ETH zürich

Scientific IT Services
High Performance Computing

Getting started with the scientific cluster

Nadia Marounina, Samuel Fux
High Performance Computing Group
Scientific IT Services, ETH Zurich

Outlook

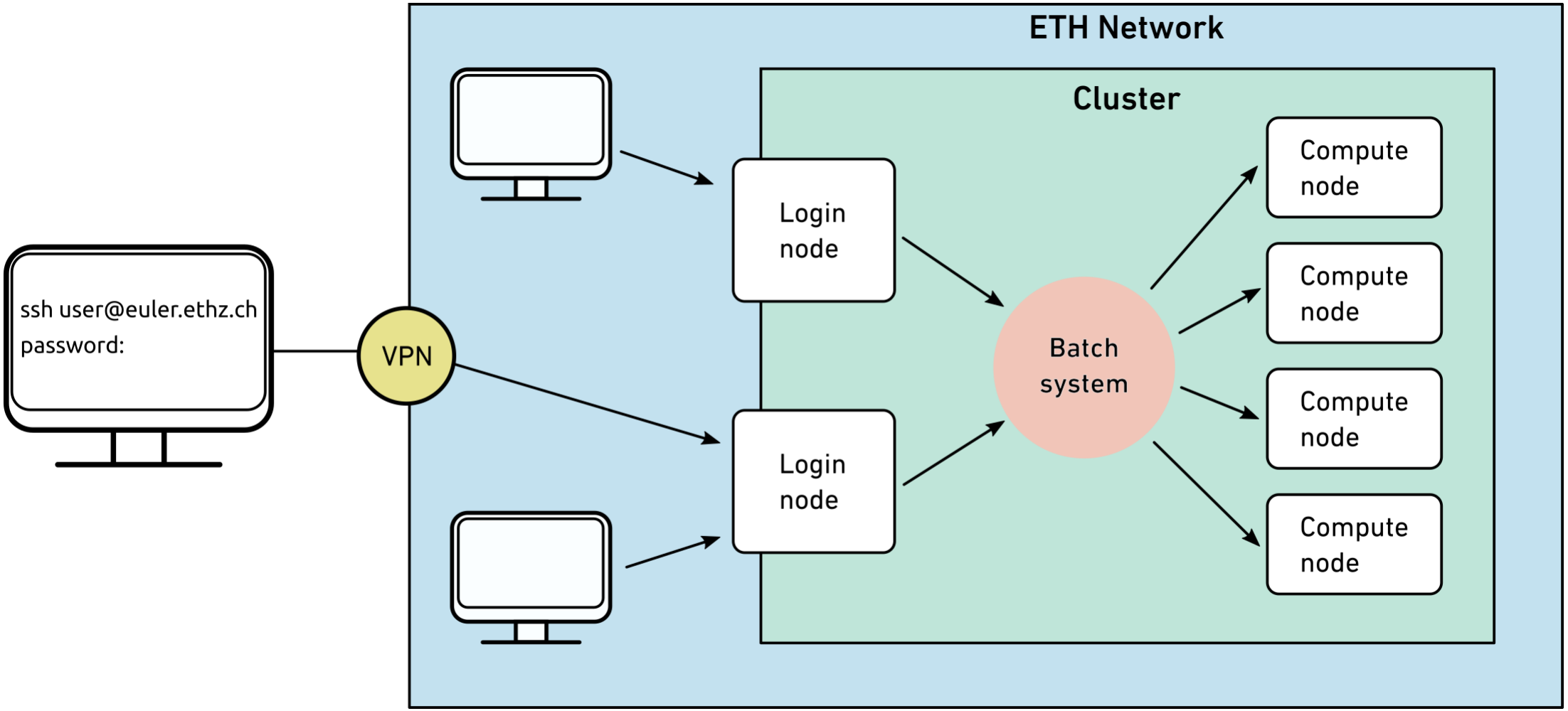
- Accessing the cluster
- Storage and data transfer
- Modules and applications
- Using the batch system

https://scicomp.ethz.ch/wiki/Main_Page

Outlook

- Accessing the cluster
- Storage and data transfer
- Modules and applications
- Using the batch system

