OOI JupyterHub Set Up

Who to contact with questions

If you have questions or encounter issues while attempting to carry out the steps in this document, please contact the OOI Helpdesk at helpdesk@oceanobservatories.org or post a question in the OOI Discourse forum (https://discourse.oceanobservatories.org).

Logging in

Access to the NSF Ocean Observatories Initiative (OOI) JupyterHub (https://jupyter.oceanobservatories.org), which creates encapsulated JupyterLab environments unique to each user, is available to researchers and students looking to interact with OOI data. The JupyterHub is set up to provide users with the option of using either Python, R, MATLAB (users need to provide their own individual or institutional license for MATLAB), or Julia to access, explore and analyze OOI data using a high-performance computing cluster co-located with the data (both the raw and processed netCDF files, which are also accessible via the OOI Gold Copy THREDDS catalog).

Additional information on JupyterHub and JupyterLab can be found online:

- JupyterHub (https://jupyterhub.readthedocs.io/en/latest/)
- JupyterLab (https://jupyterlab.readthedocs.io/en/latest/)

If you haven't already, request access to the JupyterHub via from the link on the login page. This will open an email that you can use to make the request. Please provide your name, the institution or program you are working with, an email address to assign to the account (see further information below), and a brief description of how you plan on using the data.

The email account you provide when applying for access will be used to create an account for you on the JupyterHub. OOI uses CILogon to manage access to the JupyterHub (we do not maintain a database of access credentials, rather a listing of allowed users), so you need to provide an email account that is linked to a known identity provider (e.g., your home organization).

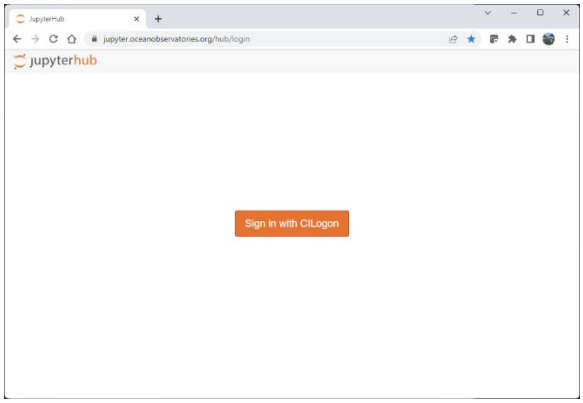

Figure 1. Initial login page for the OOI JupyterHub using CILogon.

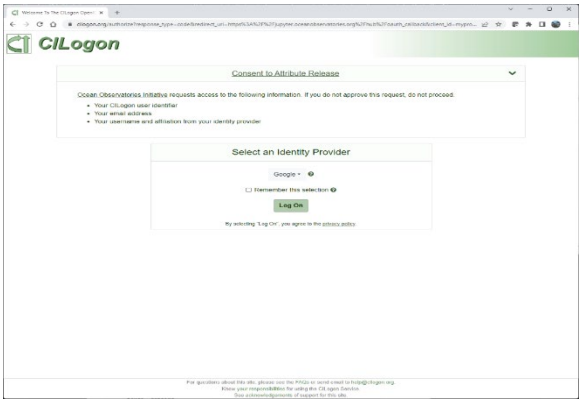

Figure 2. Selection of identity provider in CILogon; usually the user's home institution allowing them to select and use their institution's credential system.

You can login to the hub using CILogon (Figure 1) with your email account and the appropriate identity provider (use the drop-down menu in Figure 2, select your institution, and Log In). You will need to login using your home organization's credential system. Again, OOI will not store this information and has no access to it or control over it, instead the JupyterHub will use cookies set by CILogon after you are authenticated to grant access if the account is in the allowed listing.

The next step is to choose the server size and type (Figure 3). For most applications, a "Large" server will suffice. However, if you are planning on processing large volumes of data, then you are encouraged to select the "X-Large" or "XX-Large" servers. Newer servers, using the Nvidia GPUs have been added for individuals interested in processing large volumes of data for modelling or machine learning applications using PyTorch or TensorFlow.

