Azimuthal integration workflow for

2D XRD mapping at ID13

Azimuthal data integration can be performed through the Juno project provided by Loïc Huder (ESRF). This project makes use of jupyter notebooks that can be run individually and independently by users to azimuthally integrate 2D XRD data. A large part of the workflow given here has been adapted from the documentation of Loïc Huder which is available at <https://gitlab.esrf.fr/loic.huder/juno> and <https://gitlab.esrf.fr/loic.huder/juno/-/blob/main/Using_notebooks.ipynb>

Contents

[1. Start up your Juno environment 2](#_Toc95207018)

[2. First time only! 4](#_Toc95207019)

[3. Upload/Create *run\_config.yml* file 5](#_Toc95207020)

[3.1. Upload a *run\_config.yml* file 5](#_Toc95207021)

[3.2. Create a new *run\_config.yml* file 6](#_Toc95207022)

[4. Set parameters in *run\_config.yml* 8](#_Toc95207023)

[4.1. Finding the input\_file (path and name of your dataset) 11](#_Toc95207024)

[4.2. Common mistakes 13](#_Toc95207025)

[5. Run integration in *2a\_integration\_maps.ipynb* 14](#_Toc95207026)

[5.1. Integrate a single data set 14](#_Toc95207027)

[5.2. Optional 19](#_Toc95207028)

[6. Transfer and download your data 21](#_Toc95207029)

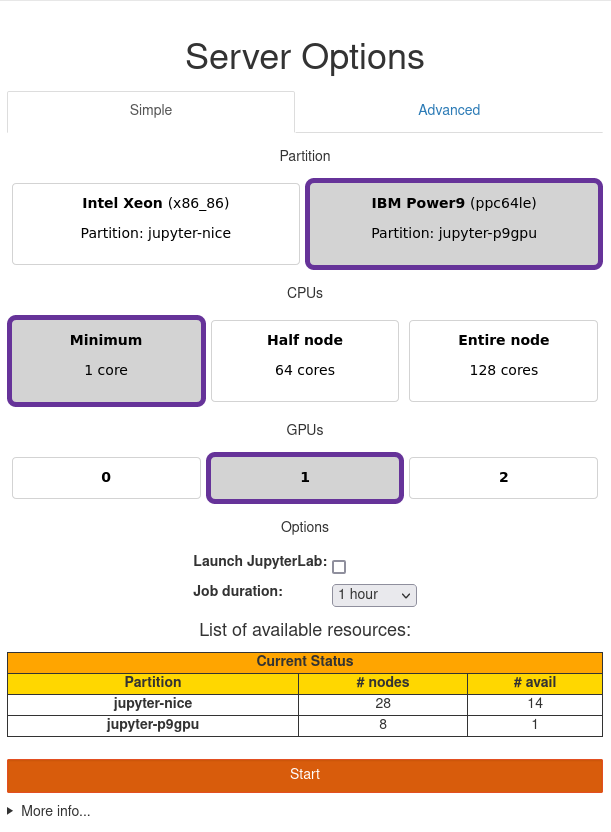
[7. Update Juno 22](#_Toc95207030)

# Start up your Juno environment

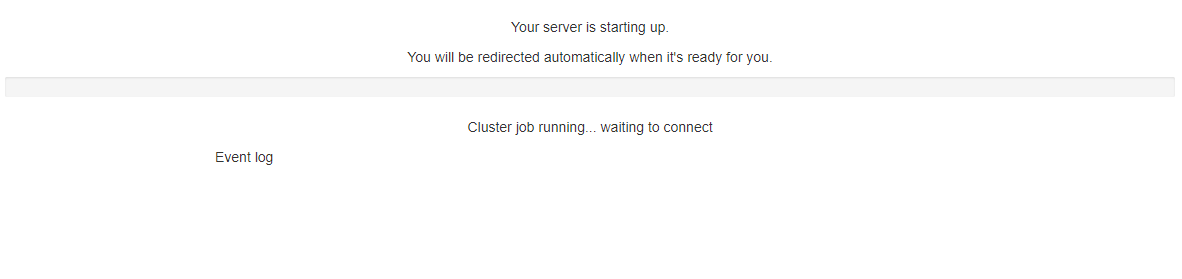
Go to [jupyter-slurm](https://jupyter-slurm.esrf.fr/hub/spawn) or <https://jupyter-slurm.esrf.fr/hub/spawn> and login with your ESRF account.

Connect to a session and request a node on the jupyter-p9gpu partition **with a GPU.**

* **Select IBM Power9 in the Partition section**
* **Select Minimum (1 core) in the CPUs section**
* **Select 1 in the GPUs section**
* **Change the job duration to a higher number if you are processing multiple datasets**
* **Press Start**

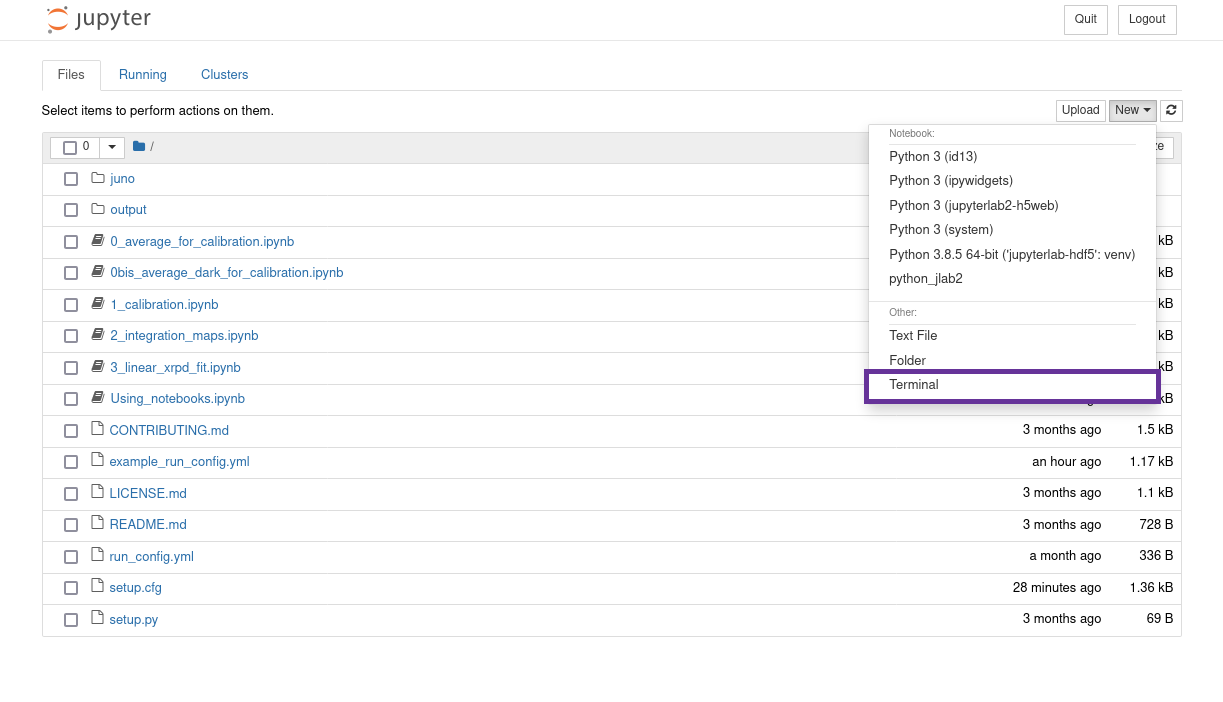


This will take you to a loading screen …

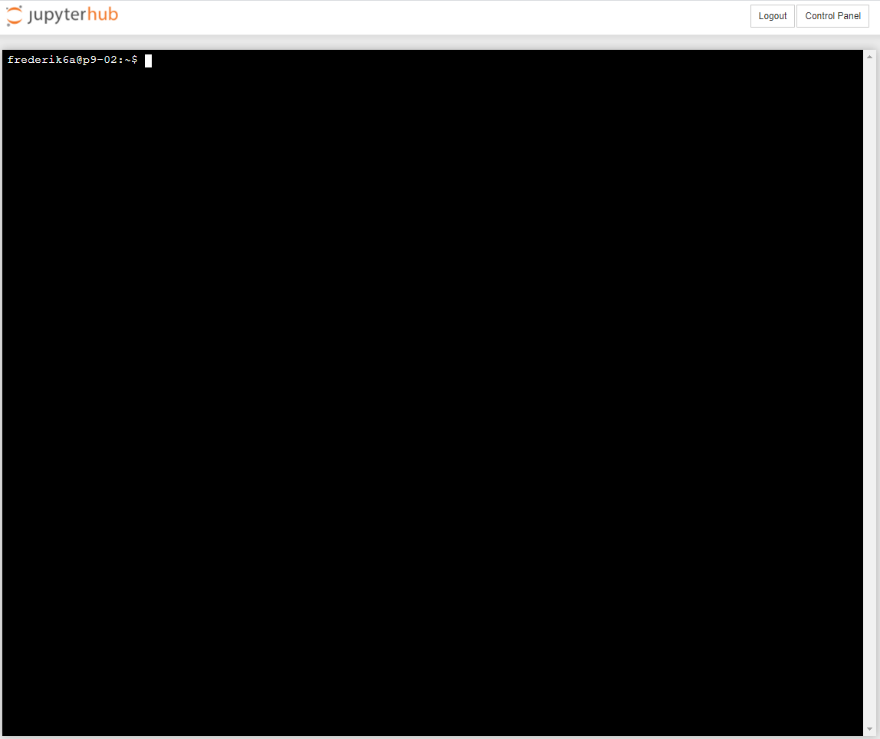


… and after some seconds a new page will open. We will name this new page the **Project window**.

In the **Project window**, open a terminal.



This will open a new window with your terminal. We will name this the **Terminal window**. You will need to work in both windows during the rest of the workflow, so do not close these.



# First time only!

Only when you started your session for the first time, you need to copy the juno project to your personal session. Anything that you change here is only for you and will not affect other users. After you close your session and start again from ‘Start up your Juno environment’ you do not need to repeat the steps below.

