



# ScaDS.AI

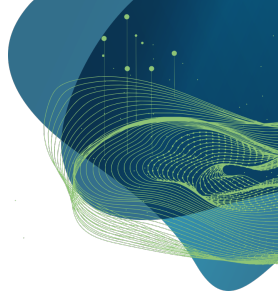
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND  
ARTIFICIAL INTELLIGENCE

## JupyterHub Training

Mariela Sanchez, Adnan Haidar, Elias Werner

Dresden, 27th of October 2022



GEFÖRDERT VOM



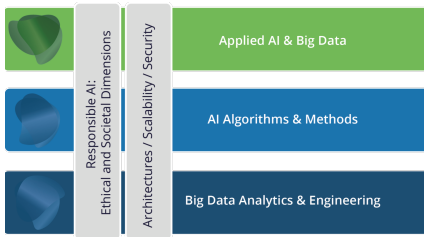
Bundesministerium  
für Bildung  
und Forschung

SACHSEN



Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

# Presenters



**Mariela Sanchez,**  
Research Associate, ZIH  
*Data Science for synthetic data evaluation*



**Adnan Haidar,**  
Research Associate, ScaDS.AI  
*Data Science, Robotics Tactile Sense using Artificial Intelligence*



**Elias Werner,**  
Research Associate, ScaDS.AI  
*performance analysis of data science workflows*

<https://scads.ai/>

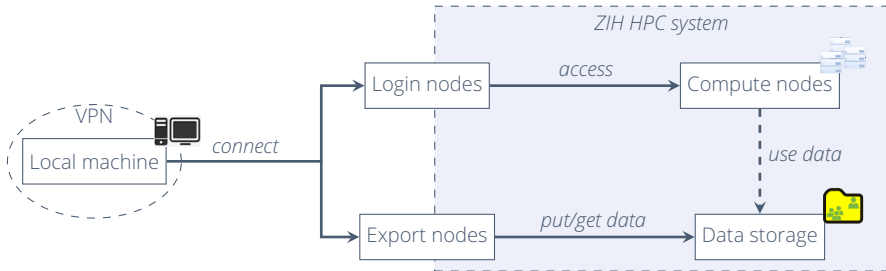
# 1 Introduction – Agenda

- 1 Introduction
- 2 Jupyter overview
- 3 Access and Basics of the ZIH HPC system
- 4 Data Management via Jupyter
- 5 Collaborative Research and Teaching with Jupyter on HPC
- 6 Hands On session: Prepare a Jupyter "environment" for collaborative working
- 7 Conclusion, Q&A and Supplementary

# 1 Recap: What is an HPC machine?

- Cluster of many single nodes (single "computers") for pooling their resources (computation and memory)
- Job scheduler *Slurm* manages resource/hardware allocation
- Software environment can be set up easily via a module system
- Collaborative working spirit, shared machine
- HPC project required for interaction

## Technical workflow:



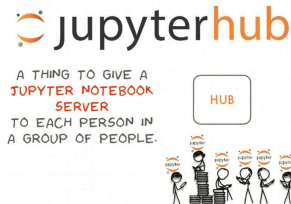
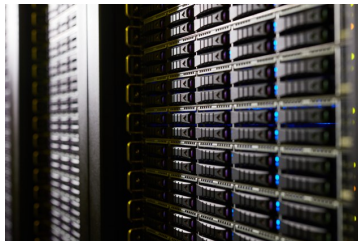
# 1 HPC system - Taurus

- **Name:** HPC-Cluster Taurus
- **Model:** The second high-performance computing/memory complex (HRSK-II)
- **Facility:** Center for Information Services and High Performance Computing (ZIH)
- **Details:**
  - ▶ More than 60,000 cores,
  - ▶ 720 GPUs,
  - ▶ Flexible storage hierarchy with about 16 PB total capacity,
  - ▶ Linux, shared login nodes, filesystems, batchsystem Slurm,
  - ▶ Perfect platform for highly scalable, data-intensive and compute-intensive applications.



# 1 JupyterHub - a Good Mean for Interacting with HPC

- Jupyter useful for data analysis
- Jupyter as an interface to an HPC system for many users
- Users access to complex computing infrastructures from the JupyterHub
- Jupyter provides remote access to Jupyter Notebooks



Taken from: [geohackweek.github.io/Introductory/05-jupyter-tutorial/](https://geohackweek.github.io/Introductory/05-jupyter-tutorial/)

# 1 Polls

Asking the audience:

- 1) What is your main discipline of work?
  - A. Humanities and Social Sciences
  - B. Natural Sciences
  - C. Life Sciences
  - D. Engineering Sciences
- 2) Which programming language you use mostly in Jupyter?
  - A. Python
  - B. R
  - C. Julia
  - D. Others
- 3) What is the OS you use to initialize and use Jupyter?
- 4) What are your main reasons for joining this training event?

# 1 Polls

Asking the audience:

- 1) What is your main discipline of work?
  - A. Humanities and Social Sciences
  - B. Natural Sciences
  - C. Life Sciences
  - D. Engineering Sciences
  
- 2) Which programming language you use mostly in Jupyter?
  - A. Python
  - B. R
  - C. Julia
  - D. Others
  
- 3) What is the OS you use to initialize and use Jupyter?
  
- 4) What are your main reasons for joining this training event?



# 1 Polls

Asking the audience:

- 1) What is your main discipline of work?
  - A. Humanities and Social Sciences
  - B. Natural Sciences
  - C. Life Sciences
  - D. Engineering Sciences
  
- 2) Which programming language you use mostly in Jupyter?
  - A. Python
  - B. R
  - C. Julia
  - D. Others
  
- 3) What is the OS you use to initialize and use Jupyter?
- 4) What are your main reasons for joining this training event?

# 1 Polls

Asking the audience:

- 1) What is your main discipline of work?
  - A. Humanities and Social Sciences
  - B. Natural Sciences
  - C. Life Sciences
  - D. Engineering Sciences
  
- 2) Which programming language you use mostly in Jupyter?
  - A. Python
  - B. R
  - C. Julia
  - D. Others
  
- 3) What is the OS you use to initialize and use Jupyter?
- 4) What are your main reasons for joining this training event?

# 1 Structure of the Presentation

- Jupyter
  - ▶ Features
  - ▶ Usability
  - ▶ Functionalities
- HPC basics
  - ▶ Access
  - ▶ Data management
  - ▶ Software management
- Collaborative working with HPC and Jupyter notebook
  - ▶ Shared system principles
  - ▶ Workflow
  - ▶ Useful features
- Hands-on session

# 1 Intention of the Presentation

- Making things understandable esp. for **frequent JH users** including teachers and researchers.
- Complete **workflow examples** are shown, here: based on **Python code** (can be used as blueprint for other software/tools).
- From the **user's perspective**: What are the most important things to work with JH on HPC?

## Note

We do **not** want to show everything that is possible. We want to show, what is needed to get started

## Note

We do **not** want to show *the* complete architecture and advanced properties of Jupyter. We want you to comprehend how Jupyter works and HPC principles behind JH tool with concrete examples

## Hint

Please ask and interrupt our presentation immediately if something is not clear! Use your mic or the chat.

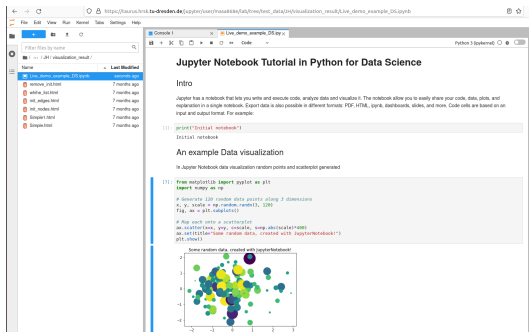
# 2 Jupyter overview – Agenda

- 1 Introduction
- 2 Jupyter overview
- 3 Access and Basics of the ZIH HPC system
- 4 Data Management via Jupyter
- 5 Collaborative Research and Teaching with Jupyter on HPC
- 6 Hands On session: Prepare a Jupyter "environment" for collaborative working
- 7 Conclusion, Q&A and Supplementary

# 2 Jupyter Overview - Why Jupyter is great

Jupyter Notebook documents:

- Dynamic notebook documents include:
  - ▶ Plots
  - ▶ Images
  - ▶ Video
  - ▶ Equations
  - ▶ Narrative text
  - ...
- Notebook web application enables to:
  - ▶ Run and edit code
  - ▶ Build hierarchical documents
  - ▶ Computations results as PDF, HTML, etc



Jupyter Notebook and Web Interface



# 2 Jupyter Overview - Why Jupyter is great

The image shows a screenshot of a Jupyter Notebook running in a browser. The browser's address bar shows the URL `localhost:8888/notebooks/Live_demo_example_DS.ipynb`. The notebook title is "Jupyter Live\_demo\_example\_DS". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for running, saving, and other actions. The notebook content is divided into sections: "Intro" and "An example Data visualization". The "Intro" section contains a code cell with `print("Initial notebook")` and its output, "Initial notebook". The "An example Data visualization" section contains a code cell with Python code for generating and plotting random data. Below the code is a scatter plot titled "Some random data, created with Jupyter Notebook" showing a distribution of points in a 2D space.

