Jupyter Notebook User Document
Professional Communications
Team: Dolphins

Jeffery B. Russell, Daniel Moore, Louden Yandow

March 3, 2020

Contents

Page

List of Figures **2**

1 Introduction **4**

2 Installation **4**

2.1 *Prerequisites* 4

2.2 *Installation on Linux* 5

2.3 *Installation on Windows* 5

3 Usage **5**

3.1 *Navigation* 5

3.2 *Creating a Notebook* 6

3.3 *Running a Notebook* 6

3.4 *Exporting Notebooks* 9

3.5 *Customization* 9

4 Advanced Usage **10**

4.1 *Multiple Kernels* 10

4.2 *Remote Connection* 11

5 Glossary **13**

6 References **14**

Index **15**

List of Figures

Page

1 Default Jupyter Lab launcher 6
2 Code before being run 7
3 Code after being run 7
4 Example of raw text 8
5 Graphing Example 8
6 Markdown Example 9
7 Settings window 10
8 Example of multiple kernels installed 11
9 Network Overview 12

1 Introduction

This document goes over the basic installation and usage of Jupyter Lab for personal development. In the advanced usage section we go over how to use Jupyter on a remote server.

Jupyter is an open-source web-based notebook tool that you can use as your development environment. A coding notebook enables you to intermix markdown and code blocks that you can execute in a single document. This is heavily used in education and research fields because it makes writing reports easy and reproducible. With Jupyter you can create content that has live code, equations, visualizations and explanatory text.

Jupyter Lab extends the basic Jypyter notebook functionality and provides you a full web environment to work in. Using this interface you can open terminals, manage files, and even tile multiple editors.

Applications of Jupyter Jab:

- Quick experimentation
- Writing a report
- Sharing code snippets for education
- Generating graphics

2 Installation

Full installation instructions can be found on the Jupyter website. [1] This user document goes over how to install Jupyter Lab.

2.1 Prerequisites

In order to install Jupyter Lab, you must have Python installed. Your version of Python **must be 3.3 or greater**. As part of having Python, you will also need to have PIP. PIP is a package manager for Python. Although PIP usually comes with your Python installation, there are some exceptions on Linux where you have to install the two separately. Go to the Python website[2] to download the latest version of Python.

We highly recommend you take the time to install Conda because it will be helpful later on with some of the more advanced features of Jupyter as well as machine learning in Python. Anaconda is a package manager for the R and Python languages aimed towards data science. Anaconda is often abbreviated as conda.

You should also have either Firefox, Chrome, or Safari because these are the only browsers Jupyter is currently known to work well with.

1. https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html
2. https://www.python.org/downloads/

2.2 Installation on Linux

There are two ways of installing Jupyter on Linux.

Using pip, the command is:.

```
$pip install jupyterlab
```

Using conda, the command is:

```
$conda install -c conda-forge jupyterlab
```

2.3 Installation on Windows

Installation on Windows is the same as installation on Linux except that you must also use the flag to install at a user level if you don't want to run the command as an administrator:

```
>pip install jupyterlab --user
```

Using conda, the command is:

```
>conda install -c conda-forge jupyterlab
```

3 Usage

To run Jupyter Lab, open your computer's command terminal and enter the following command. This will open Jupyter Lab in your default web browser.

```
jupyter lab
```

Figure 1 is what you will see upon first running Jupyter Lab. Otherwise, it will open to the most recent notebook you were working on.

3.1 Navigation

Once Jupyter Lab is running, you will see on the left side of the window a column of icons. Each icon will open a different panel to the right when clicked.

- File Browser (folder icon): displays a file browser for the user to open, move, or delete their files.
- Running Terminals and Kernels (square stop button): shows the user all currently active terminal and kernel sessions.

