Using the Datenanalysis Cluster der HAWen (DACHS)

HPC @ HAW
14.03.2025

Förderträger: Baden-Württemberg
MINISTERIUM FÜR WISSENSCHAFT, FORSCHUNG UND KUNST

Overview

The **slides** will be available as PDF, a **recording** available for download!

Today‘s topics:

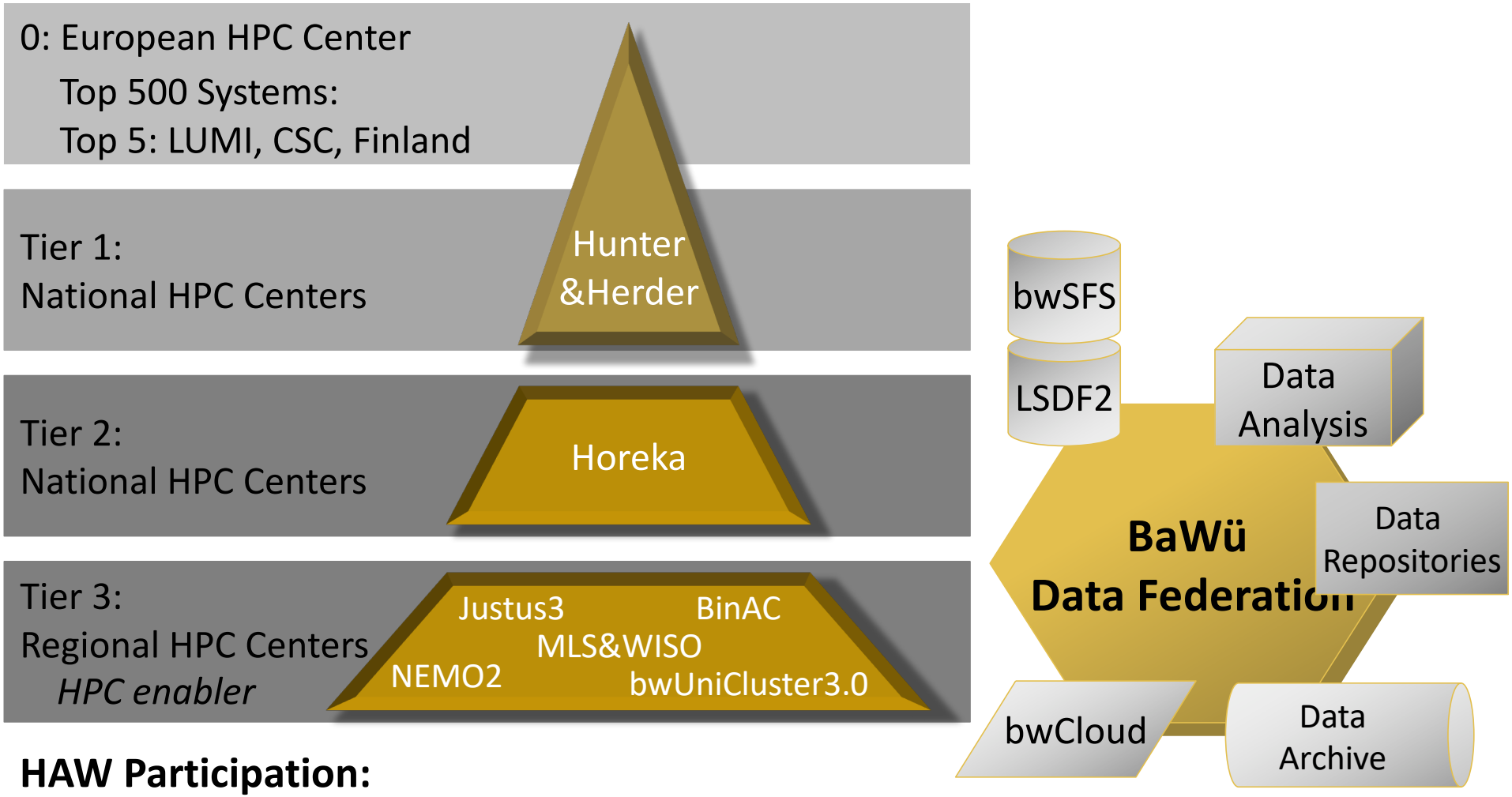
- Project Overview
- Registration
- One-Time Passwords (OTP)
- First steps using the Linux Command Line Interface (CLI)

- SLURM Queueing System
- Work-Spaces and local Scratch
- Best practices using Ollama

- Further documentation

Overview

Strategy for implementing High Performance Computing (HPC), Data intensive Computing (DIC) Large Scale Scientific Data Management (LS²DM)



HAW Participation:

1. Partnering HPC@HAW: Share of bwUniCluster (HAW as-a-whole)
2. Partnering Datenanalyse Cluster der HAW

Project HAW Datenanalyse Cluster BaWü

- Partnering as an Assocation with a cross-site installation:

1. HS Aalen
2. HS Albstadt-Sigmaringen
3. HS Esslingen
4. HS Heilbronn
5. HS Karlsruhe
6. HTWG Konstanz
7. HS Mannheim
8. HS Offenburg
9. HS Reutlingen
10. HfT Stuttgart
11. THU Ulm

Application as “Großgeräte der Länder”, reviewed positively by DFG and 50% co-funded by MWK and all partners.