Accessing the cluster

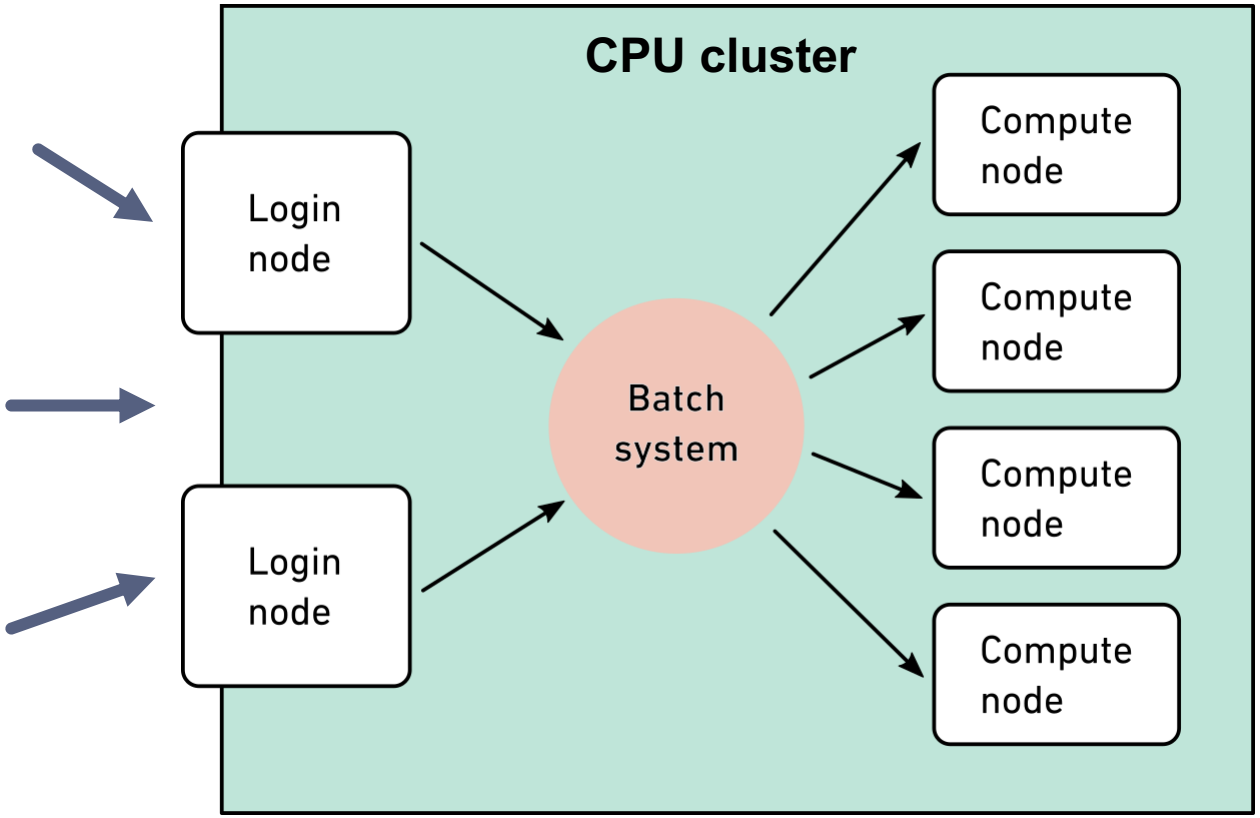


Access > Who can use the cluster > CPU cluster

Shareholders that invested into a share of the cluster resources

External collaboration partners of shareholders

All ETH members (they can use the cluster as guest users with limited resources)



Access > Prerequisites

- A valid ETH account
- Local computer with an SSH client
  - Linux and macOS contain SSH client as part of the operating system
  - Windows users need to install a third party SSH client
    - MobaXterm (https://mobaxterm.mobatek.net/) is a free open source SSH client that we recommend

- An X11 server for graphical user interface (optional)
  - Linux (https://www.xorg.com)
  - macOS (https://xquartz.org)
  - Windows (included in MobaXterm)

Access > How to access the clusters > ETH members

1. Start your SSH client
2. Use `ssh` command to connect to the login node of Euler

```
ssh username@euler.ethz.ch
```

3. Use your ETH credentials to login
4. First login
   - On first login a verification code is sent to your email address (*username*@ethz.ch)
   - By entering the verification code, your account is created automatically
   - New users must accept the cluster's usage rules