MATLAB users will want to select the "Matlab – Medium" server (note, choosing a MATLAB server does not preclude use of Python or R, those options will also be available, e.g., see Figure 4).

Click on the Start button.

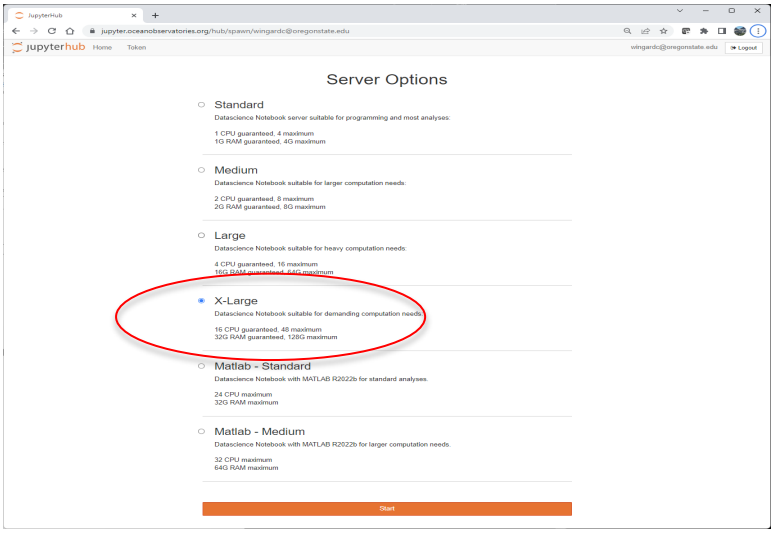

Figure 3. Selection of the server type and size. Most users will want to select a "Large", "X-Large" or "XX-Large" server (MATLAB users will want to select "MATLAB – Medium". Individuals looking to explore the use of GPUs for modeling or machine learning applications will want to select one of the Nvidia GPU servers.

Configuring JupyterLab

At this point, you should see a screen that looks like the one below in Figure 4 (assuming you selected a MATLAB server). The next step is to configure your server so that any changes you make (e.g., setting up python virtual environments) will persist between login sessions. ***Note, you will only need to run the commands in this section once. Again, only run these commands once!*** After JupyterLab is configured, re-running these commands will cause issues.

Note the `ooi` directory in the file browser on the left-hand side of the screen in the screenshot below (Figure 4). This should be the only directory you see the first time you login. This is a read-only directory that contains the raw data directories (as recorded by the different OOI platforms), and a directory called `kdata` that contains the processed netCDF files created on a regular schedule and served publicly via the OOI Gold Copy THREDDS catalog (the source files for the data shown in Data Explorer). You'll be able to access these files directly, rather than having to download them over the internet.

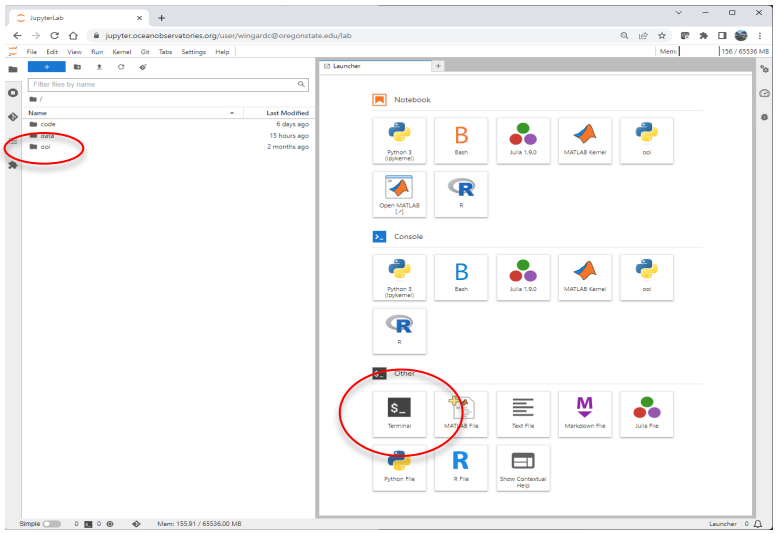

Figure 4. JupyterLab session with Launcher tab on the right showing the different notebook and console options, and a file browser to the left. The ooi directory is the third one down in the file browser on the left. This would be the only you would see the first time you set up your JupyterHub.