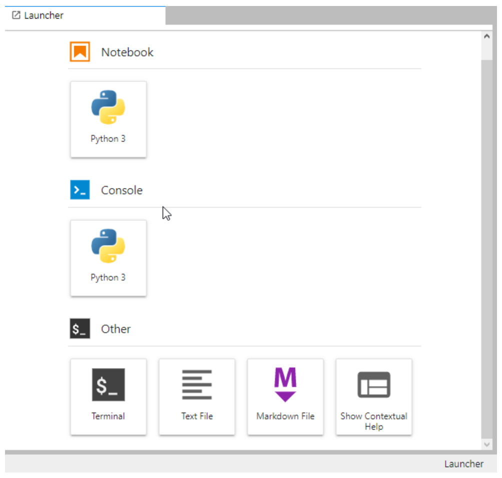


Figure 1: Default Jupyter Lab launcher

- Commands (palette icon): allows the user to enter various commands into Jupyter Lab.
- Notebook Tools (wrench icon): shows various options for the user's current notebook.
- Open Tabs (a tabbed window icon): lists all currently open tabs in Jupyter Lab.

Additionally, the top toolbar contains the following different drop-down menus: "File", "Edit", "View", "Run", "Kernel", "Tabs", "Settings", and "Help".

3.2 Creating a Notebook

To create a notebook from the launcher (figure 1), click on the Python 3 icon under the orange notebook symbol. Alternatively, if you don't have the launcher open, you can click on "File" in the toolbar, click "New", and finally click "Notebook".

This will open an empty, untitled notebook. If you right click on the tab above, or on the name of your notebook in the "Open Tabs" panel on the left, you can rename your notebook.

3.3 Running a Notebook

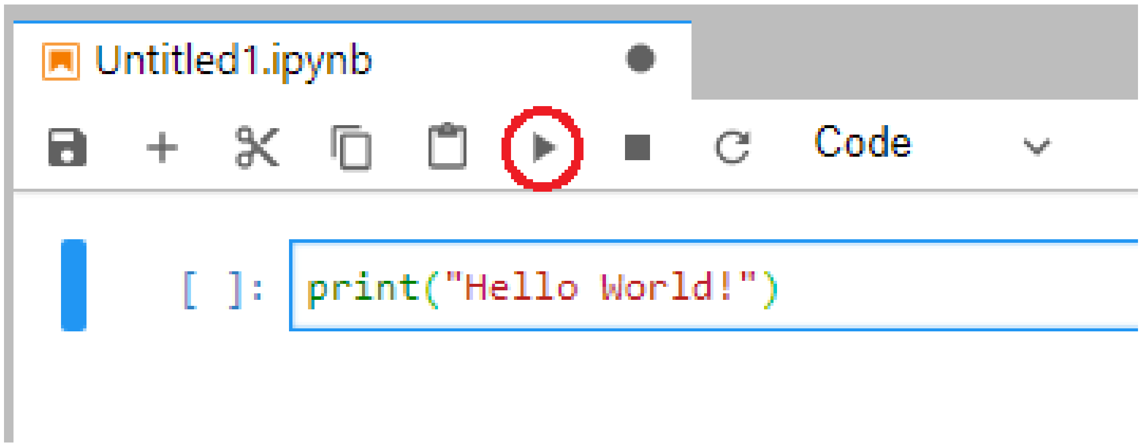
With a notebook open, you can start writing in the editor– the big empty area on the right half of the screen. Just above the editor, you will find the icon to save the open notebook.

There are also a number of icons that directly relate to the cells you are modifying. A cell may contain either Python code, markdown, or raw text. You can change what type of text a cell is by clicking on the drop-down menu just above the editor that will say either "Code", "Markdown", or "Raw".

The icons above the editor, from left to right, do the following things:

