

Introduction to



PyTorch

Étienne Plésiat

(DKRZ)

ML Workshop March 5, 2024

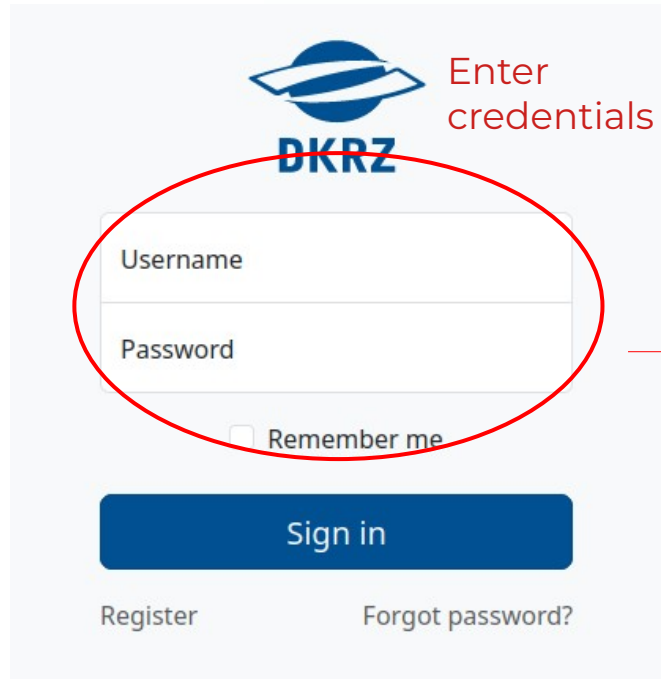
Overview

- **Setup** of the Hands-on
- What is **PyTorch**?
- Hands-on: basics of PyTorch
- Hands-on: code example

Setup the PyTorch hands-on

Setup - Step 1

Go to <https://luv.dkrz.de>



Enter credentials

DKRZ

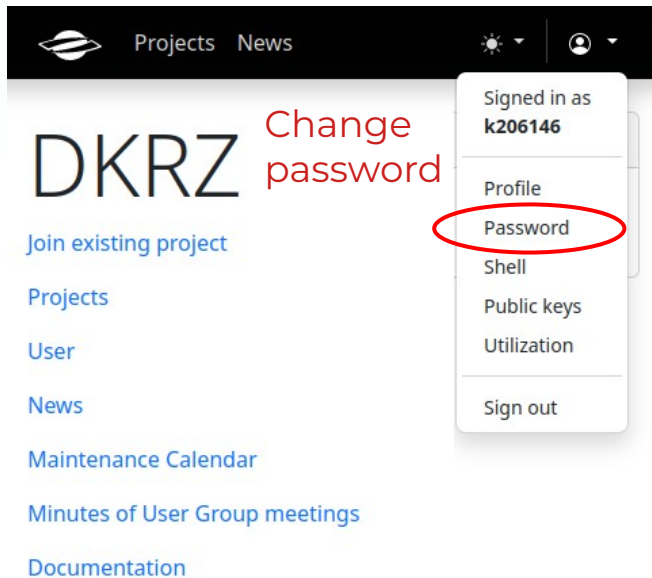
Username

Password

Remember me

Sign in

Register Forgot password?



