bw|HPC

KIT | NHR
National High-Performance Computing Center

Introduction to Jupyter at NHR@KIT

Samuel Braun, SCC, KIT

Funding: Baden-Württemberg MINISTERIUM FÜR WISSENSCHAFT, FORSCHUNG UND KUNST · Bundesministerium für Bildung und Forschung

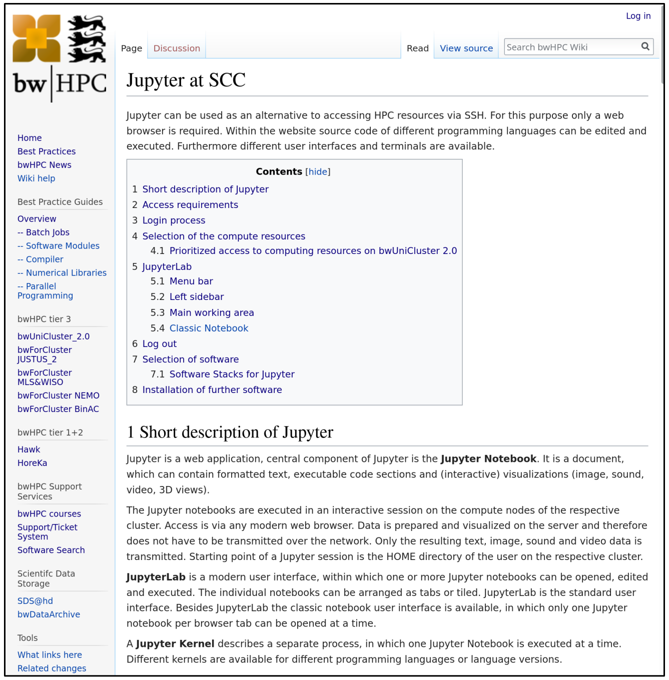
www.bwhpc.de / www.nhr.kit.edu

Outline

- Motivation
- Project Jupyter
- JupyterHub@HoreKa
- Software and Kernels
- Outlook
- Questions

Reference: Jupyter @ KIT

- Most information can be found at
  - bwHPC Wiki: https://wiki.bwhpc.de/e/Jupyter_at_SCC
  - NHR@KIT Wiki: https://www.nhr.kit.edu/userdocs/jupyter



Jupyter at SCC

Jupyter can be used as an alternative to accessing HPC resources via SSH. For this purpose only a web browser is required. Within the website source code of different programming languages can be edited and executed. Furthermore different user interfaces and terminals are available.

Contents [hide]

1 Short description of Jupyter
2 Access requirements
3 Login process
4 Selection of the compute resources
4.1 Prioritized access to computing resources on bwUniCluster 2.0
5 JupyterLab
5.1 Menu bar
5.2 Left sidebar
5.3 Main working area
5.4 Classic Notebook
6 Log out
7 Selection of software
7.1 Software Stacks for Jupyter
8 Installation of further software

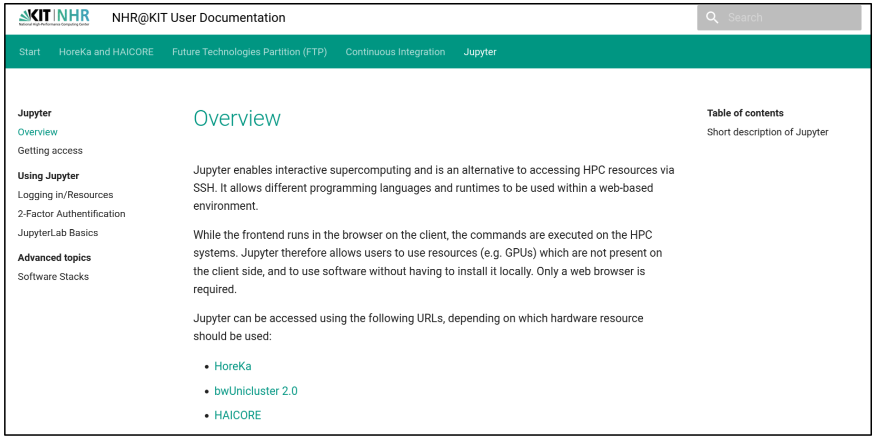
1 Short description of Jupyter

Jupyter is a web application, central component of Jupyter is the **Jupyter Notebook**. It is a document, which can contain formatted text, executable code sections and (interactive) visualizations (image, sound, video, 3D views).

The Jupyter notebooks are executed in an interactive session on the compute nodes of the respective cluster. Access is via any modern web browser. Data is prepared and visualized on the server and therefore does not have to be transmitted over the network. Only the resulting text, image, sound and video data is transmitted. Starting point of a Jupyter session is the HOME directory of the user on the respective cluster.

JupyterLab is a modern user interface, within which one or more Jupyter notebooks can be opened, edited and executed. The individual notebooks can be arranged as tabs or tiled. JupyterLab is the standard user interface. Besides JupyterLab the classic notebook user interface is available, in which only one Jupyter notebook per browser tab can be opened at a time.

A **Jupyter Kernel** describes a separate process, in which one Jupyter Notebook is executed at a time. Different kernels are available for different programming languages or language versions.



Overview

Jupyter enables interactive supercomputing and is an alternative to accessing HPC resources via SSH. It allows different programming languages and runtimes to be used within a web-based environment.

While the frontend runs in the browser on the client, the commands are executed on the HPC systems. Jupyter therefore allows users to use resources (e.g. GPUs) which are not present on the client side, and to use software without having to install it locally. Only a web browser is required.

Jupyter can be accessed using the following URLs, depending on which hardware resource should be used:

- HoreKa
- bwUnicluster 2.0
- HAICORE

Motivation

Why Jupyter?

HPC – "Classical"

- SSH
- **High Entry Hurdles**
  - Choice of resources
  - Linux
  - Tools for connection and data transfer
- Remote-Visualization
  - VNC? X11?

- State of the art for **advanced requirements!**

Why Jupyter?

HPC – Jupyter

- Web browser
  - No additional software
  - No data transfer for analysis
- **Low Entry Hurdles**
- **Intermediate** performance requirements
- **Interactive visualization** of data
- **Prototyping**