Setup Datenanalyse Cluster BaWü

The Hardware

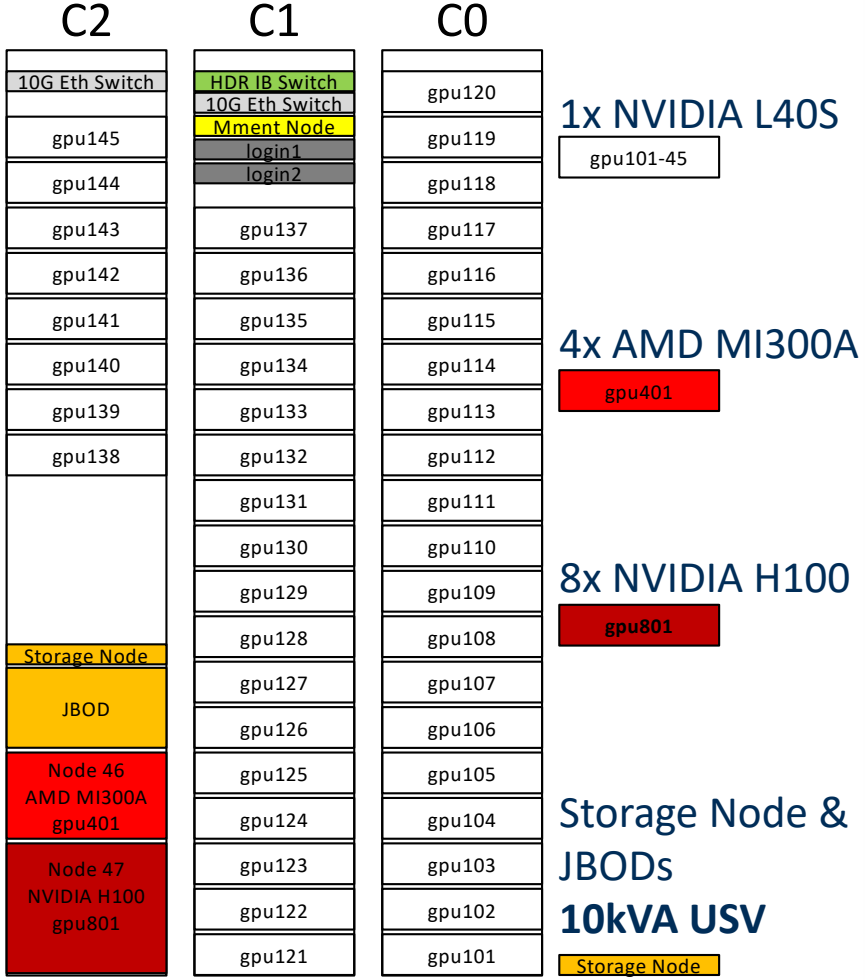
- **45** x single GPU nodes (NVIDIA L40S á 48GB)
- **1** x Quad-socket APU node
  i.e. 4x AMD MI 300A, total 512 GB HBM3 RAM
- **1** x Octo-GPU node (8x H100 á 80GB SXM5),
  Dual-AMD EPYC 9**4**54, i.e. 48 cores, 128 MB L3
  total 1,5 TB ECC-RAM
- **2 x Login** and 1x Management node
- Parallel BeeGFS filesystem with **700 TB (netto)**
- NVIDIA/Mellanox Switch IB HDR 200GBit

All nodes with:

- Dual-AMD EPYC 9254 CPU
  i.e. 24 Cores, 2.9 GHz, 128 MB L3
- 384 GB ECC RAM
- 1,92 TB local SSD (for local scratch!)

Using the available cooling infrastructure & racks.

In total **75kW peak** cooling requirement



Registration

DACHS: bwUniCluster Entitlement

- Professors of partnering Universities apply **for their** employees and students at their University's RZ for the bwUniCluster Entitlement
  (for Hochschule Esslingen the bwHPC_Antrag.pdf)

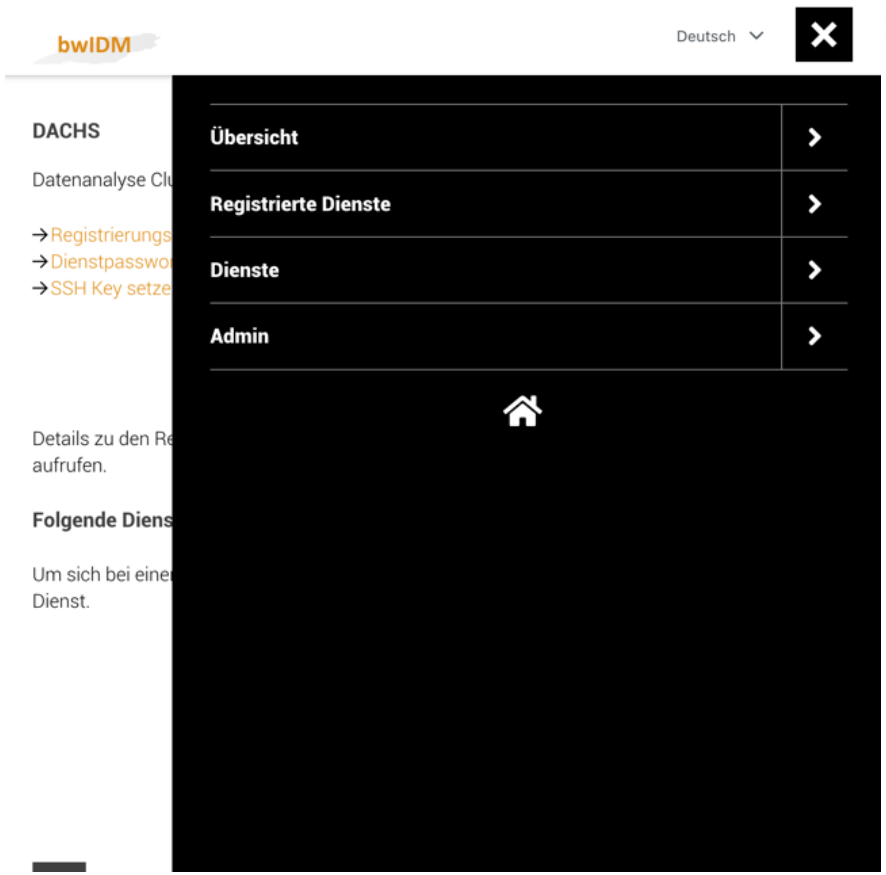
- This process is implemented at each partnering University:
  E.g. for HS Esslingen Professors fill out an PDF application form providing a project's title and short description, the names and Email Addresses of persons, together with an end date and a signature.
  More information on Services and Entitlements: https://www.bwidm.de/dienste.php

- Applicants and the receiving persons are obliged to adhere to the User guideline and specifically the German foreign trade regulations
  Parts of the HPC System are considered dual-use items and succumb to laws of export control – they are not to be used by citizens of several countries listed (Russia, Syria, Iran, North Korea).
  More information is provided by the Bundesamt für Wirtschaft und Ausfuhrkontrolle (BAFA) and the Handbook Exportkontrolle und Wissenschaft

DACHS: Log in to bwIDM

- As soon as Your Account has been issued the bwUniCluster Entitlement, You may register for the DACHS Service.
- Log into the Web-frontend https://login.bwidm.de using your University account
- You may now see the Entitlements issued to your account.