To copy the juno folder, go to the **Terminal window** and run the following command

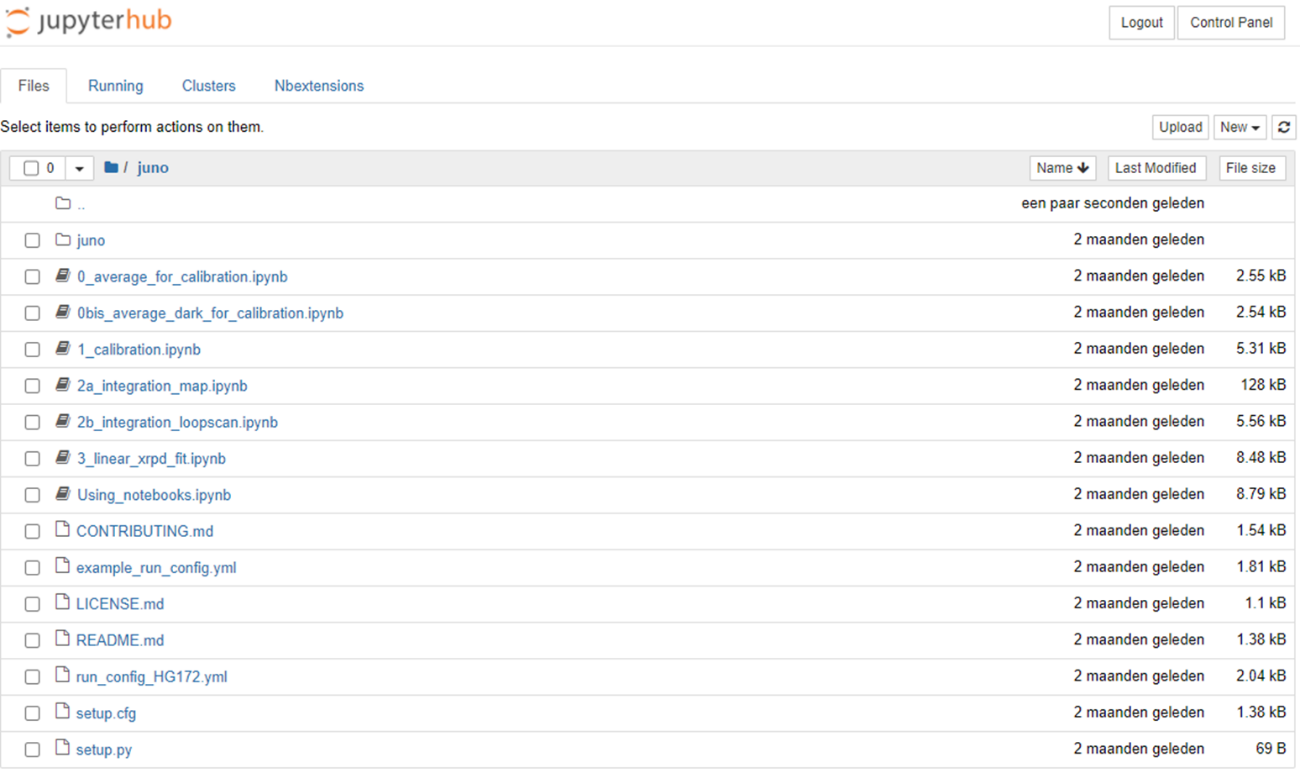
>> git clone https://gitlab.esrf.fr/loic.huder/juno

This will create a juno folder containing the notebooks and all the project files. In your **Project window** a new folder *juno* should have been created.



If you click on the *juno* folder in your **Project window**, you will be able to see all the project files.

it



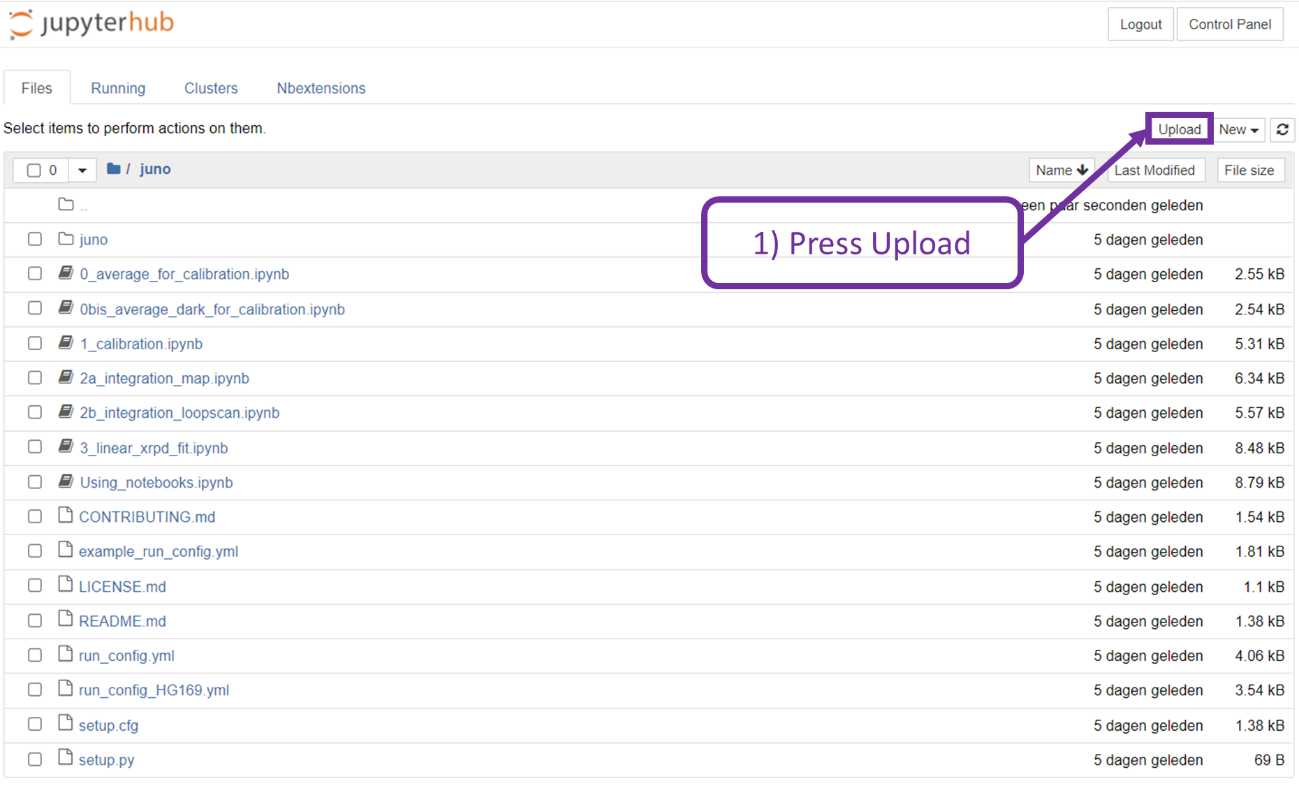
# Upload/Create *run\_config.yml* file

The *run\_config.yml* file contains several parameters for the data integration and will be needed in the next steps. In this step you will learn how to upload an existing file to your juno folder (3.1) or how to create a new config file (3.2).

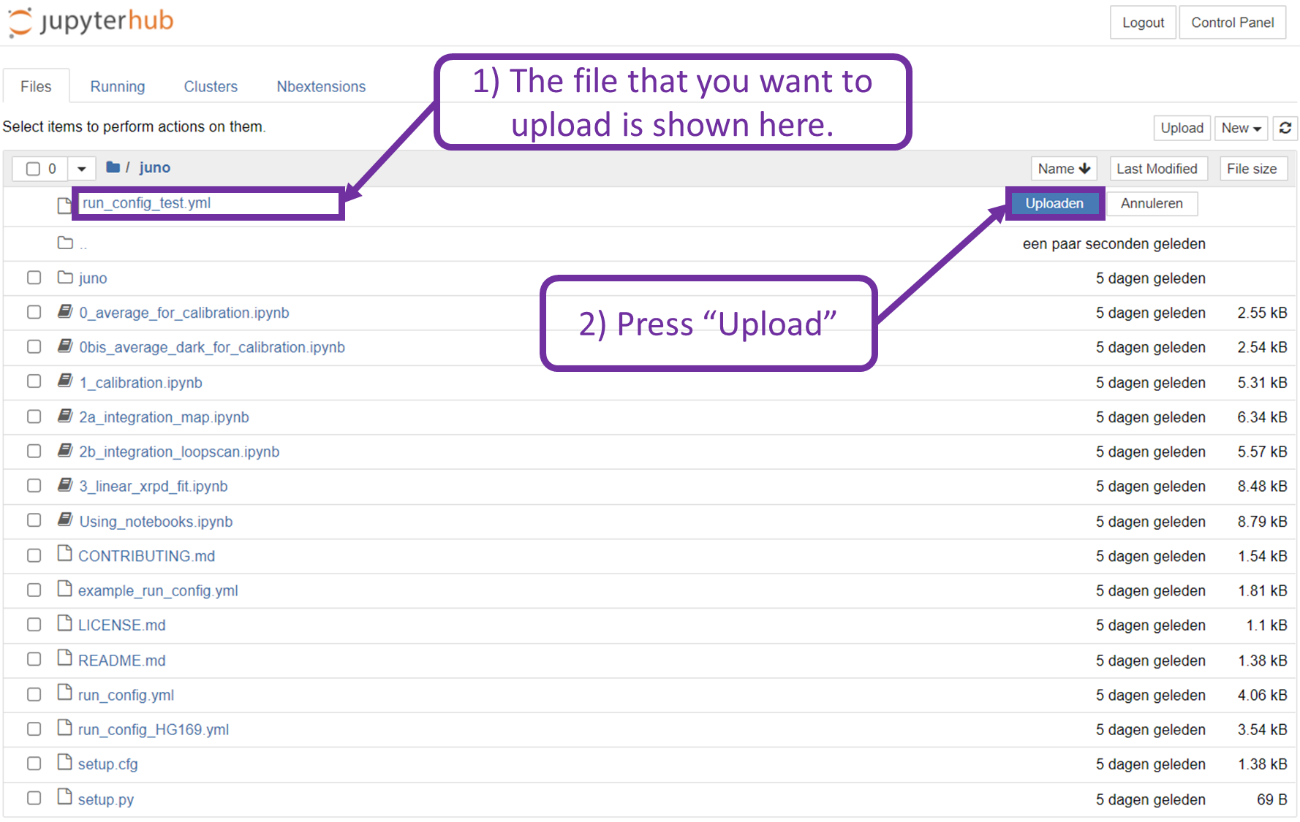
## Upload a *run\_config.yml* file

If you have received a *run\_config.yml* file (from the RSs) that is specific for your beamtime, you can upload it to your juno folder through the **Project window**. You can then use this config file for section 4 in this workflow. Typically a *run\_config.yml* file will have been created during the experiment in the XXPROCESS\_jupyter folder (/data/visitor/hg172/id13/XXPROCESS\_jupyter) and will have been uploaded to the Heritage BAG cloud space.