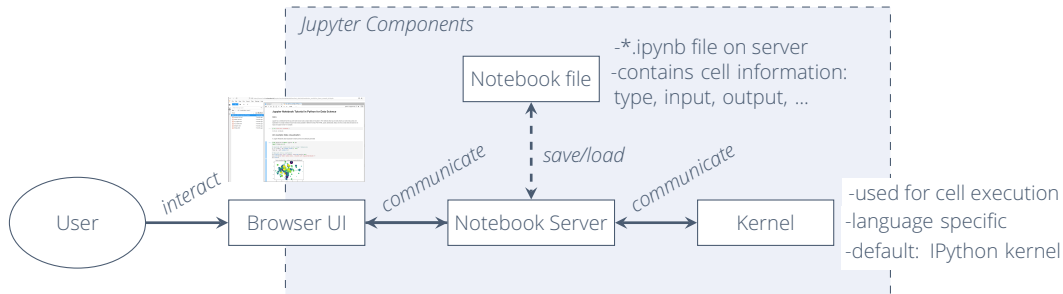
*Runs in browser*

*Texts, comments, etc*

*Lines of code (Python, Julia, R, Matlab, ...)*

*Output: values, images, plots,...*

## 2 Jupyter Overview - Why Jupyter is great



structure according to: [jupyter.readthedocs.io/.../content-architecture](https://jupyter.readthedocs.io/.../content-architecture)



## 2 Jupyter Overview - Why Jupyter is great

Jupyter Kernels:

- IPython: basic Python kernel with features for interactive computing
- IRKernel: basic R kernel
- IJulia: basic Julia kernel
- MATLAB Kernel: Basic kernel for Matlab language
- ITorch: IPython kernel with visualizations of images
- IPykernel: another Python kernel

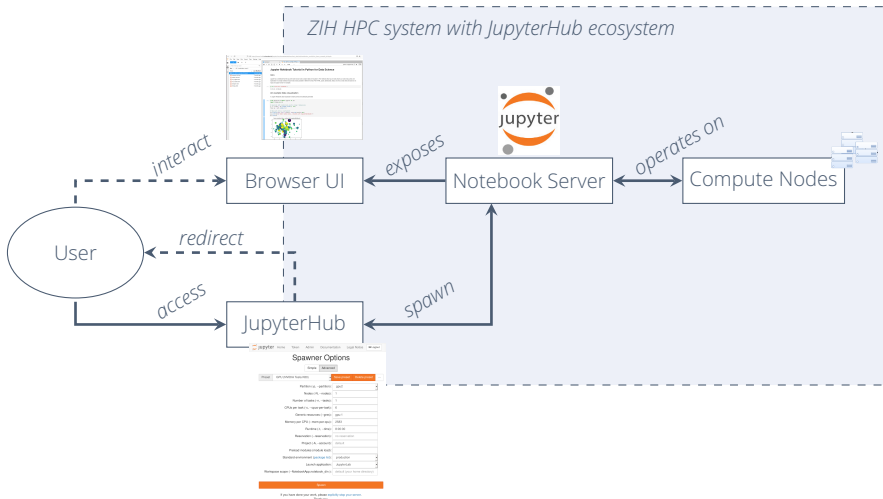
### Note

Jupyter supports over 100 programming languages. For more details: <https://github.com/jupyter/jupyter/wiki/Jupyter-kernels>

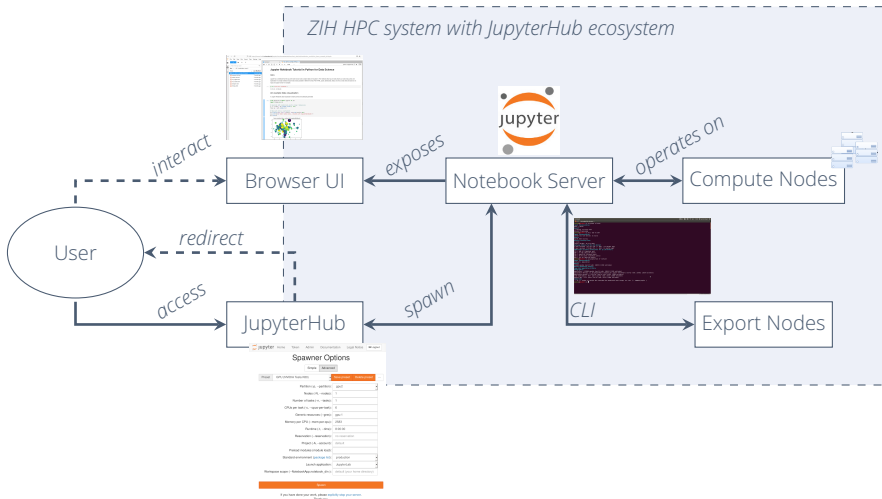




## 2 Access the ZIH System via JupyterHub



## 2 Access the ZIH System via JupyterHub



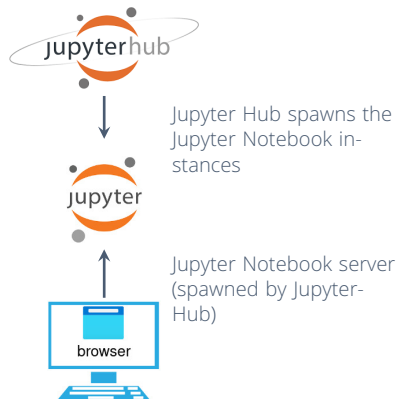
## 2 Access the ZIH System via JupyterHub

One single node of Jupyter:

- Each user a whole new instance of Jupyter software
- The instance has all necessary features to interact with the HPC:
  - ▶ Web browser
  - ▶ Jupyter Notebook
  - ▶ Kernel
  - ▶ others: Jupyter lab, etcetera

### Note

Jupyter Hub has many interesting properties to explore for intermediate and basic users.

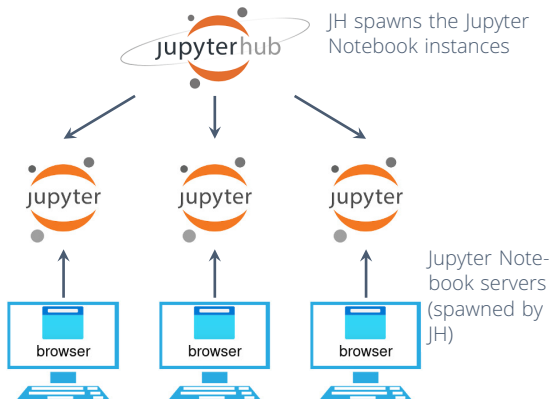


Adapted from: [opencredo.com/blogs/writing-a-custom-jupyterhub-spawner/](https://opencredo.com/blogs/writing-a-custom-jupyterhub-spawner/)

## 2 Access the ZIH System via JupyterHub

The ecosystem of JH under HPC:

- A multi-user Hub with multiple instances of single-users
- Set up in HPC system
- Built for multiple user log in
- The instances are single Jupyter structures
- A interactive interaction with HPC in every instance



Adapted from:  
[opencredo.com/blogs/writing-a-custom-jupyterhub-spawner/](https://opencredo.com/blogs/writing-a-custom-jupyterhub-spawner/)

## 2 Jupyter Scenario: Research

Paulina is a researcher,  
she has some of her  
experiments ready

1



She wants to share  
them with other  
researchers

Adapted from:  
Beg, Marijan, et al.  
"Using Jupyter for  
reproducible scientific  
workflows."

## 2 Jupyter Scenario: Research

1 Paulina is a researcher, she has some of her experiments ready



She wants to share them with other researchers



2

She describes her experiments as a Jupyter Notebook



Her Jupyter Notebook is a mix of code, visualizations, annotations, among others

Adapted from:  
Beg, Marijan, et al.  
"Using Jupyter for  
reproducible scientific  
workflows."



## 2 Jupyter Scenario: Research

1 Paulina is a researcher, she has some of her experiments ready



She wants to share them with other researchers

She sends the link to the other researchers

`https://taurus.hrsk.tu-dresden.de/jupyter/hub/user-redirect/notebooks/demo.py?base URL and page_user-redirect path to file`

3



She creates a sharable link, pointing to her experiments notebook in her Git repository

2 She describes her experiments as a Jupyter Notebook



Her Jupyter Notebook is a mix of code, visualizations, annotations, among others

Adapted from:  
Beg, Marijan, et al.  
"Using Jupyter for  
reproducible scientific  
workflows."

## 2 Jupyter Scenario: Research

1 Paulina is a researcher, she has some of her experiments ready



She wants to share them with other researchers



She sends the link to the other researchers

`https://taurus.hrsk.tu-dresden.de/jupyter/hub/user-redirect/notebooks/demo.ipynb`  
base URL and page `_user-redirect` path to file

3



She creates a sharable link, pointing to her experiments notebook in her Git repository

2 She describes her experiments as a Jupyter Notebook



Her Jupyter Notebook is a mix of code, visualizations, annotations, among others

4



Now every researcher can run and reproduce her computations

Adapted from:  
Beg, Marijan, et al.  
"Using Jupyter for  
reproducible scientific  
workflows."