DACHS: Register for the Service

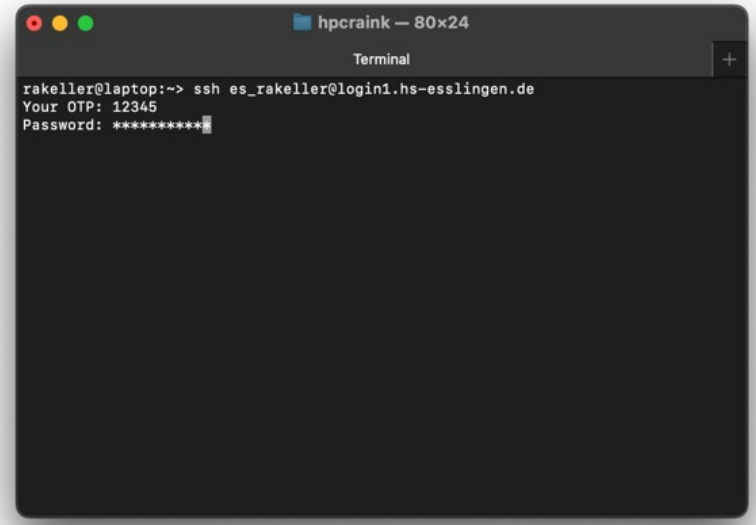


- Find the Services (Dienste) in the Hamburger Menu, ☰ respectively in the top-right corner...
- If your University is part of the alliance, You may register for the Datenanalyse Cluster der Hochschulen (DACHS).
- In Registration Info (Registrierungs-details) you see the user name for this service and You may deregister
- Please choose a **safe** Password for this service ≠ your Login-Password at your University.

- You **have to** register a 2FA token, e.g. with your phone using FreeOTP. **Please** also create a Backup TAN list, and save it **securely**, **locally**!

DACHS: Logging in

- Log into the generic login nodes (e.g. prior to using Jupyter):
  `ssh HS_ACCOUNT@dachs-login.hs-esslingen.de`
  where HS is the 2-letter abbreviation of your University and ACCOUNT is your login name
- **Attention**: For security, login is limited to BelWue-IP addresses, please use your Uni's VPN
- **Attention**: Split tunneling (OpenVPN's `route-nopull` Option) may hinder your login!
- **Hint**: There are 2 login nodes `dachs-login1` & `dachs-login2`, please use the generic one



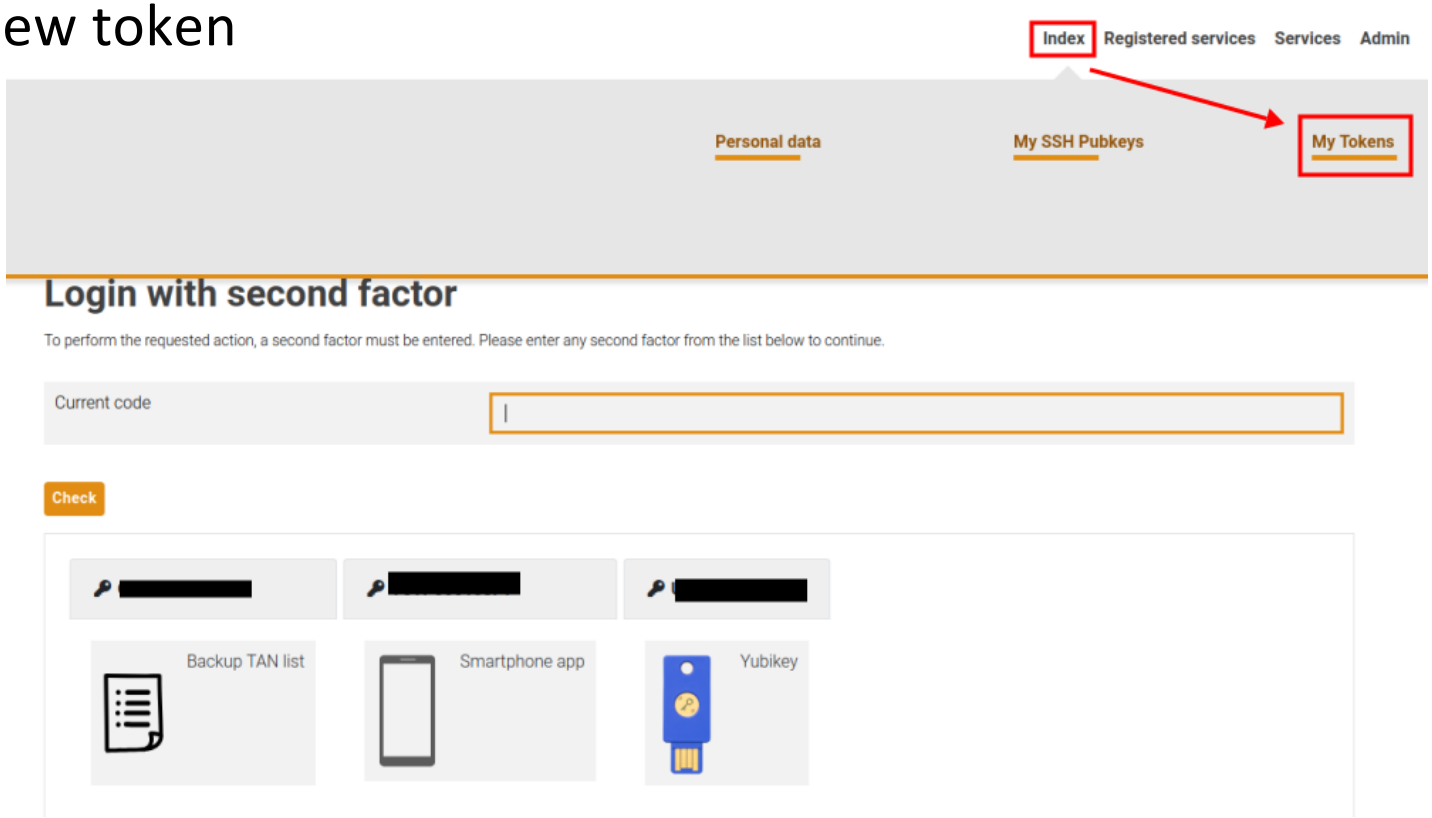
- Enter the One-Time Password (OTP) and the password of this Service (OTP is shared with bwUniCluster)
- **Hint**: The first login may take a little longer
- **Hint**: Use the Backup TAN just in case and for changing OTP in bwIDM.
- **Hint**: User from Esslingen may use the Jump host:
  `mosh comserver.hs-esslingen.de`

