

Jupyter * @ DKRZ

Interactive (Super) computing on Mistral



Dr. Sofiane Bendoukha

Deutsches Klimarechenzentrum (DKRZ)



* : [hub/notebooks/lab/kernels](https://hub.notebooks/lab/kernels)

About me

- **Background:** Computer science
- **Department:** Application Support
- **Currently:**
 - backend developer → interactive computing
 - containers

Introduction

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
```

What is, Why Jupyter? (1)

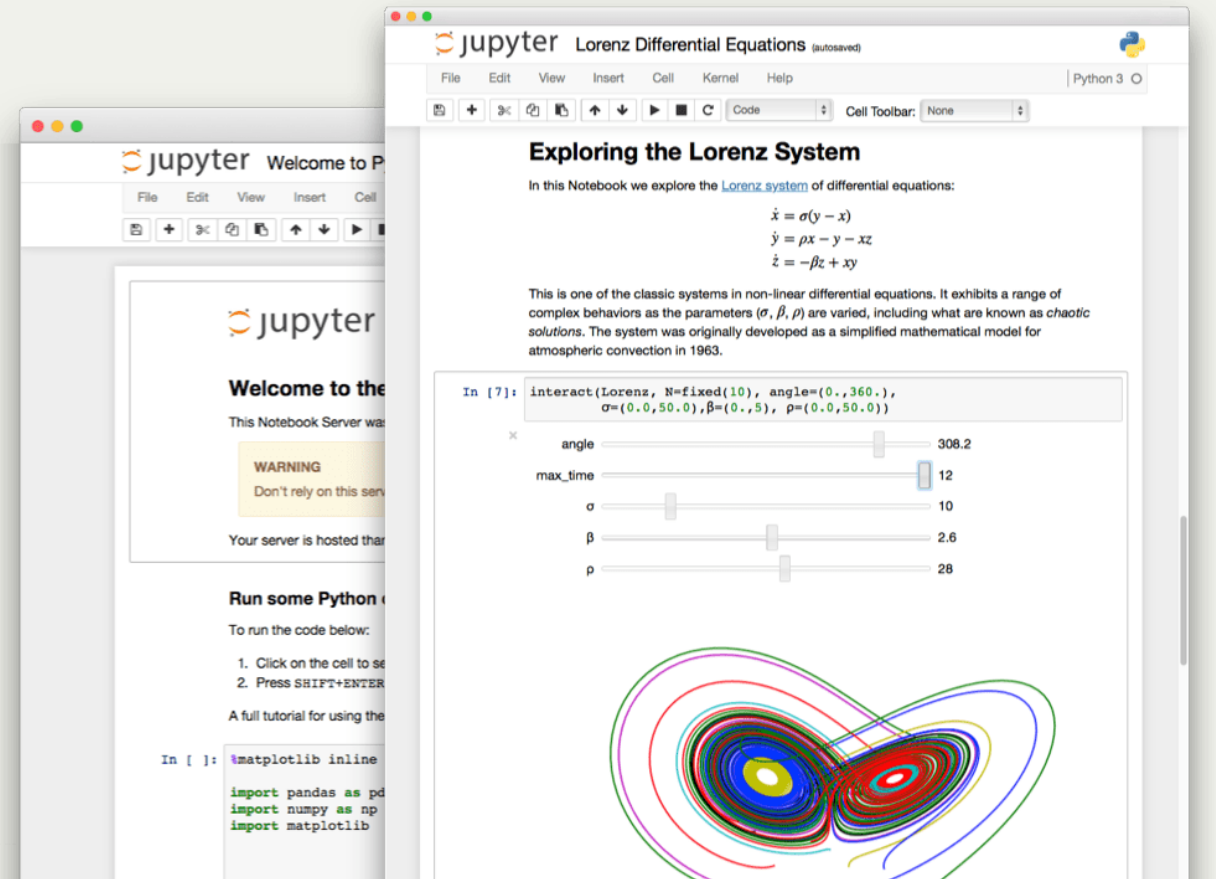
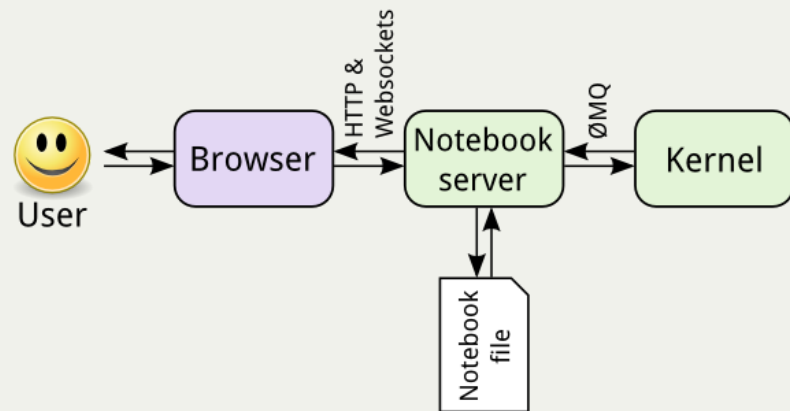
Python is popular

Aug 2020	Aug 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	16.98%	+1.83%
2	1	▼	Java	14.43%	-1.60%
3	3		Python	9.69%	-0.33%
4	4		C++	6.84%	+0.78%

<https://www.tiobe.com/tiobe-index/>

Programming Language	2020	2015	2010	2005	2000	1995	1990	1985
Java	1	2	1	2	3	-	-	-
C	2	1	2	1	1	2	1	1
Python	3	7	6	6	22	20	-	-

What is, Why Jupyter? (2)



web application that allows you to **create** and **share** documents that contain **live code**, equations, **visualizations**.

Jupyter Notebook on HPC

- **Local:**

Using Anaconda, anyone can install and run Jupyter Notebooks on their local computer.

- **HPC:** Infrastructure-specific

- security concerns
- shared file systems
- metadata transactions

Jupyter Notebook on HPC

- **Local:**

Using Anaconda, anyone can install and run Jupyter Notebooks on their local computer.

- **HPC: Infrastructure-specific**

- security concerns
- shared file systems
- metadata transactions