In the **Project window**, click on Upload and select the config file in the pop-up window.



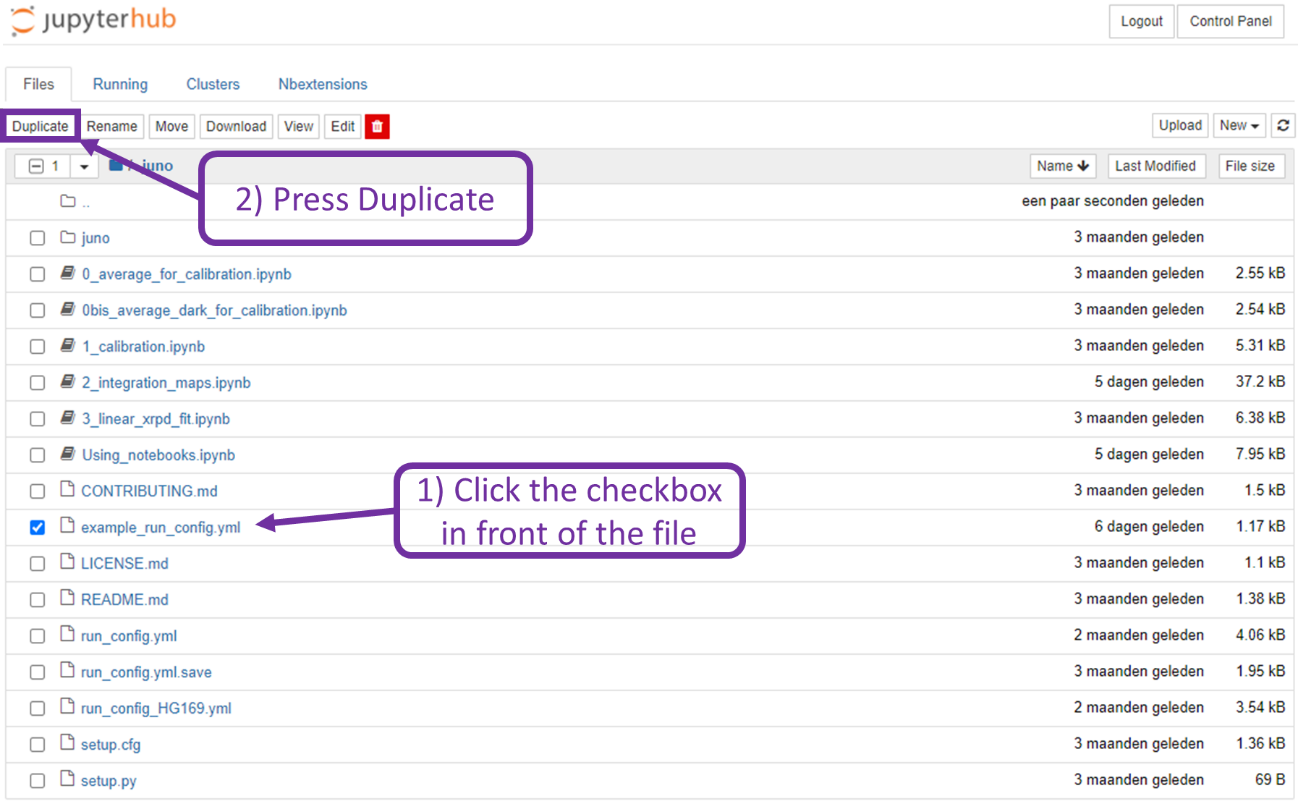
In the example below, the *run\_config\_test.yml* was selected in the pop-up window. Check to make sure that you selected the correct file and press the new blue Upload button. The new config file should now have been added to your juno folder.



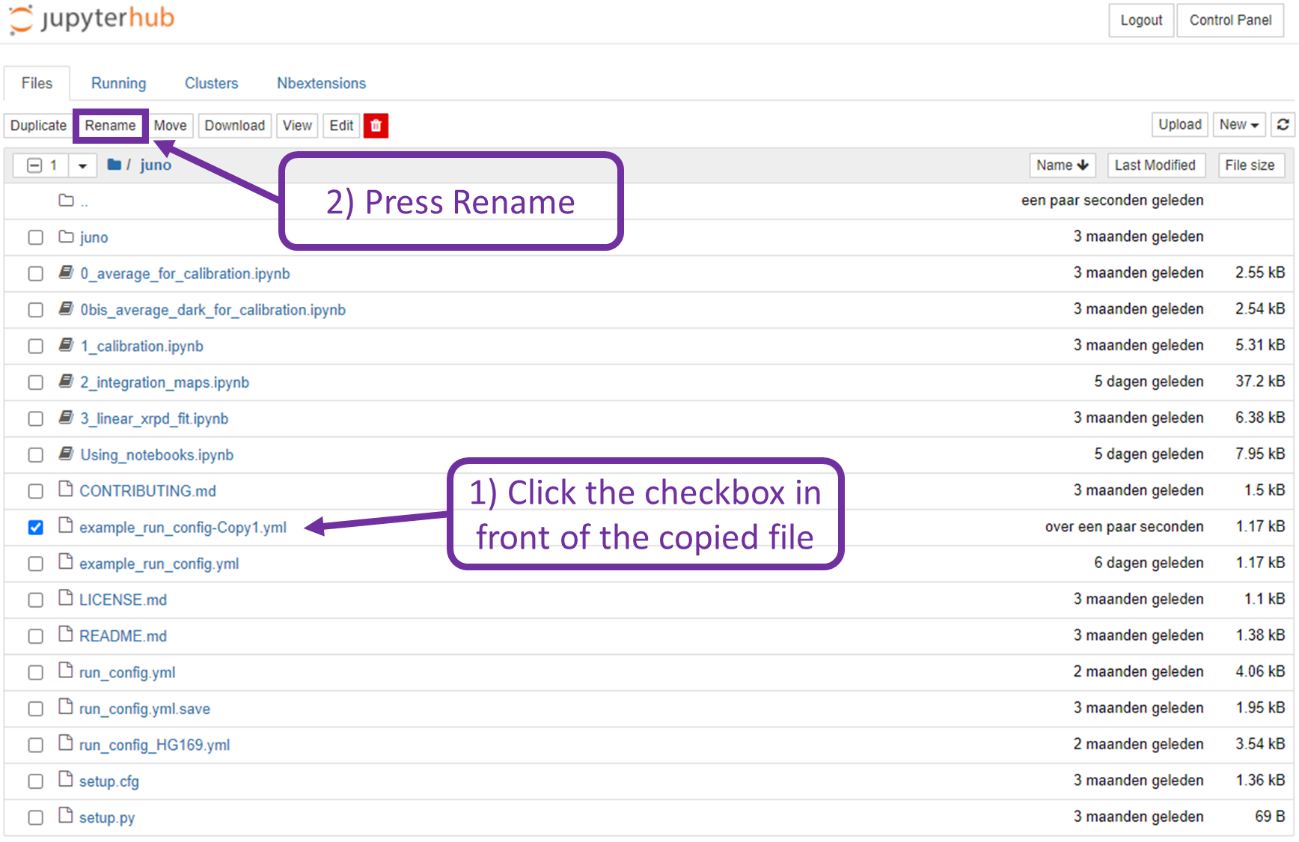
## Create a new *run\_config.yml* file

The easiest way to create the *run\_config.yml* file is by copying an existing/old *run\_config.yml* file and renaming it. In this example, the existing file *example\_run\_config.yml* will be copied and the copy will be renamed to *run\_config.yml*. There are two ways to copy and rename this file.

You can do this in the **Project window**. In the **Project window**, click on the checkbox in front of *example\_run\_config.yml* and press duplicate. This will open a pop-up window that will ask you to confirm. The filename of the copied file will be the same as the original to which “-Copy1” has been appended.



After this, click on the checkbox in front of the copied file and press rename. This will open a new pop-up window where you can give the new filename. It might be a good practice to include the experiment number to this filename (e.g., *run\_config\_HG159.yml*) since more run\_config files will start to appear as more experiments pass.



Alternatively, you can copy and rename *example\_run\_config.yml* in the **Terminal window** that you opened earlier.

In the **Terminal window**, first go to the *juno* folder

>> cd juno

Second, copy and rename the file *example\_run\_config.yml*

>> cp example\_run\_config.yml ./run\_config.yml

Congratulations! Your Juno environment and project has been set up and you can start the next steps in the data integration workflow!

# Set parameters in *run\_config.yml*

In the file *run\_config.yml* you need to set several parameters to perform the integration. You mainly need to specify the folders that contain the data and the folders to which the data needs to be saved as well as the dataset that you want to process.

In your **Project window**, click on the *run\_config.yml* file that you uploaded (section 3.1) or created (section 3.2) previously. This will open a new window with a color formatted text file that you can edit.

In this file the following color format is used

* The text in green starts with the symbol #. These are comments that provide you with more information but do not affect the integration.
* The text in blue are parameters that are called within the notebooks. Do not change these!
* The text in red are strings that you can change.

**You only need to change (some of) the text that is shown in red.**

The proposal\_folder should specify the folder that contains your datasets. Typically all data is stored in the main directory of *ID13*. The name of the datasets start with the initials of the proposer (e.g., FV\_dataset1 for the first dataset of Frederik Vanmeert). If you do not know the name of the folder that contains your data or the name of your dataset, see section 4.1.

proposal\_folder: “/data/visitor/hg172/id13”

The output folder is where the integrated data will be saved. All processed files that are created using the Jupyter notebooks should be stored in the *XXPROCESS\_jupyter* folder.

output\_folder: “/data/visitor/hg172/id13/XXPROCESS\_jupyter”

e.g., output\_folder: “/data/visitor/hg172/id13/XXPROCESS\_jupyter”

You should not change things in the sections average, dark and calibration. These have already been set for you if you uploaded the file from the RSs.