- Any errors? `ssh -vvv` or `traceroute` & `tcpdump` helps Admins
- Weitere Hilfen im Wiki, Support Portal und bei (Online)Kursen

One Time Passwords

DACHS: Configure OTP

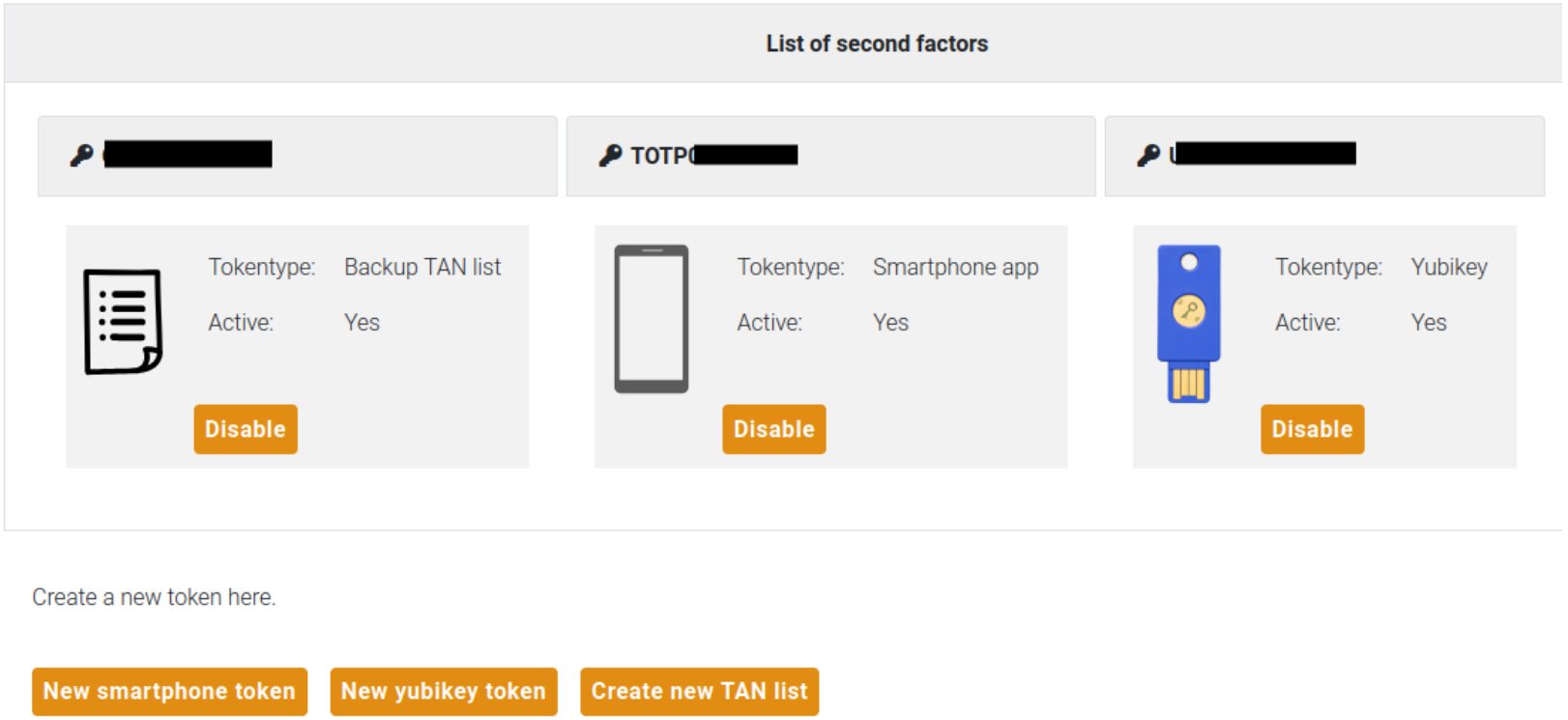
- If you already configured an OTP for BwUniCluster, you're done!
- Otherwise, go to https://login.bwidm.de/user/twofa.xhtml and add a new token



- If there are login problems, verify on this page that your OTP works

DACHS: Configure OTP

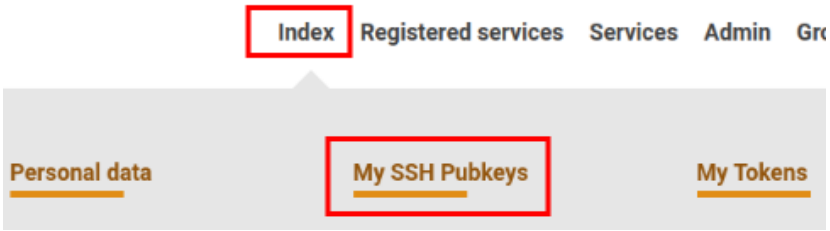
- Active second factors
- Add a new device: Yubikey, TAN list or Smartphone app
  (e.g. Google Authenticator, MS Authenticator, FreeOTP, Aegis, Sophos Authenticator)



