Jupyter in the cluster

"Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages." [1)]

"The Jupyter notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results." [2)]

"JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data." [3)]

Detailed information about various components of Project Jupyter can be accessed here.

In this section we explain Jupyter usage in the cluster.

How to start Jupyter in the cluster

On the cluster, Jupyter Notebook and JupyterLab are availabe from within the cluster web portal as OOD interactive application, which starts Jupyter sessions on a compute node by means of a SLURM job. To access the Jupyter app, login onto the portal and in the *Interavitve Apps* menu select the *Jupyter* server. This will open a web dialog page which allows you to start a Jupyter session in a SLURM job using the standard cluster-wide Conda and Module environments you can select from the drop-down *Standard environment* list. On this page, you can also specify the job parameters such as time limit, number of CPU cores, amount of allocated memory, number of GPU units and the cluster partition you want to start your Jupyter session on. Note that OOD Jupyter applications currently launch Jupyter sessions on a **single** compute node.

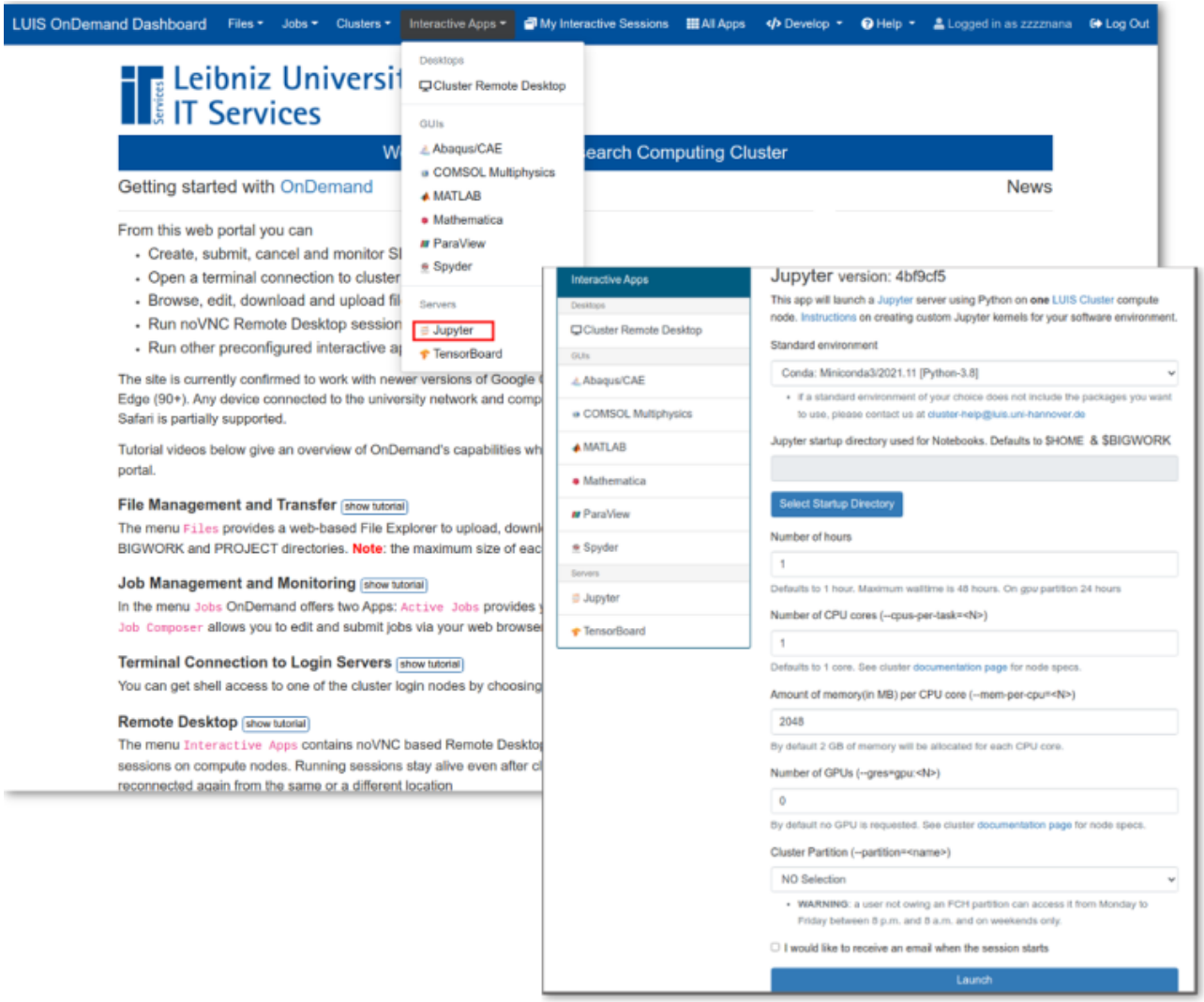

Fig. 1: Jupyter on the cluster

By selecting the standard Jupyter environment in addition to the IPython kernel providing many popular Python packages for Data Science, Machine Learning and Scientific Computing (Tensorflow, PyTorch, Pandas, NumPy, SciPy, Matplotlib, etc) you can also run the Jupyter kernels for R and MATLAB. If a standard environment of your choice does not contain packages you want to use or you would like more kernels to be included, please get in touch with the cluster group at cluster-help@luis.uni-hannover.de. You can also make the packages you need available in your JupyterLab session via a custom Jupyter kernel, see next subsection below. Check out the list of available Jupyter kernels.

Creating custom Jupyter kernels

If the standard JupyertLab environments do not contain packages you require for your project, you need a different python version or other programming languages you may provide them by creating JupyterLab kernels. In this section we explain how to install a custom python kernel for your Python virtualenv or Conda environments and how to switch between environments without resubmitting JupyterLab session SLURM job.

The installation of a new kernel is done in two steps:

1. Create an environment (Conda or Python virtualenv) and install required Python libraries and