In the section integration, you need to specify the path and file name of your dataset and the file name that needs to be given to your integrated data. The paths are relative to the paths of the proposal\_folder and output\_folder that you specified earlier, respectively for the input\_file and output\_file. You do not need to change the mask\_file if you uploaded the file from the RSs.

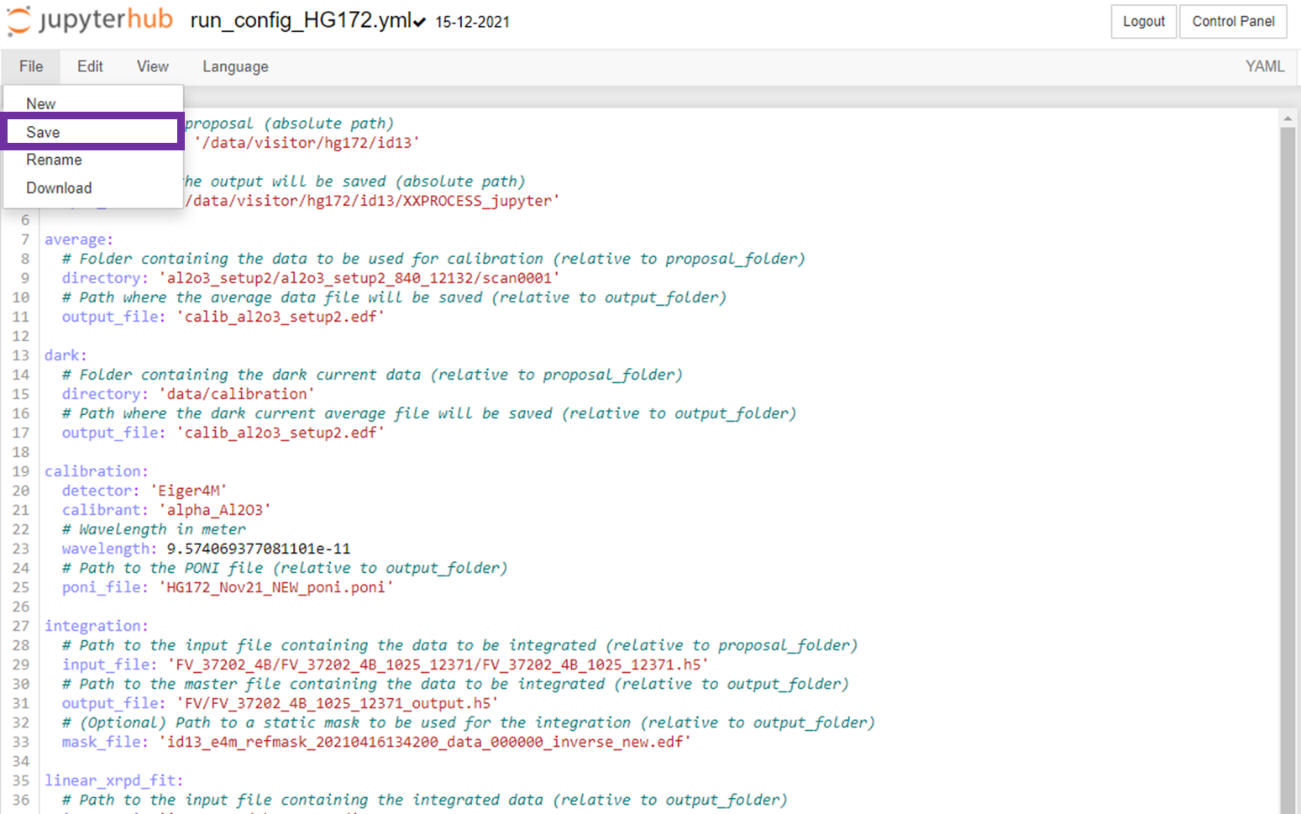
input\_file: “your\_sample/your\_dataset/your\_dataset.h5”

e.g., input\_file: “FV\_test/FV\_test\_dataset/FV\_test\_dataset.h5”

output\_file: “INITIALS/your\_dataset\_output.h5”

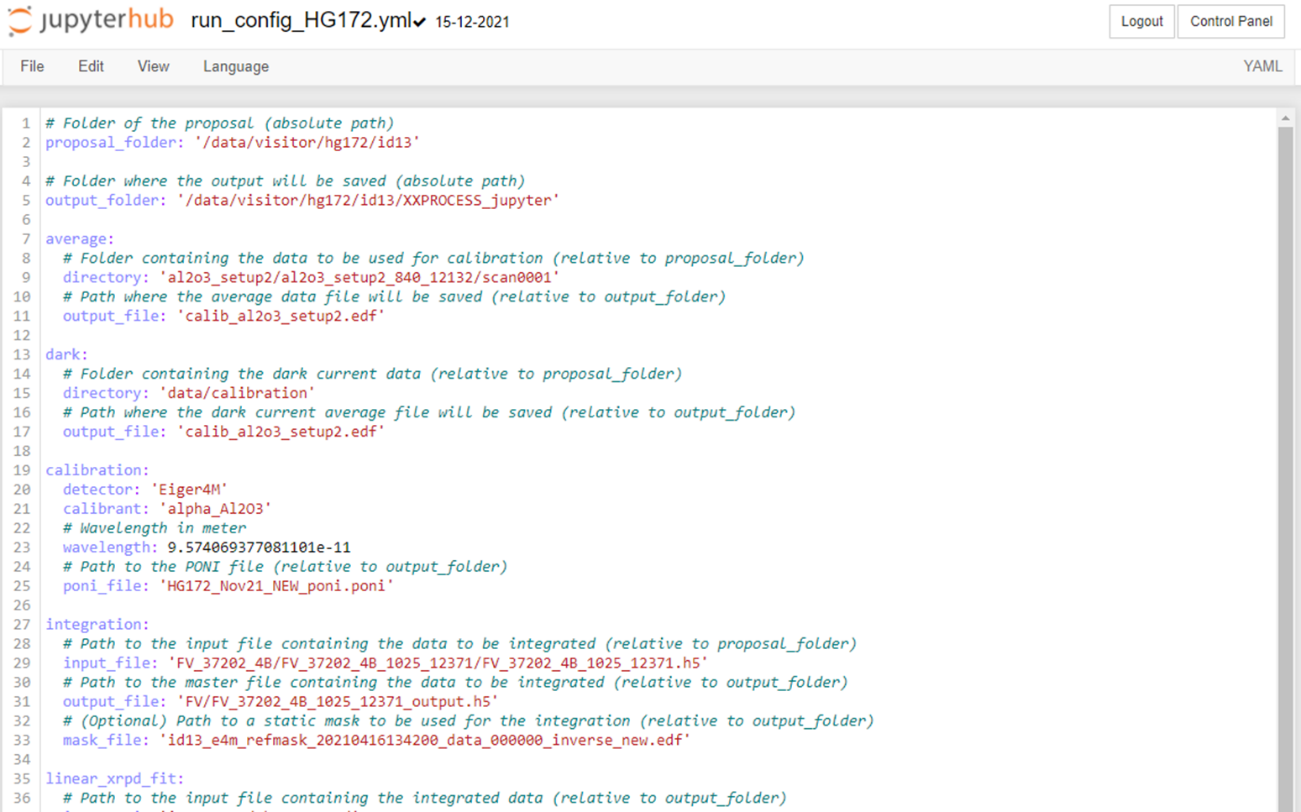
e.g., output\_file: “FV/FV\_test\_dataset\_output.h5”

Go to File -> Save to save your *run\_config.yml* file.



Be sure that all parameters are properly set as the whole config will be tested at each step!

Your final configuration file should look like this.



To keep things organized your data will be stored in a folder with your initials (as set as the output\_file in the *run\_config.yml* file). If the folder with your initials does not yet exist within the *XXPROCESS\_jupyter* folder, you first need to create it before proceeding with section 5. You will need to do this in the **Terminal window**.

In the **Terminal window**, go to the output\_folder that you specified above

>> cd /data/visitor/hg172/id13/XXPROCESS\_jupyter

Use the mkdir command to create a folder with your initials inside the output\_folder

>> mkdir “your\_INITIALS”

e.g., >> mkdir FV

This will create a folder with the name ‘FV’ in the *XXPROCESS\_jupyter* folder. This folder will contain all your processed files after performing the steps in section 5.

## Finding the input\_file (path and name of your dataset)

During the experiment, unique names and folders are created for each sample and measurement. The naming conventions are the following:

* INITIALS : the initials of the proposer to which the sample belongs
* sample\_name : the name of the sample as entered in the cloud wish list
* sample\_id : a unique number created by Daiquiri once the sample has been added
* measurement\_number : a unique number created by Daiquiri once a scan/measurement has been created

The following folders will be created:

* INITIALS\_sample\_name : for each sample
  + e.g., FV\_test
* INITIALS\_sample\_name\_sample\_id\_measurement\_number : for each measurement
  + e.g., FV\_test\_830\_10904

If you do not know where your datasets are stored (i.e., you do not know what to put for the input\_file), you can check this using the **Terminal window**.