Projects News

DKRZ

Change password

Join existing project

Projects

User

News

Maintenance Calendar

Minutes of User Group meetings

Documentation

Signed in as k206146

Profile

Password

Shell


Public keys

Utilization

Sign out

| Last name | First name | Username |
|---------------|------------|----------|
| Behncke | Jacqueline | k206090 |
| Bouarar | Idir | k206091 |
| Custódio | Danilo | k206092 |
| Fauer | Felix | k206093 |
| Filipa | Viegas | k206094 |
| Gieße | Céline | k206095 |
| Gorges | Ksenia | k206096 |
| Grawe | David | k206097 |
| Halder | Suman | k206098 |
| Hauke | Clara | k206099 |
| Hossain | Akil | k206100 |
| Kowalczyk | Lorena | k206101 |
| Krüger | Julian | k206102 |
| Linke | Olivia | k206103 |
| Lucio | Etor | k206104 |
| Mchedlishvili | Alexander | k206105 |
| Nnamchi | Hyacinth | k206106 |
| Olonscheck | Dirk | k206107 |
| Pasternack | Alexander | k206108 |
| Pohlmann | Holger | k206109 |
| Sanchez | Antonio | k206110 |
| Savita | Abhishek | k206111 |
| Vogt | Judith | k206112 |
| Wang | Xiaoxue | k206113 |
| Weinkaemmerer | Jan | k206114 |
| Winkler | Marius | k206115 |
| Zhu | Xiuhua | k206116 |

Setup - Step 2



DKRZ
DEUTSCHES
KLIMARECHENZENTRUM

Go to <https://jupyterhub.dkrz.de>

Levante Documentation

Go to DKRZ home

Sign in with your DKRZ account

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

Username:

Password:

Sign In

Enter
credentials

Tips & tricks

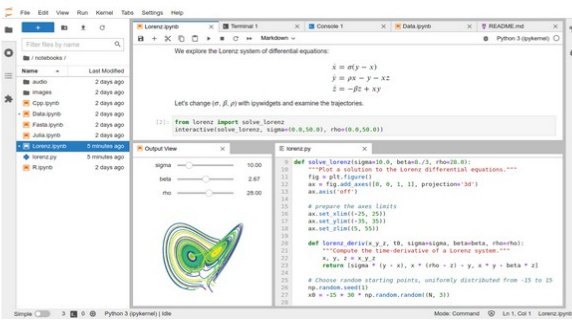
Please consult the [technical documentation](#) and also the [blog posts](#) to get started with Jupyterhub.

DKRZ System Status


Check system and other DKRZ services status [here](#). (Currently, only Mistral)

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 HPCs batch scheduling system to allocate computing resources and launch Jupyter Notebooks directly on the computing nodes. It therefore also supports the execution of parallel computation.

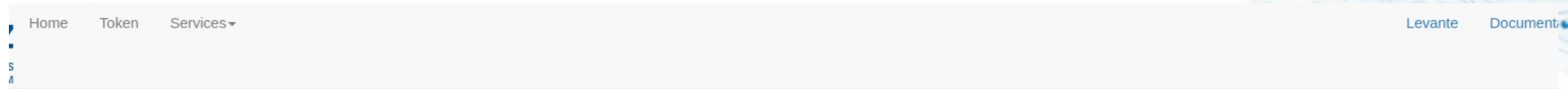


```
File Edit View Run Kernel Help
Lorenz.pyb Terminal 1 Console 1 Data.pyb README.md Python 3 (system)
We explore the Lorenz system of differential equations:
x' = sigma*(y - x)
y' = rho*x - y - x*z
z' = -rho*x + x*y
Let's change (sigma, rho, beta) with sliders and examine the trajectories.
[2]: from Lorenz import solve_Lorenz
interact(solve_Lorenz, sigma=(0.0, 50.0), rho=(0.0, 50.0))
Output View Lorenz.pyb
sigma 10.00
rho 20.00
beta 2.00
def solve_Lorenz(sigma=10.0, beta=2.0, rho=20.0):
    """Solve a solution to the Lorenz differential equations..."""
    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1, projection='3d')
    ax.set_xlabel('x')
    # prepare the axes limits
    ax.set_xlim(0, 25)
    ax.set_ylim(0, 35)
    ax.set_zlim(0, 50)
    def Lorenz_deriv(x, y, z, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x_dot = sigma*(y - x)
        y_dot = rho*x - y - x*z
        z_dot = -rho*x + x*y
    # choose random starting points, uniformly distributed from -10 to 10
    x0 = -10 + 10 * np.random.random(N, 3)
```



Notebooks on HPC nodes.

Setup - Step 3



Spawner Options



Setup - Step 4

bk1318 → Account (--account)

gpu → Partition (--partition)

mlworkshop → Reservation (--reservation) Time (hours) (--time) ← **8.00**

Number of cores (--cpus-per-task) Memory (MB) (--mem) ← **10240**

QoS (--qos)

1 x A100_80 → GPU configuration (select the gpu partition first!)