The next set of steps will configure your server so all the needed code and packages are installed, and so that those changes will persist between login sessions. For Python users, there are two files you need to create/update. From the Launcher tab, select the "Terminal" launcher under the "Other" category (see Figure 4, red circle). This is a Linux terminal running the bash shell which will be used extensively in the following examples.

At the terminal prompt (see Figure 5), copy and paste the following block of code, and then hit enter (leave the terminal open for now):

```
# Initialize the terminal to use the conda executables
cd ~
conda init bash
source .bashrc

# make sure the terminal is set to reload the conda executables the next time
# you login (all user settings would otherwise be reset)
touch .bash_profile
cat <<EOT >> .bash_profile
if [ -f $HOME/.bashrc ]; then
    source $HOME/.bashrc
fi
EOT

# configure conda so it remembers and saves any custom user environments
touch .condarc
cat <<EOT >> .condarc
envs_dirs:
  - ~/.conda/envs
  - /opt/conda/envs
EOT
```

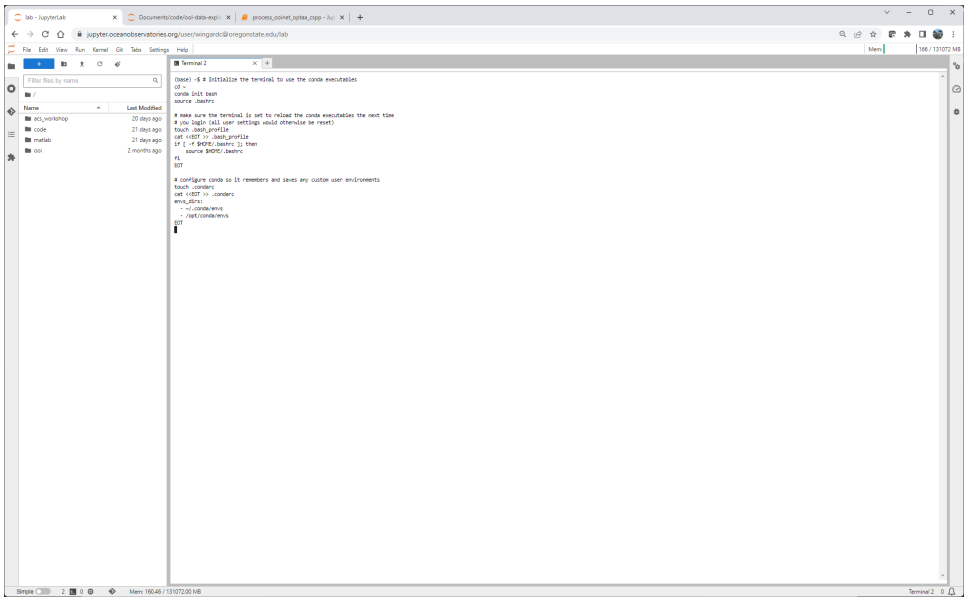

Figure 5. Terminal with code pasted in from the code block above.

For MATLAB users, the first time you launch MATLAB you will need to verify your MATLAB license using either Online Licensing or a Network License Manager (a local license will not work). From the Launcher tab, under the Notebook section, click on the “Open MATLAB” link and follow the prompts to add your license (Figure 6). This only needs to be done once. In addition, you will need to create a **`matlab`** directory in the home directory. This folder is automatically added to the MATLAB path and will be used to store some customizations (more on this below).