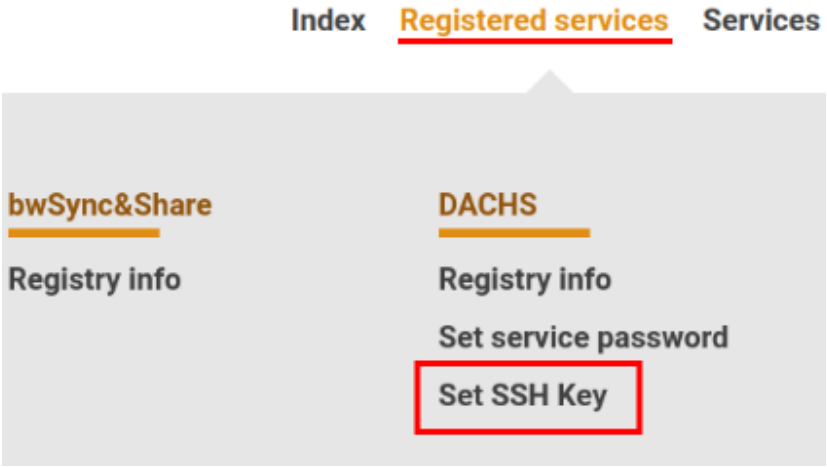
SSH Public Key Authentication

Add an SSH Public Key

- https://login.bwidm.de/user/ssh-keys.xhtml
  (or via „Index“ → „My SSH Pubkeys“)
- Add a new SSH Key in three steps



1. Generate Key pair:
   ```
   ssh-keygen -t ed25519 -f ~/.ssh/dachs/id_ed25519
   ```
2. Add the SSH public key



3. Enable this SSH key for the DACHS service („Registered services“ → DACHS „Set SSH Key“) as **interactive** key (type of usage).

SSH Configuration

- Example SSH client configuration in ~/.ssh/config

```
Host dachs
    Hostname dachs-login.hs-esslingen.de
    User es_username
    IdentityFile ~/.ssh/dachs/id_25519
```

Then on Command Line:

```
$ ssh dachs
```

- SSH key is unlocked for **1 hour** after logging in with password & OTP
- After this one hour another login with password and OTP is required

First steps using Linux

Bash

- Interactive commands using Bourne Again Shell (Bash)

- Most used commands, starting after $ (# is the comment character):

```
$ cd ./directory # change directory
$ ls -la         # list files: -l=long -a=all, including hidden files
$ NAME="Peter"   # Set Variable named NAME
$ echo "Hi $NAME" # output: Hello, Peter
$ ./program arg1 arg2 arg3
$ sbatch your_job.sh
```

- Read documentation in the manual pages
  - `$ man ls` # Information on list
  - `$ man -k printf` # (search man pages relevant to 'printf')

Bash

- Environment variables are set in the Shell and passed to any command
- Environment variables are displayed with

```
$ env
```

- Add a new environment variable:

```
$ export NAME="Peter"
```

- Define aliases/shortcuts:

```
$ alias ll='ls -l'  # now „ll“ runs the command „ls -l“
```

- Special variables

```
$ echo $?  # show status code of last command
```

and many more of these – please check `man bash`

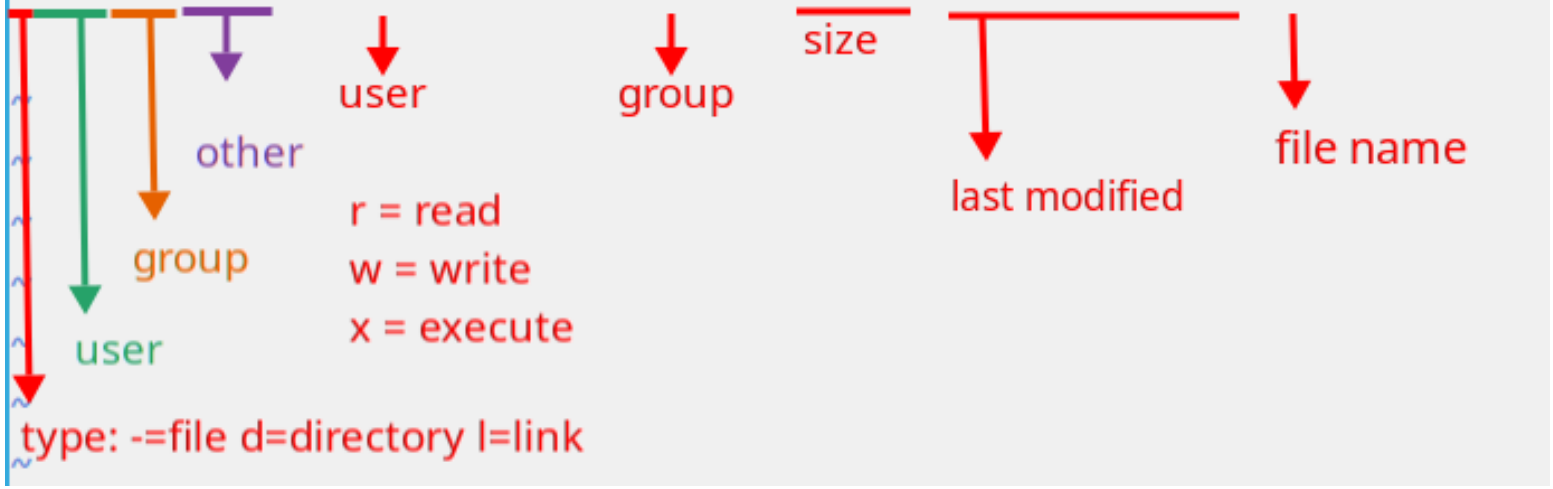
Linux Hierarchical File System

Unix Filesystem

```
/           # top level root of the hierarchical file system
|-- beegfs    # parallel file system
|   |-- bwhpc
|   |   `-- common
|   `-- scratch
|       `-- workspace
|-- bin -> usr/bin   # executable binaries
|-- etc            # configuration files
|-- home
|   |-- aa         # organized by organisation
|   |-- as
|   |-- es
|   [...]
|   `-- of
|-- localscratch    # on the compute node ~1TB
|-- opt     # further software
|-- tmp     # temporary files and directories
|-- usr     # user-installed libraries, binaries, documentation
```