## 2 Jupyter Scenario: Teaching

John is a University Professor

1



He wants to provide the lecture in Jupyter notebooks with a consistent software environment for all the students

Adapted from: Beg, Marijan, et al. "Using Jupyter for reproducible scientific workflows."

## 2 Jupyter Scenario: Teaching

1

John is a University Professor



He wants to provide the lecture in Jupyter notebooks with a consistent software environment for all the students

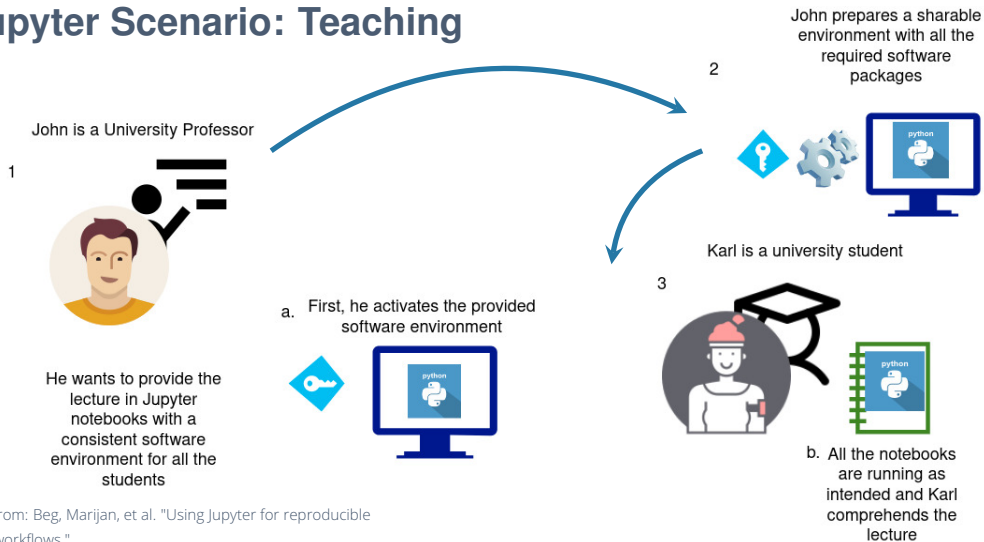
2

John prepares a sharable environment with all the required software packages



Adapted from: Beg, Marijan, et al. "Using Jupyter for reproducible scientific workflows."

## 2 Jupyter Scenario: Teaching



Adapted from: Beg, Marijan, et al. "Using Jupyter for reproducible scientific workflows."

# 3 Access and Basics of the ZIH HPC system – Agenda

- 1 Introduction
- 2 Jupyter overview
- 3 Access and Basics of the ZIH HPC system
  - Organizational, Access and Permissions, Software management
- 4 Data Management via Jupyter
- 5 Collaborative Research and Teaching with Jupyter on HPC
- 6 Hands On session: Prepare a Jupyter "environment" for collaborative working
- 7 Conclusion, Q&A and Supplementary

# 3 JupyterHub Project on the ZIH System

Working on the JupyterHub system requires an HPC project. An HPC project on the ZIH system includes:

- project directory and login with existing or without existing ZIH Login
- project members (at least project leader and project administrator)
- resource quotas for compute time (CPU/GPU hours) and storage

## Electronic HPC Project Application Form for ZIH







CONTENT ▾

### Application lists

Two application lists can be found on this page:

1. **Active applications:** Contains applications for the current call, which are not finalized yet.
2. **Finalized applications:** Shows all finalized applications for the current call.

Next to each application, buttons are displayed for the actions allowed for that record. During the application process data is saved automatically by changing a panel or clicking on the "SAVE"-Button. Please choose the application you would like to edit or create a new application by clicking on the new button. The following actions are available:

-  **copy:** Copy project data of an application and create a new one with that data.
-  **delete:** Delete the application record.
-  **edit:** Load an active application from a previous session and edit the data.
-  **files:** Show a list of files uploaded for that application.
-  **New Project Application:** Create a new application.
-  **print:** Display a summary of a finalized application as PDF for printing.

---

E-mail address of principal investigator (PI):      E-mail address of person to contact (PC):  
matthias.kraeuslein@tu-dresden.de      - none -

change PI/PC (re-login required)

# 3 JupyterHub Project on the ZIH System

There are different possibilities to work with HPC:

- create a new project
  - ▶ fill the application form: <https://hpcprojekte.zih.tu-dresden.de/>
  - ▶ find additional information in the wiki:  
[https://doc.zih.../application/project\\_request\\_form/](https://doc.zih.../application/project_request_form/)
- Join an existing project, new researchers in an existing project, teaching purposes

## Hint

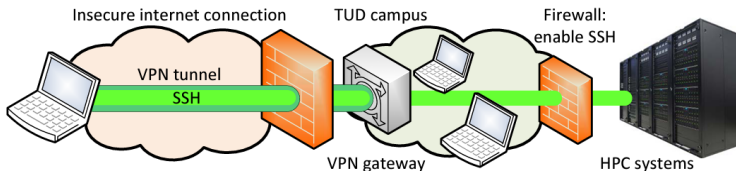
Kindly mention the HPC resource usage in the acknowledgment section of all publications  
<https://doc.zih.tu-dresden.de/application/acknowledgement/>



# 3 VPN to Access Services

To access ZIH services:

- from within the TU Dresden campus via secure shell (ssh).
- install VPN tool at your local machine.
  - ▶ OPENVPN client (recommended variant).
  - ▶ Cisco AnyConnect Secure Mobility Client

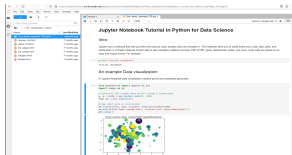


# 3 Accessing the JupyterHub

Different ways to access the ZIH system:

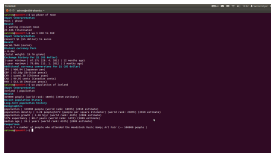
## JupyterHub

- browser based approach
- most easiest way for beginners



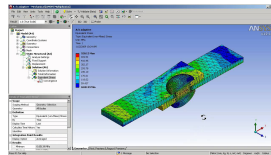
## SSH connection

- "classical" approach
- command line interface (CLI) knowledge is necessary



## desktop visualization

- esp. in case of using GUI-based software
- e.g. Ansys, Vampir,...



## Hint

For getting in touch with the system and also development purposes the approach with JupyterHub is recommended.

# 3 Accessing the JupyterHub

To access JupyterHub to use:

1. use the browser link (<https://taurus.hrsk.tu-dresden.de/jupyter>)
2. Start session by clicking Start my server button

Allocation of resources via the GUI Spawner Option

- Simple Option
- Advanced Option

Utilization Bar

- Number of GPUs/CPU's available per node

More information in the wiki:  
<https://doc.zih.../access/jupyterhub>

### Spawner Options

**Very important!**  
If you have finished your work please explicitly stop your server.

**Current Utilization**

**Utilization as of 9/19/2022, 12:12:59 PM**

- 544 cores available of 30 384 cores on haswell64
- 135 cores available of 24 320 cores on romeo

**Simple Option** → Simple Advanced ← **Advanced Option**

**CPU/GPU architecture**

| Architecture                      |                                 |
|-----------------------------------|---------------------------------|
| Intel (x86_64)                    | IBM Power (ppc64le)             |
| Intel Haswell<br>NVIDIA Tesla K80 | IBM POWER9<br>NVIDIA Tesla V100 |

**Number of CPU's**

| Minimum                        | Recommended            | Maximum                |
|--------------------------------|------------------------|------------------------|
| single core<br>(single thread) | ? cores<br>(? threads) | ? cores<br>(? threads) |

**Number of GPU's**

GPU's: 0 1 2 3 4 5 6

Spawn

# 3 Simple Option Resources

Simple form offers most important settings to start quickly:

- Intel Hashwell or IBM Power9
- Numbers of CPUs cores and threads
- GPUs

## Spawner Options

**Very important!**  
If you have finished your work please explicitly stop your server.

### Current Utilization

Utilization as of 8/8/2022, 9:59:34 AM

4 673 cores available of 31 248 cores on haswell64

1 532 cores available of 24 320 cores on romeo

The screenshot shows the 'Spawner Options' interface. At the top, there is a warning box: 'Very important! If you have finished your work please explicitly stop your server.' Below this, the 'Current Utilization' section shows 'Utilization as of 8/8/2022, 9:59:34 AM' with two bars: '4 673 cores available of 31 248 cores on haswell64' and '1 532 cores available of 24 320 cores on romeo'. The main configuration area has two tabs: 'Simple' (selected) and 'Advanced'. Under 'Architecture', there are two columns: 'Intel (x86\_64)' with 'Intel Haswell' and 'NVIDIA Tesla K80', and 'IBM Power (ppc64le)' with 'IBM POWER9' and 'NVIDIA Tesla V100'. Below this is a 'CPUs' section with a table:

| Minimum                     | Recommended         | Maximum               |
|-----------------------------|---------------------|-----------------------|
| single core (single thread) | 4 cores (4 threads) | 24 cores (24 threads) |