Project Jupyter

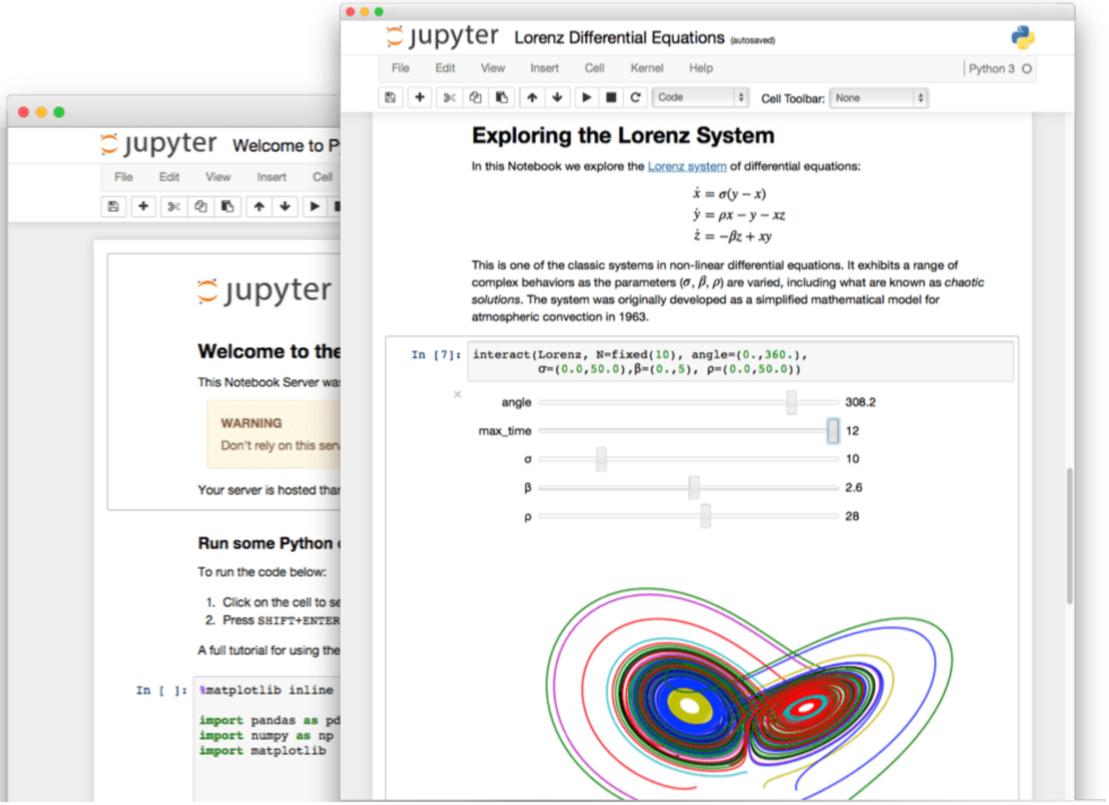
Project Jupyter

- Spin-off from project IPython
- Jupyter: Core languages
  - **Ju**lia
  - **Pyt**hon not**e**book
  - **R**
- **Language agnostic**
- Jupyter kernels
  - IPython
  - IJulia
  - IRKernel
  - >100 other kernels

https://jupyter.org/

Jupyter Notebook

- Open-source web application
- Create and share documents
  - Live code
  - Equations
  - Visualizations
  - Narrative text
  - HTML5 is the limit…
- **Execute code in browser**
  - … on HoreKa
- .ipynb file
  - JSON document



jupyter

Welcome to the

This Notebook Server wa

WARNING
Don't rely on this serv

Your server is hosted thar

Run some Python

To run the code below:

1. Click on the cell to se
2. Press SHIFT+ENTER

A full tutorial for using the

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

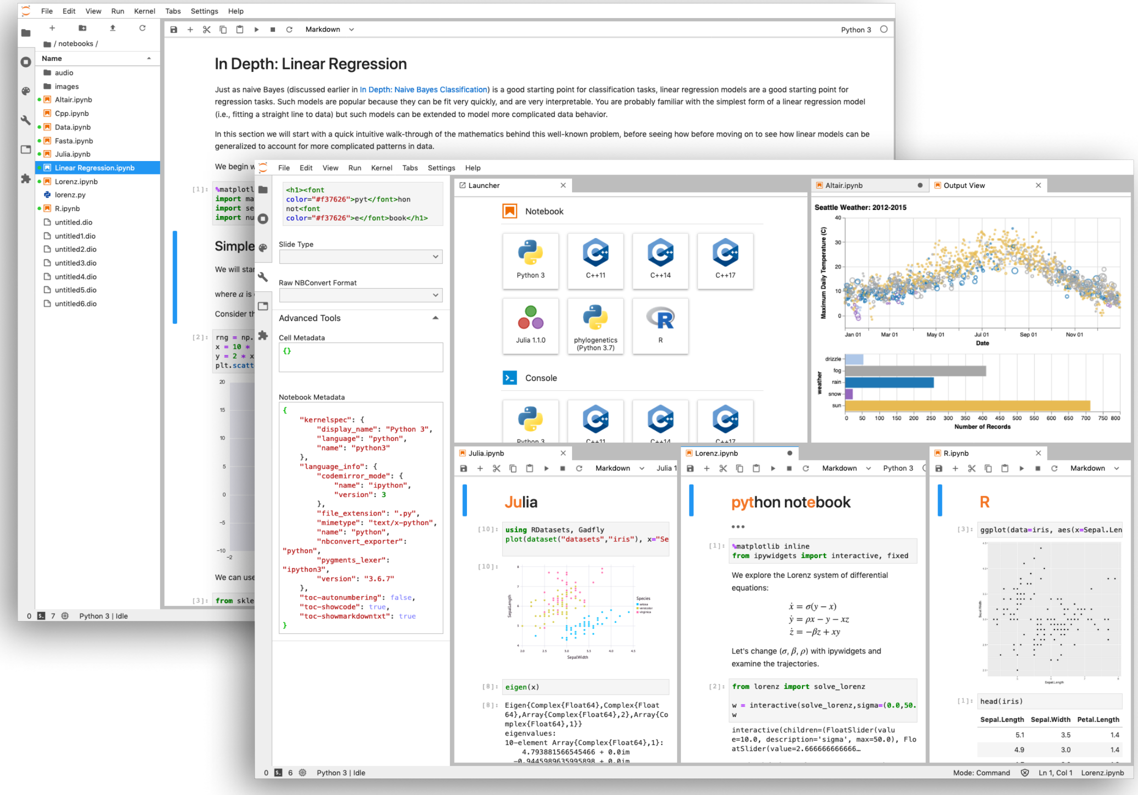
This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

https://jupyter.org/

JupyterLab

- User interface for Project Jupyter
- Arrange documents/activities in tabs/blocks
  - Notebook
  - Terminal
  - Text editor
  - File browser
  - Rich outputs
  - …



In Depth: Linear Regression

Just as naive Bayes (discussed earlier in In Depth: Naive Bayes Classification) is a good starting point for classification tasks, linear regression models are a good starting point for regression tasks. Such models are popular because they can be fit very quickly, and are very interpretable. You are probably familiar with the simplest form of a linear regression model (i.e., fitting a straight line to data) but such models can be extended to model more complicated data behavior.