Bash

```
es_makunzel@login2 ~ $  pwd
/home/es/es_es/es_makunzel
es_makunzel@login2 ~ $  ls -l
-rwxr-xr-x 1 es_makunzel es_es  17616 Feb  4 10:14 a.out
-rw-r--r-- 1 es_makunzel es_es    239 Feb  4 10:14 main.c
-rw-r--r-- 1 es_makunzel es_es  39309 Feb 26 13:51 ollama.log
-rw-r--r-- 1 es_makunzel es_es    390 Feb 21 14:09 ollama.sh
-rw-r--r-- 1 es_makunzel es_es    637 Feb 25 16:55 ollama2.sh
drwxr-xr-x 2 es_makunzel es_es     20 Mar 12 13:18 slurm
-rw-r--r-- 1 es_makunzel es_es    213 Feb  4 11:02 test.sh
```



Bash

- Further File operations are:
  - `$ chmod 700 test.sh # set rwx for owner of test.sh`
    - `Permission values: Read(4), write(2), execute(1)`
  - `$ rm test.sh # delete (remove) file`
  - `$ mkdir directory-name (make directory)`
- Create a soft-link using:

  `$ ln -s target/file/path linkname`

- Pre-installed SW is available using modules:

  `$ module avail` # shows the installed software modules

  `$ module load compiler/gnu` # loads the latest GCC

  `$ module list` # shows the loaded software modules

- Please use the E-learning module “Linux Basics” available at:

  https://**training.bwhpc.de**/ilias.php?baseClass=illmpresentationgui&cmd=resume&ref_id=310

SLURM Queueing System

SLURM: Overview Queuing System

- When logging in, You are on one of the Login-Nodes, here You may:
  - Prepare for execution, editing, programming etc.
  - Compile your application (even `make -j 48`, go for it ☺)
  - Allocate a Workspace (see later), copy files
  - But not any long-running jobs using lots of CPU-time and memory…
- For a fair-share of compute nodes we use a Batch Scheduler: SLURM
- This allows for:
  - Accounting of used resources (fair share for every partner organization)
  - Proper resource allocation, “you get, what you ask for”, i.e.:
  - Non-shared usage of the 1-GPU compute nodes with NVIDIA L40S
  - Proper shared usage of the 4x socket APU and 8x GPU server nodes
  - And only the **amount of memory** & “**generic consumable resources**” You request.
- SLURM commands start with “s” and set env.-variables with SLURM, e.g. after your 1st `srun`, do `echo $SLURM_JOBID` (use TAB-TAB)

SLURM: Resource Allocation

- For the different compute nodes, we have 3 SLURM partitions:
  - Partition `gpu1` for compute nodes with 1 NVIDIA L40S
  - Partition `gpu4` requesting (part of) the 4x AMD MI300A node
  - Partition `gpu8` requesting (part of) the compute node with 8 NVIDIA H100 GPUs
- To start: Run processes on compute nodes **interactively** using bash:

```
srun --partition=gpu1 --gres=gpu:1 --pty /bin/bash
```

Generic resource, here 1 GPU, without it, processes are limited to CPUs. Check with `nvidia-smi` / `rocm-smi`.

Forward the std. output and std. error of the 1st UNIX process to your current terminal

The actual UNIX process to run, here just a Shell, from which to start other processes

Select compute nodes (short `-p`), here from 1 partition

srun allocates resources and runs the process (here on one CPU core, binding UNIX process to this very core with SLURM default limit of memory, see below)

SLURM: Batch processes

- The real strength comes with writing *batch* scripts! Benefits are:
  - Better schedulability under high load of the whole system (please est. run time)
  - Repeatability of your jobs and results! Including documentation of science!
  - Improving scripts over time: storing Metadata of your science in the SLURM logs!
- Example script `run.slurm`, start with `sbatch run.slurm`:

```
#!/bin/bash                          ← Set the she-bang to the Bash interpreter
#SBATCH --partition=gpu8             ← Select a node in gpu8 partition
#SBATCH --gres=gpu:h100:8            ← Allocate the generic resource: 8 GPUs of type h100
#SBATCH --time=2:0:0                 ← Specify your (estimate of the) max. run time: 2hrs
#SBATCH --nodes=1                    ← Select the number of compute nodes (short -N)
#SBATCH --ntasks=96                  ← Allocate all CPU cores on this node (short -n), 2x48
#SBATCH --mem=1400G                  ← Allocate all available memory on the node: 1,4TB
#SBATCH --output=run%j.out           ← StdOut into log-file with Job-ID, slurm-%j.out
# Start processes/threads            ← Here, we start your (Bash) Shell commands
```

- Download example (for GPU1): https://www2.hs-esslingen.de/~rakeller/run_example.slurm
  or copy: `cp /tmp/run_example.slurm ~/`

SLURM: More information

- SLURM is very versatile: `sbatch` submits a “job”:
  - A job may contain `n` “job steps” (usually 1 step, may submit with `n` times `srun`)
  - A “job step” may have specific resource requirements within the job: tasks
  - Tasks are individual processes, the actual execution unit (MPI ranks, or threads)
  - Direct control of resource allocation and mappings/binding to actual hardware!
- You get information on SLURM itself using:
  - `squeue` shows (my own) SLURM jobs running (or currently exiting)
  - `sinfo_t_idle` shows idle nodes in each partition (just like on bwUniCluster)
- The example contains „debug“ information, like:
  - `free` shows the available types of (free) memory on this node
  - `ulimit -a` provides information on the „hard limit“ of the allocated memory
  - `module list` what kind of SW modules are currently loaded (see `avail`)
  - `env` lists the “environment variables”, mainly interesting `SLURM_*`
  - `ibstat` Infiniband statistics (one node 1), helpful to detect if IB-link is down!