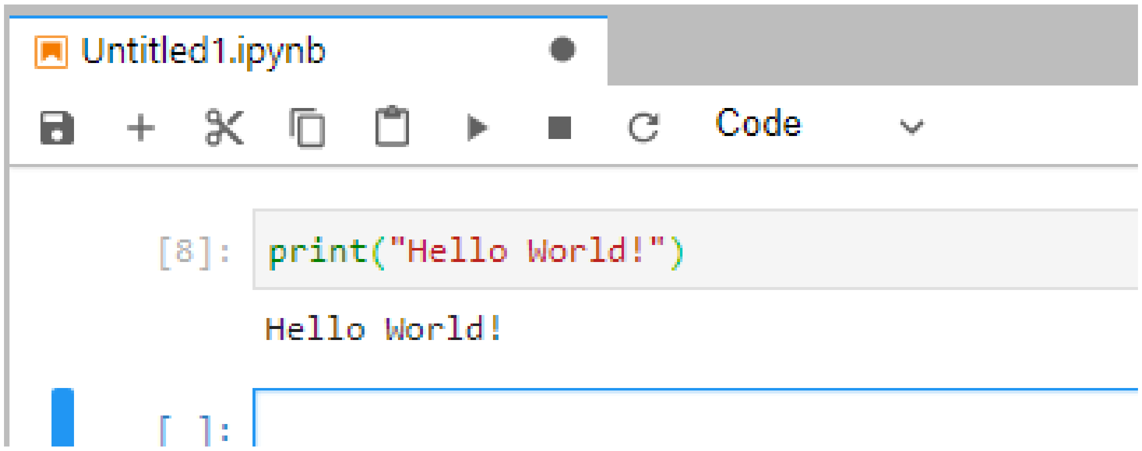
- Add a cell after the currently selected cell.
- Cut the currently selected cell.
- Copy the selected cell.
- Paste the cell from the clipboard.
- Run the selected cells and advance to the next cell.
- Interrupt the kernel.
- Restart the kernel.

The following example shows how to write code, run code, and insert raw text into the notebook. First, write some python code and click the "Run selected cells and advance" button (circled in red in the figure below). Our output is shown in figure 2. When a notebooks runs all cells, it executes top to bottom.



```
[ ]: print("Hello World!")
```

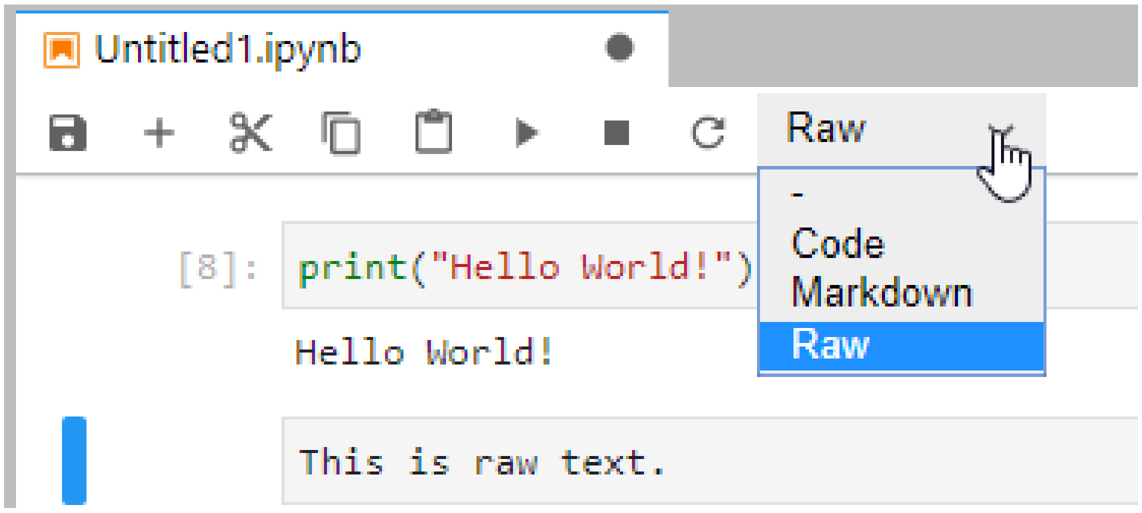
Figure 2: Code before being run



```
[8]: print("Hello World!")
Hello World!
[ ]:
```

Figure 3: Code after being run

You can also insert raw text into your document, as shown below. To do this, change the drop-down menu from "Code" (or "Markdown") to "Raw", and type what you want in the cell. This is demonstrated in figure 4.