```
1. SSH setup
2. Jupyter notebook setup
3. SSH to the remote system and start Jupyter notebook
4. Start Jupyter notebook with --no-browser and --port
5. Create an SSH "local port forward"
6. Open Jupyter notebook with your "Local" browser
```

What we provide @ DKRZ

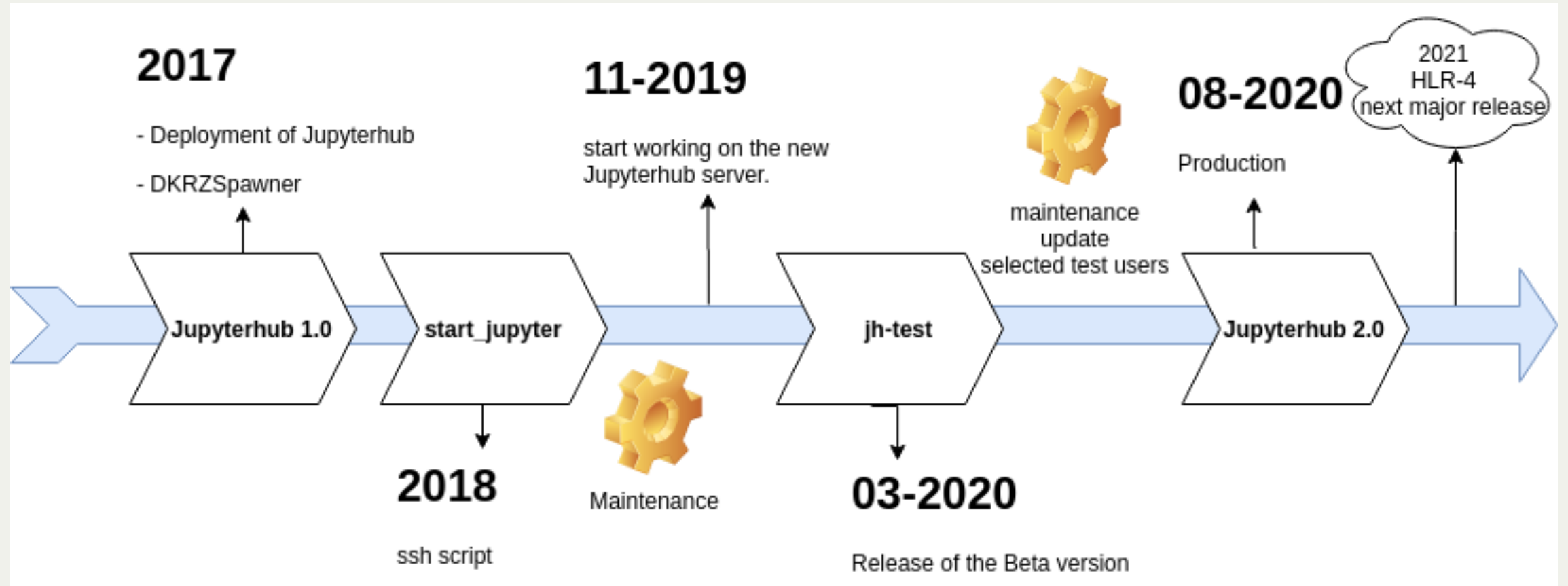
What we provide @ DKRZ

- **Convenient way: *Jupyterhub***
 - user-friendly
 - full user support
 - continuous maintenance and update

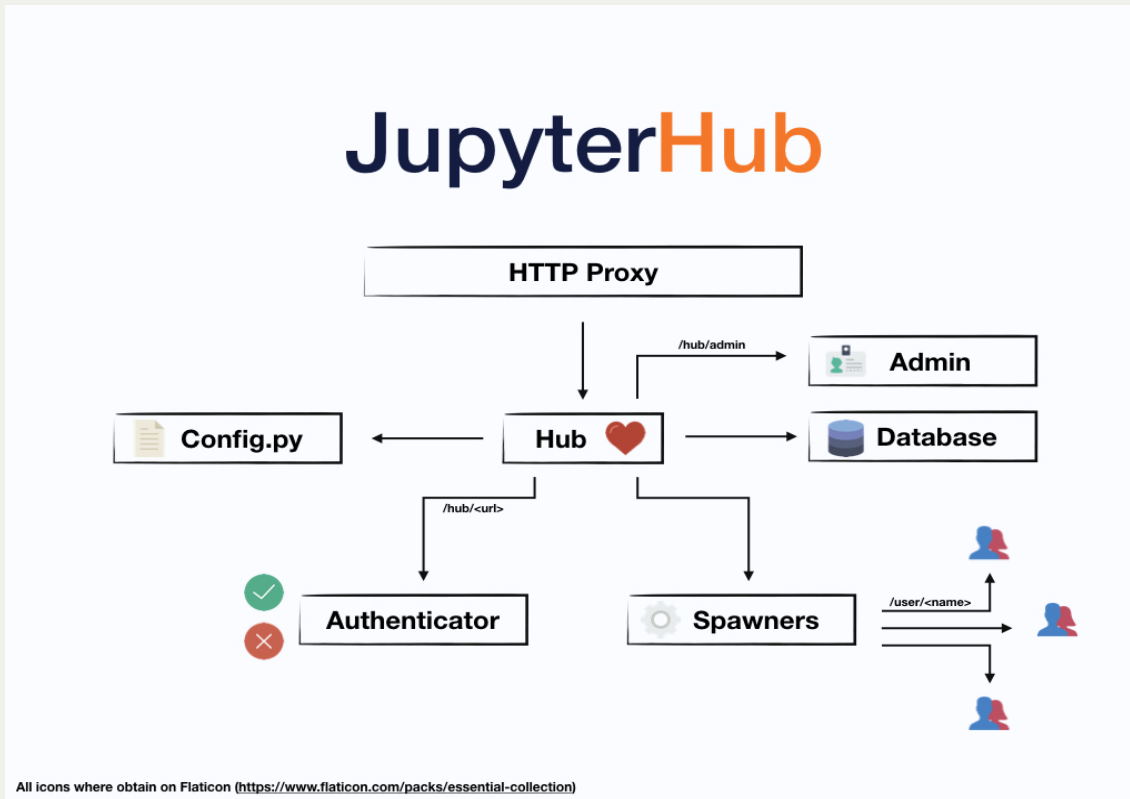
What we provide @ DKRZ

- **Convenient way:** *Jupyterhub*
 - user-friendly
 - full user support
 - continuous maintenance and update
- **Old school:** *Single Jupyter notebooks* (ssh based)
 - `./start_jupyter [Options]`
 - limited support (?)
 - port forwarding can be annoying
 - **Jupyterhub advanced spawner?** 🤔

Roadmap



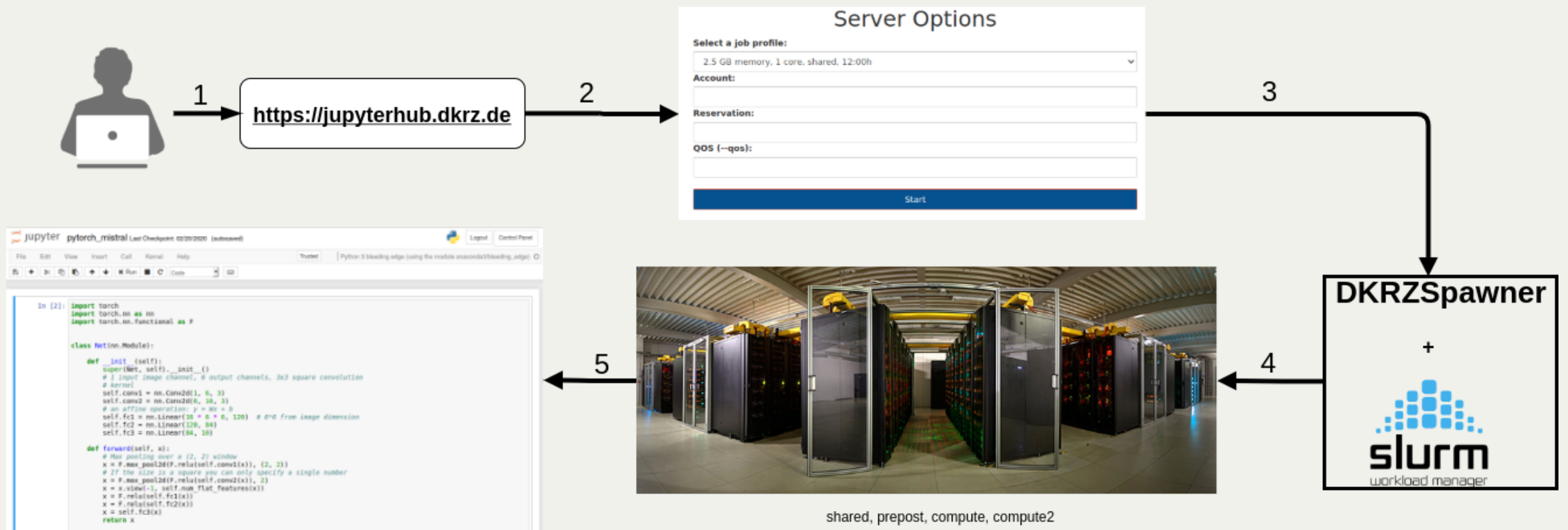
How it works?



- *Manages authentication*
- *Spawns single-user notebook servers on-demand*
- *Gives each user a complete server*

How it works on Mistral?

Spawning workflow



GUI

The screenshot shows a web browser window with the address bar displaying `jupyterhub.dkrz.de`. The page features the DKRZ logo (Deutsches Klimarechenzentrum) in the top left and navigation links for Slurm, Mistral, and Documentation in the top right. A large image of a modern building is on the left. The main content area includes the JupyterHub logo, a welcome message, a sign-in prompt for DKRZ accounts, and a login form with fields for Username and Password, and a Sign In button. The footer contains links for Contact, Legal notice, and Privacy policy, along with the copyright notice for Deutsches Klimarechenzentrum.

DKRZ
DEUTSCHES
KLIMARECHENZENTRUM

Slurm Mistral Documentation

Go to DKRZ home

Welcome to Jupyterhub @ DKRZ

Jupyterhub is a multi-user server to serve Jupyter Notebooks to a large number of users. It is integrated with our Mistral's batch scheduling system to allocate computing resources and launch Jupyter Notebooks directly on the HPC system. It therefore also supports the execution of parallel computation.

Sign in with your DKRZ account