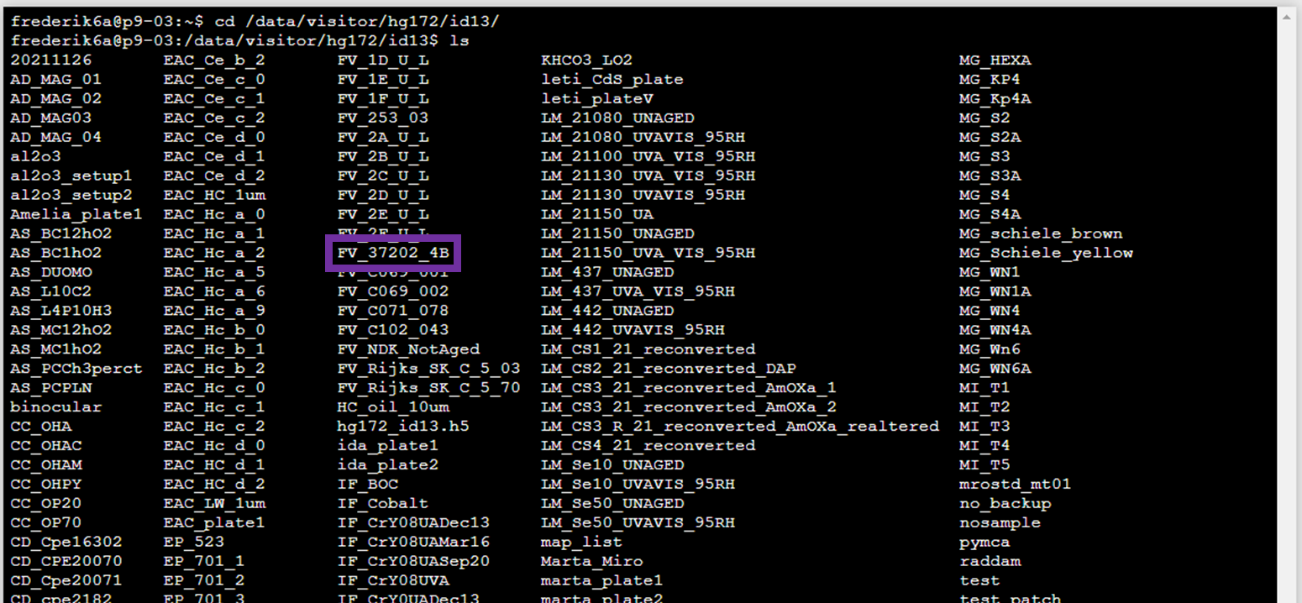
In the **Terminal window**, go to the *ID13* folder of the BAG proposal

>> cd /data/visitor/hg172/id13

Use the ls command to list all the files and directories that are in the *id13* folder

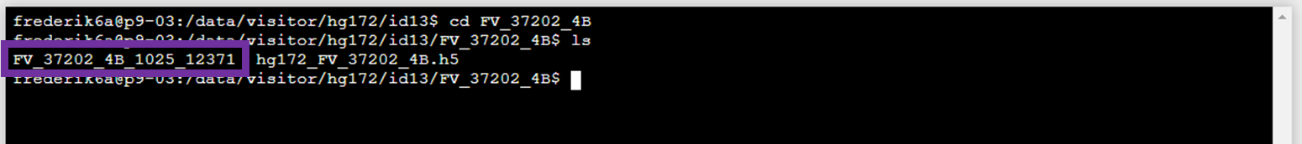
>> ls

This will give you a list of files and directories. In this list, you should find the name of the folder that contains your dataset. The example given below is the sample folder for sample 37202\_4B of proposer Frederik Vanmeert.

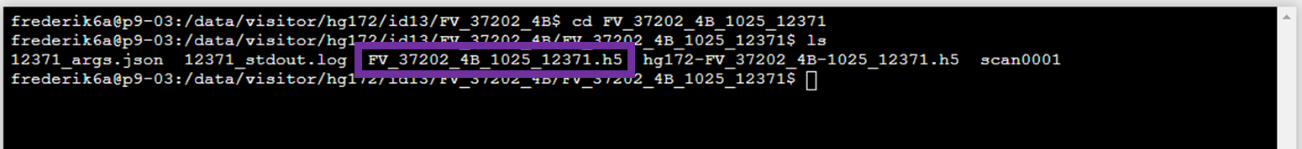


Inside each sample folder, you will find a folder for each measurement that was performed on this sample. Your dataset is inside this measurement folder.

Use the commands ‘cd’ and ‘ls’ to go inside a sample folder and list it contents and again to go inside the measurement folder and list it contents. The file that is required as input\_file is inside the measurement folder. The input\_file has the same name as the measurement folder and has the extension .h5.



In the example above, the input\_file is *FV\_37202\_4b\_1025\_12371.h5* and it is located inside the folder *FV\_37202\_4b/FV\_37202\_4b\_1025\_12371*.



## Common mistakes

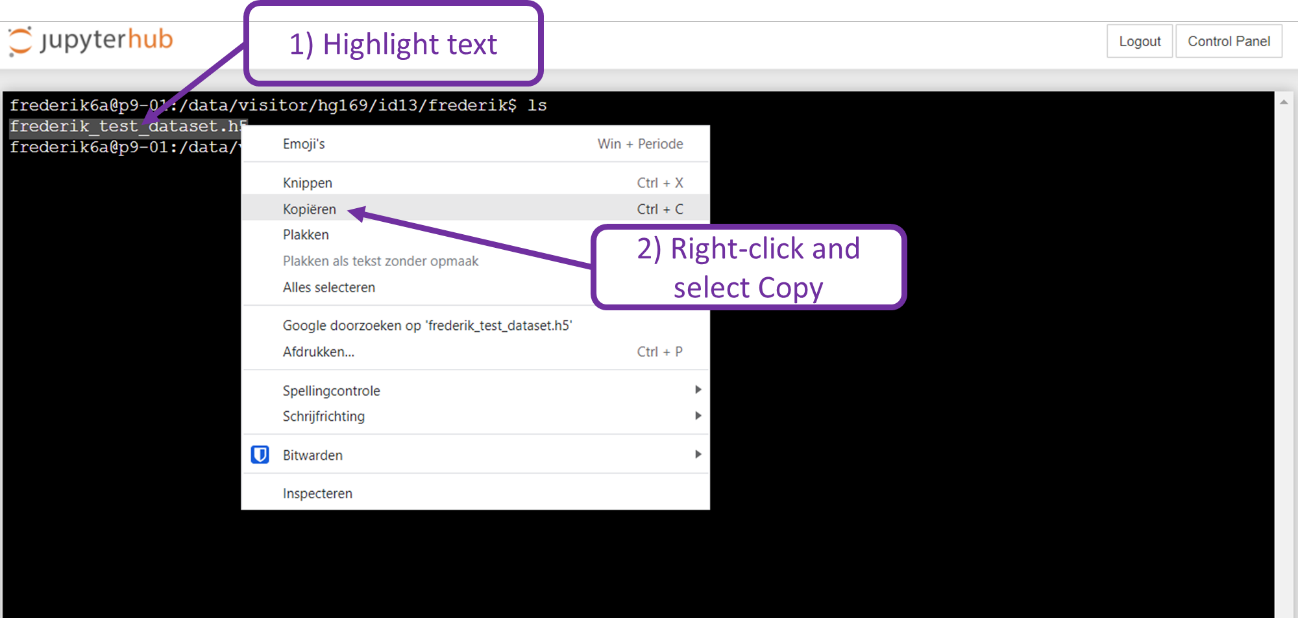
Attention!

* There is a difference between *Frederik* and *frederik*. Take care to make sure that you correctly use CAPITAL letters or small letters.
* Be careful not to make typo’s or errors in the names of the directories
* Do not add white spaces in the red text

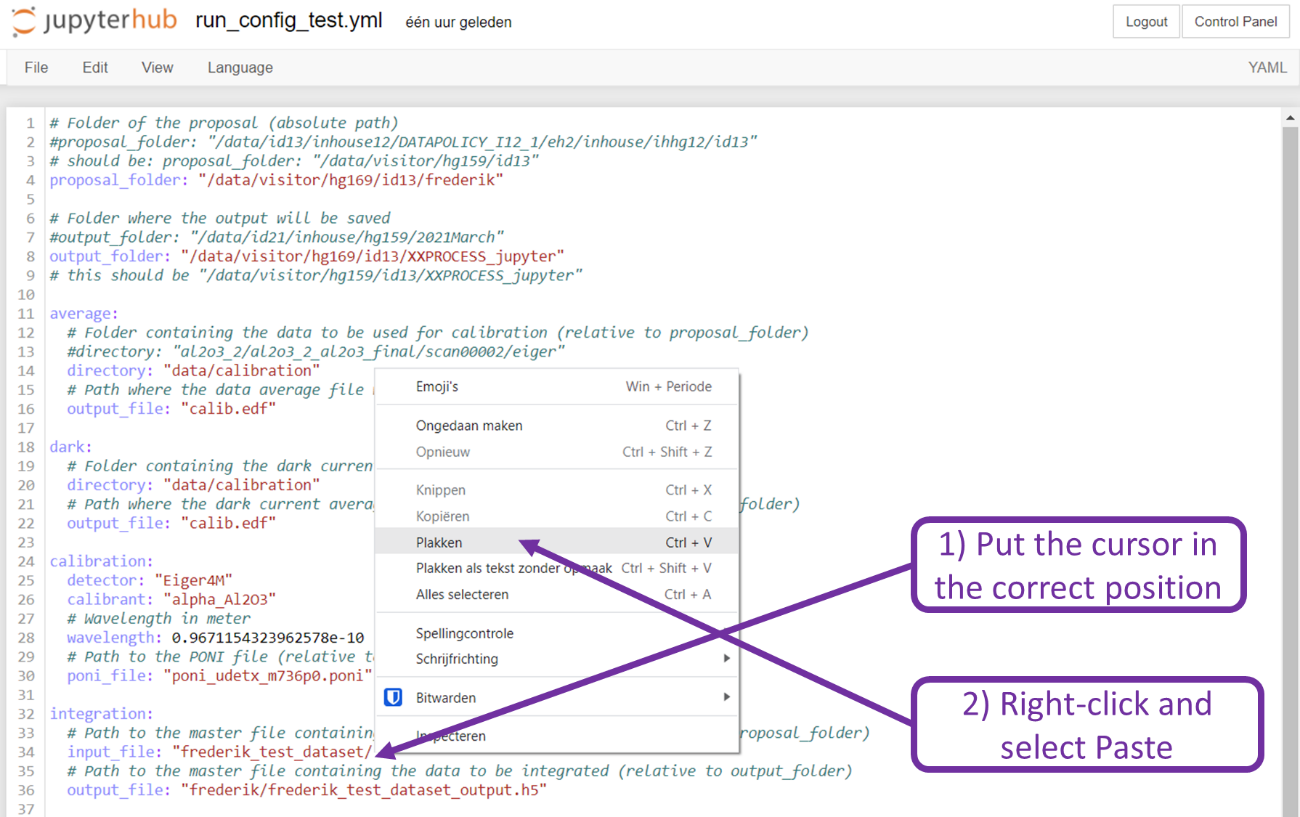
If you get an error in the next step (section 5), most likely you will have made a mistake while changing the red text.