Below the CPUs section is a 'GPUs' section with a row of buttons from 0 to 6. At the bottom is a large orange 'Spawn' button.

|                        |                                    |   |
|------------------------|------------------------------------|---|
| <b>Architecture</b>    | IBM Power9                         | Intel Haswell                           |
| <b>CPUs per node</b>   | 2 x IBM Power9 (22 cores@2.80 GHz) | 2x Intel Xeon 2680v3 (12 cores@2.50GHz) |
| <b>Memory per node</b> | 256GB RAM DDR4 2666MHz             | 64GB RAM 128 GB SSD                     |
| <b>GPUs per node</b>   | 6x NVIDIA VOLTA V100               | 4x NVIDIA Tesla K80                     |

# 3 Accessing the JupyterHub Advanced Option

Accessing the JupyterHub Advanced Option:

- Preset Option
  - ▶ Save own configurations as additional presets
  - ▶ Those are saved in browser or lost
  - ▶ Use import/export features to save as text
  - ▶ You can delete preset

The screenshot displays the JupyterHub Advanced Option interface. At the top, a 'Current Utilization' bar shows 320 cores on romeo, 256 gpus on alpha, and 896 cores on julia. Below this, a list of 'Example presets' is shown, with 'GPU (NVIDIA Ampere A100)' selected. The configuration form for this preset is visible, showing fields for Partition, Nodelist, Nodes, Number of tasks, CPUs per task, Generic resources, Memory per CPU, Runtime, Reservation, Project, Preload modules, Standard environment, Launch application, and Workspace scope. A dropdown menu is open for the Partition field, showing options: default, haswell, interactive, gpu2, hpdf, ml, dcv (selected), romeo, alpha, and julia. A red arrow points from the 'GPU (NVIDIA Ampere A100)' preset in the list to the 'Partition' dropdown menu. The 'Spawn' button is at the bottom.

# 3 JupyterHub Advanced Option

Customise available resources via advanced form:

- Every parameters of advanced form can be set.
- Partitions, Nodes, Number of tasks.
- CPU, GPUs, Memory and Run-Time.
- Project, Modules.
- Standard environment, Workspace.

## Hint

The resource parameters refer to the Slurm parameters of the `srun`, `salloc` and `sbatch` commands when using an ssh connection to the HPC system.

|  |                                   |
|--|-----------------------------------|
| Partition (-p, --partition):                           | alpha-interactive                 |
| Nodelist (-w, --nodelist):                             | no specific nodes                 |
| Nodes (-N, --nodes):                                   | 1                                 |
| Number of tasks (-n, --tasks):                         | 1                                 |
| CPUs per task (-c, --cpus-per-task):                   | 6                                 |
| Generic resources (--gres):                            | gpu:1                             |
| Memory per CPU (--mem-per-cpu):                        | 10312                             |
| Runtime (-t, --time):                                  | 8:00:00                           |
| Reservation (--reservation):                           | no reservation                    |
| Project (-A, --account):                               | default                           |
| Preload modules (module load):                         | none                              |
| Standard environment ( <a href="#">package list</a> ): | hiera_gcccore-10.2.0_python-3.8.6 |
| Launch application:                                    | JupyterLab                        |
| Workspace scope (--NotebookApp.notebook_dir=):         | default (your home directory)     |

# 3 Software Management by Loading Modules

A module system:

- is a user interface that provides utilities for the dynamic modification of a user's environment (e.g. \$PATH variable)
- enables to smoothly switch between different versions of installed software packages and libraries
- in Jupyter, modules are available via `Preload modules` parameter

## Hint

More information about module management: <https://doc.zih.../software/modules/>

# 3 Software Management - Preload Modules

- The `Preload modules` parameter for loading modules when spawning the JupyterHub server
- Preloaded modules will be available during the whole session
- After spawning, loading modules in the Jupyter terminal is not possible Only preloaded modules will be available in the Jupyter notebook files.

The screenshot shows the configuration interface for spawning a JupyterLab environment. The 'Advanced' tab is selected. The 'Preload modules (module load):' field is highlighted with a red circle and contains the following text: `PyTorch/1.6.0-fosscuda-2019b-Python-3.7.4` and `TensorFlow/2.3.1-fosscuda-2019b-Python-3.7.4`. Other configuration options include Partition, Nodelist, Nodes, Number of tasks, CPUs per task, Generic resources, Memory per CPU, Runtime, Reservation, Project, Standard environment (package list), Launch application, and Workspace scope.



# 3 Module Commands - Example

A **module** usage is quite straight forward and the following table lists the basic commands.

## ☰ Example

```
1 # Show all module options
2 marie@local$ module help
3
4 # List active modules in the user environment
5 marie@local$ module list
6
7 # Remove modules from the user environment
8 marie@local$ module purge
9
10 # Load module "modname" in the user environment
11 marie@local$ module load <modname>
12
13 # Remove module "modname" from the user environment
14 marie@local$ module unload <modname>
```

### 3 Standard Environments in JupyterHub

Standard environment through advanced option open a list with all included python-packages:

- To access the list of packages just select the "package list" link in the spawner
- Closely related to module environments

|  |                      |
|--|----------------------|
| Preload modules (module load):                         | <input type="text"/> |
| Standard environment ( <a href="#">package list</a> ): | production ▼         |
| Launch application:                                    | JupyterLab ▼         |

#### Hint

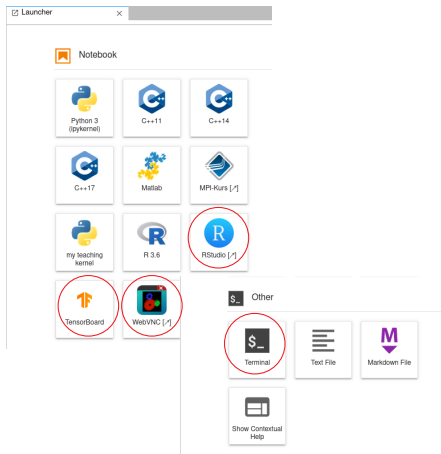
This list shows all packages of the currently selected conda environment. This depends on your settings for partition (CPU architecture) and standard environment. The default Python kernel uses conda environments.

List of available standard environments, namely:

<https://doc.zih.../access/jupyterhub/standard-environments>

# 3 Terminal, Tensorboard, WebVNC, RStudio

- JupyterHub launcher supports additional tools:
  - ▶ **Terminal:** access the compute node via CLI
  - ▶ **WebVNC:** spawn a graphical user interface on the compute node
  - ▶ **Tensorboard:** access the Tensorboard functionality
  - ▶ **RStudio:** connect to RStudio suite on the compute node
- Tools and features are updated regularly depending on the needs of the users
- Availability of features depending on the *standard environment*



# 3 CLI in JupyterHub



The terminal in the JupyterHub enables to:

- interact with ZIH system via CLI
- manage software and data with CLI commands

```
LSTM_PCA_FixedForceLevel_I X Terminal 2 X
Module Python/3.8.6, GCCcore/10.2.0 and 10 dependencies unloa
(python-env-python3.8.6-20210715-1421) ws_find
Usage: ws_find [options] [workspace]

(python-env-python3.8.6-20210715-1421) python Location_DNN.ip:
python: error while loading shared libraries: libpython3.8.so
(python-env-python3.8.6-20210715-1421) ls
2_RandForcePCA.ipynb          GRU_1FixedForce-Labels.py
ANN_1_FixedForceLevel_UR5.ipynb GRUFixedForce-MA.py
CNN_EX1.ipynb                GRU_LOWEMI.ipynb
datax.hdf5                   jupyter-session-29232985.log
datay.hdf5                   jupyter-session-29401685.log
DIR                           jupyter-session-29401907.log
Features_1FixedF.ipynb       kfold.ipynb
GRU_1FixedForce-Labels.ipynb  K Fold Task A-Multi class.ip:
GRU_1FixedForce-Labels-MA.ipynb Location_DNN.ipynb
(python-env-python3.8.6-20210715-1421) █
```

# 3 CLI Commands under the Hood: Jupyter Magic

The IPython Jupyter kernel supports *Jupyter magic commands* that:

- add additional functionality to Jupyter notebooks
- provide CLI functionality by the `%bash` and `!` magic commands

## Hint

See the Jupyter documentation to learn more about magic commands:  
[ipython.readthedocs.io/.../magics.html](http://ipython.readthedocs.io/.../magics.html)