[Forgot your password?](#) [First time user?](#)

Username:

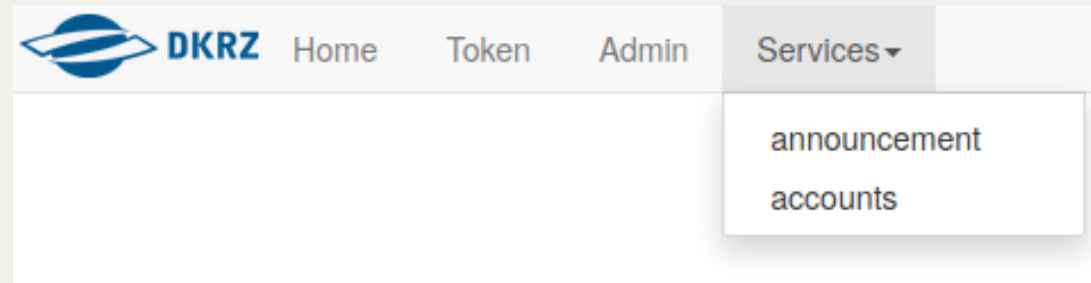
Password:

[Sign In](#)

[Contact](#) [Legal notice](#) [Privacy policy](#)

© Deutsches Klimarechenzentrum

REST Web services



Announcements (Admin)

Announcement

The prepost partition is in high demand right now. Average queueing time for prepost currently is 16 hours while on compute or compute2 it is well below one hour. Please check if your job can be run on the compute or compute2 partitions and switch if possible. These partitions also contain nodes with 128G of memory and more.

Accounts

Project/Account	QoS (comma separated)	* normal is the default qos and will have no effect when spawning your notebook.
bm0146	express	
k20200	express,normal,training	
bmX825	express	
Available features (partition, feature)		
compute,256G		
compute,64G		
compute,128G		
prepost,256G		
shared,64G		
gpu,m40,1024G		

Documentation

Overview Quick Start Spawner options Kernels Systems Tutorials Changelog DKRZ User Portal

Search the docs ...

Welcome to Jupyterhub @ DKRZ!

Jupyterhub is a multi-user server for Jupyter notebooks that allows to execute the notebook directly on the DKRZ HPC system Mistral. It therefore also supports the execution of parallel computation.

On this page

- Welcome to Jupyterhub @ DKRZ!
- Indices and tables

Technical documentation
Please consult the technical documentation to get started with Jupyterhub.

Note
JupyterHub is available at <https://jupyterhub.dkrz.de> for all DKRZ users who have access to Mistral and who are allowed to submit batch jobs.

Warning
DKRZ operates the jupyterhub server in extended testing phase, which means that we do not guarantee fulltime availability of the system and some more limitations documented [here](#).

Any input/request is welcome!

JupyterLab (1)

The screenshot displays the JupyterLab interface with a dark theme. On the left is a file browser showing a directory structure with files like 'conda-envs', 'dask-worker-space', and 'machine-learning.ipynb'. The main area contains a code editor for 'machine-learning.ipynb' with the following code and output:

```
[1]: from dask.distributed import Client, progress
```

```
[10]: client = Client(processes=False, threads_per_worker=4,
                  n_workers=3, memory_limit='2GB')
      client
```

Client	Cluster
Scheduler: Inproc://10.50.64.54/25547/68	Workers: 3
Dashboard: [redacted]	Cores: 12
	Memory: 3.22 GB

```
[11]: from sklearn.datasets import make_classification
      from sklearn.svm import SVC
      from sklearn.model_selection import GridSearchCV
      import pandas as pd
```

```
[12]: X, y = make_classification(n_samples=1000, random_state=0)
      X[:5]
```

```
[12]: array([[ -1.06377997,  0.67640868,  1.06935647, -0.21758002,  0.46021477,
            -0.39916689, -0.07918751,  1.20938491, -0.78531472, -0.17218611,
            -1.08535744, -0.99311895,  0.30693511,  0.06405769, -1.0542328 ,
```

On the right side, there are three panels: 'Dask Graph', 'Dask Workers', and 'Dask Task Stream', all of which are currently empty.

JupyterLab (2)

switching between classic/lab

The screenshot shows a web browser window with the address bar displaying `jh-test.dkrz.de:8000/user/k204213/preset/tree/home/dkrz/k204213`. The browser's address bar includes navigation icons (back, forward, refresh, home) and a list of bookmarks: Most Visited, DKRZ, Python, Github, Dask for Parallel Com..., JupyterHub, JupyterHub, JupyterHub, AAI, JupyterHub, and Slurm. The DKRZ logo is visible in the top left of the page content. Below the logo, there are tabs for 'Files', 'Running', 'Clusters', and 'Nbextensions'. A message says 'Select items to perform actions on them.' Below this is a file browser interface showing a directory listing for `/home/dkrz/k204213`. The listing includes a search bar with '0' and a 'Name' dropdown. The files and folders listed are: `..`, `intake-esm`, `sing_cache`, `sing_tmp`, `singularity`, `DYMOND_BENCHMARK.ipynb`, `pytorch_mistral.ipynb`, `g300043_python.txt`, `jupyter_preload`, `jupyterhub_slurmspawner_20072599.log`, `jupyterhub_slurmspawner_20150913.log`, `jupyterhub_slurmspawner_20150958.log`, `jupyterhub_slurmspawner_20461470.log`, `jupyterhub_slurmspawner_20461597.log`, and `jupyterhub_slurmspawner_20462764.log`.

Agenda

- ~~Introduction~~
- Spawning process
- Kernels
- Extensions
- Q & A

Spawner options

Overview



DKRZ
DEUTSCHES
KLIMARECHENZENTRUM

[Home](#)

[Token](#)

[Admin](#)

[Services](#) ▾

[Slurm](#)

[Mistral](#)

[Documentation](#)

[Logout](#)

Spawner Options

Preset

Advanced

Preset profiles

[start from preset profiles](#)

Preset options form

Choose from the list

Available profiles

Server Options

Select a job profile:

2.5 GB memory, 1 core, shared, 12:00h

Account (--account):

Reservation (--reservation):

QoS (--qos):

Start

Select a job profile:

- ✓ 2.5 GB memory, 1 core, shared, 12:00h
- 5 GB memory, 2 cores, shared, 12:00h
- 10 GB memory, 4 cores, shared, 12:00h
- 64 GB memory, 24 cores, 1 node, compute, 8:00h
- 5 GB memory, 1 core, prepost, 12:00h
- 10 GB memory, 2 core, prepost, 12:00h
- 20 GB memory, 4 core, prepost, 12:00h
- 256 GB memory, 24 core, 1 node, prepost, 12:00h
- 5 GB memory, 1 core, miklip, 12:00h

Advanced options form (1)

Server Options

Account (--account)
bm0146

Partition (--partition)
Shared

Reservation (--reservation) None **Time (hours) (--time)** 1.00

Number of cores (--cpus-per-task) 1 **Memory (MB) (--mem)** 1024

QoS (--qos)

Log File Name
jupyterhub_slurmspawner

Request Features/Constraints (--constraint)

User interface
Jupyter Notebook

Start

Annotations:

- Display a list of your accounts (points to Account dropdown)
- List of available partitions (points to Partition dropdown)
- Partition QoS (points to QoS text input)
- You can rename the log file (points to Log File Name text input)
- SLURM features related to partitions (points to Request Features/Constraints dropdown)
- Notebook
- Jupyter Lab
- Terminal (points to User interface dropdown)

