# USING JUPYTERHUB IN THE CLASSROOM: SETUP AND LESSONS LEARNED

Jeff Brown

Department of Mathematics and Statistics, University of North Carolina Wilmington

***ABSTRACT***

*Jupyter notebooks, formerly known as iPython notebooks, are widely used for data analysis and other areas of scientific computing. Notebooks can contain formatted text, images, LaTeX formulas, as well as code that can be executed, edited and executed again. A jupyter hub is a multi-user server for jupyter notebooks, and setting up a jupyter hub is a complex endeavour that involves many steps. The instructions found online for setup often have to be customized for different operating systems, and there is not one source that covers all aspects of setup. This paper describes the details of setting up a jupyter hub environment on a server running CentOS 7, and includes a discussion of lessons learned from using this system in data science classes.*

***KEYWORDS***

*Data analysis, Python, Jupyter, Jupyterhub*

## 1. INTRODUCTION

### 1.1. IPYTHON AND JUPYTER NOTEBOOKS

The iPython notebook was created in 2011 by a team of researchers from the University of California, Berkeley, and the California Polytechnic State University [1]. Notebooks can contain live code from a wide range of languages, as well as visualizations, explanatory text written in HTML or Markdown, and formulas written in LaTeX.

A jupyter notebook runs in a Web browser and consists of a series of cells. *Markdown* cells display formatted text, images, formulas, and video clips. *Code* cells contain programs that can be executed, showing the results in the notebook. Figure 1 below shows three cells from a notebook. At the top is a Markdown cell that states the trapezoidal rule for approximating a definite integral. Below the Markdown cell are two Code cells. The first code cell defines a function named *trap* that implements the trapezoidal rule. After the code defining the trap function has been executed, that function may be used in other code cells. The second code cell uses the trap function to approximate the definite integral from 0 to pi of the sine function.

DOI: 10.5121/ijsea.2018.9201 

## Example: Trapezoidal Rule

The trapezoidal rule provides a numerical approximation of a definite integral.

To approximate $\int_a^b f(x)dx$ using $n$ subintervals let $x_0, x_1, \ldots, x_n$ be the endpoints of the $n$ subintervals and let $\Delta x = \frac{b-a}{n}$, the length of each subinterval.

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}(f(x_0) + 2 * f(x_1) + \ldots + 2 * f(x_{n-1}) + f(x_n))$$

In [1]:
```
import numpy as np

def trap(f,a,b,n):
    '''
    Arguments: f a function, a and b the limits of integration, n the number of subintervals
    Return: approximation of the definite integral by the trapezoidal rule
    '''
    x = np.linspace(a,b,n+1)
    y = f(x)
    y[1:-1] *= 2
    return y.sum()*(b - a)/(2*n)
```

In [2]:
```
trap(np.sin,0,np.pi,20)  # Exact value is 2
```

Out[2]:
```
1.9958859727087148
```

Figure 1. Three cells from a jupyter notebook using a python kernel

In 2014 Project Jupyter was created as a non-profit, open source, spin-off of the iPython project. Project Jupyter took over development of the notebook and other aspects of the iPython project that are not specifically related to python. The iPython project is continuing and it supplies the python kernel used by jupyter. Jupyter notebooks support kernels for many languages other than python, including Julia, R, Matlab, and SAS.

Researchers in many fields, including data analysis [2], ocean science [3] and biomedicine [4], use notebooks to share reproducible results with colleagues. A typical journal article may include discussion of theory and tables or visualizations showing the results of numerical experiments, but reproducing those experiments may require quite a bit of work. A notebook can contain the discussion, the tables and visualizations, as well as the code necessary to produce the results.

Students in our data analytics class spend more time working on problems than in lecture. The problem based learning (PBL) paradigm has increased in popularity in recent years, particularly in the sciences and engineering [5][6]. We find jupyter notebooks very useful for this style of learning, because the student can be presented with examples, and then open new cells to extend or modify the given content. The notebook can be used as an outline for a project, guiding the student through the discovery process. The basic notebook environment is also being extended for use by other educators [7][8].

### 1.2 JUPYTERHUB

Jupyter hub is a Web application that runs on a server. In the author's environment the server is a linux machine running CentOS 7. In this paper server always refers to the machine on which the hub is running. Users interact with the hub through a Web browser on a client machine, and the browser is the only software needed on the client machine.


Figure 2. Jupyterhub home screen

Figure 2 shows the hub home screen. The files listed on the left are notebooks. The file extension .ipynb stands for iPython notebook. On the right you see the objects that you can create: notebooks with a choice of four kernels, a text file, a new folder or terminal for executing shell commands.

The property of allowing access to computing resources without requiring special software on the client machine makes the jupyter hub ideal for use in a classroom [7]. In section 2 we discuss our experience using a jupyter hub in data science classes. We explain how the jupyterhub was used and lessons learned from the experience. In section 3 we provide detailed instructions on how to setup a jupyter hub on a CentOS 7 server, and in section 4 we discuss configuration. Section 5 contains conclusions.