packages including `ipykernel`
2. Install a custom Python kernel for JupyterLab. The kernel will be located in a sub-directory of `$HOME/.local/share/jupyter/kernels`

Please note: since the installation of Python packages requires an access to internet, the configuration of a kernel must be done on a login node.

Conda environment

To make your conda environment available in a JupyterLab session, follow the instructions below.

For details about creating conda environments, see the conda usage in the cluster.

1. Create a conda environment (skip this step if the environment already exists)

```
[username@login01 ~]$ module load Miniforge3
[username@login01 ~]$ conda create -n my_env
```

2. Activate the environment, install the `ipykernel` library and create a Python kernel for JupyterLab:

```
[username@login01 ~]$ conda activate my_env
(my_env)[username@login01 ~]$ conda install ipykernel
(my_env)[username@login01 ~]$ make-ipykernel --name my_conda_env --display-
name "My Software Env"
```

The configuration file, `kernel.json`, of the `my_conda_env` kernel will be created in the directory `$HOME/.local/share/jupyter/kernels/my_conda_env`.

Note: The `--name` option must uniquely identify the kernel.

3. Install additional packages you need:

```
(my_env)[username@login01 ~]$ conda install numpy matplotlib sympy
```

Python virtualenv

1. Select your preferred Python version by loading the appropriate module:

```
[username@login01 ~]$ module load GCC/10.2.0 Python/3.8.6
```

2. Create a Python virtual environment named `myvenv` at the specified location (skip this step if the environment already exists):

```
[username@login01 ~]$ virtualenv $HOME/myvenv
```

3. Activate the environment, install the `ipykernel` library and create a Python kernel for JupyterLab:

```
[username@login01 ~]$ source $HOME/myvenv/bin/activate
(myvenv)[username@login01 ~]$ pip install ipykernel
(myvenv)[username@login01 ~]$ make-ipykernel --name my_venv --display-name
"My Software Env"
```

The configuration file, `kernel.json`, of the `my_venv` kernel will be created in the directory `$HOME/.local/share/jupyter/kernels/my_venv`.

Note: The `--name` option must uniquely identify the kernel.

4. Install other Python packages

```
(myvenv)[username@login01 ~]$ pip install numpy sympy
```

A few seconds after creating the kernel, your software environment will appear in your active JupyterLab session listed as `My Software Env` on the launcher page. To make the kernel immediately available in JupyterLab, just refresh the browser window.