In this section we will start with a quick intuitive walk-through of the mathematics behind this well-known problem, before seeing how before moving on to see how linear models can be generalized to account for more complicated patterns in data.

```
<h1><font
color="#f37626">pyt</font>hon
not<font
color="#f37626">e</font>book</h1>
```

Cell Metadata

```
{}
```

Notebook Metadata

```
{
    "kernelspec": {
        "display_name": "Python 3",
        "language": "python",
        "name": "python3"
    },
    "language_info": {
        "codemirror_mode": {
            "name": "ipython",
            "version": 3
        },
        "file_extension": ".py",
        "mimetype": "text/x-python",
        "name": "python",
        "nbconvert_exporter":
"python",
        "pygments_lexer":
"ipython3",
        "version": "3.6.7"
    },
    "toc-autonumbering": false,
    "toc-showcode": true,
    "toc-showmarkdowntxt": true
}
```

Julia

```
[10]: using RDatasets, Gadfly
      plot(dataset("datasets","iris"), x="Se
```

```
[8]: eigen(x)
[8]: Eigen{Complex{Float64},Complex{Float
     64},Array{Complex{Float64},2},Array{Co
     mplex{Float64},1}}
     eigenvalues:
     10-element Array{Complex{Float64},1}:
         4.793881566545466 + 0.0im
       -0.9445989635995898 + 0.0im
```

python notebook

```
[1]: %matplotlib inline
     from ipywidgets import interactive, fixed
```

We explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

Let's change (σ, β, ρ) with ipywidgets and examine the trajectories.

```
[2]: from lorenz import solve_lorenz

     w = interactive(solve_lorenz,sigma=(0.0,50.
     w
     interactive(children=(FloatSlider(valu
     e=10.0, description='sigma', max=50.0), Flo
     atSlider(value=2.666666666666…
```

R

```
[3]: ggplot(data=iris, aes(x=Sepal.Len
[1]: head(iris)
```

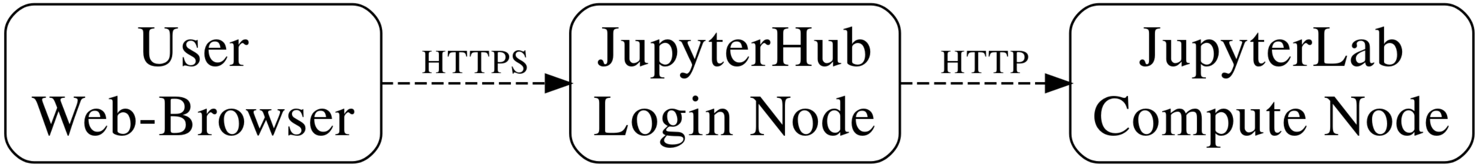
Sepal.Length	Sepal.Width	Petal.Length
5.1	3.5	1.4
4.9	3.0	1.4

https://jupyter.org/

JupyterHub

- Multi-user server for Jupyter Notebooks
- User management and authentication
- Spawning and proxying
- HPC context
  - Choice of resources
  - Slurm integration
  - Authentication

https://jupyter.org/



Jupyter: How-To

Local Jupyter

- Requirements:
  - Python / Anaconda (Windows users)
  - (nodejs + npm: JupyterLab extensions)
- Install locally

```
python -m venv env
source env/bin/activate
pip install jupyterlab
```

- Start
  - Jupyter Notebook: `jupyter notebook`
  - JupyterLab: `jupyter lab`
- Use
  - Open http://127.0.0.1:8888 in web browser

Live Demo

- Jupyter Notebook example mentioned in last c't [1]
- Install

  ```
  python -m venv env
  source env/bin/activate
  pip install jupyterlab
  ```

  Additional Packages

  ```
  pip install matplotlib ipympl
  jupyter labextension install jupyter-matplotlib
  ```

- Start
  - Notebook

    ```
    jupyter notebook
    ```

  - JupyterLab

    ```
    jupyter lab
    ```

- Get the notebook:

  ```
  git clone https://github.com/pinae/BresenhamLidar
  ```

[1]:

Jupyter@HoreKa

- Login to HoreKa, start **interactive job**

```
ssh <userID>@hk.scc.kit.edu
salloc -p accelerated --gres=gpu:4 --time=30:00
```

- Wait till job is running, remember compute node hostname (nodeID) and install and/or **start** Jupyter Notebook or JupyterLab …

```
module load jupyter/base
jupyter notebook --no-browser --port=8888 --ip 0.0.0.0
```

- From your local terminal: Establish **SSH tunnel** to compute node

```
ssh -L 8888:<nodeID>:8888 hk.scc.kit.edu
```

- Open in web browser: http://127.0.0.1:8888

JupyterHub@HoreKa

Registration Process – HoreKa

- **Registration @ HoreKa**
  - Online proposal form (Jards)
  - Peer reviewed proposal
  - HoreKa access form for each coworker
  - Web registration

- Set **service password**
  - FeLS → HoreKa → Set service password

- Register a software or hardware token (alias **2FA**)
  - FeLS → My Tokens
  - KIT users: https://my.scc.kit.edu/token

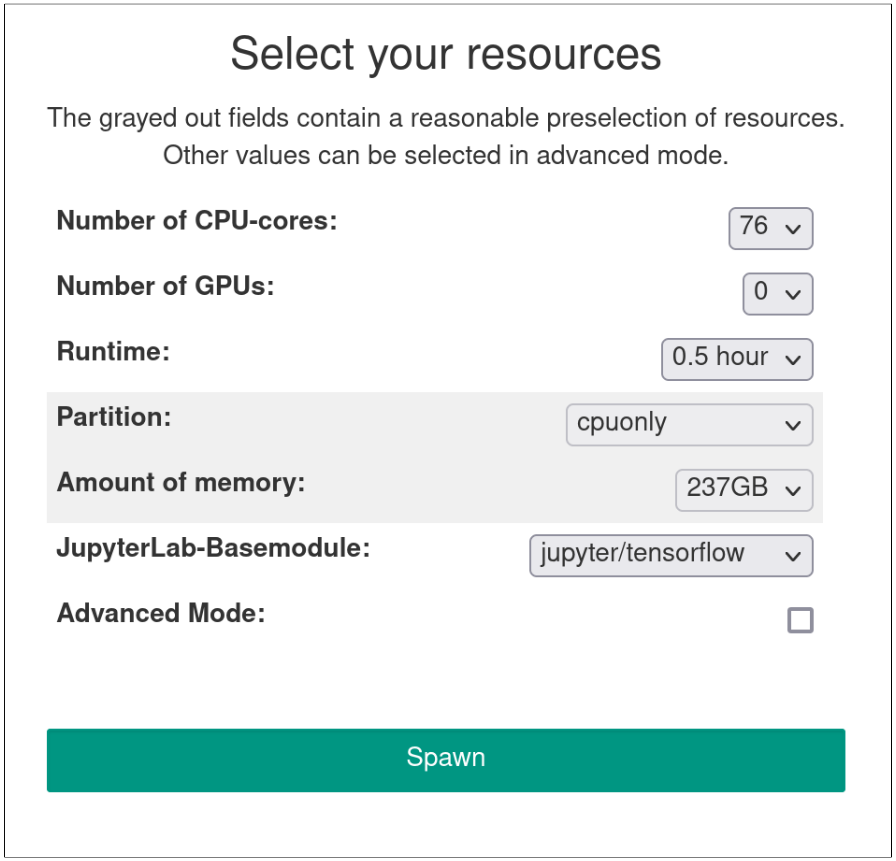
Accessing HoreKa

- Only within **network**
  - … of **KIT**
  - … of your **home institution**
- … otherwise establish **VPN** connection

- **SSH**
  - `ssh <userid>@hk.scc.kit.edu`
  - TOTP prompt (first)
  - Service password (second)

- **Jupyter**
  - (modern) Web browser: https://hk-jupyter.scc.kit.edu
  - **ONE successful login via SSH required**
    - … otherwise there is no `$HOME`
    - … spawning will fail (timeout)