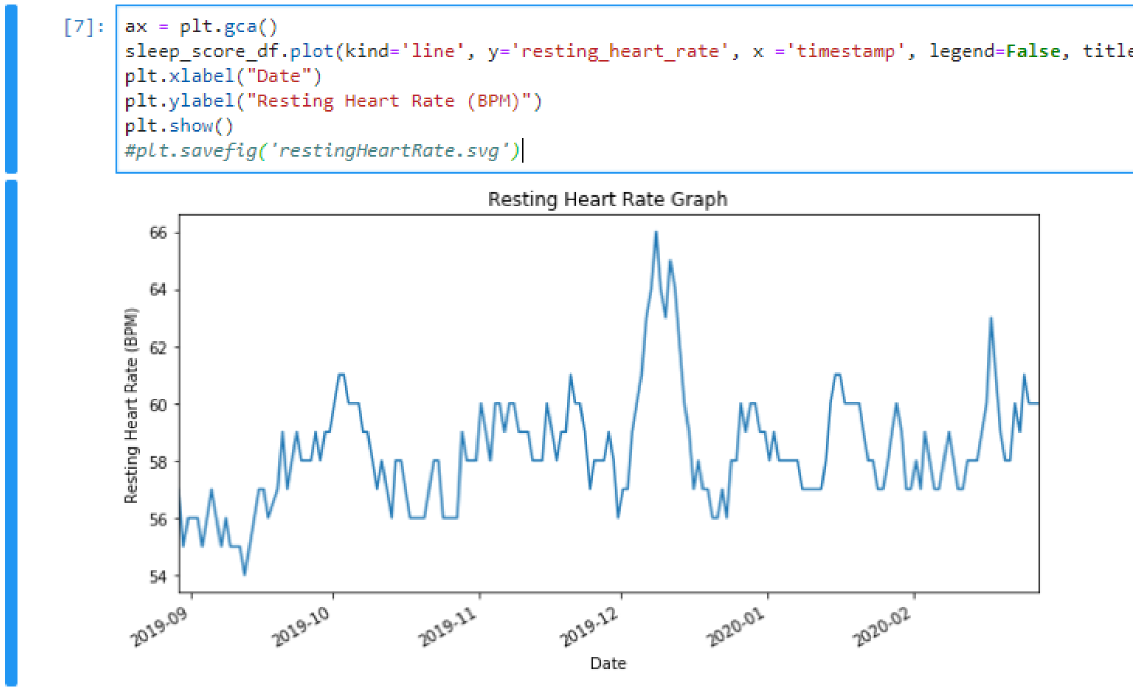
```
[8]: print("Hello World!")

Hello World!

This is raw text.
```

Figure 4: Example of raw text

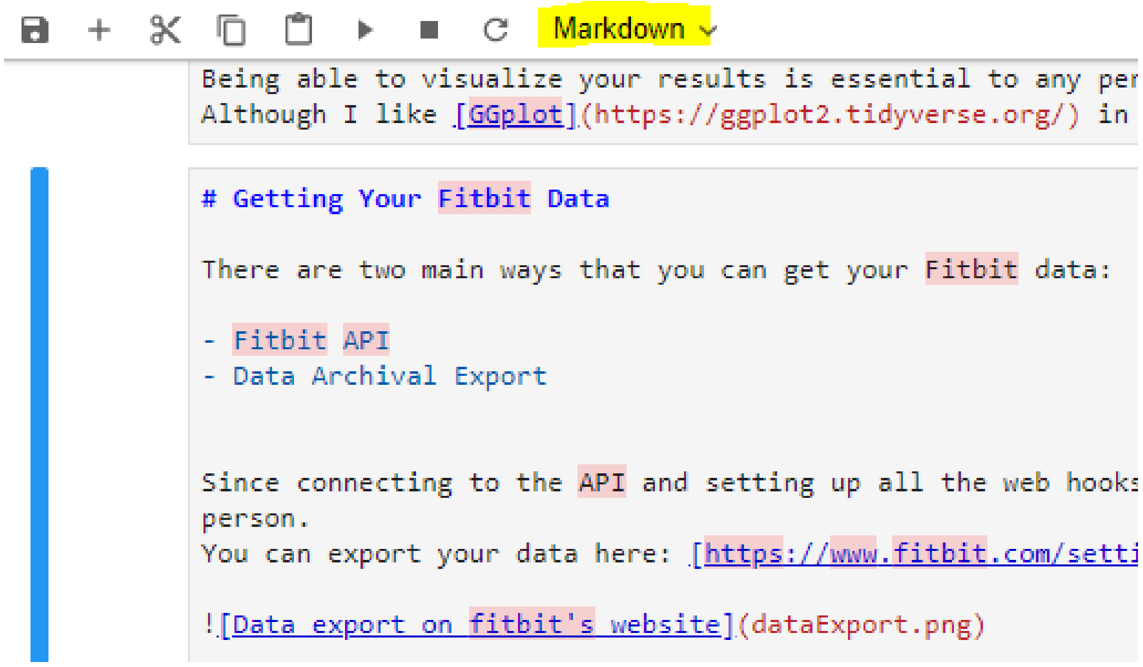
Figure 5 demonstrates the true power of Jupyter which is in its ability to produce graphs and other graphics inline. This is useful for people doing data visualization or data science.



```
[7]: ax = plt.gca()
     sleep_score_df.plot(kind='line', y='resting_heart_rate', x ='timestamp', legend=False, title
     plt.xlabel("Date")
     plt.ylabel("Resting Heart Rate (BPM)")
     plt.show()
     #plt.savefig('restingHeartRate.svg')
```