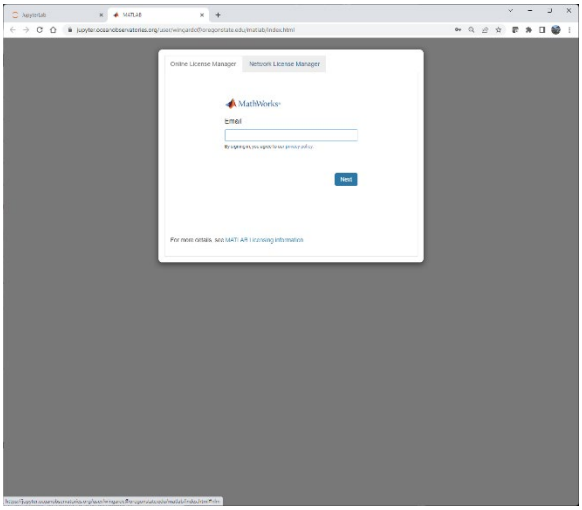

Figure 6. MATLAB login screen to verify user license.

For R users, a directory and user file will need to be created to set up a user-customizable R environment. Those will be created below as part of setting up access credentials and installing some example R libraries.

Setting up Access Credentials

OOI requires access credentials to download data and/or metadata (e.g., calibration coefficients) via the M2M interface, which a lot of the code and examples described below will require. Directions on how to obtain the access credentials, in addition to details about the M2M system, are available on the OOI website. As above, you will only need to run the commands in this section once. ***Once you have your credentials in place, you do not need to re-run these commands.***

- If you haven't already done so, either create a user account on the OOI Data Portal (original OOI website and API server for the OOI M2M system), or use the CILogon button with an academic or Google account (login button is towards the upper right corner of the web page) to login to the portal (see Figure 7 below).
- After you login, the "Log In" text will change to your username.
- Click on your username and then on the "User Profile" element of the drop down.
- Copy and save the following values from the user profile: **`API Username`** and **`API Token`**.

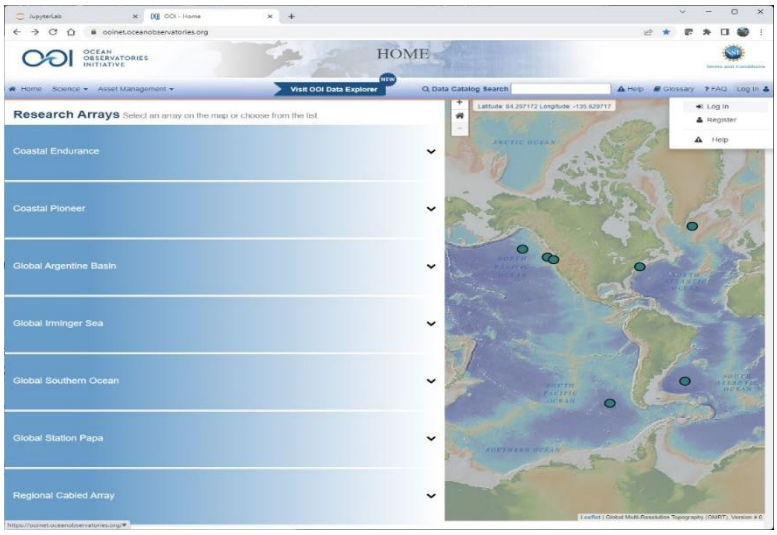

Figure 7. Log in link on the OOI Data Portal (OOINet) where you can retrieve your API credentials.

The code described below uses the netrc utility to obtain your access credentials. Users need to create a **`.netrc`** file in their home directory to store these access credentials. Using the text block below, create a **`.netrc`** file (replacing the **`API_Username`** and **`API_Token`** text below with the corresponding values from your login credentials for the OOI Data Portal):

```
# Set your OOINet credentials (make sure to replace API_Username and
# API_Token below with your credentials!!)
touch .netrc
chmod 600 .netrc
cat <<EOT >> .netrc
machine ooinet.oceanobservatories.org
    login API_Username
    password API_Token
EOT
```

To make this easy you could copy/paste the above into a text document, edit the API Keys, and then copy/paste the edited block into the terminal opened earlier.