Log File Name (--output)

Request Features/Constraints (--constraint)

JupyterLab → User interface

Setup - Step 5

The screenshot shows the JupyterLab Launcher interface. The main area displays a grid of icons for different environments and file types. The 'Terminal' icon, which is a black square with a white '\$_' symbol, is circled in red. Below the grid, the word 'select' is written in red. The interface includes a top menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Git', 'Tabs', 'Settings', and 'Help'. A search bar is located at the top left of the main area. The bottom status bar shows 'Simple' mode, memory usage of 115.39 MB, and the word 'Launcher' in the bottom right corner.

home/k/l

Launcher

Filter files by name

Name Last Modified

Notebook

- R 4.1.2
- ESMValTool (based on the
- Julia 1.7.0
- Python 3 (based on the module
- VNC DESKTOP [1]

Console

- R 4.1.2
- ESMValTool (based on the
- Julia 1.7.0
- Python 3 (based on the module

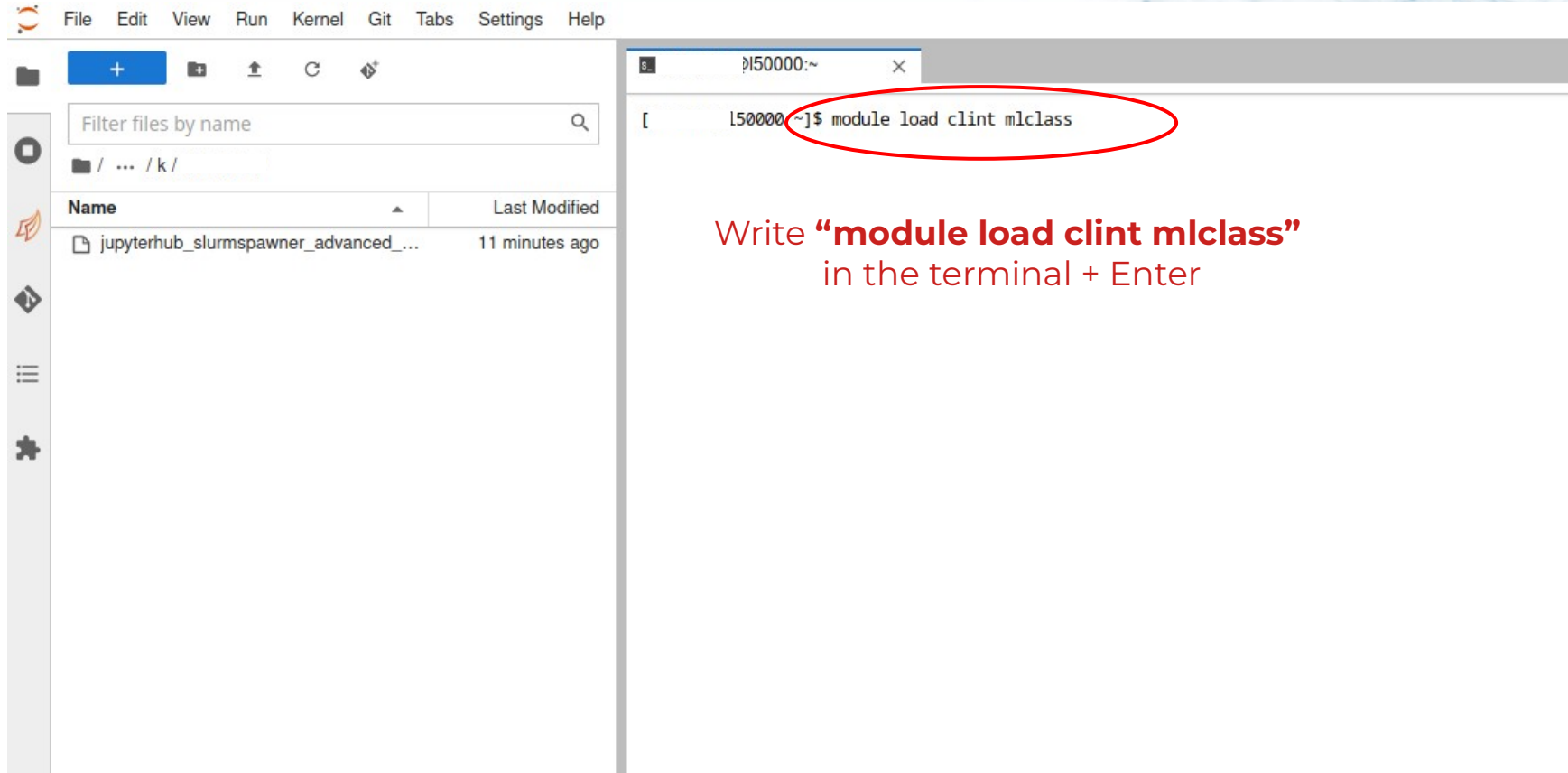
Other

- Terminal
- Text File
- Markdown File
- Julia File
- Python File
- R File
- Show Contextual Help

Simple 0 0 Mem: 115.39 MB Launcher

select

Setup - Step 6



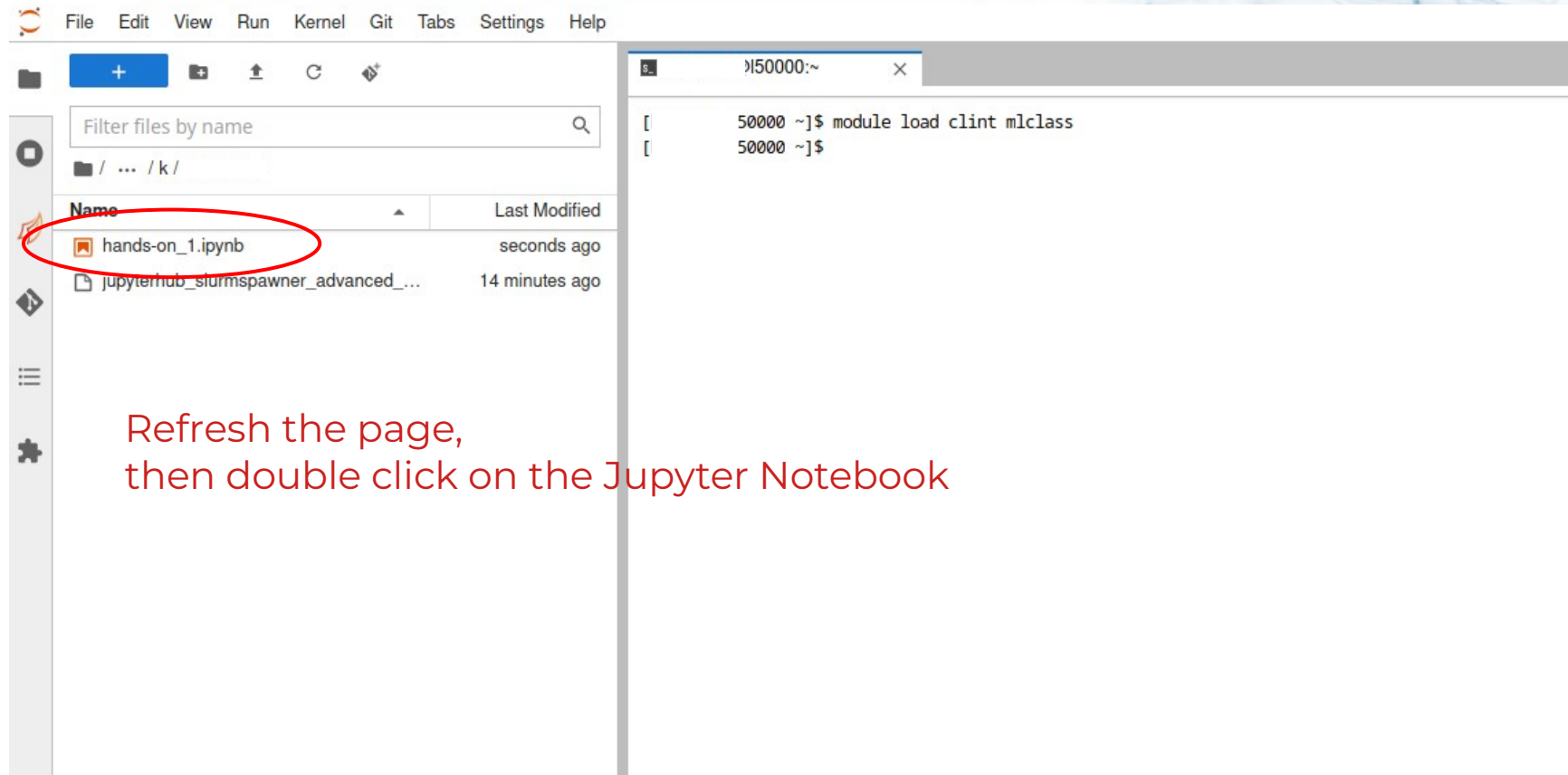
The screenshot shows the JupyterLab interface. On the left is a file browser with a search bar and a table of files. The main area is a terminal window with the prompt `[150000 ~]$` and the command `module load clint mlclass` entered. The command is circled in red.