Selection of Resources – Normal



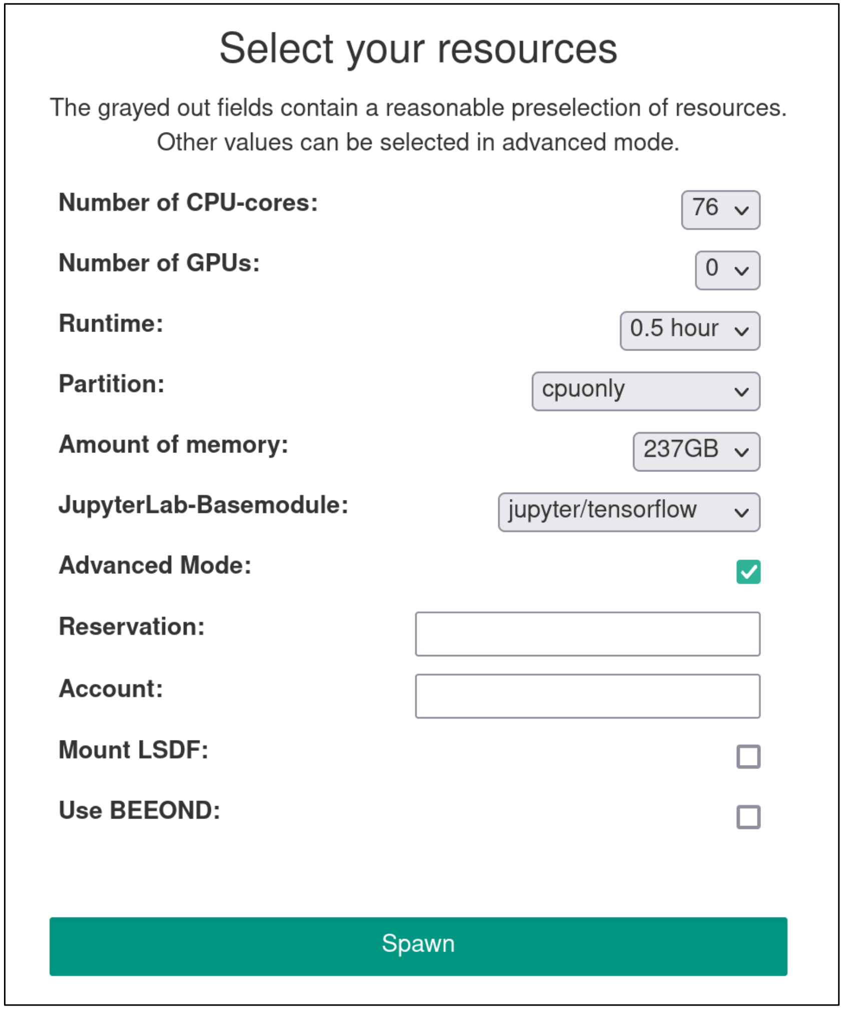
Select your resources

The grayed out fields contain a reasonable preselection of resources. Other values can be selected in advanced mode.

Number of CPU-cores:	76
Number of GPUs:	0
Runtime:	0.5 hour
Partition:	cpuonly
Amount of memory:	237GB
JupyterLab-Basemodule:	jupyter/tensorflow
Advanced Mode:	□

- "**Normal**" mode
  - Number of CPU cores **OR** GPUs
  - Runtime
  - Jupyter Basemodule
  - Grayed out fields: Sane pre-selection of resources
- **Spawn**
  - Starts JupyterLab in interactive Slurm session
  - Connects/proxies to that session

Selection of Resources – Advanced



Select your resources

The grayed out fields contain a reasonable preselection of resources. Other values can be selected in advanced mode.

Number of CPU-cores:	76
Number of GPUs:	0
Runtime:	0.5 hour
Partition:	cpuonly
Amount of memory:	237GB
JupyterLab-Basemodule:	jupyter/tensorflow
Advanced Mode:	☑
Reservation:	
Account:	
Mount LSDF:	□
Use BEEOND:	□

- „**Advanced**“ mode
  - Free choice of resources
  - No grayed out fields
  - No auto reservation

- Reservation
- Account
- LSDF
- BEEOND

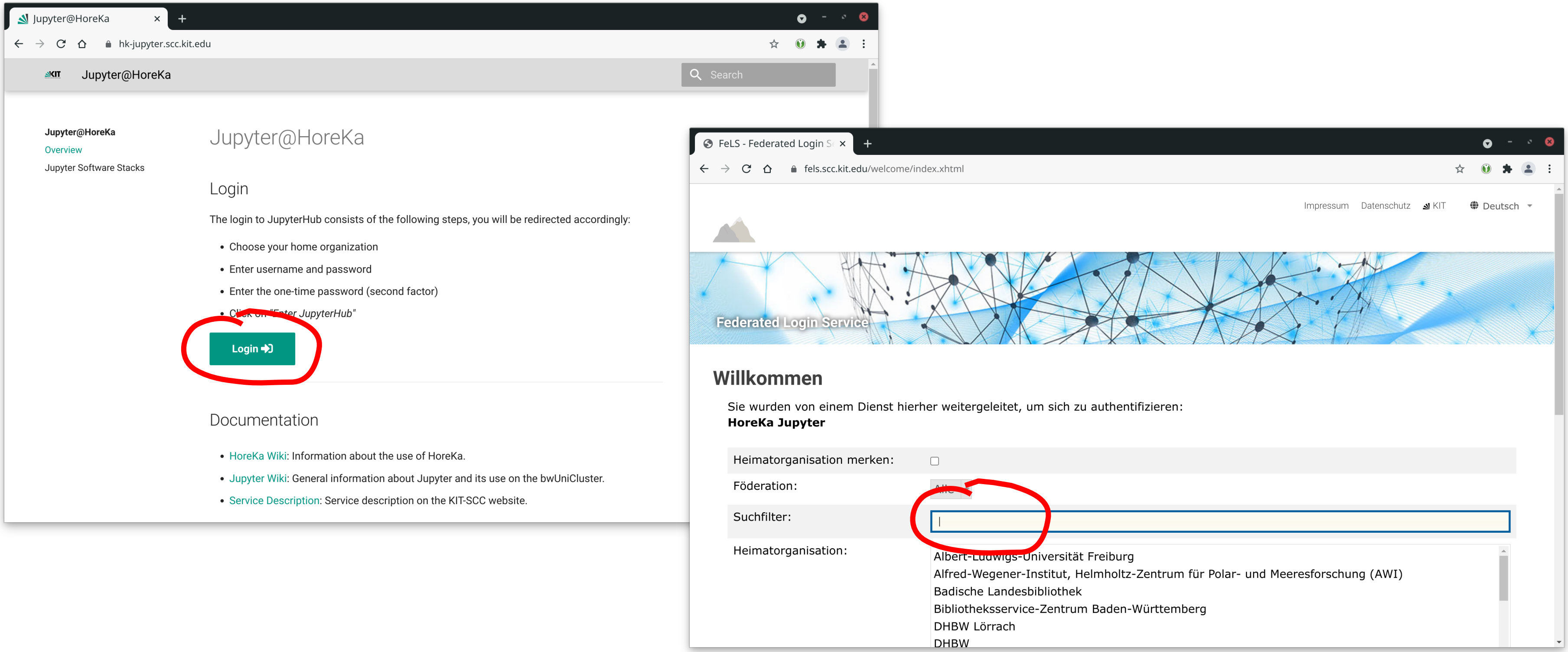
Jupyter Software Stacks

- Lmod modules
  - `jupyter/base`
  - `jupyter/tensorflow`
- JupyterLab lives inside venv
  - `--system-site-packages` enabled/visible
  - Possible interference with `pip --user` installs (!)
- Access via
  - Drop-down menu in JupyterHub: “**JupyterLab-Basemodule**“
  - `module load jupyter/base` or `jupyter/tensorflow`