https://scicomp.ethz.ch/wiki/New_account_request_process_for_HPC_clusters

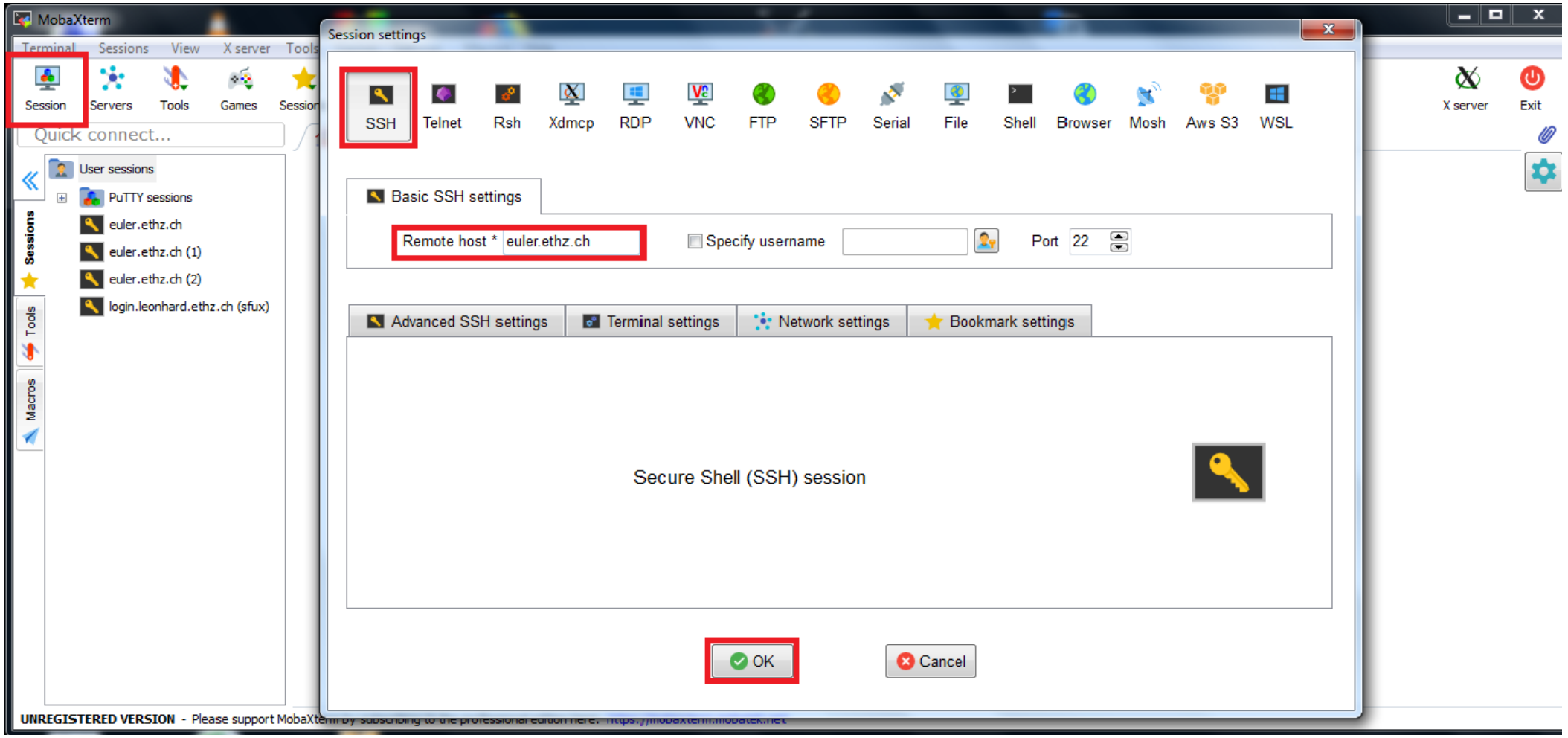
Access > Legal compliance

- The HPC clusters are subject to ETH's acceptable use policy for IT resources (Benutzungsordnung für Telematik, BOT, https://rechtssammlung.sp.ethz.ch/Dokumente/203.21en.pdf), in particular:
  - Cluster accounts are **strictly personal**
  - DO NOT share your account (password, ssh keys) with anyone
  - DO NOT use someone else's account, even if they say it's OK
  - If you suspect that someone used your account:
    - change your password at https://password.ethz.ch
    - contact cluster-support@id.ethz.ch
- Consequences
  - In case of abuse, the offender's account may be blocked temporarily or closed
  - System administrators are obliged by law to investigate abusive or illegal activities and report them to the relevant authorities