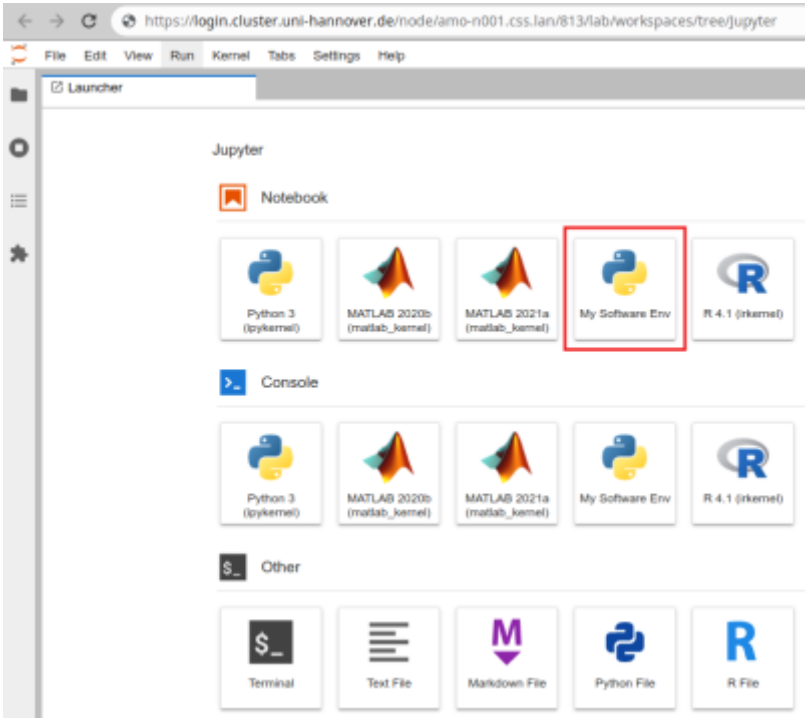

Fig. 2: Python Jupyter kernel

You can switch kernels using the menu *Kernel* → *Change Kernel…* The kernel `Python3 (ipykernel)` corresponds to the standard cluster-wide environment you selected when submitting your JupyterLab session.

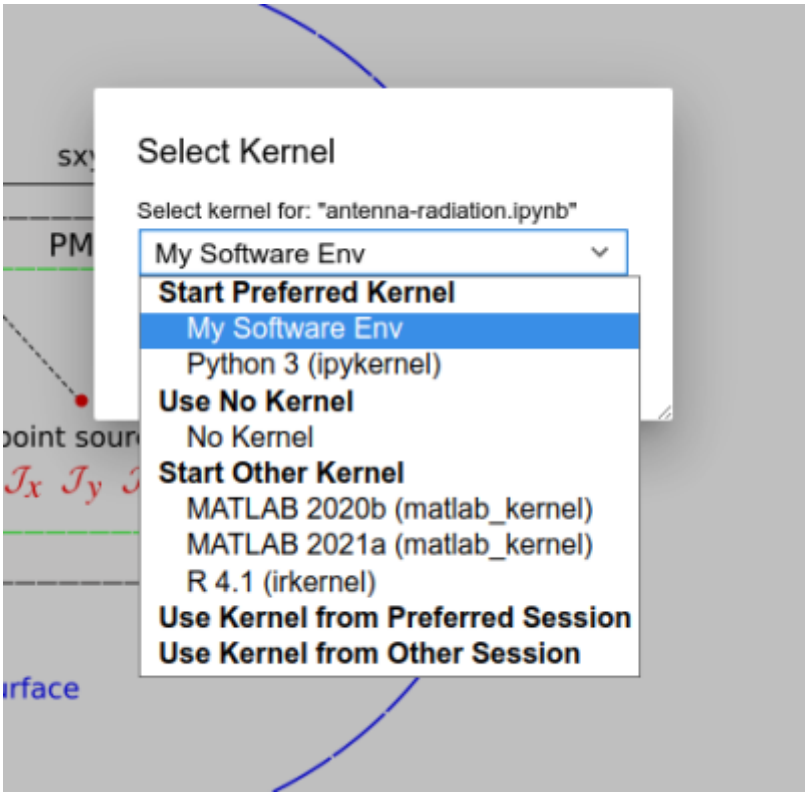

Fig. 3: Change Kernel

Classic Jupyter notebook: Your kernel is listed in the menu *New*:

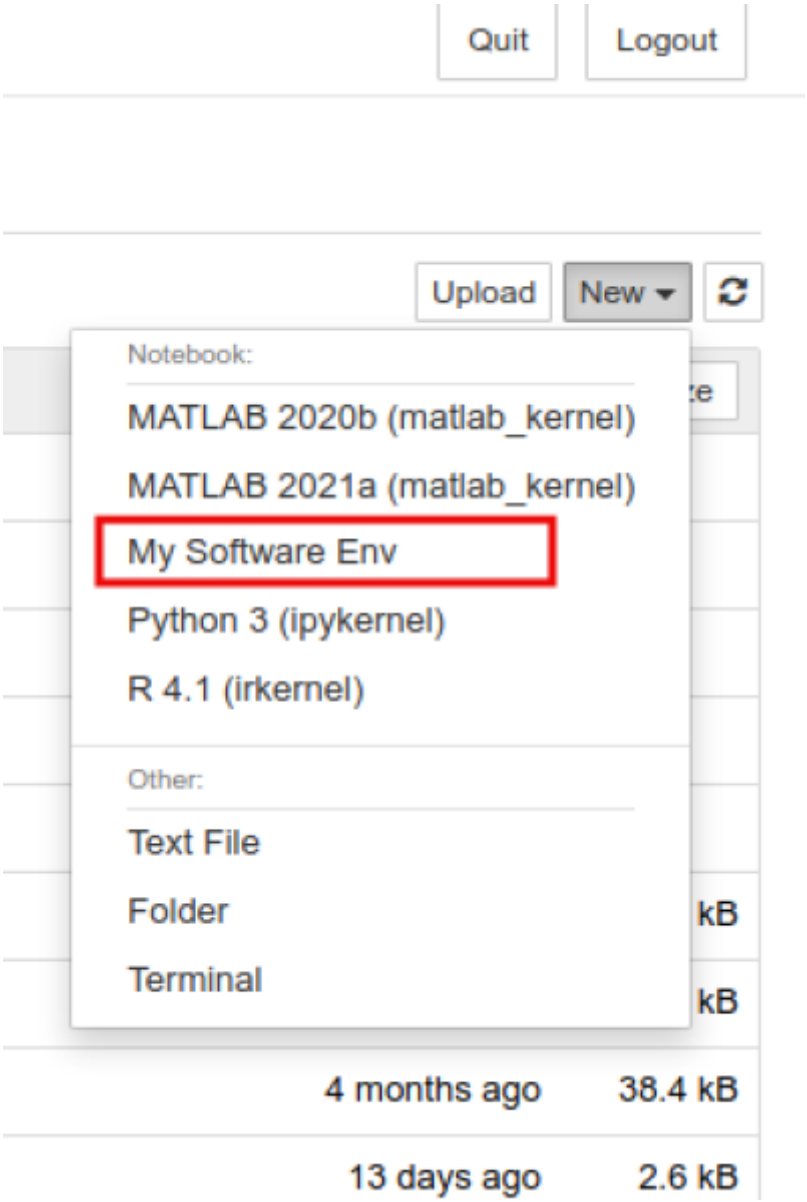

Fig. 4: Python Jupyter kernel

Classic Jupyter notebook: To switch kernels visit the menu *Kernel*:

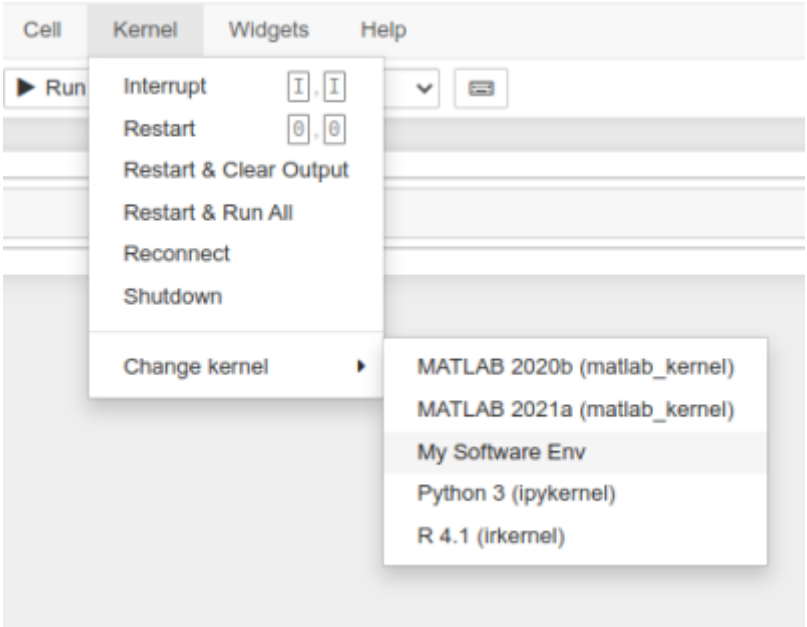

Fig. 5: Switch kernels

1)

Project Jupyter webpage

2)

Jupyter Notebook documentation

3)

JupyterLab documentation