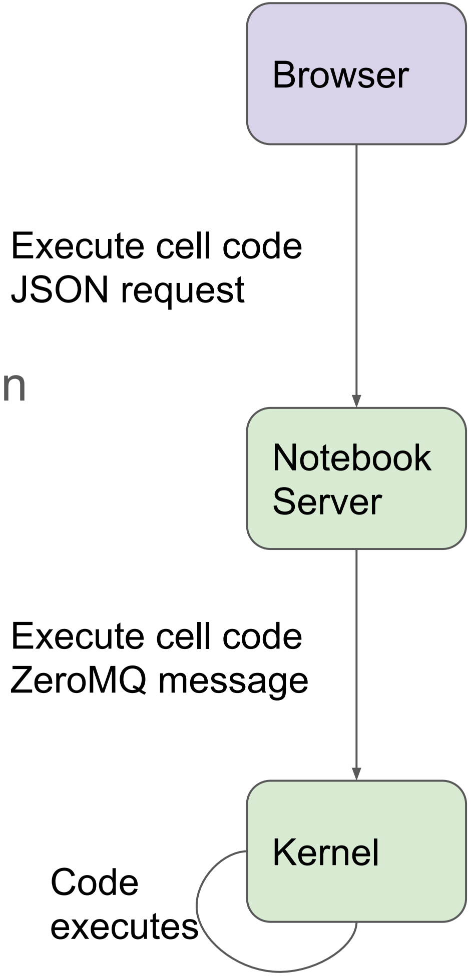
- REST API
- Handles Notebook File Menu requests (Download as)
- IPyparallel cluster management

Kernel


User
Browser
HTTP &
Websockets
Notebook
server
ØMQ
Kernel
Notebook
file

Kernel

- Where user code execution happens
- Separate process from the Notebook Server
- Also handles other types of requests, like code completion

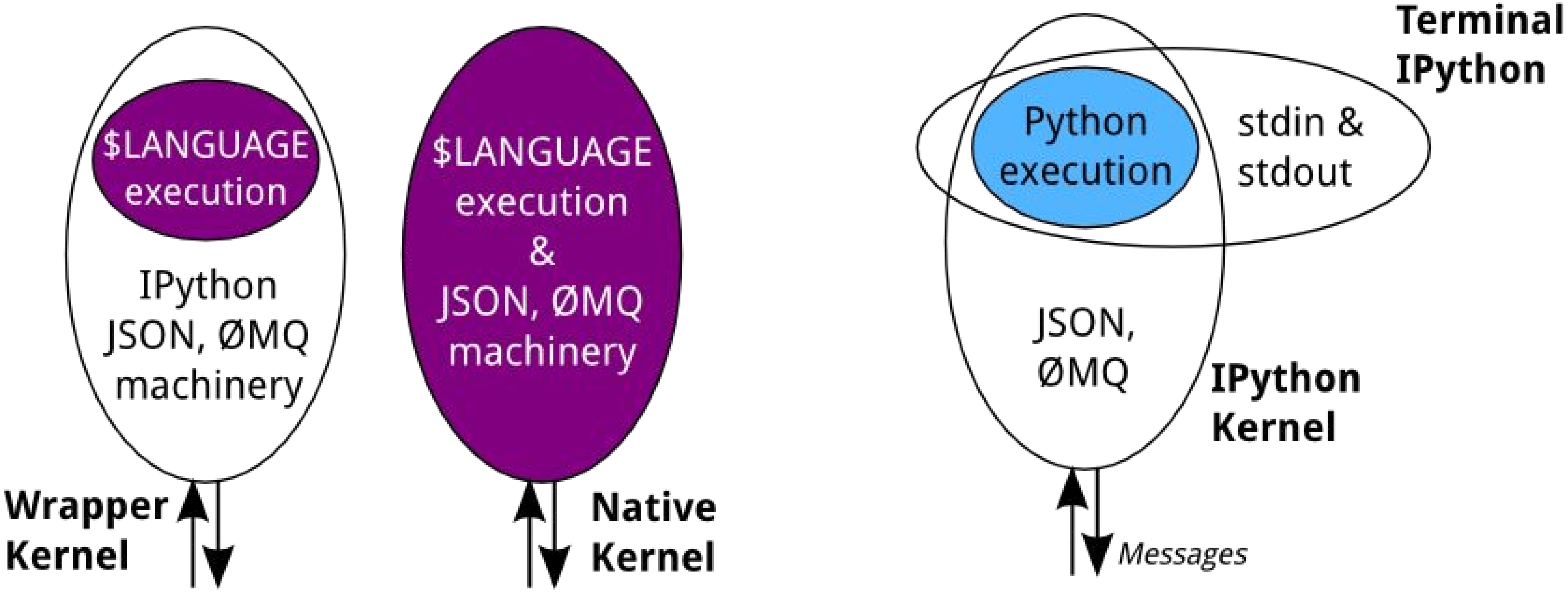


More about Kernels

- Jupyter messaging protocol
- Multiple clients can connect with different views
    - Qt console
    - Interactive shell/Terminal
    - Notebook

Support for programming languages

Kernel spec


$LANGUAGE execution
IPython JSON, ØMQ machinery
Wrapper Kernel
$LANGUAGE execution & JSON, ØMQ machinery
Native Kernel
Terminal IPython
Python execution
stdin & stdout
JSON, ØMQ
IPython Kernel
Messages