Jupyterhub

Jupyter Notebook is a web-based application for interactive work on Jupyter Notebook documents. These documents can combine source code, the source code's output, and additional text and images.

The system is accessible via a web browser. The Jupyter Notebook software connects a notebook document with an R or Python interpreter to allow interactive execution of the source code.

The software comes with preinstalled interpreters and a predefined set of libraries for Python 2.7, Python 3.6, and R 3.6 in a multiuser configuration called Jupyterhub. These environments are named "Anaconda-Python2.7", "Anaconda-Python3.6", and "R".

Users cannot modify this basic set of libraries. Instead, users can create their custom environments for Python 2.7 or Python 3.6, stored in the user's home folder, and add additional libraries therein. For R, users can extend the basic library set by installing additional libraries in a local package repository, also stored in the user's home folder.

Central instances of the Jupyter Notebook Server are available under https://jupyterhub.awi.de for all AWI users, and under https://jupyterhub.mosaic-data.org for MOSAiC. Additionally, personal instances of Jupyterhub are available for users or workgroups on request. They have a limited life span, after which they are deleted automatically. As described before, custom environments remain, as they are stored in the user's home folder and remain available in all instances of Jupyterhub at AWI.

The default user interface is the newly designed Jupyterlab UI. However, you can switch to the old interface by clicking "Help" in the menu bar and then "Launch Classic Notebook". The old documentation can be found here.

Getting started

The left sidebar contains tabs for a file browser, a list of running notebooks and terminals, a list of open tabs, etc. It can be collapsed and expanded by a click on the tabs icon.

Existing notebooks can be found by the "file browser" tab in the left side panel (first icon) and opened by a double click. A right-click on an item in the file browser opens a context menu for file download, copy, rename, etc.

The "Launcher" starts new notebook documents. It is accessible either by click on "File" -> "New Launcher" in the menu bar or by click on the plus-symbol in the file browser tab in the left side panel.

The Launcher shows an icon for a new notebook for each installed conda environment (see below). The complete name of an environment is shown when you place the mouse pointer over the icon for a second.

The official Jupyterlab manual can be found at https://jupyterlab.readthedocs.io/en/stable/user/interface.html

Where to find your data

The included file browser (first icon in the left icon bar) shows your personal home folder.

The central storage is available in folder /isibhv and includes projects (/isibhv/projects), netscratch (/isibhv/netscratch), platforms-data (/isibhv/platforms), etc.

You can directly use those paths within your scripts and notebooks, but the folders are not directly accessible within the file browser.

You can place a link into your home folder:

Open a "terminal" and enter the following command:

```
ln -s /isibhv/projects/myOwnProject ~/myProjectLink
```

"/isibhv/projects/myOwnProject" is the folder you want to link to and "myProjectLink" is the name under which the link will appear within your home folder (indicated by the "~" symbol).

How to close a notebook document

Whenever a kernel is attached to a notebook document, the notebook becomes a running program and thus occupies system memory.

However, such a running notebook/program does not automatically terminate when you log out from the Jupyterhub or close the notebook tab. Instead, you need to stop the notebook!

You can stop a running notebook by clicking "File" and "Close and shutdown notebook".

The tab "Running Terminals and Kernels" (second icon on the left icon bar) lists all running notebooks. They can also be terminated by a click on "SHUT DOWN".

Please stop all notebooks after you are done with your work!

Working with notebooks

A notebook document needs a connected kernel/environment to be executable. The active kernel is shown in the upper right corner. A click on that name opens a menu to change the kernel.

All input in a notebook document is organized in cells, of which different types exist:

Code cells contain (Python or R) source code, they can be executed interactively by the selected kernel. The output is shown below the cell.

Markdown cells contain formatted text. The formatted code replaces the markdown code when the cell is executed.

Raw cells are formatted like code cells but are not executed.

A cell can be executed by clicking the “Play” button, by the shortcut “CTRL” + “Enter”, or by click on an entry within the “Run” menu in the menu bar.

Each code cell gets a sequential number to indicate the order in which the cells have been executed.

The selected cell has a blue bar on the left side. In “editing” mode it also has a blue frame around it. The modes can be switched by the “Esc” and “Enter” key respectively.

Several shortcuts are predefined, e.g. in the command mode:

Up/down keys: scroll up and down the cells

- A: Create a new cell above the selected one
- B: Create a new cell below the selected one
- M: Treat the active cell as a markdown cell
- R: Treat the active call as a raw cell (like a code cell, but won’t be executed)
- Numbers 1-6: Format cell to heading
- Y: Treat the active cell as a code cell
- DD: Delete the active cell
- Z: Undo a cell delete
- Shift + M: Merge the active and the following cell

In edit mode:

CTRL + Enter: Execute the cell

CTRL + SHIFT + “-“ (minus): Split the cell at the cursor position

Preinstalled kernels are:

Python [conda env:Anaconda-Python2.7] : Python 2.7 and a set of Anaconda’s default libraries

Python [conda env:Anaconda-Python3.6] : Python 3.6 and a set of Anaconda’s default libraries

R : R 3.6 and default libraries

Conda environments

A conda environment combines an interpreter (e.g. for Python or R) and installed libraries under an explicit name. Such an environment can be available centrally on a server, which makes it available for all users of that system, or locally in the user’s home folder. Centrally stored environments are read-only. You can use them, but you cannot add or change libraries. If you need further libraries or specific library versions, you need to create a custom environment in your home folder.

Handling environments with the graphical packages manager

The graphical package manager is started by clicking “Settings” -> “Conda Packages Manager” in the menu bar.

Here you can create, update, delete, export, and import custom environments.

Please note: The centrally stored, shared environments are also listed, but cannot be modified!

Handling environments in the terminal

Conda paths need to be set once by the following command:

/opt/miniconda3/bin/conda init

You can change the active environment with the command „conda activate“. Python’s package manager pip always refers to the activated environment. Conda package manager refers to the activated environment, except you specify another one by the –n flag.

Create a new environment

```
conda env create -n envName ipykernel
conda activate -n envName
conda install library1 library2 (…)
```

Export an environment

```
conda activate myEnv
conda env export > myEnv.yaml
```

Import an environment

```
conda env create –f myEnv.yaml
```

Install libraries

```
conda install library1 –n myEnv
```

Or

```
conda activate myEnv
conda install library1
```

If you use Pip, please activate the environment first:

```
conda activate myEnv
pip install library1
```