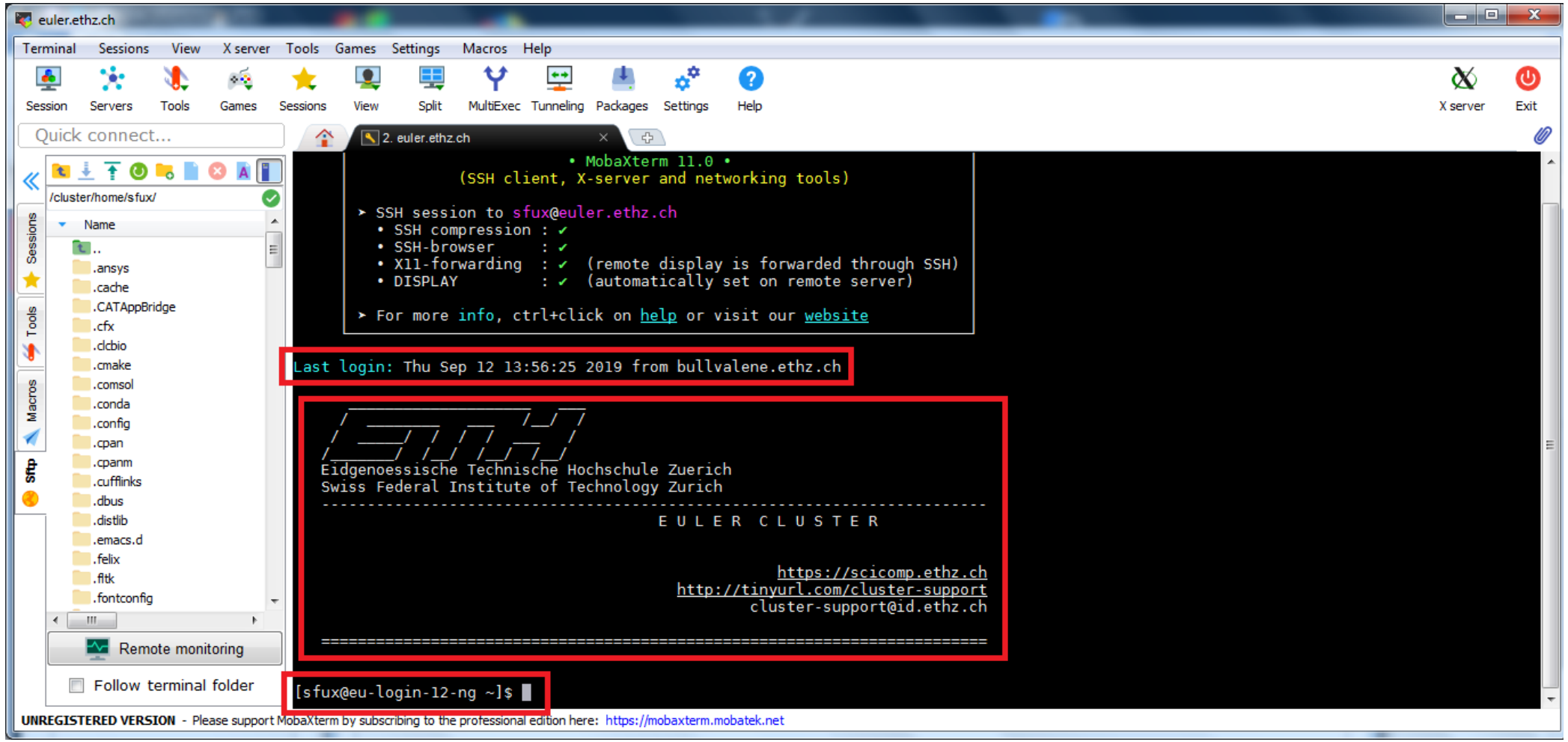
Access > SSH connection > Linux, macOS

```
samfux@bullvalene:~$ ssh sfux@euler.ethz.ch
sfux@euler.ethz.ch's password:
Last login: Fri Sep 13 07:33:57 2019 from bullvalene.ethz.ch

      ____________________   ___
     /  _________   ___   /__/  /
    /  /____/  /  /  /  /___/  /
   /_______/  /__/  /__/  /__/
   Eidgenoessische Technische Hochschule Zuerich
   Swiss Federal Institute of Technology Zurich
   ------------------------------------------------------------------
                                  E U L E R  C L U S T E R


                                                  https://scicomp.ethz.ch
                                              http://www.smartdesk.ethz.ch
                                                cluster-support@id.ethz.ch

   ==================================================================

[sfux@eu-login-19 ~]$
```

Access > SSH connection > Windows



Access > SSH connection > Windows



Access > SSH keys

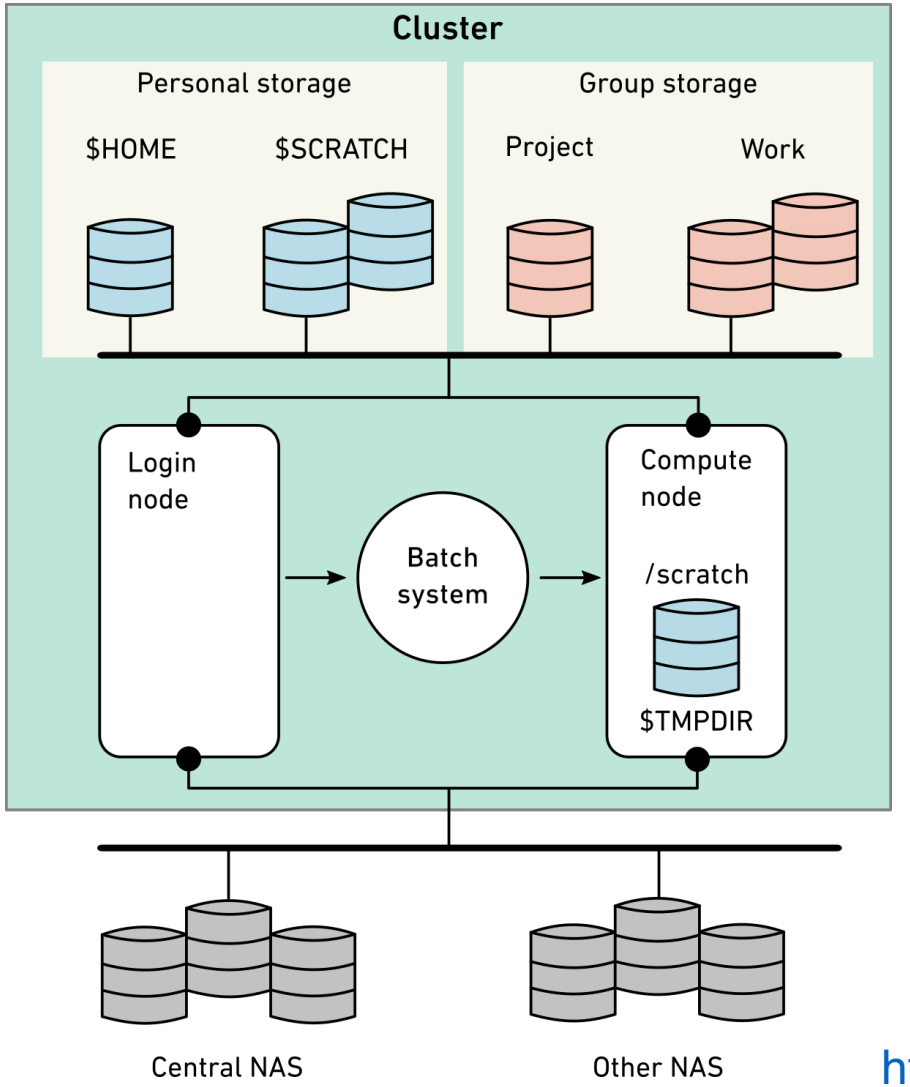
- SSH keys allows passwordless login
  - Useful for file transfers and automated tasks
  - When used properly, SSH keys are much safer than passwords