Advanced options form (2)

Parameter	Mandatory	Description
Account	Yes	Project that should be charged
Partition	Yes	Partition to run the job
Reservation	No	Resources reserved for certain time/accounts
Time	No (Default: 1 Hour)	The maximum amount of time your job can take before Slurm forcefully kills it.
Number of cores	No (Default: 1 core)	Number of threads (logical cores) per task.
Memory	No (Default: 1024 MB)	The total amount of RAM to allocate.
QoS	No	Quality of Service often puts the job in high priority queue (e.g. training).
Log File Name	No (Default: <code>jupyterhub_slurm-spawner_advanced</code>)	Notebook log
Request Features	No	Node-features requested for the job.
User interfaces	No (Default: notebook)	Notebook/Lab/Terminal

Advanced options form (3)

Server Options

Error: sbatch: error: CPU count per node can not be satisfied sbatch: error: Batch job submission failed: Requested node configuration is not available

Start

Server Options

Error: sbatch: error: you have to specify less than 48 CPUs for 'shared' partition - current specification is 72 sbatch: error: Batch job submission failed: Error generating job credential

Start

Number of cores (per task)

Value must be less than or equal to 72

73

Memory (MB)

Value must be less than or equal to 1024000

5000000

Advanced options form (3)

Server Options

Error: sbatch: error: CPU count per node can not be satisfied sbatch: error: Batch job submission failed: Requested node configuration is not available

Start

Server Options

Error: sbatch: error: you have to specify less than 48 CPUs for 'shared' partition - current specification is 72 sbatch: error: Batch job submission failed: Error generating job credential

Start

Number of CPUs (per task)

Value must be less than or equal to 72

Memory (MB)

Value must be less than or equal to 1024000

Named servers

Server Options

Select a job profile:

2.5 GB memory, 1 core, shared, 12:00h

Account:

Reservation:

QOS (--qos):

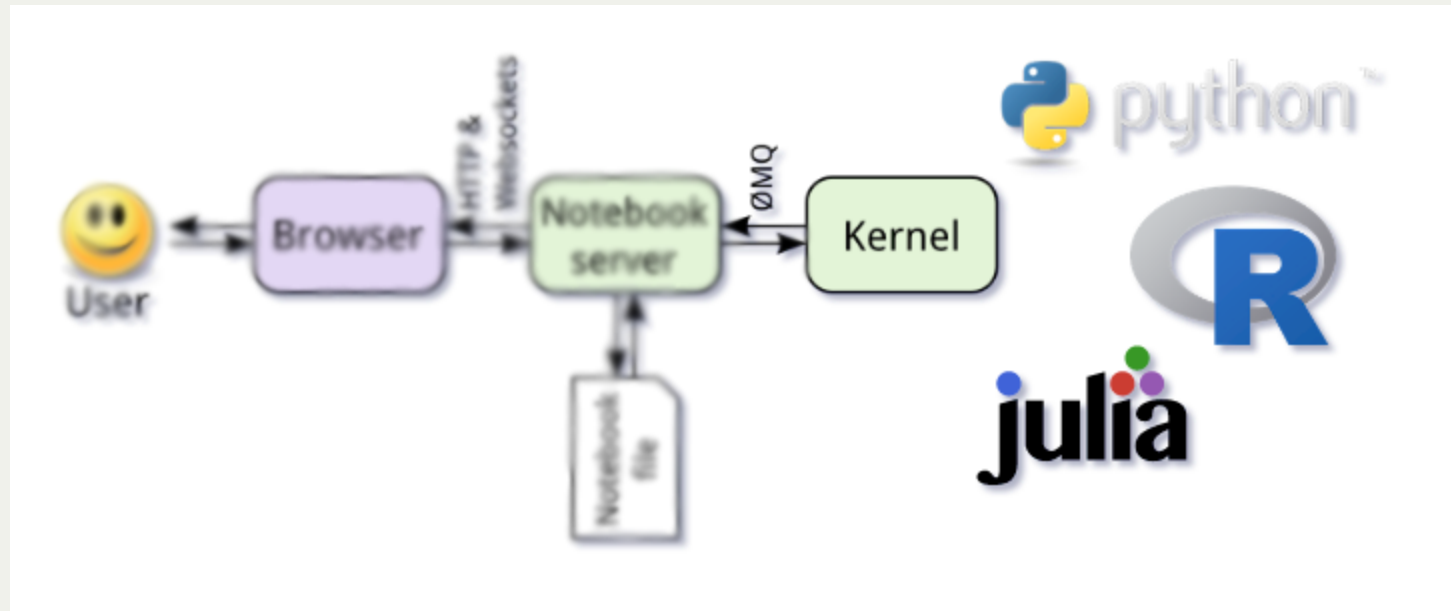
Start

- *Named servers allow you to have more than one server running in the same time*
- *Currently 2 allowed: **preset** and **advanced***
- *Extendable --> HLRE 4 --> more*

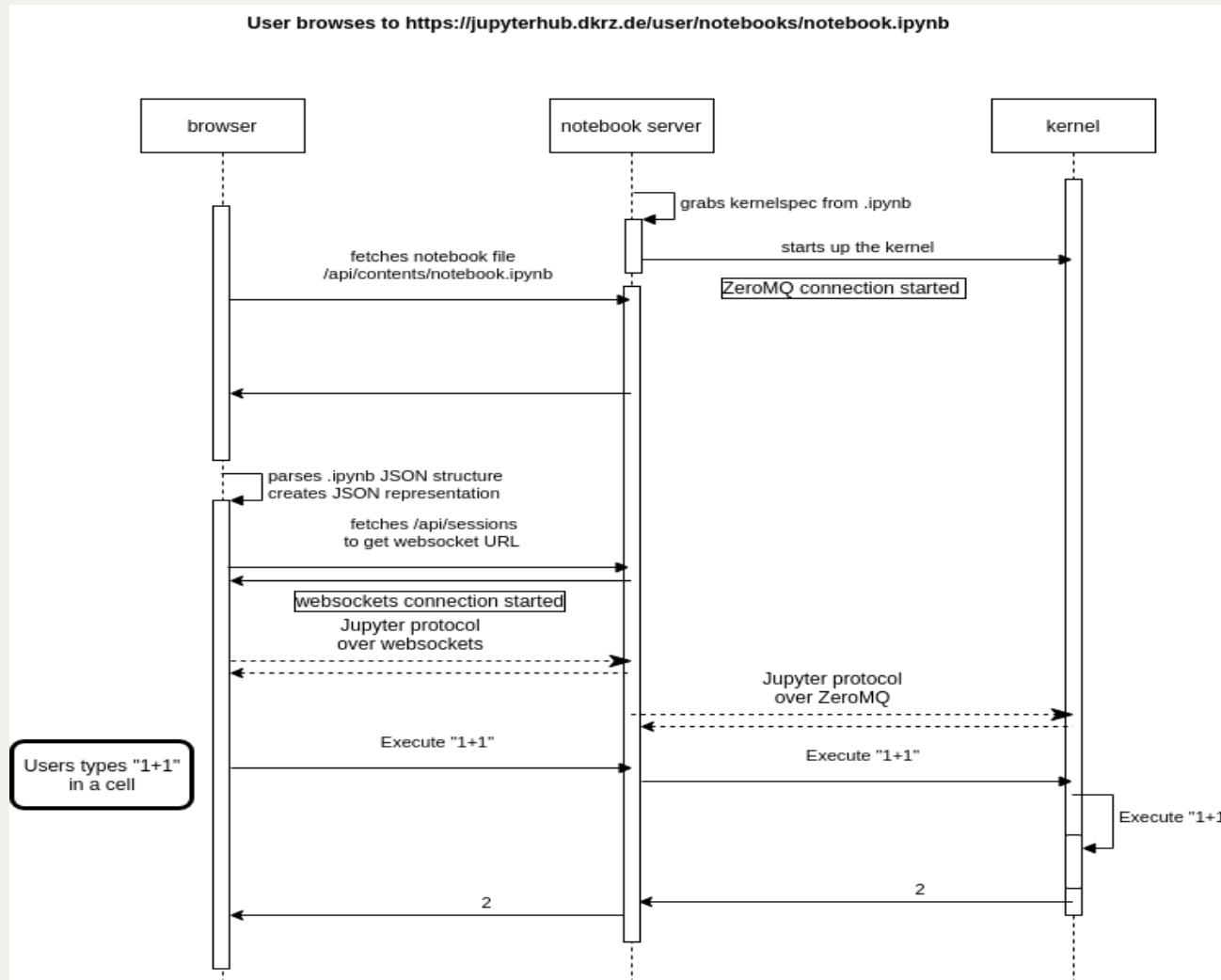
Stop your server

The screenshot shows a web browser window with the URL `https://jh-test.dkrz.de/user/k204213/preset/lab`. The browser tabs include "Discover - Kibana", "JupyterLab", "home/dkrz/k204213/", and "Overview — JupyterHub". The browser's address bar shows the URL and navigation icons. Below the browser window, the JupyterLab interface is visible. On the left, there is a file browser showing a directory structure with files like `dask-worker-space`, `intake-esm`, `sing_cache`, `sing_tmp`, `singularity`, `DYMOND_BENCHMARK.ipynb`, `g300043_python.txt`, `jupyter_preload`, `pytorch_mistral.ipynb`, `test.ipynb`, and `tutorial.ipynb`. The main area is titled "Launcher" and shows the path `home/dkrz/k204213`. It is divided into three sections: "Notebook", "Console", and "Other". The "Notebook" section contains five Python environment icons: "Python 2 (using the module)", "Python 2 bleeding edge", "Python 3 (using the module)", "Python 3 bleeding edge", and "Python 3 unstable (using)". The "Console" section also contains five identical Python environment icons. The "Other" section contains four icons: "Terminal", "Text File", "Markdown File", and "Show Contextual Help".

Kernels



What is happening in the background



Default system kernels

Kernel	Source module
Python 2	python/2.7.12
Python 2	anaconda2/bleeding_edge
Python 3	anaconda3/bleeding_edge
<i>Python 3 unstable</i>	<i>python3/unstable</i>

"We will have a new Python about every half year, which is going to be called 'python3/YYYY.MM-compiler-version'. The first one is 'python3/2020.02-gcc-9.1.0'".

Bring your own environment (1)

- Conda