Figure 5: Graphing Example

Figure 6 demonstrates using the markdown in Jupyter. Markdown is a Lightweight markup-language – used to define layout and style for a document. This enables you to add links, headers, images, and lists to your document. Jupyter's markdown also supports embedded Latex math – used to display complex mathematical equations. If you want to learn how to use markdown, Github has excellent tutorials on markdown.[3]



```
Being able to visualize your results is essential to any per
Although I like [GGplot](https://ggplot2.tidyverse.org/) in
```

```
# Getting Your Fitbit Data

There are two main ways that you can get your Fitbit data:

- Fitbit API
- Data Archival Export


Since connecting to the API and setting up all the web hooks
person.
You can export your data here: [https://www.fitbit.com/setti

![Data export on fitbit's website](dataExport.png)
```

Figure 6: Markdown Example

3.4 Exporting Notebooks

In order to export the document, click the **File** tab, mouse down to **Export Notebook As...** and then click a desired format– Jupyter supports a lot of format types. The most popular export formats are PDF, Markdown, and HTML.

3.5 Customization

In general, customization of Jupyter is easy.

To change the Jupyter theme, go to the "Settings" tab and drop down to Jupyter Lab Theme. This will allow you to change from light to dark mode as well as the font sizes for the code, content and UI.

Scrolling down the rest of the settings you see many other things that can be customized.

3. https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

Advanced settings allows you to customize many aspects of Jupyter Lab such as keyboard shortcuts, terminal settings, and a myriad of others. Figure 7 shows the configuration menu for Jupyter Lab.