| Name | Last Modified |
|--------------------------------------|----------------|
| jupyterhub_slurmspawner_advanced_... | 11 minutes ago |

```
[ 150000 ~]$ module load clint mlclass
```

Write **“module load clint mlclass”**
in the terminal + Enter

Setup - Step 7



The screenshot shows the JupyterLab interface. On the left, a file browser displays a list of files. The file `hands-on_1.ipynb` is highlighted with a red circle. On the right, a terminal window shows the following output:

```
[ 50000 ~]$ module load clint mlclass  
[ 50000 ~]$
```

Refresh the page,
then double click on the Jupyter Notebook

Setup - Step 8

The screenshot shows a JupyterLab environment with a notebook titled "hands-on_1.ipynb". The notebook content includes a welcome message, an outline, and the start of a section titled "1 - Packages". A modal dialog box titled "Select Kernel" is open, showing a dropdown menu with "ML Class" selected. The dialog also has "No Kernel" and "Select" buttons. The notebook content includes the following code cell:

```
[1]: # Importing the packages might take some minutes
import numpy as np
import torch
```

The notebook also shows the following text:

1 - Packages

Let's first import the `numpy` and `pytorch` packages.

2 - Initialization

PyTorch tensors are similar to Numpy arrays. In fact, many methods from Numpy exist for PyTorch tensors as well (e.g., `numpy.zeros` = `torch.zeros`, `numpy.shape` = `torch.shape`). The main difference is that PyTorch tensors support GPU acceleration.

Tensors can be initialized in different ways:

- `torch.Tensor` to create a tensor without assignment:

Setup - Step 9

The screenshot shows the JupyterLab interface with a notebook titled "hands-on_1.ipynb". The kernel is set to "ML Class". The notebook content includes a welcome message, an outline of the tutorial, and the start of the first section, "1 - Packages".

Check that "ML class" kernel is selected

```
[1]: # Importing the packages might take some minutes
import numpy as np
import torch
```

Setup - Step 10

Hands-on 1: PyTorch Tutorial

Welcome to your first hands-on session! During 1 hour, we will learn the basics features of PyTorch.

Outline

- 1 - Packages
- 2 - Initialization
- 3 - Indexing
- 4 - Arithmetic operations
- 5 - Backpropagation
- 6 - GPU computation

1 - Packages

Let's first import the `numpy` and `pytorch` packages.

```
[1]: # Importing the packages might take some minutes
import numpy as np
import torch
```

/work/bk1310/mlclass/ewm/mlclass/tib/python3.10/site-packages/tqdm/auto.py:22: TqdmWarning: IPProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
from .autonotebook import tqdm as notebook_tqdm

2 - Initialization

PyTorch tensors are similar to Numpy arrays. In fact, many methods from Numpy exist for PyTorch tensors as well (e.g., `numpy.zeros` \equiv `torch.zeros`, `numpy.shape` \equiv `torch.shape`). The main difference is that PyTorch tensors support GPU acceleration.