```
% mkdir $HOME/kernels
% conda create --prefix $HOME/kernels/tensorflow ipykernel python=3.x
% source activate $HOME/kernels/tensorflow
% python -m ipykernel install --user --name tensorflow --display-name="ten
% conda deactivate
```

- Virtualenv

```
% python -m pip install --user virtualenv
% python -m virtualenv --system-site-packages /path/to/new-kernel
% source /path/to/new-kernel/bin/activate
% pip install ipykernel
% python -m ipykernel install --user --name new-kernel --display-name="new
```

Bring your own environment (2)

Default

```
{
  "argv": [
    "/path/to/kernel/bin/python",
    "-m",
    "ipykernel_launcher",
    "-f",
    "{connection_file}"
  ],
  "display_name": "new-kernel",
  "language": "python"
}
```

Customized

```
{
  "argv": [
    "/path/to/kernel/bin/python",
    "-m",
    "ipykernel_launcher",
    "-f",
    "{connection_file}"
  ],
  "display_name": "new-kernel",
  "language": "python",
  "env": {
    "variable": "value",
  }
}
```

Bring your own environment (2)

Default

```
{
  "argv": [
    "/path/to/kernel/bin/python",
    "-m",
    "ipykernel_launcher",
    "-f",
    "{connection_file}"
  ],
  "display_name": "new-kernel",
  "language": "python"
}
```

Customized

```
{
  "argv": [
    "/path/to/kernel/bin/python",
    "-m",
    "ipykernel_launcher",
    "-f",
    "{connection_file}"
  ],
  "display_name": "new-kernel",
  "language": "python",
  "env": {
    "variable": "value",
  }
}
```

more advanced: use executable scripts! --> check the doc

Using/Changing the kernels

The image shows the JupyterLab interface with a file browser on the left and a main workspace on the right. The file browser displays a directory structure with files like `conda-envs`, `jupyterhub_module`, `kernels`, `notebooks`, `singularity`, `dask-jobqueue.ipynb`, `requirements.txt`, and `tensorflow_2.0.0a0-gpu-py3.sif`. A context menu is open over the `dask-jobqueue.ipynb` file, showing options: `Interrupt Kernel`, `Restart Kernel...`, `Restart Kernel and Clear All Outputs...`, `Restart Kernel and Run All Cells...`, `Shut Down Kernel`, `Shut Down All Kernels...`, and `Change Kernel...`. The text **Change kernel** is overlaid on the menu.

The main workspace shows the **Available kernels** section with a yellow header. It contains six kernel icons: `Python 3`, `new-kernel`, `Python 2 (using the module)`, `Python 2 bleeding edge`, `Python 3 (using the module)`, and `Python 3 bleeding edge`. Below this is a **Console** section with a **Select Kernel** dialog box. The dialog box has a title **Select** and a dropdown menu showing `Python 3`. Below the dropdown are three sections: **Start Preferred Kernel** (with `Python 3` selected), **Use No Kernel** (with `No Kernel`), and **Start Other Kernel** (with `new-kernel`, `Python 2 (using the module python/2.7.12)`, `Python 2 bleeding edge (using the module anaconda2/bleeding_edge)`, `Python 3 (using the module python/3.5.2)`, and `Python 3 bleeding edge (using the module anaconda3/bleeding_edge)` listed).

Using/Changing the kernels

The image shows a screenshot of the Classic Jupyter web interface. At the top, the title "Classic Jupyter" is displayed. Below the title, there are buttons for "Upload", "New", and a refresh icon. A large text overlay "New -->" points to the "New" button. A dropdown menu is open, showing options for "Notebook" and "Other". The "Notebook" section lists several Python kernels: "Python 2 (using the module python/2.7.12)", "Python 2 bleeding edge (using the module anaconda2/bleeding_edge)", "Python 3 (using the module python/3.5.2)", "Python 3 bleeding edge (using the module anaconda3/bleeding_edge)", "Python 3 unstable (using the module python3/unstable)", and "work-env". The "Other" section lists "Text File", "Folder", and "Terminal". Below the menu, a table shows a list of notebooks, including one named "jobque" with a size of "30.1 kB" and a creation time of "3 months ago".

New -->

Cell Kernel Help

Run Interrupt Restart Restart & Clear Output Restart & Run All Reconnect Shutdown Change kernel

Notebook:

- Python 2 (using the module python/2.7.12)
- Python 2 bleeding edge (using the module anaconda2/bleeding_edge)
- Python 3 (using the module python/3.5.2)
- Python 3 bleeding edge (using the module anaconda3/bleeding_edge)
- Python 3 unstable (using the module python3/unstable)
- work-env

Other:

- Text File
- Folder
- Terminal

jobque 3 months ago 30.1 kB

<-- Change kernel

Python 2 (using the module python/2.7.12)

Python 2 bleeding edge (using the module anaconda2/bleeding_edge)

Python 3

Python 3 (using the module python/3.5.2)

Python 3 bleeding edge (using the module anaconda3/bleeding_edge)

new-kernel