Login: Step-by-Step

Step-by-Step Jupyter@HoreKa (1/4)

- Go to https://hk-jupyter.scc.kit.edu and click on „**Login**“
  - …or go directly to https://hk-jupyter.scc.kit.edu/hub/login
- Choose your **home organization** and continue



Jupyter@HoreKa

hk-jupyter.scc.kit.edu

Jupyter@HoreKa

Jupyter@HoreKa
Overview
Jupyter Software Stacks

Jupyter@HoreKa

Login

The login to JupyterHub consists of the following steps, you will be redirected accordingly:

- Choose your home organization
- Enter username and password
- Enter the one-time password (second factor)
- Click on *"Enter JupyterHub"*

Login

Documentation

- HoreKa Wiki: Information about the use of HoreKa.
- Jupyter Wiki: General information about Jupyter and its use on the bwUniCluster.
- Service Description: Service description on the KIT-SCC website.

FeLS - Federated Login S

fels.scc.kit.edu/welcome/index.xhtml

Impressum Datenschutz KIT Deutsch

Federated Login Service

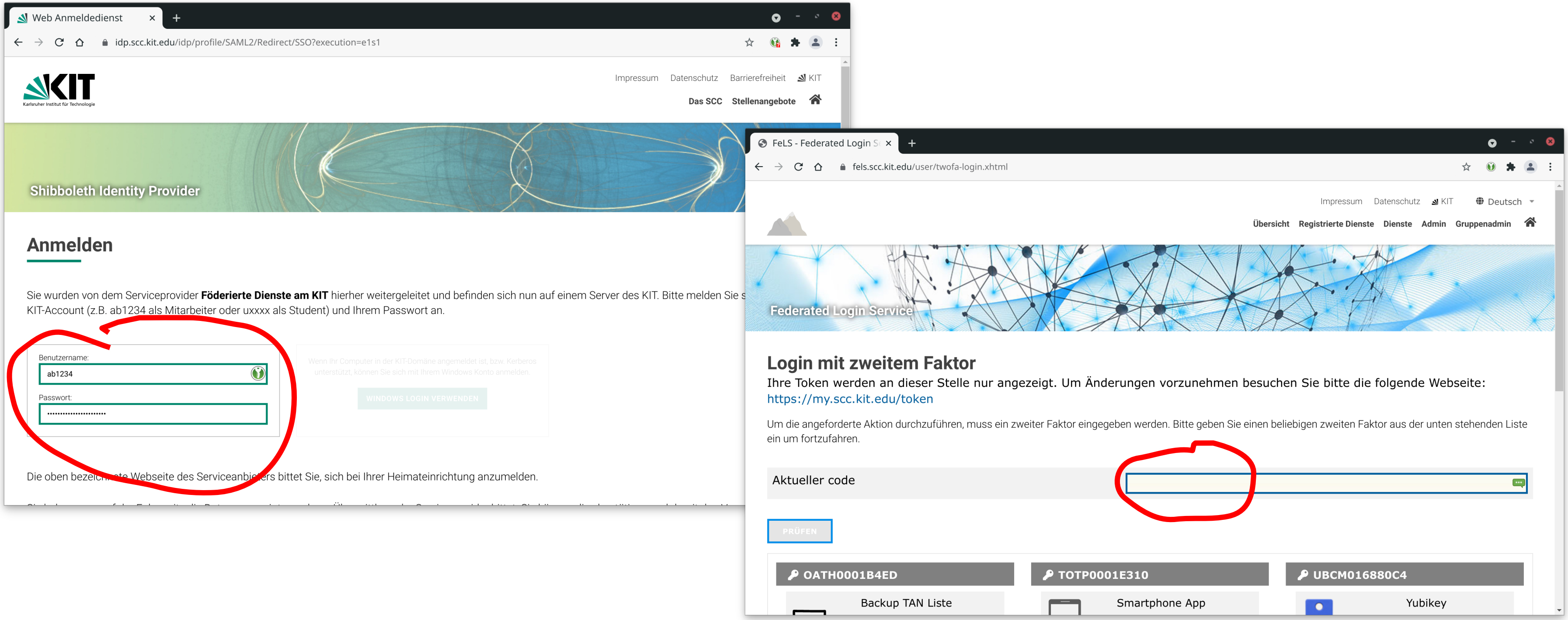
Willkommen

Sie wurden von einem Dienst hierher weitergeleitet, um sich zu authentifizieren:
HoreKa Jupyter

Heimatorganisation merken:	□
Föderation:	Alle
Suchfilter:	
Heimatorganisation:	Albert-Ludwigs-Universität Freiburg Alfred-Wegener-Institut, Helmholtz-Zentrum für Polar- und Meeresforschung (AWI) Badische Landesbibliothek Bibliotheksservice-Zentrum Baden-Württemberg DHBW Lörrach DHBW

Step-by-Step Jupyter@HoreKa (2/4)

- Login to your home organization
  - **Username + password**
  - **2FA**



Web Anmeldedienst
idp.scc.kit.edu/idp/profile/SAML2/Redirect/SSO?execution=e1s1
Impressum
Datenschutz
Barrierefreiheit
KIT
Das SCC
Stellenangebote
Shibboleth Identity Provider

Anmelden

Sie wurden von dem Serviceprovider **Föderierte Dienste am KIT** hierher weitergeleitet und befinden sich nun auf einem Server des KIT. Bitte melden Sie s KIT-Account (z.B. ab1234 als Mitarbeiter oder uxxxx als Student) und Ihrem Passwort an.

Benutzername: ab1234

Passwort: ························

Wenn Ihr Computer in der KIT-Domäne angemeldet ist, bzw. Kerberos unterstützt, können Sie sich mit Ihrem Windows Konto anmelden.

Die oben bezeichnete Webseite des Serviceanbieters bittet Sie, sich bei Ihrer Heimateinrichtung anzumelden.