SLURM: Multi-node jobs / MPI

- Let's prepare a MPI job:

```
cp /tmp/mpi_stub.c $HOME/
cp /tmp/mpi_stub.slurm $HOME/
module load mpi/openmpi
mpicc -Wall -O2 -o mpi_stub mpi_stub.c
```

And submit the script: `sbatch mpi_stub.slurm`

```
#!/bin/bash
#SBATCH --nodes=2                  ← Allocate 2 nodes (of any partition)
#SBATCH --ntasks-per-node=48       ← Allocate 48 tasks per node (all cores per gpu1)
#SBATCH --time=1                   ← This job runs for 1 minute (well only seconds)
#SBATCH --mail-user=me@i.de        ← SLURM mails about job start, fail & end
module load mpi/openmpi            ← Load the same MPI module, may need version
mpirun ~/mpi_stub                  ← Let MPI-environment figure out the rest
```

- Or start the MPI-process directly using `srun`:

```
srun --mpi=pmix --nodes=2 --ntasks-per-node=48 ./mpi_stub
```

SLURM: Multi-node jobs / MPI + OpenMP

- The more advanced option is to use MPI+X, e.g. OpenMP:

```
cp /tmp/mpi_openmp.c $HOME/
cp /tmp/mpi_openmp.slurm $HOME/
module load mpi/openmpi
mpicc -Wall -O2 -fopenmp -o mpi_openmp mpi_openmp.c
```

And submit the script: `sbatch mpi_openmp.slurm`

```
#!/bin/bash
#SBATCH --nodes=2                  ← Allocate 2 nodes (of any partition)
#SBATCH --sockets-per-node=2       ← Hint to restrict to nodes with 2 sockets/node
#SBATCH --cores-per-socket=24      ← Hint to restrict to nodes with 24 cores/sockets
#SBATCH --ntasks-per-node=2        ← Run 2 (MPI)tasks per node
#SBATCH --cpus-per-task=24         ← Run each task with 24 cores (one Thread/core)
module load mpi/openmpi
export OMP_NUM_THREADS=24
mpirun -np 4 ~/mpi_openmp          ← 2 Nodes with 2 MPI processes
```

SLURM: Advanced options

- SLURM has an abundance of features, you may request:
  - „licenses“, to schedule SW like ANSYS available to your HS (currently unused)
  - A different “account” for an industry-collaboration project (talk to us)
  - To create a reservation, e.g. for classes using 8 nodes for a certain time
  - Job chains…
- CPU-distribution and binding is essential for good performance:
  “The default distribution on multi-core/multi-threaded systems is equivalent to `-m block:cyclic` with `--cpu-bind=thread`”

```
--cpu-bind=socket     # Good for MPI+OpenMP
--cpu-bind=verbose    # To review the setting
```

- Details about Your job (or partition):

```
scontrol show job
```

- Find out when Your job is scheduled to start:

```
squeue --start
```

Work Spaces and LocalScratch

Workspace Tools

- Workspaces are stored on BeeGFS
- Default duration: 30 days (extendable up to 90 days)
- Basic commands
  - Create: `ws_allocate <name> <days>`
  - Extend: `ws_extend <name> <days>`
  - Delete: `ws_release <name>`
  - Find storage path: ws_find <name>
    - `$ ws_find test_workspace`
    - `/beegfs/scratch/workspace/xx_use-test_workspace`
  - List your workspaces: `ws_list`

Workspace Tools

- Email reminder start being send **1 week** before expiry
- After expiry workspace is kept for another 14 days
  - Restore using `ws_restore`

```
$ ws_restore -l                  # list restorable workspaces

$ ws_allocate new-ws             # allocate a new workspace

$ ws_restore <old> new-ws        # restore the  under new name
```

More examples in the user guide
https://github.com/holgerBerger/hpc-workspace/blob/master/user-guide.md

Local Scratch

- Every node has ~1TB on a NVME SSD for user jobs!
- XFS file system for Users mounted under
  `/localscratch/tmpdir.${SLURM_JOB_ID}`

- Use if your programs needs to write and/or read frequently from disk!
- Especially many small files will be better put onto these

Examples:

- Copy (recursively) a directory from your `$HOME` there:
  `cp -r $HOME/dir /localscratch/tmpdir.${SLURM_JOB_ID}`
- Unpack file from Workspace into directory (attention, 2 lines):

```
unzip `ws_find my_workspace`/file.zip
            -d /localscratch/tmpdir.${SLURM_JOB_ID}
```

Best practices using Ollama

Ollama: Preparations

- Ollama runs multiple / different LLMs using CPUs/GPUs on llama.cpp

- One may download different models – which may be huge (404GB!)
  Please **do not store** in Your HOME in `~/.ollama` (soft-limit: 200GB)
  (The reduced model `deepseek-r1:70b` fits NVIDIA L40S perfectly
- Instead, create a work-space for these Ollama models (for 60 days), and link into your HOME directory:

```
ws_allocate ollama_models 60
ln -s `ws_find ollama_models`/ ~/.ollama
```

- This will keep these huge files out of Your HOME…
  Instead of creating a soft-link, specify the environment variable:

```
export OLLAMA_MODELS=`ws_find ollama_models`/models/
```

Ollama: Running the server

- Runs the server on a GPU node `ollama_example.slurm`:

```
#!/bin/bash
#SBATCH --partition=gpu1
#SBATCH --gres=gpu
#SBATCH --nodes=1
#SBATCH --time=2:0:0
#SBATCH --ntasks=48
#SBATCH --mem=350G
#SBATCH --job-name=ollama
#SBATCH --mail-type=BEGIN
#SBATCH --mail-user=m@me.de
module load cs/ollama
export OLLAMA_HOST=0.0.0.0
export OLLAMA_LOAD_TIMEOUT=0
export OLLAMA_KEEP_ALIVE=0
ollama serve
```

Allocate one NVIDIA L40S node, requesting the GPU for 2 hours – and use all cores and the maximum amount of memory.

Set the job name to ollama.

Mail me, when the job begins.

Load the latest version (currently 0.5.13)

The server should bind to the global network

Disable load timeout – if loading takes too long

Do not unload the model (default is 5 minutes)

Run the server; use `--help` for other options

Ollama: Running the client

- Now with a second login/terminal to DACHS, one may pull & rune the largest (reduced) model of deepseek,fitting into the VRAM of L40S:

```
module load cs/ollama
export OLLAMA_HOST=gpu101
ollama pull deepseek-r1:70b
ollama list
ollama run deepseek-r1:70b
```

← Please adapt to the node allocated to you.