```
mem_pos
, line in
f(' --men
return 1
SLURMcluster(cores=2,
               memory="10GB",
               processes=4,
               queue="compute",
               project="k20200",
               interface="ib0")
```

Debugging

Debugging

- preset spawner
 - **jupyterhub_slurmspawner_preset_{slurm_job_id}.log**

Debugging

- preset spawner
 - **jupyterhub_slurmspawner_preset_{slurm_job_id}.log**
- advanced spawner
 - default: **jupyterhub_slurmspawner_advanced_{slurm_job_id}.log**
 - customized: **you_name_it.log**

Debugging

- preset spawner
 - **jupyterhub_slurmspawner_preset_{slurm_job_id}.log**
- advanced spawner
 - default: **jupyterhub_slurmspawner_advanced_{slurm_job_id}.log**
 - customized: **you_name_it.log**



Extensions

Jupyter extensions (1)

Nbextensions

The screenshot shows the Jupyter Nbextensions configuration page. At the top, there are navigation tabs: Files, Running, Clusters, and Nbextensions. Below the tabs is a title "Configurable nbextensions" and a checkbox to "disable configuration for nbextensions without explicit compatibility (they may break your notebook environment, but can be useful to show for nbextension development)". A search filter box is present with the text "filter: by description, section, or tags". The main area contains a grid of 40 extension options, each with a checkbox. Some options are highlighted in yellow, and two are checked.

Extension Name	Checked
<input type="checkbox"/> (some) LaTeX environments for Jupyter	
<input type="checkbox"/> Autopep8	
<input type="checkbox"/> Code Font Size	
<input type="checkbox"/> CodeMirror mode extensions	
<input type="checkbox"/> datestamper	
<input type="checkbox"/> Exercise	
<input type="checkbox"/> Gist-it	
<input type="checkbox"/> Hide input all	
<input type="checkbox"/> Initialization cells	
<input type="checkbox"/> jupytermod/main	
<input type="checkbox"/> 2to3 Converter	
<input type="checkbox"/> AutoSaveTime	
<input type="checkbox"/> Code prettify	
<input type="checkbox"/> Collapsible Headings	
<input type="checkbox"/> Equation Auto Numbering	
<input type="checkbox"/> Exercise2	
<input type="checkbox"/> Help panel	
<input type="checkbox"/> Highlight selected word	
<input type="checkbox"/> isort formatter	
<input type="checkbox"/> Keyboard shortcut editor	
<input type="checkbox"/> AddBefore	
<input type="checkbox"/> Autoscroll	
<input type="checkbox"/> Codefolding	
<input type="checkbox"/> Comment/Uncomment Hotkey	
<input type="checkbox"/> ExecuteTime	
<input type="checkbox"/> Export Embedded HTML	
<input type="checkbox"/> Hide Header	
<input type="checkbox"/> highlighter	
<input checked="" type="checkbox"/> jupyter_server_proxy/tree	
<input type="checkbox"/> Launch QTConsole	
<input type="checkbox"/> appmode/main	
<input type="checkbox"/> Cell Filter	
<input type="checkbox"/> Codefolding in Editor	
<input checked="" type="checkbox"/> contrib_nbextensions_help_item	
<input type="checkbox"/> Execution Dependencies	
<input type="checkbox"/> Freeze	
<input type="checkbox"/> Hide input	
<input type="checkbox"/> Hinterland	
<input checked="" type="checkbox"/> jupyter_tensorboard/tree	
<input type="checkbox"/> Limit Output	

Jupyter extensions (2)

Enabling/Disabling extensions

Enable:

```
jupyter nbextension enable <nbextension require path>
```

Example:

```
jupyter nbextension enable appmode/main
```

Disable:

```
jupyter nbextension disable <nbextension require path>
```

Jupyter extensions (3)

Dask Labextension

The screenshot shows the JupyterLab interface with the Dask Labextension. The central code editor contains the following Python code:

```
[ ]: from dask.distributed import Client, progress

[ ]: client = Client(processes=False, threads_per_worker=4,
                  n_workers=3, memory_limit='2GB')
    client

[ ]: from sklearn.datasets import make_classification
    from sklearn.svm import SVC
    from sklearn.model_selection import GridSearchCV
    import pandas as pd

[ ]: X, y = make_classification(n_samples=1000, random_state=0)
    X[:5]