For MATLAB users, the credentials are stored in a `weboptions` object that the code can load and use as needed. Open MATLAB, and then run the following commands, again making sure to replace `API_Username` and `API_Token` with the values from your API Keys:

```
% create the MATLAB home directory
mkdir('/home/jovyan', 'matlab')

% create the OOINet Credentials object and save it in the home directory
% make sure to replace API_Username and API_Token below with your credentials
username = 'API_Username';
password = 'API_Token';
options = weboptions('Timeout', 180, 'HeaderFields', {'Authorization', ...
                    ['Basic ' matlab.net.base64encode([username ':' password])]});
save('/home/jovyan/matlab/ooinet.credentials.mat', 'options');
```

For R users, the API Keys will be stored in the `.Rprofile` file. Copy/paste the following text block into the terminal:

```
# Make sure we are in the home directory
cd ~

# Create a user folder to store R packages beyond those provided
# by the server.
mkdir -p .Ruser/lib

# Set up an .Rprofile file for user customizations
touch .Rprofile
chmod 600 .Rprofile

# Set up your OOINet credentials (make sure to replace API_Username and
# API_Token below with your credentials!!)
cat <<EOT >> .Rprofile
# Add the API Keys for the OOI system
options(OOI_Username = "API_Username")
options(OOI_Password = "API_Token")

EOT
```

Adding Code and Examples

OOI Data Team members have been developing code for accessing and processing data from multiple instruments for various data reviews and assessments and the production of QC test limits. This code is freely available on GitHub under the OOI Data Explorations repository and includes several utilities we've found useful over the years. The code snippet below will download that code from GitHub and make it available to the different applications that might use it (Python, MATLAB and R). For the purposes of some OOI JupyterHub demonstrations, you will need to add that code and set it up in a virtual environment (note, this workflow could be changed to suit any custom environment you'd want to create) using the following text block (copy/paste into the terminal and hit enter; this may take a while to execute).

Again, you will only need to run the commands described in this section once. After the code is downloaded and the environment(s) are created, re-running these commands may cause issues.

```
# Download the ooi-data-explorations code and create the ooi environment
cd ~
```

```
mkdir code
cd code
git clone https://github.com/oceanobservatories/ooi-data-explorations.git
cd ooi-data-explorations/python

# Configure the OOI python environment
conda env create -f environment.yml
conda activate ooi

# install the package as a local development package
conda develop .

# add the newly created ooi environment to the list of JupyterLab kernels
python -m ipykernel install --user --name=ooi
```

After the last step, you should see “ooi” as an option in the “Notebook” and “Console” categories on the Launcher tab (you may need to log out and log back in, see Figure 4). This environment contains Python modules you can use to work with OOI data. More importantly, all the customizations from above will persist between login sessions, so you won’t need to re-create them every time you log back in.

For MATLAB users, from the launcher, click on “Open MATLAB” under the “Notebook” category. This will launch MATLAB in a new tab after you click on the “Start MATLAB Session” button. Navigate to the `/home/jovyan/matlab` folder and create a `startup.m` file with the following contents (and see Figure 8):

```
% startup.m
% Set up default user path and add the OOI Data Explorations code
userpath('/home/jovyan/matlab')
addpath('/home/jovyan/code/ooi-data-explorations/matlab', ...
        '/home/jovyan/code/ooi-data-explorations/matlab/utilities', ...
        '/home/jovyan/code/ooi-data-explorations/matlab/examples', '-end')
savepath

% Use a lower-level graphics renderer, otherwise plots don't work
set(groot, 'DefaultFigureRenderer', 'painters')
```