## Example

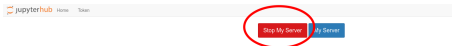
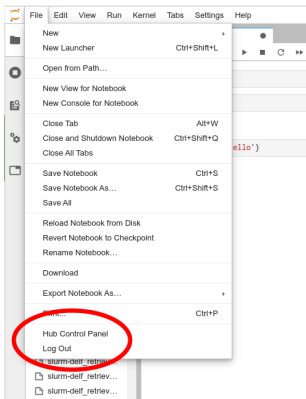
```
[3]: %%bash
     cd /beegfs/ws/1/s4122485-jh_ws
     mkdir test_magic
     ls -l

total 42757
drwxr-xr-x 4 s4122485 p_scads      2 Oct 17 15:21 JPEGIm
-rw-r--r-- 1 s4122485 p_scads      789 Oct 17 13:22 LICENSE
-rw-r--r-- 1 s4122485 p_scads 43780258 Oct 17 13:21 8000_0
```

```
[5]: !cd /beegfs/ws/1/s4122485-jh_ws && ls -l

total 42757
drwxr-xr-x 4 s4122485 p_scads      2 Oct 17 15:21 JPEGIm
-rw-r--r-- 1 s4122485 p_scads      789 Oct 17 13:22 LICENSE
-rw-r--r-- 1 s4122485 p_scads 43780258 Oct 17 13:21 8000_0
```

# 3 JupyterHub Stop a Session



## Warning

Work finished do not forget to logout and stop your server! Otherwise, the resources will not be available to others and account for your **CPU quota!**

# 4 Data Management via Jupyter – Agenda

- 1 Introduction
- 2 Jupyter overview
- 3 Access and Basics of the ZIH HPC system
- 4 Data Management via Jupyter  
Data Management via Jupyter
- 5 Collaborative Research and Teaching with Jupyter on HPC
- 6 Hands On session: Prepare a Jupyter "environment" for collaborative working
- 7 Conclusion, Q&A and Supplementary

## 4 Data Management on HPC System

- many systems to store data that differ in capacity, bandwidth, IOPS rate etc
- organization of these storage systems in **files systems**
- many file systems, users, data require mechanisms for data management (duration, size, permission)
- the concept of work-spaces is used on the ZIH system for data management

### Hint

for this presentation: a file system refers to a "place" to store data and a **workspace** refers to the "access" to that place



## 4 Filesystem Storage Overview

- Each filesystems serves as respective capacity, performance, duration, lifetime.
- User creates a workspace with defined expiration date.
- A workspace can be extended multiple times, depending on the filesystem.

| scope            | filesystem      | speed | size | duration |
|------------------|-----------------|-------|------|----------|
| local            | /tmp            | +++   | ---  | ---      |
| Global temporary | /beegfs         | ++    | --   | --       |
| Global temporary | /ssd            | +     | -    | --       |
| Global temporary | /scratch        | -     | +    | +        |
| Global permanent | /projects /home | --    | ++   | ++       |
| archiving        | /warm_archive   | ---   | +++  | +++      |

### Note

/projects and /home are permanent filesystems without expiration date

## 4 Workspace management in JupyterHub

To manage workspaces in Jupyter, use dedicated `ws` commands via:

- the CLI from the JupyterHub
- Jupyter magic commands within a notebook cell to execute bash scripts

| Description   | Command                     |
|---|-----------------------------|
| Find available workspace filesystems                          | <code>ws_find --list</code> |
| Allocate workspace  | <code>ws_allocate</code>    |
| List your workspaces and get information (duration, path,...) | <code>ws_list</code>        |
| Extend workspace  | <code>ws_extend</code>      |
| Delete workspaces   | <code>ws_release</code>     |

### Note

Find additional commands in the docs: <https://doc.zih....data/lifecycle/workspaces/>

## 4 Workspace - Example

A **workspace** is a directory, with an associated expiration date, created on behalf of a user in a certain filesystem.

### ☰ Example

```
1 # Show available file systems for using workspaces
2 marie@compute$ ws_find -l
3
4 # List all allocated workspaces
5 marie@compute$ ws_list
6
7 # Allocate a workspace "test-workspace" in filesystem "scratch"
8 marie@compute$ ws_allocate -F scratch test-workspace
9
10 # Extend lifetime of a workspace called "test-workspace"
11 marie@compute$ ws_allocate -F scratch test-workspace 10
12
13 # Delete workspace
14 marie@compute$ ws_release test-workspace
```

# 4 Accessing Data: Linking Workspaces

- It might be valuable to link workspaces within a certain directory, e.g., your home directory.
- The command "ws\_register" DIR to create link of workspace within the Home DIR directory
- via the Workspace scope parameter, set a workspace as the default directory for the notebook data

## Warning

The **workspace scope** parameter is only for setting a specific working directory as default through the spawn menu. It is not related to the workspace principles on the system.

### Spawner Options

**Very important!**  
If you have finished your work please explicitly stop your server.

**Current Utilization**

- 2 998 cores available of 24 320 cores on romeo
- 7 gpus available of 256 gpus on alpha
- 56 cores available of 896 cores on julia

Simple Advanced

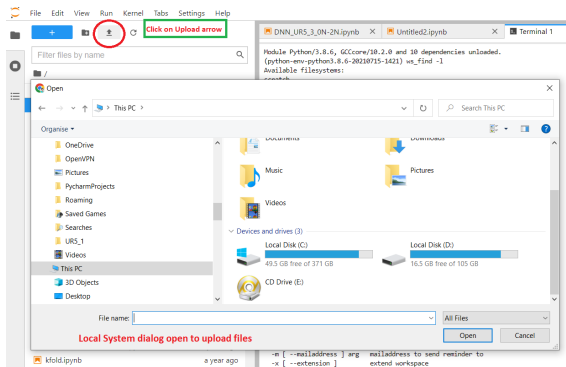
Preset GPU (NVIDIA Ampere A100) Save preset Delete preset ...

|  |                                   |
|--|-----------------------------------|
| Partition (-p, --partition):                   | alpha                             |
| NodeList (-w, --nodelist):                     | no specific nodes                 |
| Nodes (-N, --nodes):                           | 1                                 |
| Number of tasks (-n, --tasks):                 | 1                                 |
| CPUs per task (-c, --cpus-per-task):           | 6                                 |
| Generic resources (--gres):                    | gpu.1                             |
| Memory per CPU (--mem-per-cpu):                | 10312                             |
| Runtime (-t, --time):                          | 8:00:00                           |
| Reservation (--reservation):                   | no reservation                    |
| Project (-A, --account):                       | default                           |
| Preload modules (module load):                 |                                   |
| Standard environment (package list):           | hiera_gcccore-10.2.0_python-3.8.6 |
| Launch application:                            | JupyterLab                        |
| Workspace scope (--NotebookApp.notebook_dir=): | < Your Workspace >                |

Spawn

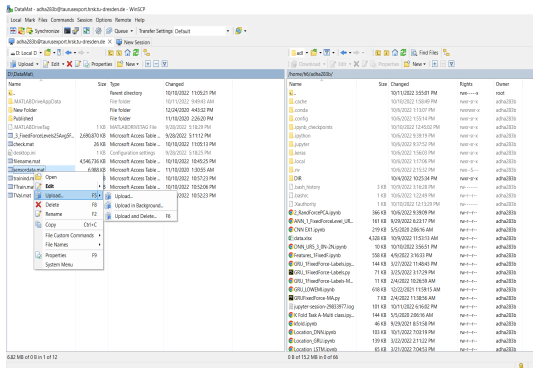
# 4 Data transfer to the JupyterHub

With JupyterHub GUI, data can be uploaded, downloaded with into workspace or home directory.



# 4 Data transfer to the ZIH System, Windows

With an WinSCP software, data can be uploaded, downloaded with a GUI to workspace or home directory for windows



## 4 Data Transfer on HPC System

There are two methods for exchanging data on HPC:

- scp, rsync, and sftp
- Datamover, to move data faster inside the ZIH
- for datamover you have to use commands prefixed with dt: dtcp, dtwget, dtmv, dtrm, dtrsync, dttar, dtls

More information in the wiki: <https://doc.zih.../access/data/transfer/>

**Example:** Access From Linux using scp

### ☰ Example

```
1 # Copy a file from your workstation to ZIH systems
2 marie@local$ scp <file> taurusexport:<target-location>
3 # Add -r to copy whole directory
4 marie@local$ scp -r <directory> taurusexport:<target-location>
5 # Copy a file from ZIH systems to your workstation
6 marie@local$ scp taurusexport:<file> <target-location>
```

# 4 Data Transfer on HPC System

Command examples prefixed with dt for transfer files from one filesystem to another filesystem

## ☰ Example

```
1 # dtcp for copying data from one filesystem to another filesystem
2 marie@login$ dtcp -r <fs_origin>/results <fs_destination>/.
3
4 # dtmv for moving data from one filesystem to another filesystem
5 marie@login$ dtmv <fs_origin>/ws/results <fs_destination>/ws/ws-archive/.
6
7 # dttar for archive data "results.gz" from one filesystem to another filesystem
8 marie@login$ dttar -czf <fs_destination>/results.tgz <fs_origin>/ws/results
```