- The procedure to create and use the SSH keys is detailed here : https://scicomp.ethz.ch/wiki/Accessing_the_clusters#SSH_keys

Outlook

- Accessing the cluster
- Storage and data transfer
- Modules and applications
- Using the batch system

Data > Available storage systems



- Cluster wide storage systems
  - Home (personal)
  - Global scratch (personal)
  - Work (group)
  - Project (group)
- Local storage inside the compute node
  - Local scratch
- External storage
  - NAS
  - CDS
  - LTS

https://scicomp.ethz.ch/wiki/Storage_systems#External_storage

Data > Personal storage (every user) > Home

```
$ cd $HOME
$ pwd
/cluster/home/username
```

- Safe, long-term storage for critical data (program source, scripts, etc.)
- Accessible only by the user (owner); other people cannot read its contents
- Disk quota of 16/21 GB and a maximum of 80’000/200’000 files (soft/hard quota). Quota can be checked with the command `lquota`
- Contents saved every hour/day/week using snapshot. Users can access these snapshots in the hidden `.snapshot` directory

https://scicomp.ethz.ch/wiki/Storage_systems#Home

Data > `lquota`

```
[sfux@eu-login-02 ~]$ lquota
+----------------------+------------+-----------+---------------+---------------+
| Storage location:    | Quota type: | Used:     | Soft quota:   | Hard quota:   |
+----------------------+------------+-----------+---------------+---------------+
| /cluster/home/sfux   | space      |    8.85 GB |      17.18 GB |      21.47 GB |
| /cluster/home/sfux   | files      |      25610 |        160000 |        200000 |
+----------------------+------------+-----------+---------------+---------------+
| /cluster/shadow      | space      |    4.10 kB |       2.15 GB |       2.15 GB |
| /cluster/shadow      | files      |          2 |         50000 |         50000 |
+----------------------+------------+-----------+---------------+---------------+
| /cluster/scratch/sfux | space      |  237.57 kB |       2.50 TB |       2.70 TB |
| /cluster/scratch/sfux | files      |         29 |       1000000 |       1500000 |
+----------------------+------------+-----------+---------------+---------------+
```

Data > Personal storage (every user) > Global scratch vs. local scratch

Global scratch

```
$ cd $SCRATCH
$ pwd
/cluster/scratch/username
```

- Fast, short-term storage for computations running on the cluster
- Created automatically upon first access and visible (mounted) only when accessed
- Disk quota of 2.5/2.7 TB and a maximum of 1m/1.5m files (soft/hard quota). Quota can be checked with the command `lquota`
- Strict usage rules; see `$SCRATCH/__USAGE_RULES__` for details
- No backup

Data > Group storage (only shareholders) > Project vs. Work

Project

`/cluster/`**`project`**`/`*`groupname`*

- Similar to home, but for groups
- Safe, long-term storage for critical data

Work

`/cluster/`**`work`**`/`*`groupname`*

- Similar to global scratch, but without purge
- Fast, short- or medium-term storage for large computations
- Visible (mounted) only when accessed

- Shareholders can buy as much space as they need
- The access rights are managed by the owner
- Quota can be checked with `lquota`
- Backed up multiple times per week