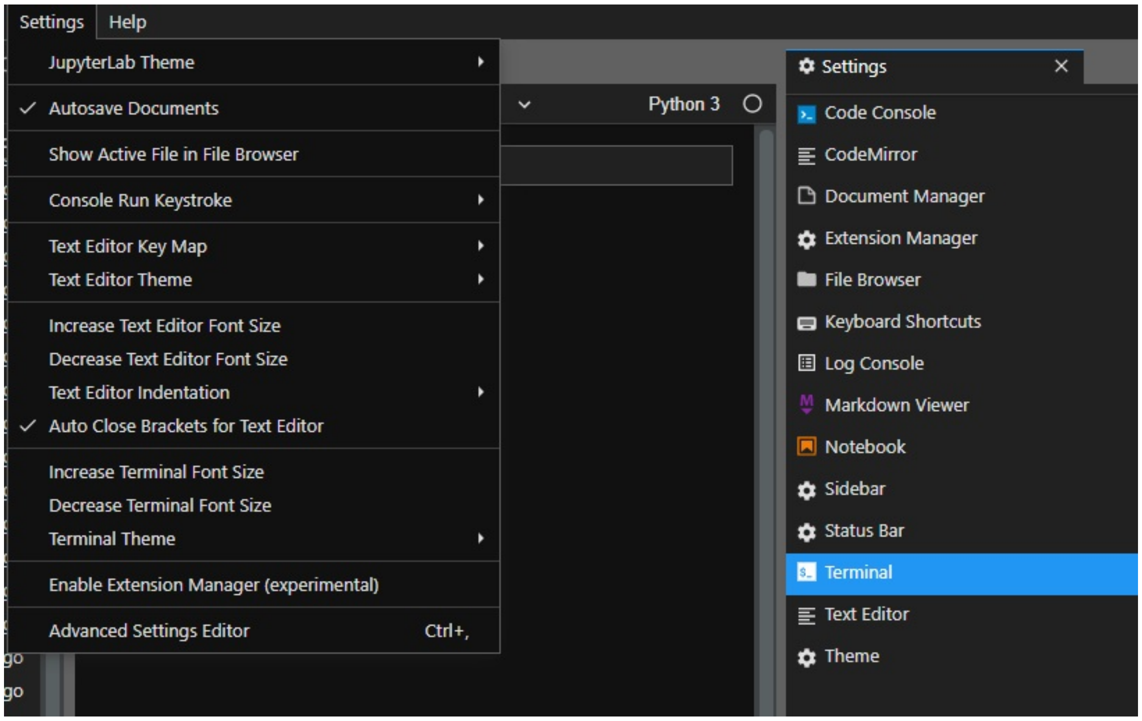


Figure 7: Settings window

4 Advanced Usage

This section goes over how to use multiple programming languages in Jupyter and how to connect to your Jupyter Lab instance remotely.

4.1 Multiple Kernels

Adding a Kernel in Jupyter enables you to program with another programming language, such as R or Scala. Up to this point, we have been working with the Python kernel which enables you to make python notebooks. The ability to use multiple kernels is useful for education and having cohesion within one IDE.

You can view a complete list of kernels can be found on Jupyter's Github wiki.[4]

Figure 8 shows an example of what your launcher would look like with multiple kernels installed.

4. `https://github.com/jupyter/jupyter/wiki/Jupyter-kernels`

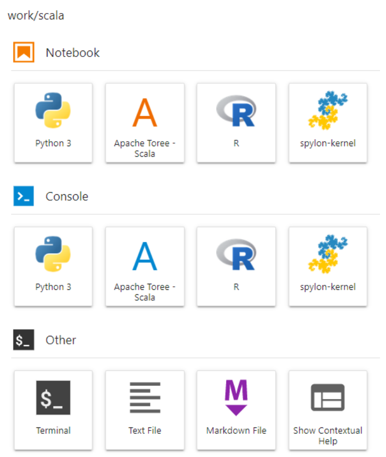


Figure 8: Example of multiple kernels installed

The most common way to install a kernel is by using the Anaconda prompt – pip support is limited here. Once you start adding multiple kernels, it is best if you start running Jupyter Lab through Docker because it would make porting it to another computer easier. A popular Docker Jupyter lab instance with Scala, Python, and R can be found on Jupyter's Docker hub profile[5].

4.2 Remote Connection

A prerequisite for this section is having a Linux machine with SSH installed on it.

If you have a firewall, Jupyter Lab will only be reachable on your local machine at "localhost:8888". However, it is possible to connect to Jupyter Lab from remote computers. Remote access is helpful since it enables multiple people to connect to the same Jupyter instance. This would also save you resources on your local machine so you can program on a lightweight chrome-book that would not be able to run a full IDE like Pycharm.

The first step to enable remote host would be to set a password that you can connect to the notebook using. You can set a password that you use to log into the website by executing the following command in a terminal:

```
$jupyter notebook password
```

The second step would be to launch the Jupyter Lab instance in a headless environment – it never launches a web browser. A headless environments means that you never launch a GUI and it operates solely as a console application.

```
$jupyter lab --no-browser --port=6000
```

5. https://hub.docker.com/u/jupyter/

The final step is to connect to the Jupyter Lab instance from your remote computer. The easiest way to do this is via a local port forward in SSH. The command at listing 1 forwards all of the traffic from your local machine to a specific port on a remote computer over a SSH connection. The main benefit of SSH is that all the traffic over the connection is encrypted.