Or unpack the file using:

```
unzip /tmp/ollama_deepseek_r1_70b.zip
```

← Shows all the Models known on the server

← Allows quering the model / setting parameters

- Once it has loaded (about 240seconds), You may ask questions:

```
>>> When did Elvis die?
Elvis Presley died on August 16, 1977.
>>> Where did he die?
Elvis Presley died at his home, Graceland, in Memphis, TN.
>>> Was he married at that time?
At the time of his death, Elvis Presley was not married.
His divorce from Priscilla Ann Beaulieu had been finalized
on October 3, 1973.
```

Ollama: Connecting from your Laptop

- Employing the power of the GPUs from home requires forwarding the Port – so from Your own Laptop You may log in:
  `ssh -L 11434:gpu101:11434 HS_ACCOUNT@dachs-login.hs-esslingen.de`
- This will create a TCP Socket on Port 11434 on `localhost` and tunnel any connections to the gpu101 node on the same port there.
- Now, you may use even local Python programming:

```
python -m venv ollama_test
source ollama_test/bin/activate
python -m pip install ollama
```

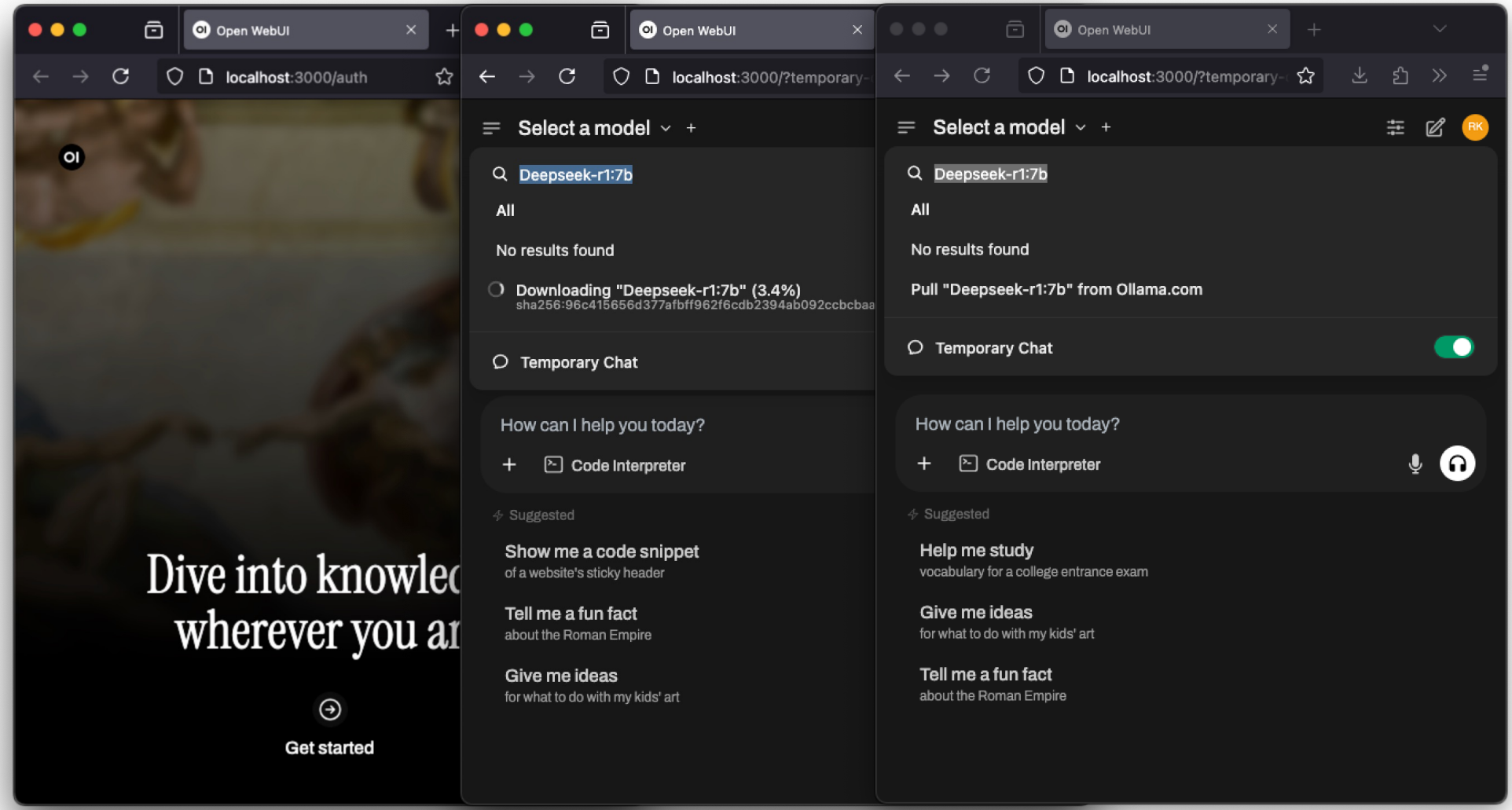
  and connect to it running Python code:

```
import ollama
response = ollama.chat(model='deepseek-r1:70b', messages=[ {
'role': 'user', 'content': 'why is the sky blue?'},])
print(response)
```

- **Attention**: Ports on compute nodes are open to anyone (within the Cluster), so anyone may connect to ”your” Ollama server, however, they have their own context – but will take “your” resources.

Ollama: Open WebUI

- Open WebUI runs a docker-based a web-frontend client locally:



More information

DACHS: More Information

Please take note of the below Links to our Wiki:

- In your `$HOME` please only store the most important data – we have a Hard Quota per organisation, we mail on hitting the Soft Quota per User: 200 GB!
- Please use the Work Space mechanism (Scratch) on the parallel BeeGFS using `ws_allocate` and other `ws_*` tools.
- **Most performant file access for AI workloads:** node-local `/localscratch`
- On login nodes: no long-running processes / no huge memory…
- Please use SLURM Batch jobs, see `man squeue` & `sinfo_t_idle`
- Please use Environment Modules, s. `module avail` et al
- In case of questions, please e-mail dachs-admin@hs-esslingen.de
- For SW-Installations, longer support, please open a Ticket in Support-Portal https://www.bwhpc.de/supportportal/, select **Support Unit**: DACHS

- **Please take note of the E-Training platform:** https://training.bwhpc.de

Fragen?

Bei weiteren Fragen: dachs-admin@hs-esslingen.de

Oder ein Ticket auf: https://www.bwhpc.de/supportportal