Data > File system comparison

File system	Life span	Snapshot	Backup	Max size	Small files	Large files
`/cluster/home`	Permanent	Yes	Yes	21 GB	+	o
`/cluster/scratch`	2 weeks	-	-	2.7 TB	o	++
`/cluster/project`	4 years	Optional	Yes	Flexible	+	+
`/cluster/work`	4 years	-	Yes	Flexible	o	++
local `/scratch`	Job	-	-	800 GB	++	o
central NAS	Flexible	Yes	Optional	Flexible	+	+

Retention time
Snapshots: up to 1 week
Backup: up to 90 days

Data > Copying data from/to the cluster (command line)

Secure copy (`scp`) is most commonly used to transfer files

```
scp [options] source destination
```

Examples: All the following examples need to be run on your local computer

- Upload a file from your workstation to Euler

```
scp local_file username@euler.ethz.ch:/path/to/remotedir
```

- Download a file from Euler to your workstation

```
scp username@euler.ethz.ch:/path/to/remote_file /path/to/localdir
```

- Copy a whole directory

```
scp -r localdir username@euler.ethz.ch:remotedir
```

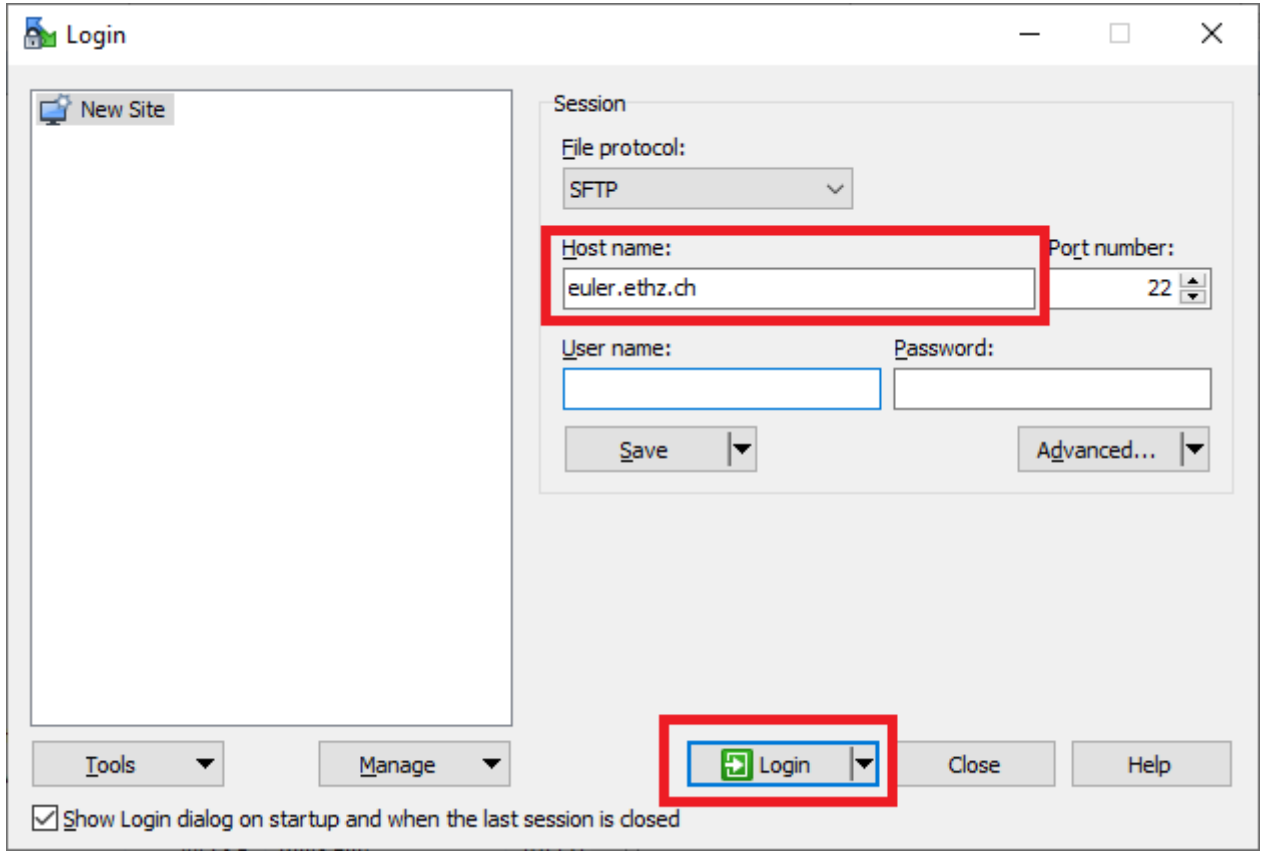
Alternatives to `scp`: `sftp`, `rsync`, `svn`, `git`, `wget`

Data > Copying data from/to the cluster (graphical user interface)