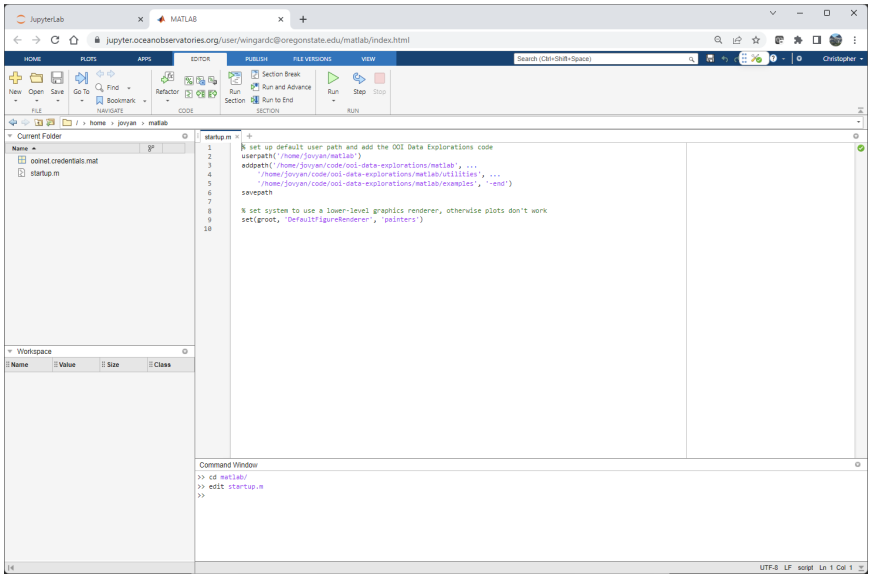

Figure 8. MATLAB launched from JupyterLab setting up the startup.m file for user customizations.

For R users, copy/paste the following code block into the terminal, appending needed customizations to the `.Rprofile` file created earlier:

```
# Make sure we are in the home directory
cd ~

# Add user customizations to .Rprofile
cat <<EOT >> .Rprofile
# Set up default user path and location to install local packages
.libPaths(c("/home/jovyan/.Ruser/lib", .libPaths()))

# Add the OOI Data Explorations code
.libPaths(c(.libPaths(), "/home/jovyan/code/ooi-data-explorations/R"))

.First <- function(){
    cat("\nWelcome at ", date(), "\n")
}

# Use .Last() function to save the workspace image to a consistent directory
.Last <- function(){
    save.image("/home/jovyan/.Ruser/data")
    cat("\nGoodbye at ", date(), "\n")
}
EOT

# Add the OOI M2M R package
conda activate ooi  # need the netCDF4 library from Python.
Rscript -e "devtools::install_github("oceanobservatories/ooim2mr")"
```

Testing the Environment

The final step is to test that everything is set up. This will entail copying an example notebook out of the ooi-data-explorations directory and into a folder that you will control, and then running that notebook to confirm everything is working as expected. Paste the following code block into the terminal:

```
# Make sure we are in the home directory
cd ~

# Create a local folder for working with different OOI data sets
mkdir -p data/adhoc/testing

# Copy some example notebooks over from the code directory
cd code/ooi-data-explorations/python/examples/notebooks/phsen/
cp * ~/data/adhoc/testing
cd ~/data/adhoc/testing
```

Note, you can also use the file browser on the left to accomplish the above (can create folders and files and copy/paste). Use the file browser on the left to navigate to the top folder and then into `adhoc/testing` folder. Double-click on the `creating_annotations.ipynb` file to start the python notebook. You will need to change the kernel that is running in that notebook to `ooi` before running the notebook. In the upper right corner of the notebook (Figure 9, red oval), you will see text indicating the name of the Python kernel being used. By default, it will be `Python 3 (ipykernel)`. Click on that to change the kernel to `ooi`.