## 2. JUPYTERHUB IN THE CLASSROOM

Our hub has been used in classes on computational physics, linear algebra and data analytics with python. In this section we discuss our experience with the data analytics classes.

### 2.1. FILE MANAGEMENT

Class content was generated on the hub, and we used a script to copy the content into the home directories of all the students. The script also made the students the owners of the copied files.

For graded assignments students were given a due date, and told to save and checkpoint their work by the due date. As administrators of the jupyter hub, the instructors can open and edit the students' notebooks. The instructor writes feedback in the notebook for the students to review. This is much more efficient than having the students turn in their work through email or some other means.

### 2.2. MEMORY MANAGEMENT

In [7] the authors state that when a class involves minimal computing, then they expect that "a standard desktop computer could support many dozens of students." Our data analytics classes require more than minimal computing, and server memory is an issue. We have supported forty

students on a dual processor (Xeon E5) machine with 128 GB of ram, and we had resources to spare. If memory is limited then students and faculty need to be taught how to conserve memory.

Each notebook runs in a tab on the browser. Simply closing the tab does not halt the notebook process and does not free memory. The *Running* view inside the jupyter hub home screen shows the terminals and notebooks that are currently running.

Figure 3 below shows five running notebooks and provides buttons to shut them down.


Figure 3. View of processes running for one user

Under the File menu in the notebook you may select 'Close and Halt,' and this should be used instead of simply closing the notebook's tab. Another way to make sure that system resources are released is the schedule the shutdown and restarting of the hub.

The hub configuration file allows you to limit the number of cores or percentage of a core that a notebook process can consume. You may also put a limit on the amount of memory a notebook may use.

## 3. JUPYTERHUB INSTALLATION

### 3.1. OVERVIEW

We will describe the installation and setup on a server running CentOS 7.4.1708.
JupyterHub requires python 3.4 or later. We find it convenient to use the Anaconda platform, which provides python and associated utilities as well as many packages used for scientific computing, such as numpy, matplotlib, and pandas.

The client browser communicates with the hub through an http proxy, and the proxy communicates with multiple single-user notebook servers.

Installation will require the use of three different package managers. The CentOS package manager *yum* will be used to install node.js, which is a JavaScript run-time environment. Along with node.js you get *npm*, which is another package manager that will be used to install the http proxy. Finally we use *pip*, a python package manager, to install the hub and notebook.

### 3.2. ANACONDA

On a linux machine Anaconda is installed through a shell script. We downloaded the file Anaconda3-5.0.1-Linux-x86_64.sh from https://anaconda.com.

By default, the installer will create a directory named anaconda3 in your current directory. We decided to have the installer create the directory /opt/anaconda3. The default settings only allow root to write to /opt, so we used **sudo** to do the install. As a result, root owns the directory and installing more packages there will also require sudo. Here is the installation command, which is run in the directory containing the anaconda installation script.

```
sudo bash Anaconda3-5.0.1-Linux-x86_64.sh
```

The installer has you accept the license, accept the default installation location or enter your own, and offers to update the user's .bashrc file so their command path will contain the new software. The following lines were added to root's .bashrc file by the installation script.

```
# added by Anaconda3 installer
export PATH="/opt/anaconda3/bin:$PATH"
```

Other users who need access to the new software should add the export line to their .bashrc files. Note that the new component added to the path needs to come first because the original path may include older versions of python.

### 3.3. NODEJS AND CONFIGURABLE-HTTP-PROXY

EPEL stands for 'Extra Packages for Enterprise Linux,' and it is a repository of add-on packages for Red Hat Enterprise Linux and spin-offs such as CentOS. EPEL is used to install NodeJS

```
sudo yum install epel-release
sudo yum install nodejs
```

NodeJS includes the package manager npm and we use it to install the http proxy.

```
sudo npm install --g configurable-http-proxy
```

### 3.4. JUPYTERHUB AND NOTEBOOK

Anaconda includes the python package manager pip, and we used it to install JupyterHub and Notebook with these commands.

```
sudo /opt/anaconda3/bin/pip install jupyterhub
sudo /opt/anaconda3/bin/pip install --upgrade notebook
```

Note that we used the full path to the pip program, because sudo does not preserve command paths.

# 4. CONFIGURATION

## 4.1. jupyterhub_config.py

We put the configuration files in the directory /etc/jupyterhub. The following commands create that directory and generate a default configuration file named jupyterhub_config.py.

```
sudo mkdir /etc/jupyterhub
cd /etc/jupyterhub
sudo /opt/anaconda3/bin/jupyterhub --generate-config
```

The JupyterHub requires https, so you have to acquire a certificate for your server. Our server is behind our university firewall and is not associated with a domain name, so getting a certificate for it is not something a faculty member cannot easily do. Originally we used a self-signed certificate, but then our users were warned about security and had to allow a security exception in order to access the hub. Our IT department acquired a wild-card certificate and using that has eliminated the security warnings.