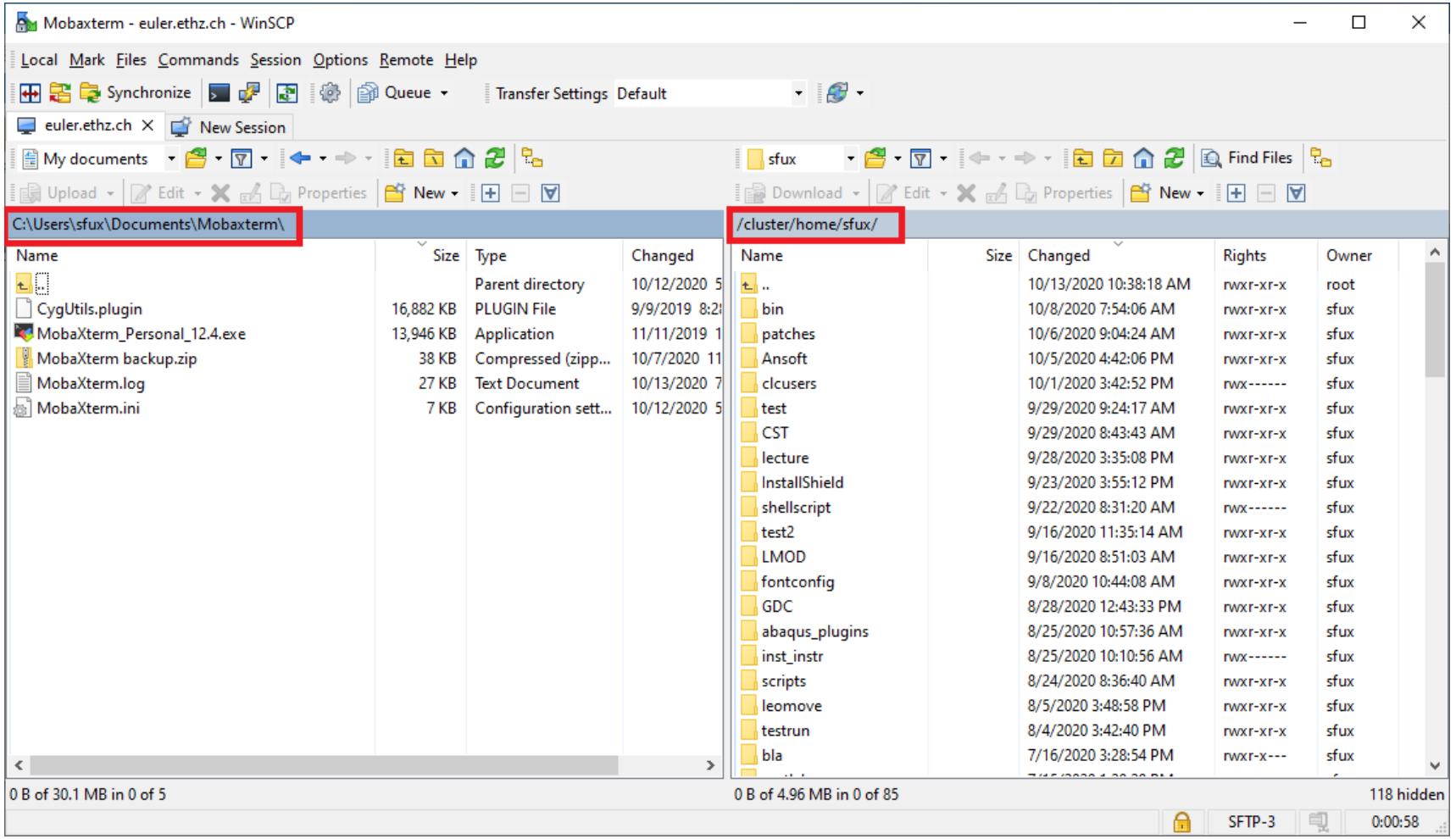
Graphical file transfer programs

Linux	macOS	Windows
FileZilla	FileZilla Cyberduck	WinSCP PSCP FileZilla Cyberduck

Data > Copying data from/to the cluster > WinSCP



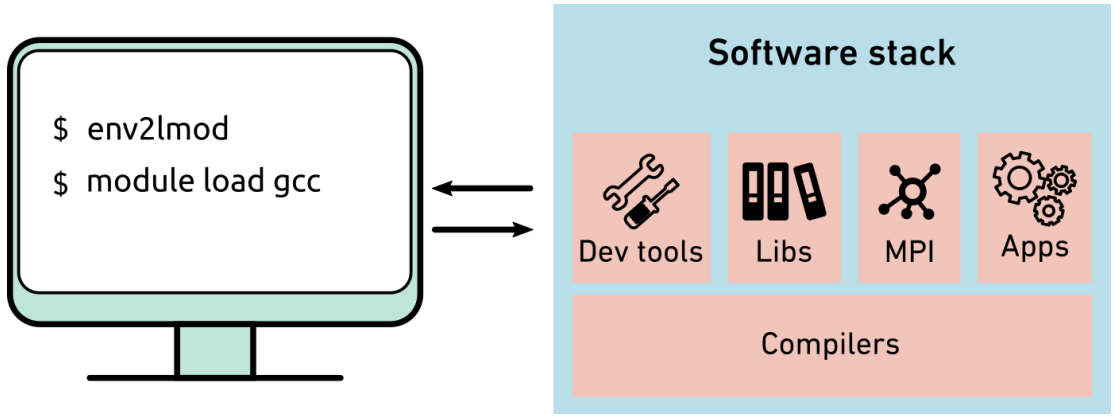
Data > Copying data from/to the cluster > WinSCP