Login mit zweitem Faktor

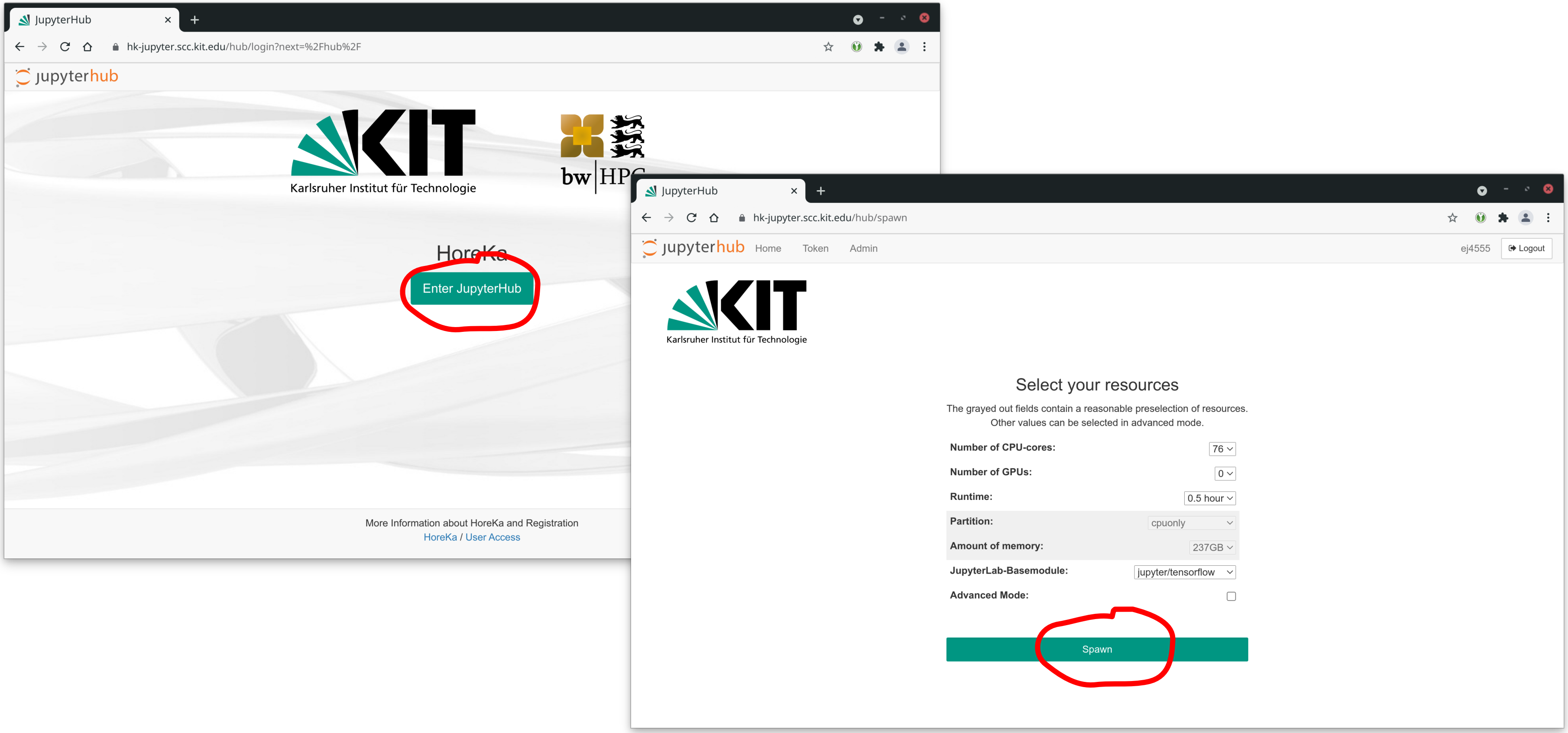
Ihre Token werden an dieser Stelle nur angezeigt. Um Änderungen vorzunehmen besuchen Sie bitte die folgende Webseite: https://my.scc.kit.edu/token

Um die angeforderte Aktion durchzuführen, muss ein zweiter Faktor eingegeben werden. Bitte geben Sie einen beliebigen zweiten Faktor aus der unten stehenden Liste ein um fortzufahren.

Aktueller code

Step-by-Step Jupyter@HoreKa (3/4)

- Click “**Enter JupyterHub**“
- Select resources and click “**Spawn**”



JupyterHub
hk-jupyter.scc.kit.edu/hub/login?next=%2Fhub%2F
jupyterhub
Karlsruher Institut für Technologie
bw|HPC
HoreKa
Enter JupyterHub
More Information about HoreKa and Registration
HoreKa / User Access

JupyterHub
hk-jupyter.scc.kit.edu/hub/spawn
jupyterhub
Home
Token
Admin
ej4555
Logout
Karlsruher Institut für Technologie

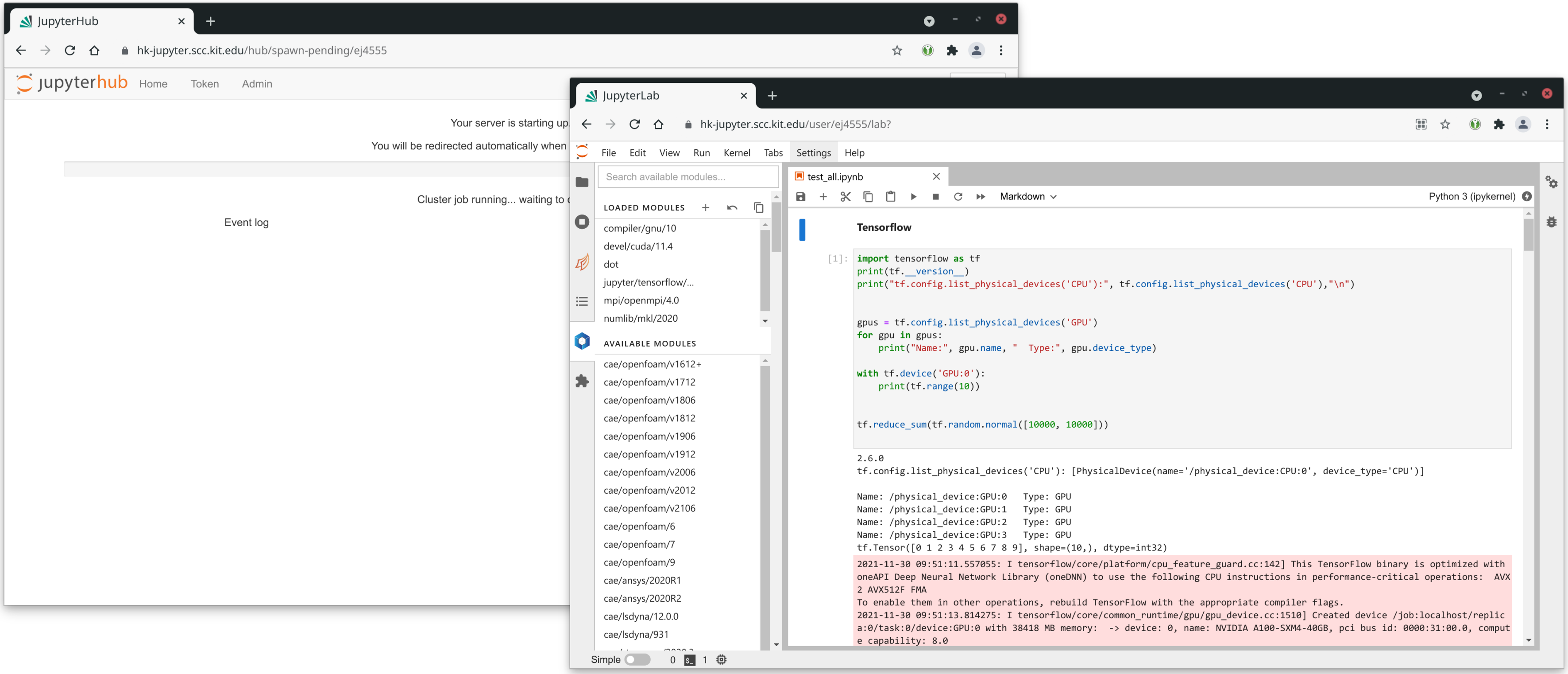
Select your resources

The grayed out fields contain a reasonable preselection of resources.
Other values can be selected in advanced mode.