The following commands create a directory named keys in the /etc/jupyterhub and make it readable only by root. Put your certificate and key in the keys directory. The key file should also be readable only by root.

```
sudo mkdir keys
sudo chmod 700 keys
```

There are many configuration options in jupyterhub_config.py. Here are the changes we made to make the hub work. Other changes are optional. For each of these changes remove the # at the beginning of the line and enter values. These values are python strings, so they must be quoted. The last one is a python list that contains a string.

- c.JupyterHub.ip is the numeric IP address of the server.
- c.JupyterHub.ssl_cert is the path to your ssl certificate.
- c.JupyterHub.ssl_key is the path to your ssl key.
- c.Spawner.cmd default value is [‘jupyterhub-singleuser’]. We had to replace that with the full path [‘/opt/anaconda3/bin/jupyterhub-singleuser’].

The hub uses a 32-byte key, encoded as hex, to encrypt cookies. This key can be stored in the configuration file or in an environment variable or in a file whose default location is /etc/jupyterhub/jupyterhub_cookie_secret. We used the file option.

Because the second line below executes a command and writes the output to a file, it is not convenient to use sudo. So we used ‘sudo su’ to become root. While still in /etc/jupyterhub, execute these commands.

```
sudo su
openssl rand -hex 32 > jupyterhub_cookie_secret
chmod 700 jupyterhub_cookie_secret
exit
```

The ‘exit’ command quits the root shell and takes you back to your user shell.

### 4.2. FIREWALL

The default port for accessing the hub is 8000, so you need to allow access through the firewall on that port. There are graphical tools for this, but it is easy to do with commands.
sudo firewall-cmd --zone=public --add-port=8000/tcp --permanent
sudo firewall-cmd –reload

### 4.3. Security-Enhanced Linux (SELinux)

CentOS runs SELinux by default, and your hub will not function without making some changes to the SELinux environment. The simplest solution is to disable SELinux, but we do not recommend that option because SELinux is an important enhancement. Here are the steps we took to allow the hub to run under SELinux.

- Put SELinux in permissive mode. This means it does not prevent any activities, but it still logs activities that would have been prevented.

sudo setenforce 0

- Start the hub from the command line as root, telling it where to find the configuration file.

sudo /opt/anaconda3/bin/jupyterhub -f /etc/jupyterhub/jupyterhub_config.py

- Login to the hub from a remote machine by using this URL in a browser.

https://<address of server>:8000

- The SELinux audit log file is /var/log/audit/audit.log. Use grep to find the lines in the log file that contain the word denied.

sudo grep denied /var/log/audit/audit.log

- For us the output of the grep command produced several lines, and they all contained ‘comm=jupyterhub’. The comm field gives the name of the command that resulted in the denied activity. Next we use these lines to tell SELinux to allow these activities. That is why you first check that you are only allowing activities associated with the jupyterhub command.

- Pipe the audit log lines to the program audit2allow and give the module a name. We named it jh-module.

sudo grep denied audit.log | audit2allow -M jh-module

- This command produces two files: jh-module.pp and jh-module.te, and it prompts you to make the new policy active with the following command.

sudo semodule -I jh-module.pp

- Now set SELinux back to enforce mode.

sudo setenforce 1

Now you should be able to start and use the jupyter hub.

## 5. CONCLUSIONS

- Jupyter notebooks are widely viewed as valuable pedagogical tools.
- A jupyter hub is particularly useful for the classroom. It allows you to provide a fully configured computing environment that only requires a Web browser on the student computer.
- Depending on the computational demands of the course, a server with limited resources may be sufficient. Steps should be taken to conserve server memory.
- Setup and configuration of a jupyter hub is a complex process, but it will probably become easier as the software becomes more widely used.

## REFERENCES

[1] Shen, H. (2014). "INTERACTIVE NOTEBOOKS: SHARING THE CODE." Nature, 515(7525), 151-2

[2] McKinney, W. (2012). "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython." O'Reilly Media, Inc.

[3] Signell, R. P., Fernandes, F., & Wilcox, K. (2016). "Dynamic reusable workflows for ocean science." Journal of Marine Science and Engineering, 4(4)

[4] Grüning BA, Rasche E, Rebolledo-Jaramillo B, Eberhard C, Houwaart T, Chilton J, et al. (2017) "Jupyter and Galaxy: Easing entry barriers into complex data analyses for biomedical researchers." PLoS Comput Biol 13(5)

[5] Regalado-Menendez, Cid-Rodriguez, Baez-Gonzalez (2010) "Problem Based Learning (PBL): Analysis of Continuous Stirred Tank Chemical Reactors with a Process Control Approach" International Journal of Software Engineering & Applications (IJSEA) 1(4)

[6] Regalado-Menendez, Cid-Rodriguez, Baez-Gonzalez (2010) "Problem Based Learning (PBL): Analysis of Continuous Stirred Tank Chemical Reactors with a Process Control Approach" International Journal of Software Engineering & Applications (IJSEA) 1(4)

[7] O'Hara, K. J., Blank, D., Marshall, J. (2015) "Computational Notebooks for AI Education" Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference, Hollywood, Florida

[8] Perez, F., Grainger, B. E. (2007) "IPYthon: A System for Interactive Scientific Computing" Computing in Science and Engineering 9(3)