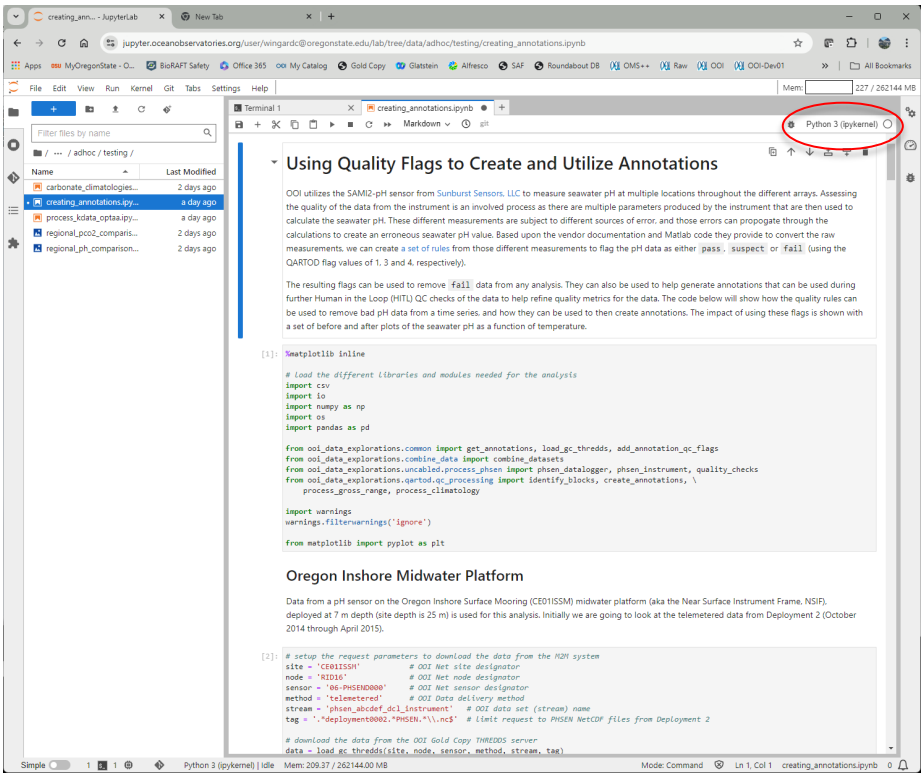

Figure 9. Change the kernel to **`ooi`**. *Clicking on "Python 3 (ipykernel)" shown in the red circle will open a small window with a drop-down menu that you can use to select the ooi kernel.*

At that point you can run through the notebook (**`shift+enter`** will advance through each cell highlighted by the blue vertical bar, Figure 9).

If for some reason this step doesn't work, indeed if any of the above doesn't work, please contact either the OOI HelpDesk (helpdesk@oceanobservatories.org) or post a question in the OOI Discourse forum

Additional and Sundry

GitHub updates

The ooi-data-explorations code is a work in process and is being updated and refined regularly. To make sure you have the most recent code, copy/paste the following code block into a terminal:

```
cd ~/code/ooi-data-explorations
git checkout master
git pull
```

CILogon Reset

As noted above, the OOI JupyterHub uses CILogon for access credentials. If by chance you make a mistake in setting these up, you will need to delete the cookies CILogon creates as part of the validation process. This is relative straightforward: open the following URL in your browser (https://cilogon.org/me/) and delete the browser cookies.

Hub Control Panel

If by chance you choose the “wrong” server type, you can easily change to a different one. In the JupyterLab, under the File menu, click on the Hub Control Panel. This will bring up a page where you can opt to stop your server. Wait for the server to stop, close any open JupyterHub or JupyterLab tabs and then you can login to the JupyterHub selecting a different server type (see Figures 10 and 11).