Number of CPU-cores:	76
Number of GPUs:	0
Runtime:	0.5 hour
Partition:	cpuonly
Amount of memory:	237GB
JupyterLab-Basemodule:	jupyter/tensorflow
Advanced Mode:	□

Step-by-Step Jupyter@HoreKa (4/4)

- Spawning may take a while
  - … timeout after 10 minutes
- JupyterLab runs on compute node on HoreKa



Your server is starting up

You will be redirected automatically when

Cluster job running... waiting to c

Event log

Tensorflow

```
[1]: import tensorflow as tf
     print(tf.__version__)
     print("tf.config.list_physical_devices('CPU'):", tf.config.list_physical_devices('CPU'),"\n")


     gpus = tf.config.list_physical_devices('GPU')
     for gpu in gpus:
         print("Name:", gpu.name, "  Type:", gpu.device_type)

     with tf.device('GPU:0'):
         print(tf.range(10))


     tf.reduce_sum(tf.random.normal([10000, 10000]))
```

```
2.6.0
tf.config.list_physical_devices('CPU'): [PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU')]

Name: /physical_device:GPU:0   Type: GPU
Name: /physical_device:GPU:1   Type: GPU
Name: /physical_device:GPU:2   Type: GPU
Name: /physical_device:GPU:3   Type: GPU
tf.Tensor([0 1 2 3 4 5 6 7 8 9], shape=(10,), dtype=int32)
2021-11-30 09:51:11.557055: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with
oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:  AVX
2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-11-30 09:51:13.814275: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Created device /job:localhost/replic
a:0/task:0/device:GPU:0 with 38418 MB memory:  -> device: 0, name: NVIDIA A100-SXM4-40GB, pci bus id: 0000:31:00.0, comput
e capability: 8.0
```

Hands-On: Login

~10min

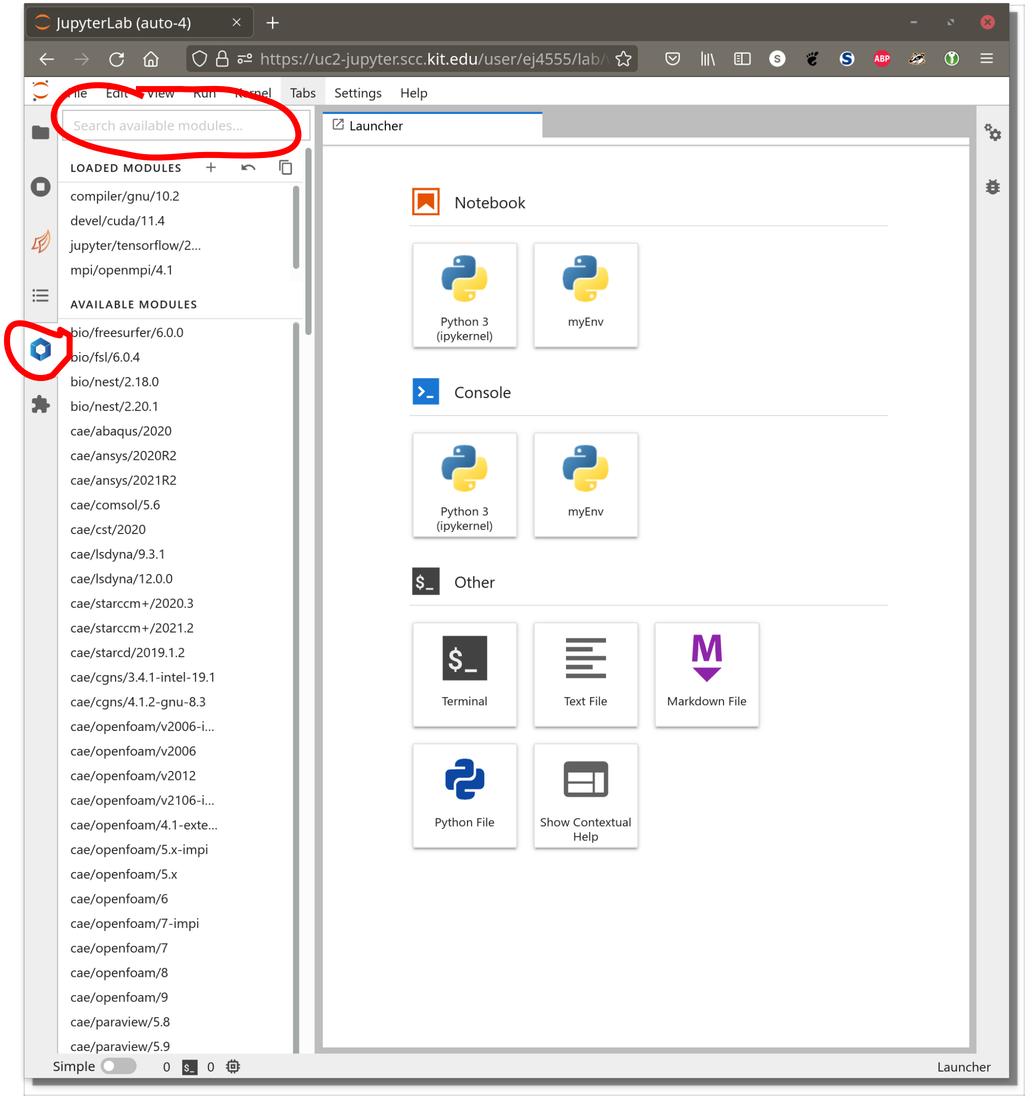
Hands-On: Login

- Start a session at https://hk-jupyter.scc.kit.edu
  - Hint:
    choose “accelerated” partition

Software and Kernels

Select Software

- Activate Lmod software modules
  → blue button
  → search field
- Kernel restart required



Search available modules...

LOADED MODULES

compiler/gnu/10.2
devel/cuda/11.4
jupyter/tensorflow/2...
mpi/openmpi/4.1

AVAILABLE MODULES

bio/freesurfer/6.0.0
bio/fsl/6.0.4
bio/nest/2.18.0
bio/nest/2.20.1
cae/abaqus/2020
cae/ansys/2020R2
cae/ansys/2021R2
cae/comsol/5.6
cae/cst/2020
cae/lsdyna/9.3.1
cae/lsdyna/12.0.0
cae/starccm+/2020.3
cae/starccm+/2021.2
cae/starcd/2019.1.2
cae/cgns/3.4.1-intel-19.1
cae/cgns/4.1.2-gnu-8.3
cae/openfoam/v2006-i...
cae/openfoam/v2006
cae/openfoam/v2012
cae/openfoam/v2106-i...
cae/openfoam/4.1-exte...
cae/openfoam/5.x-impi
cae/openfoam/5.x
cae/openfoam/6
cae/openfoam/7-impi
cae/openfoam/7
cae/openfoam/8
cae/openfoam/9
cae/paraview/5.8
cae/paraview/5.9