# 5 Collaborative Research and Teaching with Jupyter on HPC – Agenda

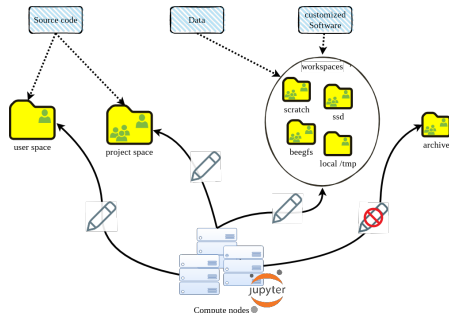
- 1 Introduction
- 2 Jupyter overview
- 3 Access and Basics of the ZIH HPC system
- 4 Data Management via Jupyter
- 5 Collaborative Research and Teaching with Jupyter on HPC
- 6 Hands On session: Prepare a Jupyter "environment" for collaborative working
- 7 Conclusion, Q&A and Supplementary

# 5 Collaborative Principles on HPC

- consider access/permission restrictions on the cluster
- level 1: permissions of files/directories need to be configured (esp. data and customized software environments)
- level 2: read/write connection between parts of the cluster (login nodes, compute nodes, storage)

Distinguish between: data, software environment, source code:

- data: in workspaces with group access
- software environment:
  - ▶ module system
  - ▶ in workspaces with group access
- source code:
  - ▶ user space or directory in project space
  - ▶ collaborative management via external version control, e.g. Git



# 5 Accessing Data and Sharing Software: Permissions Management on ZIH system

On the ZIH system, we use Linux permissions:

- distinguish **r**ead, **w**rite and **e**xecute permissions
- assignable to **u**ser/owner, **g**roup (HPC project), **o**ther, **a**ll
- works on directory level and file level

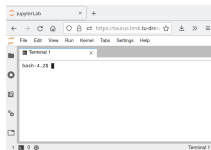
## Hint

The `chown` and `chgrp` command change the owner and group of the directory/file

Reading permissions with `ls -l` command:

### Example

```
1 # Check permissions of directory
2 marie@compute$ ls -l /scratch/ws/1/Marie-ml-ws/
3 # Output
4 -rwxr-xr-- 1 marie marie_group 10 11. Nov 11:11 textfile.txt
5 drwx----- 1 marie marie_group 4096 11. Nov 11:12 marie_dir
```



# 5 Accessing Data and Sharing Software: Permissions Management on ZIH system

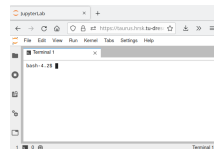
Permission management is done by the 'chmod' command via CLI. Syntax:

```
chmod [ugoa]*([-+]=([rwxXst]*|[ugo]))+|[-+]=[0-7]+ myfile
```

- ugoa = user/owner, group, other, all
- -+= = remove(-), add(+), set(=) permissions
- rwx = read, write, execute
- for more information: `man chmod`

## Example

```
1 # Give read and execute access to the group of the file
2 marie@compute$ chmod g+rx /scratch/ws/1/Marie-ml-ws/marie_dir -R
3 # Check permissions of directory
4 marie@compute$ ls -l /scratch/ws/1/Marie-ml-ws/
5 # Output
6 drwxr-x--- 1 marie marie_group 4096 11. Nov 11:12 marie_dir
```



# 5 Customized Software Environments with Python

- Python virtual environments create isolated working environments
- You can install packages, without interfere with other environments
- Multiple environments can be installed on one system simultaneously
- `conda` or `virtualenv` are state-of-the-art systems to create virtual environments

## Warning

`conda` virtual environments may infer with the module system on HPC. Therefore we recommend using `virtualenv` and `pip` instead of `conda`

More information in the wiki: [https://doc.zih.../software/python\\_virtual\\_environments](https://doc.zih.../software/python_virtual_environments)

# 5 Shared Python Environment and Jupyter Kernels

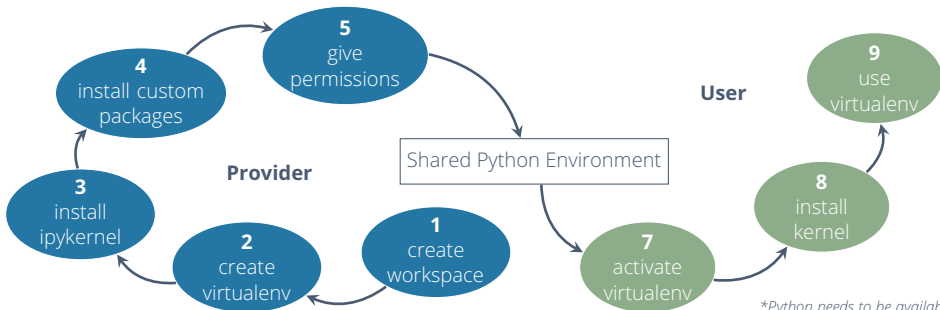
We assume two roles for collaboration:

- **Provider:**

- ▶ provide a consistent Python environment
- ▶ share environment with collaborators
- ▶ e.g. teacher, researcher

- **User:**

- ▶ uses a Python environment to reproduce results or studying
- ▶ e.g. pupil, research collaborator



*\*Python needs to be available for following the workflow*

# 5 Shared Python Environment: Example - Provider

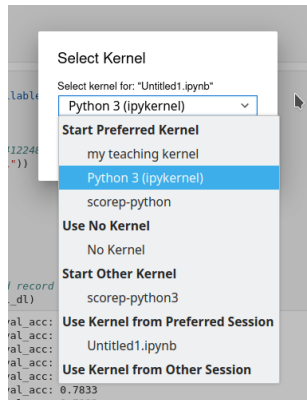
## ☰ Example

```
1 # Create a virtual environment in the workspace
2 marie@compute$ virtualenv --system-site-packages jh_venv
3
4 # Activate the environment
5 marie@compute$ source jh_venv/bin/activate
6
7 # Install the ipykernel package to use the environment in JupyterHub
8 (jh_venv) marie@compute$ pip install ipykernel
9
10 # Install additional packages
11 (jh_venv) marie@compute$ pip install [...]
12
13 # Set permissions of the workspace to share environment with collaborators
14 marie@compute$ chmod g+rx /scratch/ws/1/jh_ws/ -R
```

# 5 Shared Python Environment: Example - User

## Example

```
1 # Activate the environment
2 marie@compute$ source /scratch/ws/1/jh_ws/jh_venv/bin/activate
3
4 # Install the kernel for Jupyter
5 (jh_venv) marie@compute$ python -m ipykernel install --user --
   name my-teaching-kernel --display-name="my teaching kernel"
6
7 # Deactivate environment
8 (jh_venv) marie@compute$ deactivate
```



After installing the kernel, it is available in the drop down menu



# 5 Persistence of Environments

- Goal: long-term storage and distribution of a Python software environment
- Conda and pip provide mechanisms to persist environments via a file
- Permanently store files in Git repository, e.g. repository for lecture series
- Restore environments by creating new environments based on the file

## ☰ Example

**pip:**

```
1 # Store packages of "env_pre" in "req.txt" and restore in fresh environment "env_post"
2 (env_pre) marie@compute$ pip freeze > req.txt
3 (env_post) marie@compute$ pip install -r req.txt
```

# 5 Share a Spawn Link

- JupyterHub allows to share spawn options, e.g. Slurm parameters or work directory, via a link
- Every parameter of advanced form can be set as parameter
- If a parameter is not set, the default value will be loaded
- Setting the parameters in the form also sets the parameters in your browsers URL line:

  [https://taurus.hrsk.tu-dresden.de/jupyter/hub/spawn#/~\(partition~'romeo](https://taurus.hrsk.tu-dresden.de/jupyter/hub/spawn#/~(partition~'romeo) 

|                                      |                      |             |                   |
|--------------------------------------|----------------------|-------------|-------------------|
| Simple                               |                      | Advanced    |                   |
| Preset                               | ▼                    | Save preset | Delete preset ... |
| Partition (-p, --partition):         | romeo                | ▼           |                   |
| Nodelist (-w, --nodelist):           | no specific nodes    |             |                   |
| Nodes (-N, --nodes):                 | 1                    |             |                   |
| Number of tasks (-n, --tasks):       | 1                    |             |                   |
| CPUs per task (-c, --cpus-per-task): | 4                    |             |                   |
| Generic resources (--gres):          | no generic resources |             |                   |
| Memory per CPU (--mem-per-cpu):      | 2500                 |             |                   |

## Example

[https://taurus.hrsk.tu-dresden.de/jupyter/hub/spawn#/~\(partition~'romeo~cpuspertask~'4~mempercpu~'2500](https://taurus.hrsk.tu-dresden.de/jupyter/hub/spawn#/~(partition~'romeo~cpuspertask~'4~mempercpu~'2500)

base URL and page "spawn"

encoded spawn parameters

# 5 Git Pull Feature

- Clone a public git repository into users work directory via `nbgitpuller` extension
- Distribute notebooks, data and other material to students or colleagues
- Use [nbgitpuller link generator](#)

## ☰ Example

[https://taurus.hrsk.tu-dresden.de/jupyter/hub/user-redirect/...](https://taurus.hrsk.tu-dresden.de/jupyter/hub/user-redirect/)

base URL and page "user-redirect"

...[git-pull?repo=https://github.com/jdwittenauer/ipynotebooks&urlpath=/tree/ipynotebooks/notebooks/language/Intro.ipynb](https://github.com/jdwittenauer/ipynotebooks)

call to `nbgitpuller` extension with "repo" and "urlpath" parameter

- clone repository *ipynotebooks* into work directory
- open *Intro.ipynb* notebook

# 5 Git Pull Feature: NBGitpuller Link Generator

[JupyterHub](#) [Launch from Canvas](#) [Binder](#)

`https://taurus.hrsk.tu-dresden.de/jupyter/hub/user-redirect/git-pull?repo=https%3A%2F%2Fgithub.com%2Felwer%2FJH_HPC_demo&urlpath=lab`

JupyterHub URL  ✓  
The JupyterHub to send users to. [nbgitpuller](#) must be installed in this hub.