The best way to avoid the above mistakes is to copy and paste as much as possible the text that you need to change. You can do this directly from the **Terminal window** into the *run\_config.yml* file. In the example below I am changing my input\_file by copying and pasting the filename of my dataset into the *run\_config.yml* file.

In the **Terminal window**, you can select text using your mouse by clicking and highlighting the text that you need. You can then right-click and select Copy.



In the *run\_config.yml* file, you can paste the text by right-clicking your mouse and selecting paste.



# Run integration in *2a\_integration\_maps.ipynb*

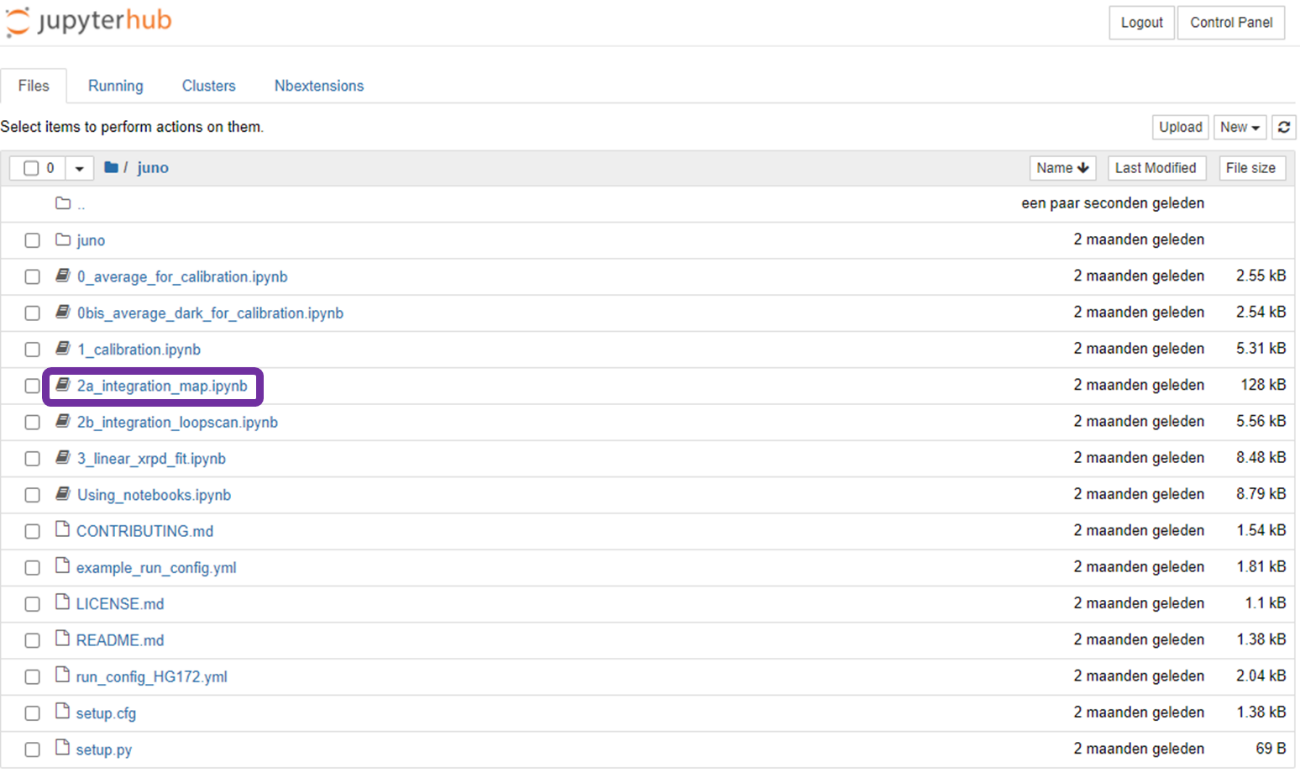
## Integrate a single data set

In your **Project window** you will find several jupyter notebooks.

* *0\_average\_for\_calibration.ipynb*: to make an average 2D XRD pattern to use for calibration
* *0bis\_average\_dark\_for\_calibration.ipynb*: to make an average dark image for the calibration
* *1\_calibration.ipynb*: determines the geometry of your detector and creates a .PONI file to be used during the integration
* *2a\_integration\_maps.ipynb*: azimuthally integrates your dataset
* *3\_linear\_xrpd\_fit.ipynb*: performs a linear combination fit of your data with given reference files

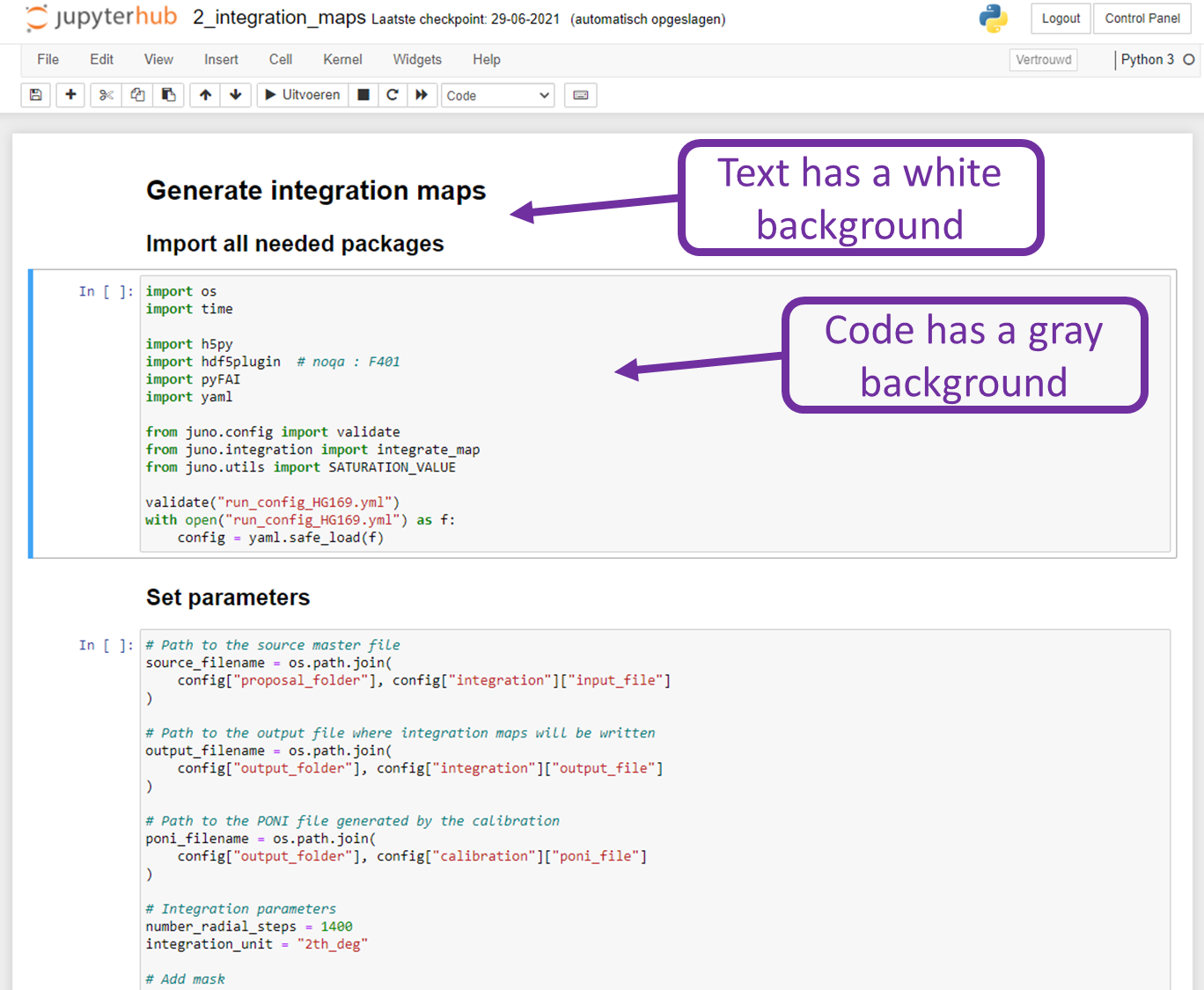
Optional: More information about these notebooks can be found if you click *Using\_notebooks.ipynb* in your **Project window**.

In this workflow, you only need to use the notebook *2a\_integration\_maps.ipynb*. If you click on it a new window will open with the notebook.



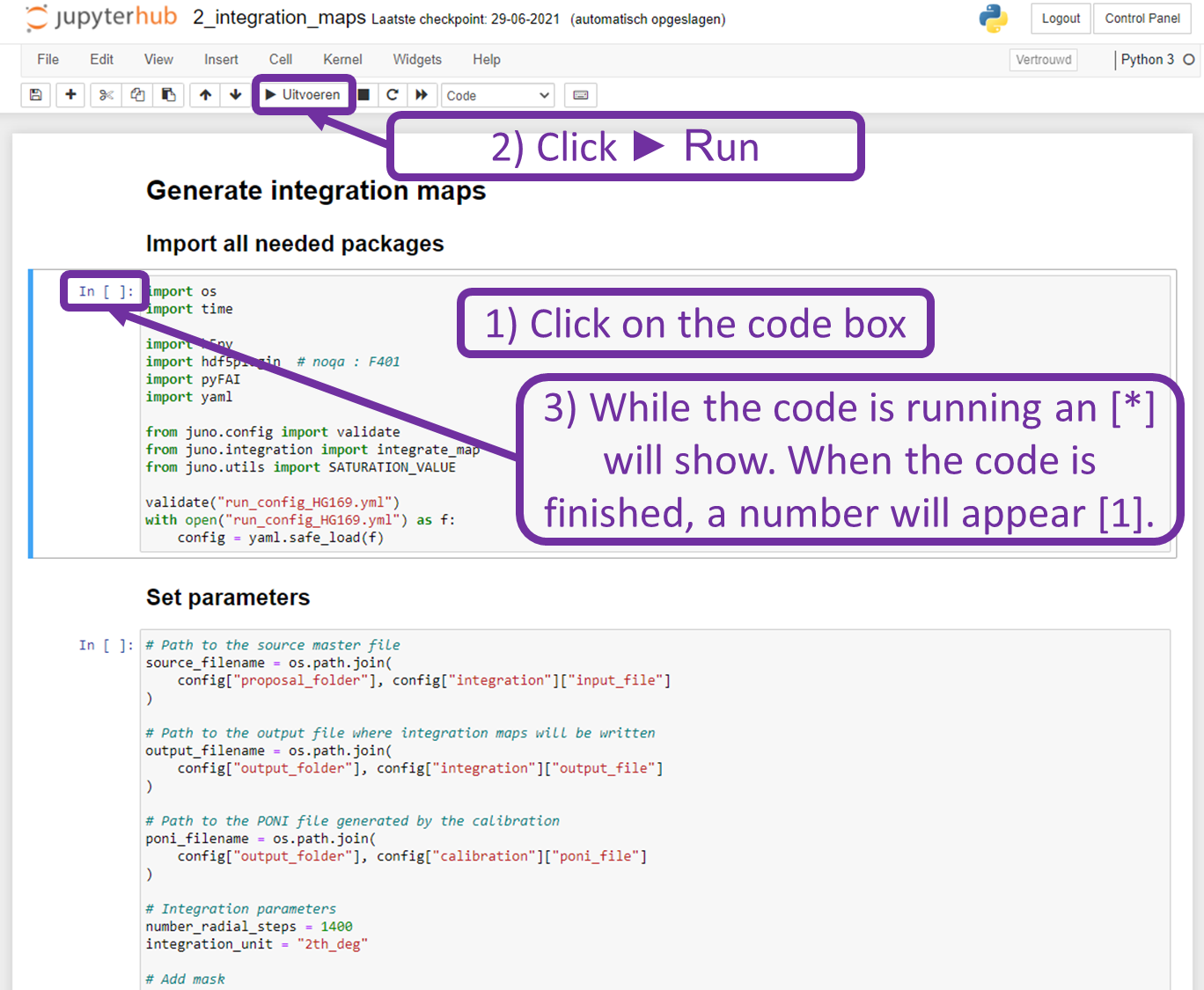
A notebook typically contains several sections with text and code. These sections are called cells. The code has a gray background and is grouped in a single cell.