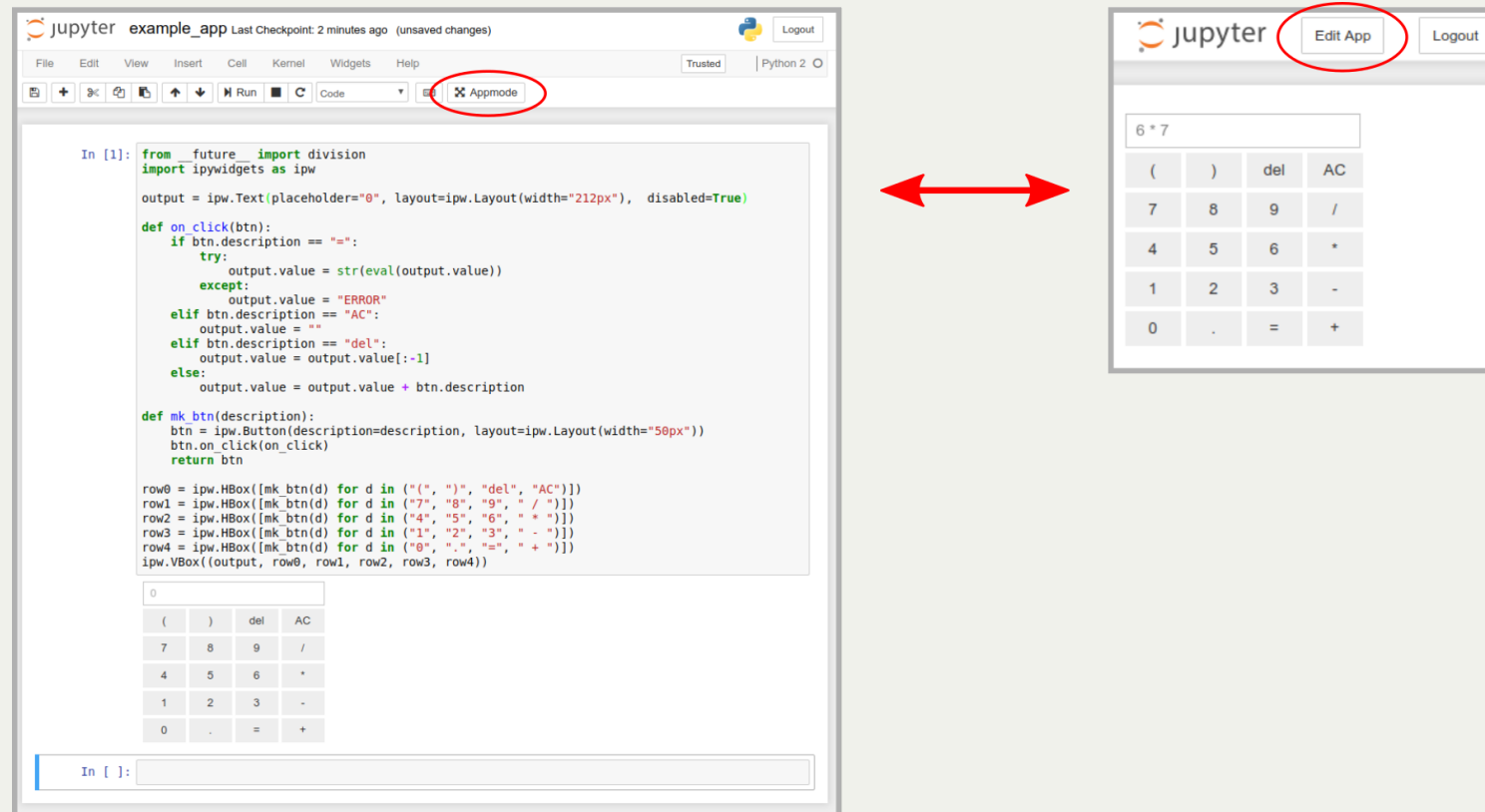
[ ]: param_grid = {"C": [0.001, 0.01, 0.1, 0.5, 1.0, 2.0, 5.0, 10.0],
                  "kernel": ['rbf', 'poly', 'sigmoid'],
                  "shrinking": [True, False]}

    grid_search = GridSearchCV(SVC(gamma='auto', random_state=0, probability=True)
                              param_grid=param_grid,
                              return_train_score=False,
                              iid=True,
                              cv=3,
```

The interface also features a sidebar with monitoring tools such as 'AGGREGATE TIME PER ACTION', 'COMPUTE TIME PER KEY', 'BANDWIDTH WORKERS', 'BANDWIDTH TYPES', 'PROFILE', 'GRAPH', 'MEMORY BY KEY', 'WORKERS', 'TASK STREAM', 'PROFILE SERVER', 'CPU', 'NPROCESSING', 'PROGRESS', 'NBYTES', and 'CLUSTER MAP'. The bottom panel shows 'Dask Workers' and 'Dask Task Stream'.

Jupyter extensions (4)

<https://github.com/oschuett/appmode>

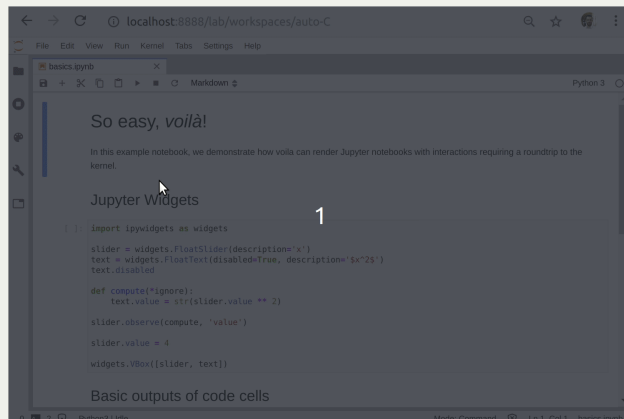


A Jupyter extensions that turns notebooks into web applications.

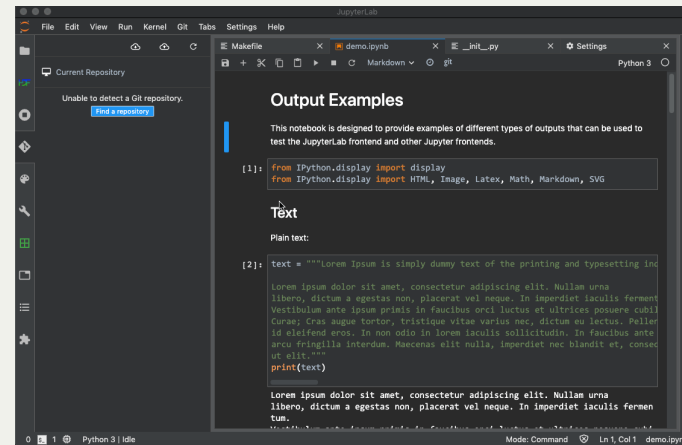
Jupyter extensions (5)

Do you need more?

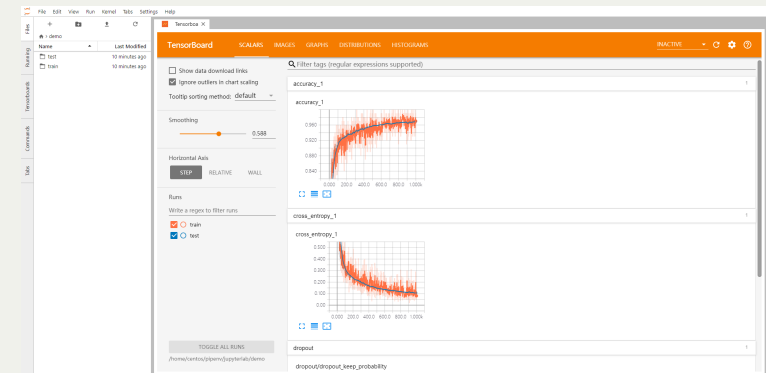
Voilà



JupyterLab Git



TensorBoard



... support@dkrz.de

“The JupyterLab development team is excited to have a robust third-party extension community. However, we do not review third-party extensions, and some extensions may introduce security risks or contain malicious code that runs on your machine.”

Future work

1. Sharing services/extensions
 - shared notebooks repositories (Git/hub/lab)
 - external sharing services
 - WPS
2. Enhanced spawning queue for Jupyterhub
3. More dedicated system kernels and extensions (e.g. ML)
4. Speed up loading Python packages (HLR4)
5. Containerize Jupyterhub
6. Binder for Mistral?

Feedback? Questions?



- Jupyterhub @ DKRZ
- Technical documentation
- support@dkrz.de