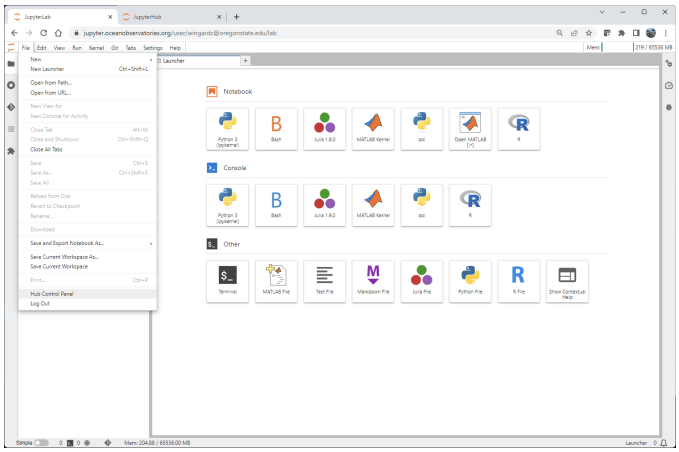

Figure 10. To exit from the current server (if you need to change server types), select File > Hub Control Panel from the JupyterLab menu.

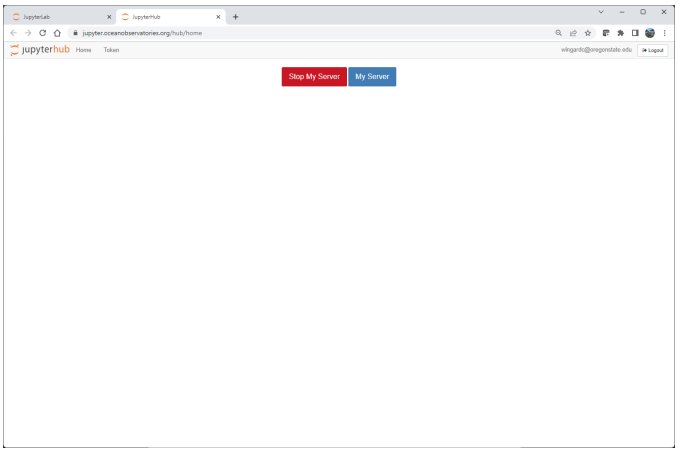

Figure 11. Select “Stop My Server” to end the current server session and then close the JupyterLab and JupyterHub tabs once the server has stopped.

MATLAB JVM Heap Memory

If you are using MATLAB, it is a good idea to increase the Java Runtime Environment heap size. The default is 2 GB. You are encouraged to increase this to 8 GB. You can adjust this under the Preferences > General > Java Runtime Environment > Java Heap Size (Figure 12).

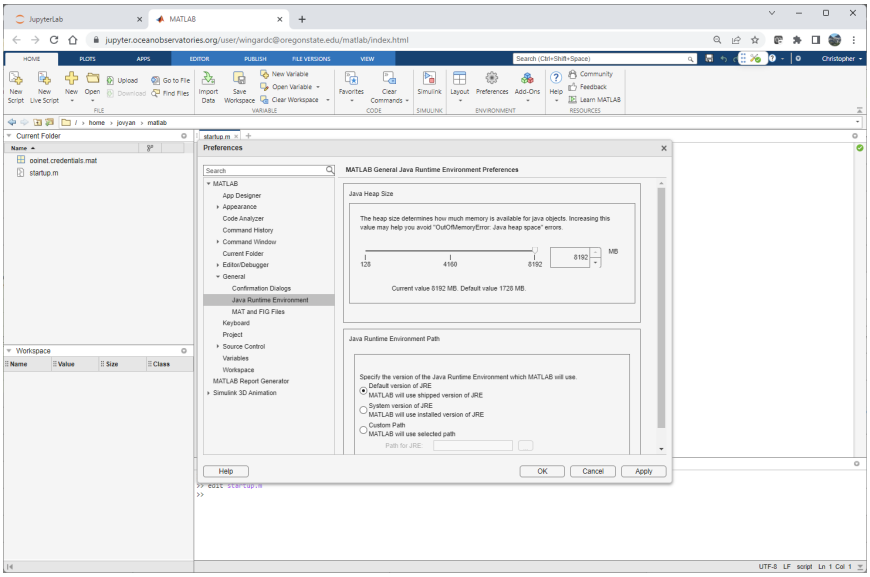

Figure 12. If using MATLAB – How to increase the Java Runtime Environment heap size