Add Python Packages + Custom Kernel

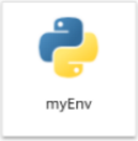
- Python: Use virtual environments

```
python -m venv myEnv
source myEnv/bin/activate
pip install <myPackage>
```

- Install kernel → IPython docs

```
python -m ipykernel install \
     --user \
     --name myEnv \
     --display-name "Python (myEnv)"
```

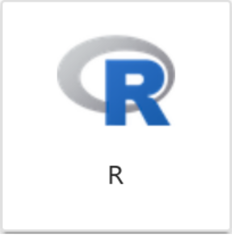
- You will get this:



R and Julia Kernel

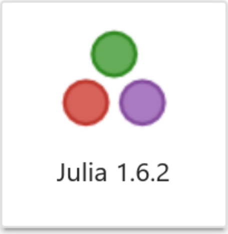
You want to use R

- Load module `math/R`
- Open terminal
- `R`
  - `install.packages('IRkernel')`
  - `IRkernel::installspec()`



You want to use Julia

- Load module `devel/julia/1.6.2`
- Open terminal
- `julia`
  - `]`
  - `add IJulia`



Resetting everything

`rm -r …`

- Jupyter
  - … `~/.local/share/jupyter/kernels`
  - … `~/.jupyter`
- Python packages
  - … `~/.local/lib/python3.*`
- R
  - … `~/R`
- Julia
  - … `~/.julia`

Check state

- Bash
  - `~/.bashrc`

Hands-On: First Steps

~10min

Hands-On: First Steps

- Run the c’t example on HoreKa
  - Hint 1:
    `git clone https://github.com/pinae/BresenhamLidar`
  - Hint 2:
    Replace `%matplotlib notebook` by `%matplotlib widget`

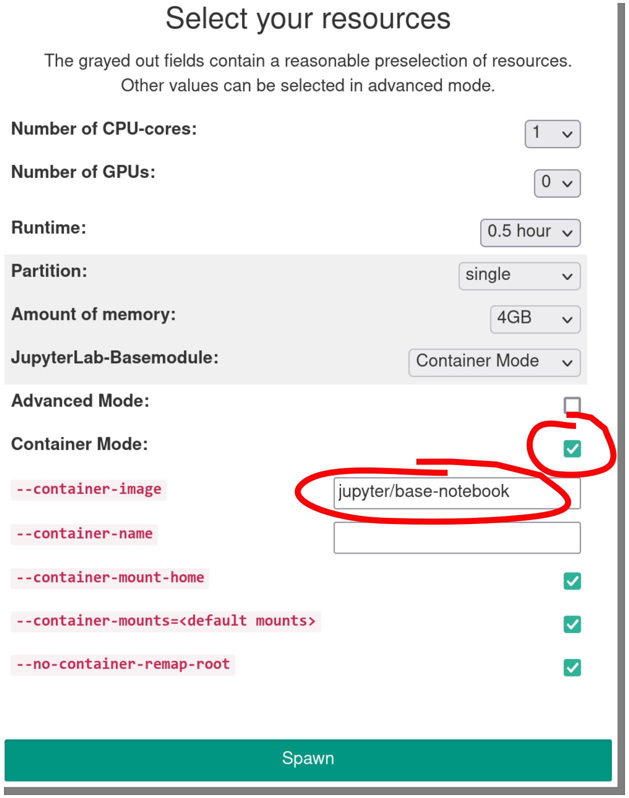
- Install a Julia Kernel
  - Compute `1+1` in a Julia Notebook
  - Try out some examples, e.g.: https://rosettacode.org/wiki/Factorial#Julia

Outlook

WIP: BYO Jupyter Container

- Connect **containerized** Jupyter with JupyterHub@HoreKa
- **Docker images** from any registry
  - For complicated/intrusive software stacks
  - Optimized software stacks
    - Intel, e.g.
      `intel/intel-optimized-tensorflow`
    - Nvidia, e.g.
      `nvcr.io#nvidia/tensorflow:21.10-tf2-py3`
    - AMD, e.g.
      `rocm/tensorflow:rocm4.3.1-tf2.6-dev`
- Possible **root access** (sic!)
  - **Yes, you can**

```
sudo apt-get install <myPackage>
```

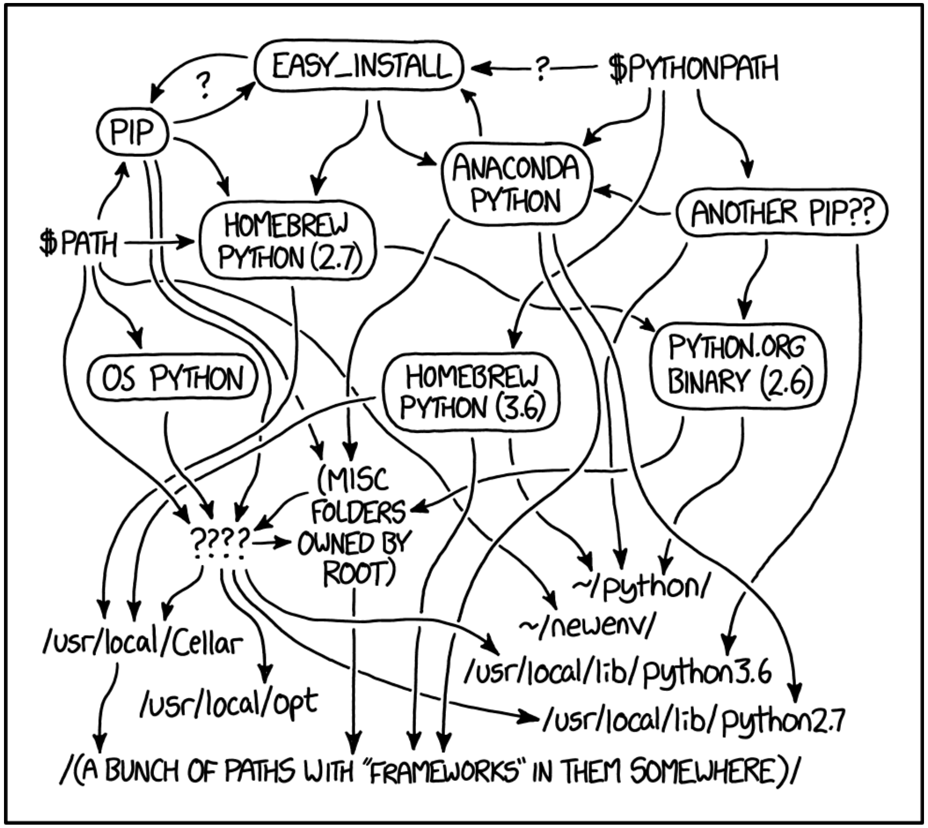


Select your resources

The grayed out fields contain a reasonable preselection of resources.
Other values can be selected in advanced mode.

Number of CPU-cores:	1
Number of GPUs:	0
Runtime:	0.5 hour
Partition:	single
Amount of memory:	4GB
JupyterLab-Basemodule:	Container Mode
Advanced Mode:	□
Container Mode:	☑
--container-image	jupyter/base-notebook
--container-name	
--container-mount-home	☑
--container-mounts=<default mounts>	☑
--no-container-remap-root	☑

Thank you for your attention!
Questions?



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

https://xkcd.com/1987