Git Repository URL  ✓   ✓  
Use **main** instead of **master** for new GitHub repositories

File to open  ✓  
This file or directory from within the repo will open when user clicks the link.

Application to Open

- Classic Jupyter Notebook
- RetroLab
- JupyterLab
- RStudio
- Shiny
- Custom URL

# 5 Git Pull Feature: Private Repository

- Git pull feature also possible for private repositories, e.g. repository in GitLab Chemnitz
- Access tokens provide access to your repository for the "outer world"
- Permissions and expiration of tokens definable by yourself

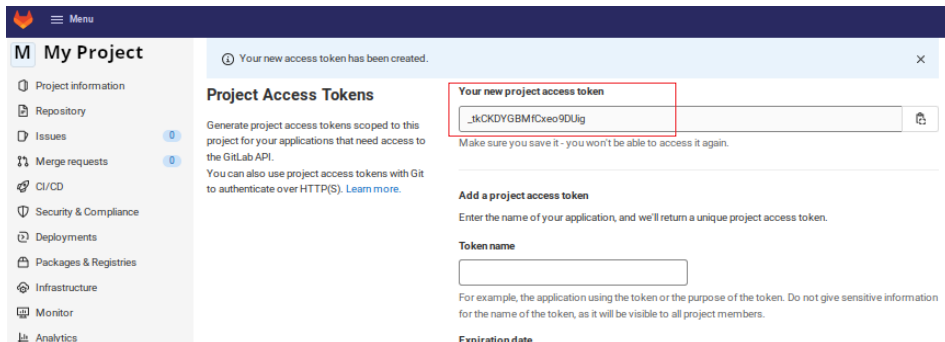
## Workflow:

1. Create access token in repository settings
2. Adapt link to repository for Git HTTPS cloning:  
`https://<username>:<access-token>@<repository-url>`
3. Use [nbgitpuller link generator](#)

### Note

GitLab Chemnitz creates a "Bot user" when creating an access token. Use this user for the link generation. The username is created as follows: `project_project_id_bot`. You can find the `project_id` in your repository settings. More information: [gitlab.hrz.tu-chemnitz.de/.../project\\_access\\_tokens](https://gitlab.hrz.tu-chemnitz.de/.../project_access_tokens)

# 5 Git Pull Feature: Private Repository - Example(1)



Menu

## M My Project

- Project information
- Repository
- Issues 0
- Merge requests 0
- CI/CD
- Security & Compliance
- Deployments
- Packages & Registries
- Infrastructure
- Monitor
- Analytics

Your new access token has been created.

### Project Access Tokens

Generate project access tokens scoped to this project for your applications that need access to the GitLab API.

You can also use project access tokens with Git to authenticate over HTTP(S). [Learn more.](#)

**Your new project access token**

`_tkCKDYGBMfCxeo9DUlg`

Make sure you save it - you won't be able to access it again.

**Add a project access token**

Enter the name of your application, and we'll return a unique project access token.

**Token name**

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

**Expiration date**

## Warning

The token is only visible at creation time. On the page you can find multiple options for configuring a specific token (access role, scopes, expiration date) as well as revoking access for existing tokens.

# 5 Git Pull Feature: Private Repository - Example(2)

ScaDS-AI > My Project



## My Project

Project ID: 99 [Leave project](#)

27 Commits 2 Branches 0 Tags 3.3 MB Project Storage



### Auto DevOps

It will automatically build, test, and deploy your application based on a

Learn more in the [Auto DevOps documentation](#)

[Enable in settings](#)

### Clone with SSH

### Clone with HTTPS

### Open in your IDE

Visual Studio Code (SSH)

Visual Studio Code (HTTPS)

IntelliJ IDEA (SSH)

IntelliJ IDEA (HTTPS)

MLP\_MNIST

my\_project / +

Find file

Web IDE

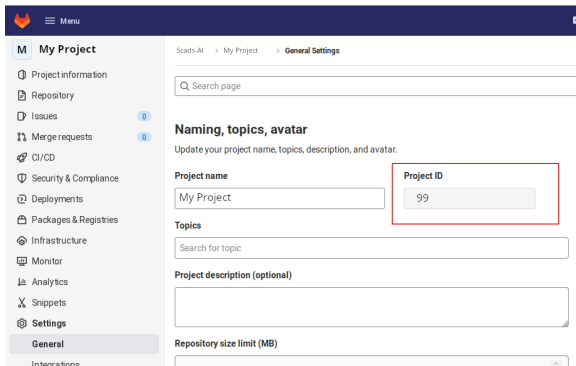


Clone

# 5 Git Pull Feature: Private Repository - Example(3)

## Workflow Example:

- get access token
- get git repository link
- get project id Bot username:  
project\_99\_bot
- now: create link for Git HTTPS cloning  
`https://project_99_bot:_tkagshd63Sd@gitlab.hrz.tu-chemnitz.de/scads-ai/my_project.git`
- use the link in the nbgitpuller link generator





# 5 Combine Parameter Sharing and Git Pull

- You can also combine the *parameter sharing* and the *git pull* feature via a single link
- Concatenation of links can be done manually or:
  1. create the *parameter sharing* link
  2. insert the link as *JupyterHub URL* in the nbgitpuller link generator and add *git pull* functionality

## Warning

When **combining** the link via the nbgitpuller link generator, the link might be defective and needs to be fixed.

e/jupyter/hub/spawn/hub/user-redirect/git-pu  
↓  
e/jupyter/hub/user-redirect/git-pu

## Example

[https://taurus.hrsk.tu-dresden.de/jupyter/hub/user-redirect/...](https://taurus.hrsk.tu-dresden.de/jupyter/hub/user-redirect/)

base URL and page "user-redirect"

[...git-pull?repo=https://github.com/jdwittenauer/ipynb-puller&urlpath=/tree/ipynb-puller/notebooks/language/Intro.ipynb...](https://github.com/jdwittenauer/ipynb-puller)

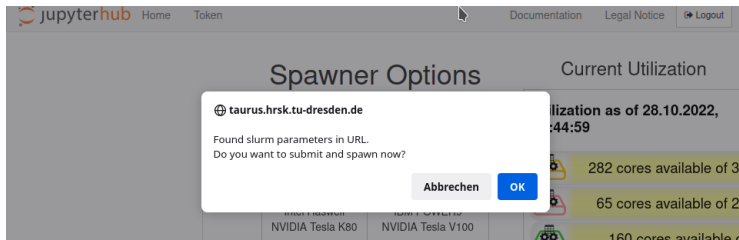
call to nbgitpuller extension with "repo" and "urlpath" parameter

[...#/~\(partition~'romeo~cpuspertask~4~mempercpu~2500\)](#)

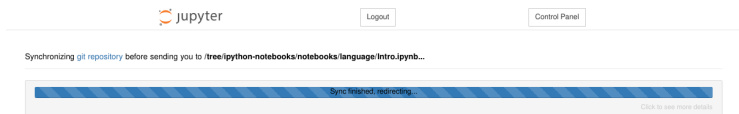
encoded spawn parameters

# 5 JupyterHub Link Feedback

Feedback of the parameter sharing feature:



Feedback of the git pull feature:



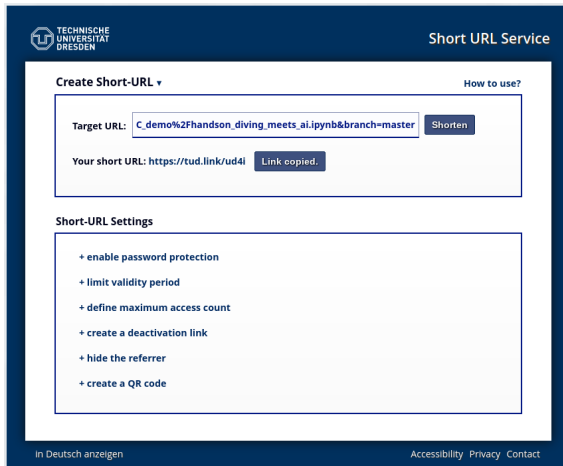
# 5 Short URLs

## Hint

Short URL services shorten your links and may provide further functionality, e.g.