In this notebook you do not need to change anything. What we need to do is to select the different cells of code by clicking on them once. When you have selected a code cell, a blue or green vertical line will appear on the left side of the cell. In the example below, the first code cell has been selected and shows a blue vertical line on the left side.



We need to select the code cells and run them one after each other to go through the integration notebook, see the scheme below.

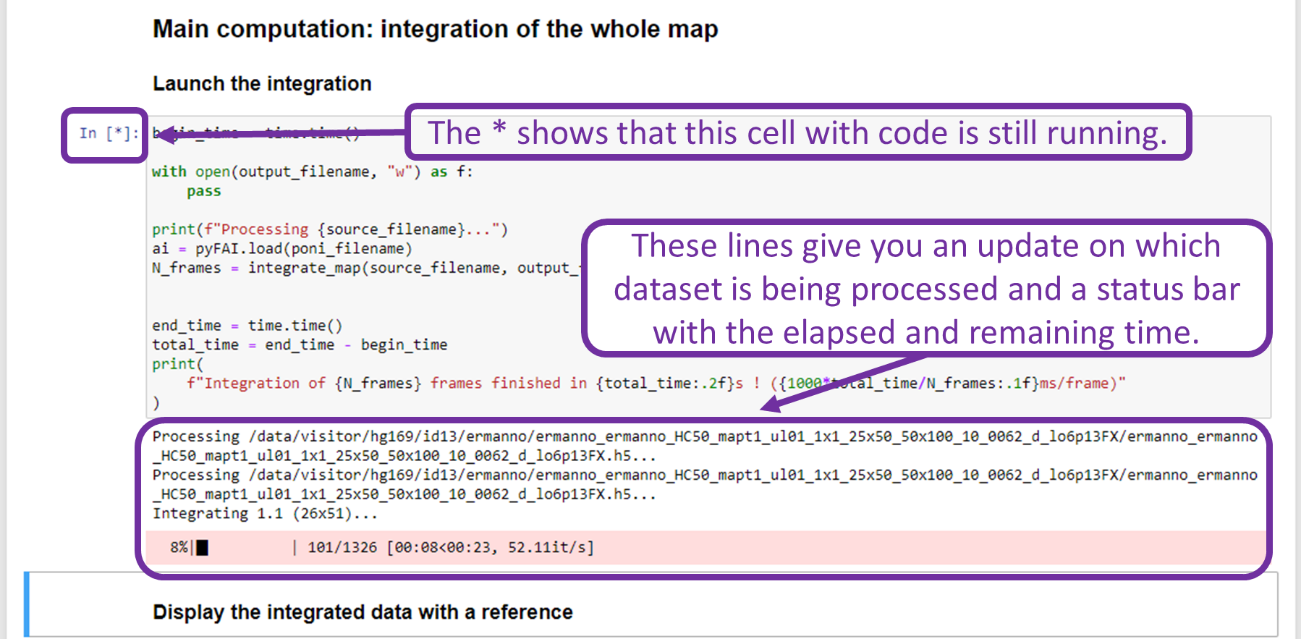
1. First, you need to select the first code cell underneath ‘**Import all needed packages**’ by clicking somewhere in the gray area.
2. Then, you need to click on the ► Run. Since the language depends on your browser settings, this may read something different than ‘Run’.
3. While the code is running an asterisk will appear between the brackets [\*] and when the code has finished a number will appear in between the brackets [1].



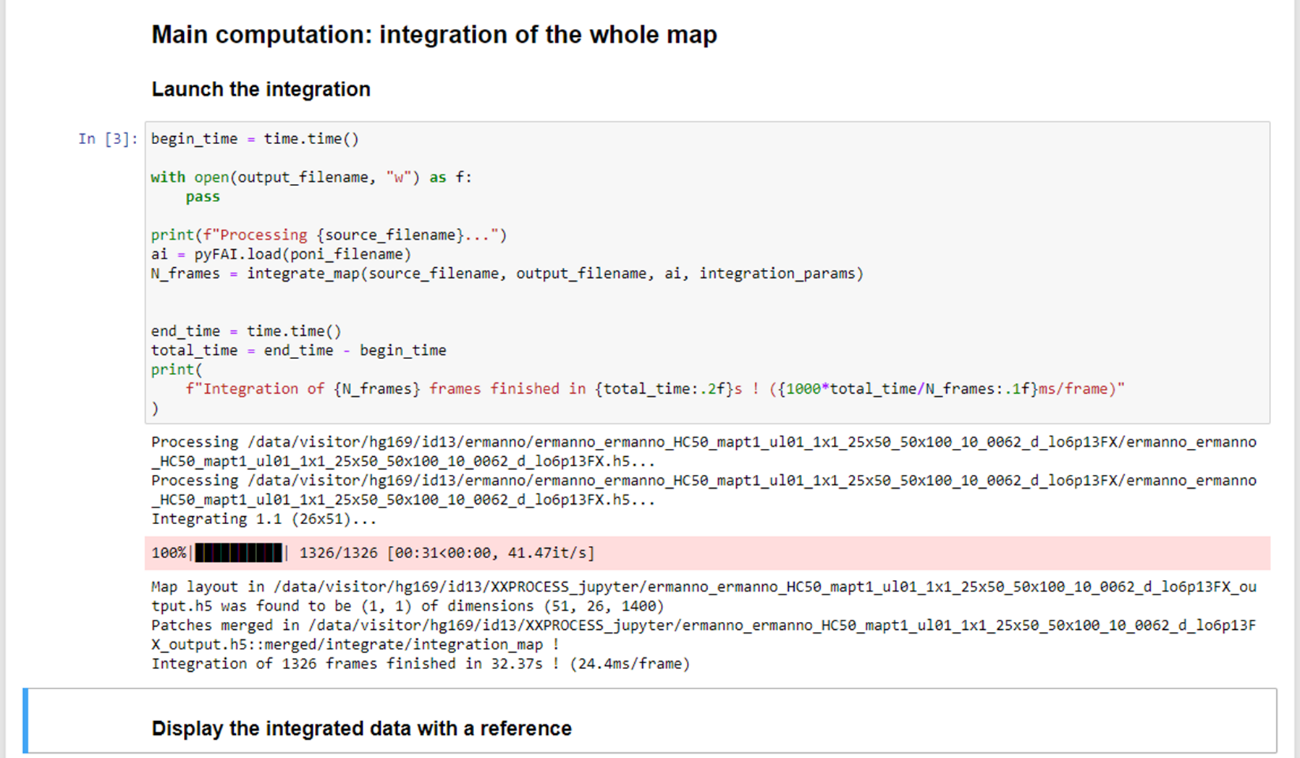
After running the first cell, all the packages that the jupyter notebook needs have been loaded.

In the second cell with code the parameters for the integration will be set. Again, you do not need to change anything in the code. You need to follow the 3 steps shown above, but now for the code cell that is shown underneath ‘**Set parameters**’ (i.e., click on the gray cell with code, press ► Run and wait for the code to finish).

The third code cell (the cell under ‘**Launch integration**’) will perform the actual integration. Again, click on the cell and press ► Run. Since this cell performs the actual integration, it will take some time for the code to complete. A status bar is shown in red just above the next cell.



After the code has completed the integration, it will show the following message. Your data will have been saved in the folder *XXPROCES\_jupyter* in a subfolder that you specified in the *run\_config.yml* for the output\_file (see section 4 of this workflow).



Congratulations! You have now successfully performed the azimuthal integration of your 2D XRD map. In the next step (Transfer and download your data) you can see how you can download this data.

Optionally, you can run several other cells in the current notebook (*2a\_integration\_maps.ipynb*), see 5.2.