Figure 9 shows an overview of what the network architecture looks like.

Listing 1: SSH command for local port forwarding

```
$ssh -L 6000:localhost:6000 user@remote-host
```

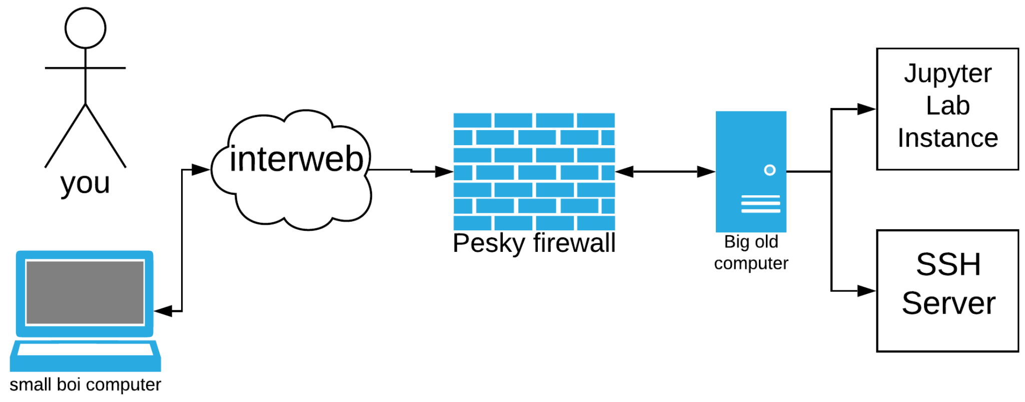


Figure 9: Network Overview

After you execute the command above on your remote computer, you will be able to access your Jupyter Lab instance at "localhost:6000" even though it is only running on the remote computer at port 6000.

5 Glossary

Anaconda: Anaconda is a package manager for the R and Python languages aimed towards data science" [6].

IDE: Interactive Development Environment– some popular IDEs are Pycharm, Netbeans, and Visual Studio.

Jupyter: Nonprofit organization created to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages" [7].

JVM: Java Virtual Machine

LaTeX: LaTeX is a document preparation system

Markdown(MD): Lightweight markup-language – used to define layout and style for a document [8].

pip: Tool for installing and managing python packages [9].

Pycharm: A popular versatile Python IDE developed by Jetbrains.

Python: High-level interpreted, general purpose programming language [10].

R: Programming language for statistical computing and graphics [11].

Scala: General purpose functional programming language that runs on the JVM [12].

SSH: Secure Socket Shell – used in connecting to a remote computer over a encrypted channel.

6. `https://www.anaconda.com`
7. `https://jupyter.org/`
8. `https://en.wikipedia.org/wiki/Markdown`
9. `https://pypi.org/project/pip/`
10. `https://www.python.org/`
11. `https://www.r-project.org/`
12. `https://scala-lang.org/`

6 References

1. Developer Tools for Professionals and Teams. (n.d.). Retrieved March 3, 2020, from `https://www.jetbrains.com/`.
2. Markdown (2020, March 1). Retrieved March 3, 2020, from `https://en.wikipedia.org/wiki/Markdown`
3. PIP (n.d.). Retrieved March 3, 2020, from `https://pypi.org/project/pip/`.
4. Project Jupyter. (n.d.). Retrieved March 3, 2020 `https://jupyter.org/`
5. Python.(n.d.). Retrieved March 3, 2020 from `https://www.python.org/`
6. The R Project for Statistical Computing. (n.d.). Retrieved March 3, 2020, from `https://www.r-project.org/`.
7. The Scala Programming Language. (n.d.). Retrieved March 3, 2020, from `https://scala-lang.org/`.

Index

anaconda, 4, 11, 13

docker, 11

IDE, 10, 11, 13

jupyter, 4, 9, 10, 13

latex, 9, 13

markdown, 6, 9, 13

pip, 5, 11, 13
pycharm, 11
python, 4, 6, 10, 13

r, 10, 13

scala, 10, 13
ssh, 12, 13