- expiration date
- maximum number of accesses

We recommend to use the TUD Short link service for that purposes:  
<https://tud.link/create/>



The screenshot shows the 'Short URL Service' interface from Technische Universität Dresden. It features a dark blue header with the university logo and name. The main content area is white with a dark blue border. At the top left of the content area is the text 'Create Short-URL' with a dropdown arrow, and at the top right is a link 'How to use?'. Below this is a form with a 'Target URL' input field containing 'C\_demo%2Fhandson\_diving\_meets\_ai.ipynb&branch=master' and a 'Shorten' button. Below the input field, it displays 'Your short URL: https://tud.link/ud4i' with a 'Link copied.' button. Underneath is a 'Short-URL Settings' section with a list of options, each preceded by a plus sign: 'enable password protection', 'limit validity period', 'define maximum access count', 'create a deactivation link', 'hide the referrer', and 'create a QR code'. At the bottom of the interface, there are links for 'In Deutsch anzeigen', 'Accessibility', 'Privacy', and 'Contact'.

# 6 Hands On session: Prepare a Jupyter "environment" for collaborative working – Agenda

- 1 Introduction
- 2 Jupyter overview
- 3 Access and Basics of the ZIH HPC system
- 4 Data Management via Jupyter
- 5 Collaborative Research and Teaching with Jupyter on HPC
- 6 Hands On session: Prepare a Jupyter "environment" for collaborative working
- 7 Conclusion, Q&A and Supplementary

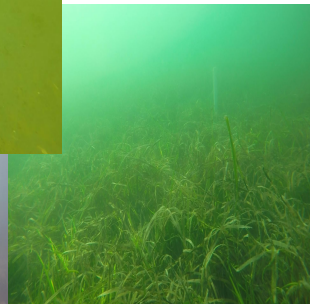
# 6 HandsOn: Diving meets AI

**Dataset:** NOAA Puget Sound Nearshore Fish 2017-2018

- 77.739 images from underwater video
- 30.384 images with fish and crustaceans, annotated as *animal*
- remaining images annotated as *empty*
- more information about the dataset: [lila.science/.../noaa-puget-sound-nearshore-fish](https://lila.science/.../noaa-puget-sound-nearshore-fish)

## ⚠ Warning

Please use your personal ZIH login credentials we sent you for the following tasks! Use the reservation `p_scads_jupyterhub_773` for the `--reservation` parameter to reduce scheduling time.



# 6 Setting

## Hint

We distinguish between *source code*, *data* and *software*. In this example, we already prepared the source code in a git repository.

- A Jupyter notebook with the source code is available in a Git repository:  
[https://github.com/elwer/JH\\_HPC\\_demo](https://github.com/elwer/JH_HPC_demo)
- The NOAA fish dataset is available in the workspace:  
`/beegfs/ws/1/s4122485-jh_ws/NOAA`
- **Aim:** Share your experiences with the dataset with your colleagues
  
- The example consists of two parts:
  1. Prepare and share your setup (data and software)
  2. Test the setup

# 6 Part 1a: Prepare the workspace and the data

Provider

## Concrete tasks (data part):

1. Open the browser and spawn a Jupyter server:
  - ▶ Advanced option (see slide 30)
  - ▶ Partition: alpha
  - ▶ Runtime: 2hours
  - ▶ Reservation: p\_scads\_jupyterhub\_773
2. Open a terminal after spawning (see slide 37)
3. Create a workspace `teaching_ws` on `beegfs` file system (see slide 44)
  - ▶ create a directory `data` for the data in `teaching_ws`
  - ▶ create a directory `sw` for the software in `teaching_ws`
  - ▶ note down/remember the workspace directory (it is required later)
4. Copy the data from `/beegfs/ws/1/s4122485-jh_ws/NOAA` to `[...]/teaching_ws/data` either via the `datamover` nodes (see slide 49) or simple `cp` command

# 6 Part 1b: Prepare the software

## Concrete tasks:

Provider

1. Load the Python module: `module load modenv/scs5 Python/3.9.5`
2. Create a Python virtual environment in `[...]/teaching_ws/sw` and activate it (see slide 56)
3. Install the `ipykernel` package (see slide 56)
4. Install additional packages (see slide 56):
  - ▶ `pip3 install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu113`
  - ▶ `pip install tqdm matplotlib`
5. Give the group `read` and `execute` permissions to `[...]/teaching_ws` and all subdirectories (see slide 53)
6. Log out from your session and stop it(see slide 39)

### ⚠ Warning

Important: This example is only for demonstration purposes. When interacting with the HPC, always prefer pre-installed Modules instead of installing the software on your own.



# 6 Part 1c: Prepare for sharing

Provider

## Concrete tasks:

- Via the spawner advanced interface: Create a link for sharing the resources (see slide 59):
  - ▶ partition: alpha
  - ▶ nodes: 1
  - ▶ CPUs: 6
  - ▶ GPUs: 1
  - ▶ memory per CPU: 2500M
  - ▶ runtime: 2hours
  - ▶ reservation: p\_scads\_jupyterhub\_773
  - ▶ preload modules: Python/3.9.5
  - ▶ standard environment: production
  - ▶ workspace\_scope: your workspace directory
- Insert the link in the nbgitpuller website as *JupyterHub URL* (see slide 60/61)
- Add the *Git Repository URL* and the *File to open* to combine both links (see slide 61/62)
- Shorten the link (see slide 64) and communicate the link to the group via the chat

## 6 Part 2: Test the setup

User

### Concrete task:

1. Use some link from the chat to start a JupyterHub (not your own)
2. Activate and install the virtual environment from the workspace of your group members either via CLI or Jupyter magic (see slide 38)
3. Select the installed Kernel
4. Execute the Jupyter notebook and explore the data set

### Warning

This part will only work, if the person that created your link, carefully followed all the steps in part 1.

## 6 Basic CLI Commands

| Command                                | Description   |
|--|---|
| ls                                     | list files and/or directories                               |
| cd <path_destination>                  | change directory  |
| mkdir <directory_name>                 | create a new directory                                      |
| pwd                                    | Show full path of the current directory                     |
| cp <path_source> <path_destination>    | copy a file   |
| cp -r <path_source> <path_destination> | copy a directory  |
| mv <path_source> <path_destination>    | move a file or directory                                    |
| rm <path_source>                       | delete a file or directory                                  |
| wget                                   | get datasets from the web (using HTTP, HTTPS, FTP and FTPS) |

## 6 Troubleshooting

- Problems when spawning:
  - ▶ delete cookies
  - ▶ delete access tokens
  - ▶ check whether your spawner is still running via `squeue --me` command in CLI (ssh connection)
- General:
- check the `jupyter-session-XXXXXXXXX.log` file in your HOME directory
- it gives a lot of insights of your notebook servers behaviour (e.g. if module (pre-)load was successful)
- Pika Job Monitoring tool shows you utilization statistics of your notebook server
- ssh connection to the node with the notebook server software (get name of the node via `squeue --me`), then `ssh taurusi_nodeID`, lets you investigate your computing node(s) further

### Hint

Having a look in the compendium is always useful: [doc.zih.tu-dresden.de](http://doc.zih.tu-dresden.de)

# 7 Conclusion, Q&A and Supplementary – Agenda

- 1 Introduction
- 2 Jupyter overview
- 3 Access and Basics of the ZIH HPC system
- 4 Data Management via Jupyter
- 5 Collaborative Research and Teaching with Jupyter on HPC
- 6 Hands On session: Prepare a Jupyter "environment" for collaborative working
- 7 Conclusion, Q&A and Supplementary

# 7 Conclusions

- The Jupyter universe is big (extensions, plugins, use cases, people)
- JupyterHub offers an easy access to HPC resources
- HPC is a complex task - JupyterHub abstracts some of the complexity
- Versatile features for collaborative working
- We proposed one approach but typically there is not only one solution

## 7 Conclusions

- In case of questions start at the official docs for the ZIH system: <https://doc.zih.tu-dresden.de/>
- Software ecosystems are changing dynamically, esp. in ML: with HPC systems it is hard to react immediately
- Contact us with your own ideas, experiences and wishes!

### Note

logins (scads0xx) will work for further 10 days

- if you are interested afterwards please apply for your own HPC project
- for starting purposes we recommend a “Schnupperprojekt” without a detailed project description

**ScaDS.AI**  
**DRESDEN LEIPZIG**

# Thank you

Contact us on:

[elias.werner@tu-dresden.de](mailto:elias.werner@tu-dresden.de),

[adnan.haidar@tu-dresden.de](mailto:adnan.haidar@tu-dresden.de),

[mariela\\_rossana.sanchez\\_figueroa@tu-dresden.de](mailto:mariela_rossana.sanchez_figueroa@tu-dresden.de)