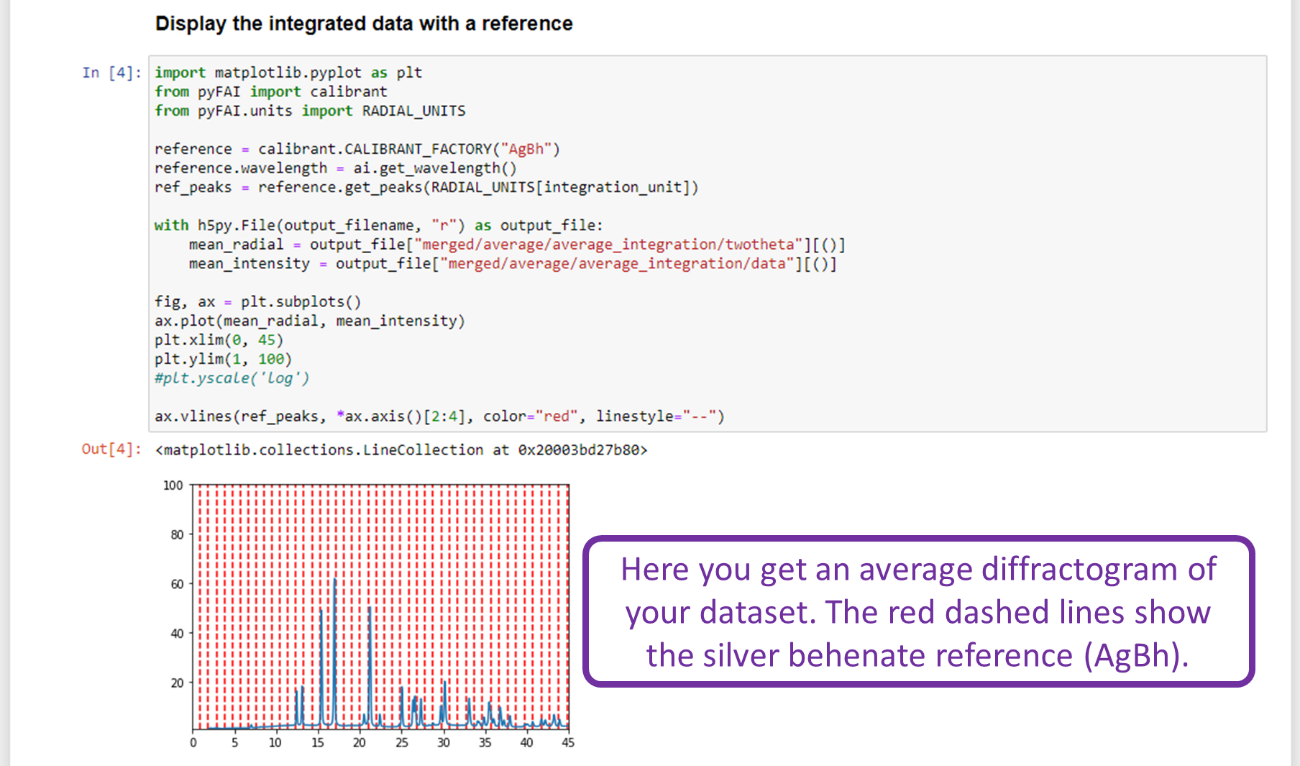
If you want to integrate another data set, you need to restart the procedure starting from section 4. As long as your datasets are in the same proposal\_folder, you only need to change the input\_file and output\_file in your *run\_config.yml* file.

## Optional

In the notebook *2a\_integration\_maps.ipynb* several other code cells are present that already give you some first information about your sample. You can take a look at the average diffractogram of your dataset and get a first image of your map.

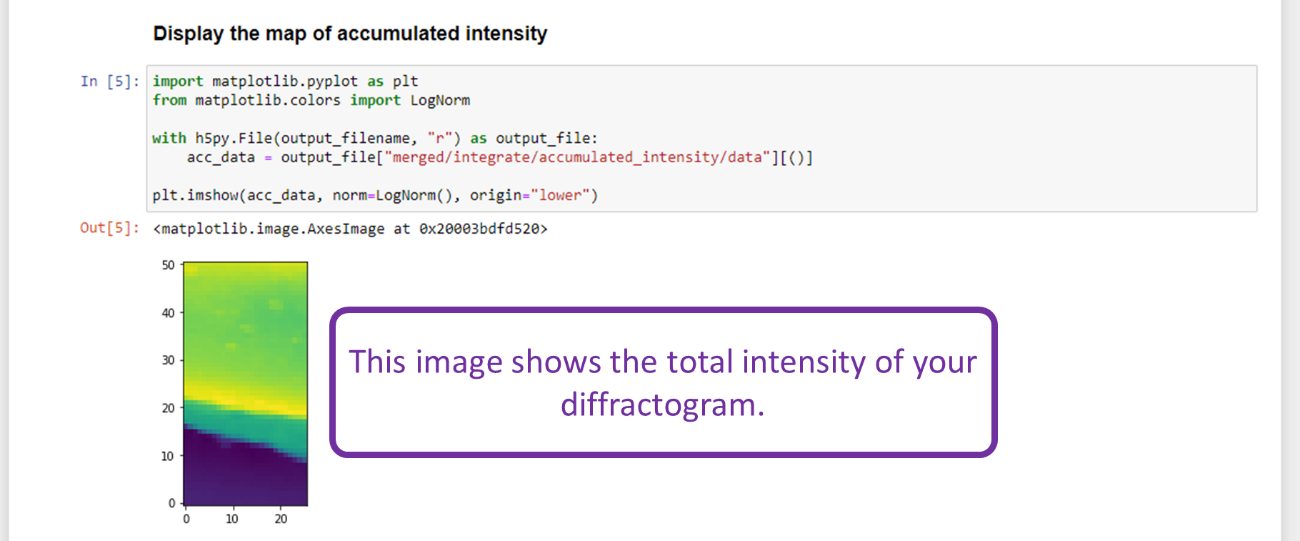
**Averaged diffractogram**

In order to get an average diffractogram of your dataset, click on the cell just below ‘**Display the integrated data with a reference**’ and run this cell. By default the reference silver behenate is selected (AgBh) and shown as red dashed lines over your diffractogram.



**Display your map**

In the next cell, you can take a first look at your map.



**Convert data format**

And lastly, you can convert the default file format of the integration (.h5) to the .edf format. The edf format can be loaded into the XRDUA software package for further processing of your data.



# Transfer and download your data

In the **Terminal window** run the following command to activate additional functions

>> source /data/id21/inhouse/id21rc/bootstraprc

Go to the directory that contains the integrated files. This is the directory that you specified as output\_file in the *run\_config.yml* file.

>> cd /data/visitor/hg172/id13/XXPROCESS\_jupyter/”your\_INITIALS”

e.g., >> cd / data/visitor/hg172/id13/XXPROCESS\_jupyter/FV

Optional: In case you processed multiple datasets you first have to make a single zip archive that contains these multiple files. The ‘ftpshare’ command in the next step can only transfer one file at a time. In order to zip multiple files and directories into one zip archive run the following command. This will make a zip archive of all the files in the current working directory. This may take some time depending on the size of your processed data.

>> zip -r “your\_data\_archive”.zip \*.\*

e.g., >> zip -r frederik\_data.zip \*.\*

If you do not want to add everything to the zip archive, you can specify individual folders and/or datasets by separating them with a space.

>> zip -r “your\_data\_archive”.zip “folder1” “dataset1”.h5 “dataset2”.h5

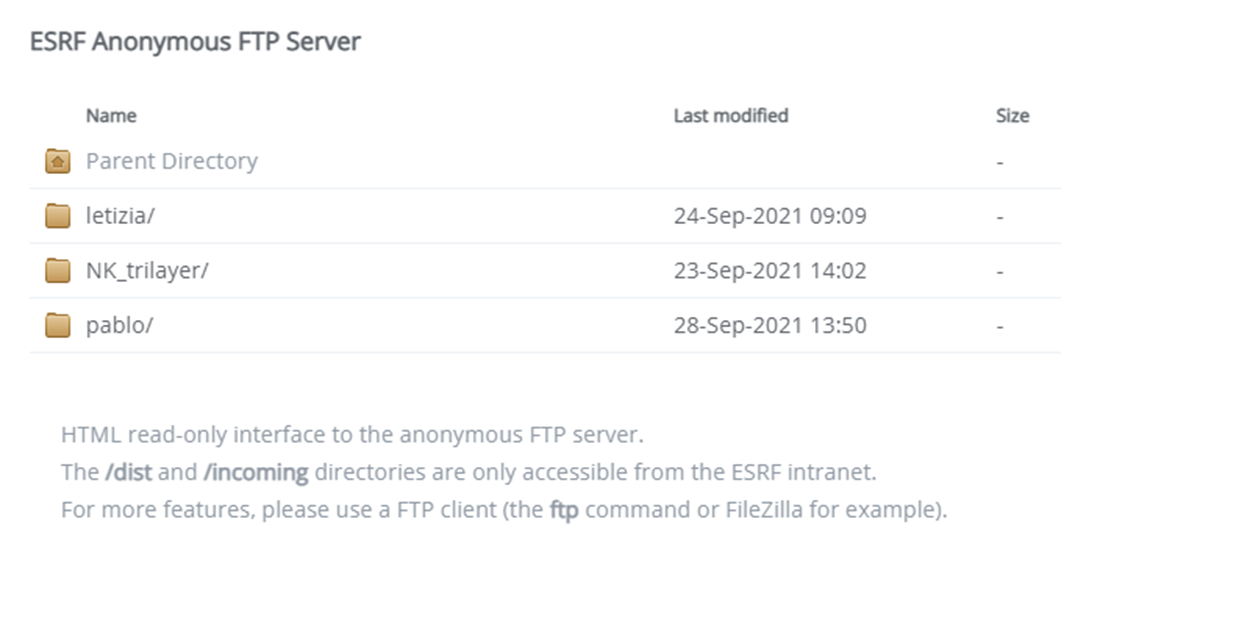
e.g., >> zip -r frederik\_data.zip folder1 dataset1.h5 dataset2.h5

You can then transfer your data to the ESRF ftp server. It is best to put the data in a folder with your name.

>> ftpshare “your\_data\_archive”.zip “your\_ftp\_folder” FV/.

e.g., >> ftpshare frederik\_data.zip FV/.

You can use an ftp client to download your data from the ESRF ftp server or you can use your favorite web browser to go to <http://ftp.esrf.fr/tmp/>. Your data is stored in the folder tmp/”your\_name”.



# Update Juno

To update your Jupyter notebooks with the last modifications implemented by Loïc, please go to your juno folder and run the following commands in the **Terminal window**:

Go to your Juno folder

>> cd juno

“Stash” any changes that you might have made to your juno folder, so that you can put them back after the update

>> git stash

Get the update

>> git pull

Restore your own changes

>> git stash pop

The Jupyter scripts in your personal Juno folder are now up to date.

You can check the version of your Jupyter scripts using the following command

>> git log