Outlook

- Accessing the cluster
- Storage and data transfer
- Modules and applications
- Using the batch system

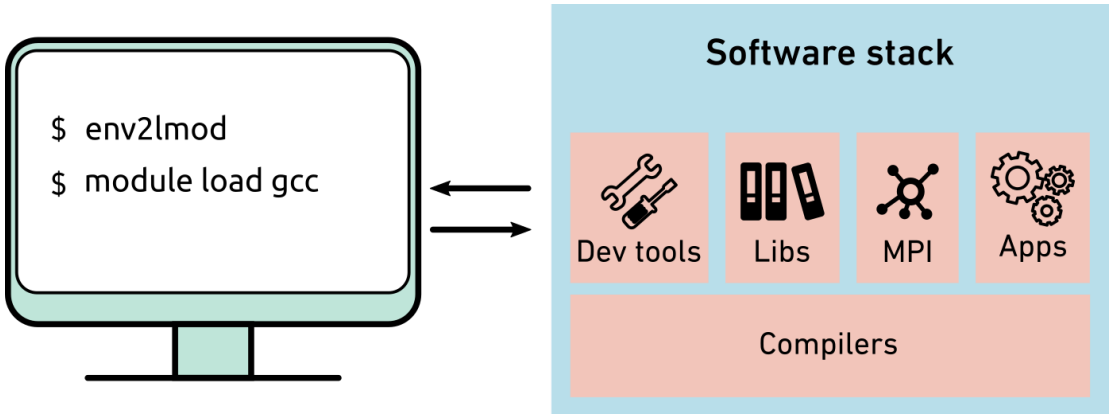
Modules



A Modules package is a tool to let users easily configure the computing environment which includes

- Development tools
- Scientific libraries
- Communication libraries (MPI)
- Third-party applications

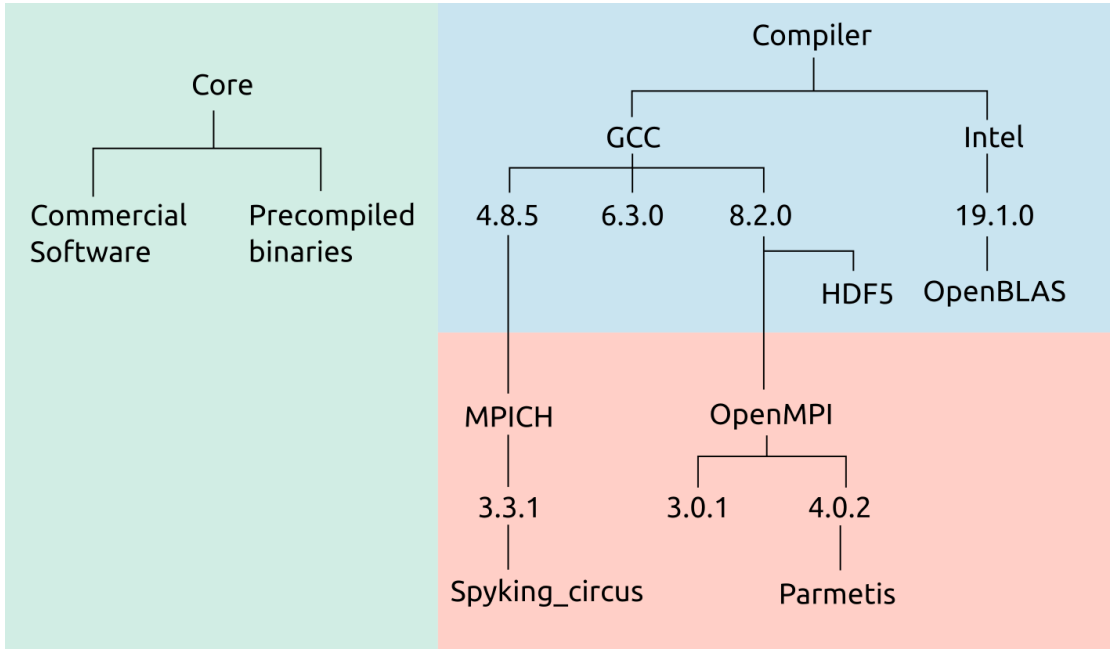
Modules



Advantages:

- Automatic configuration
- Different versions of the same software can co-exist and can be selected explicitly
- You can easily try out different tools, switch between versions, to find out which one works best for you

Modules > LMOD modules



Module hierarchy with 3 layers:

- Core layer

```
$ module load comsol/5.6
```

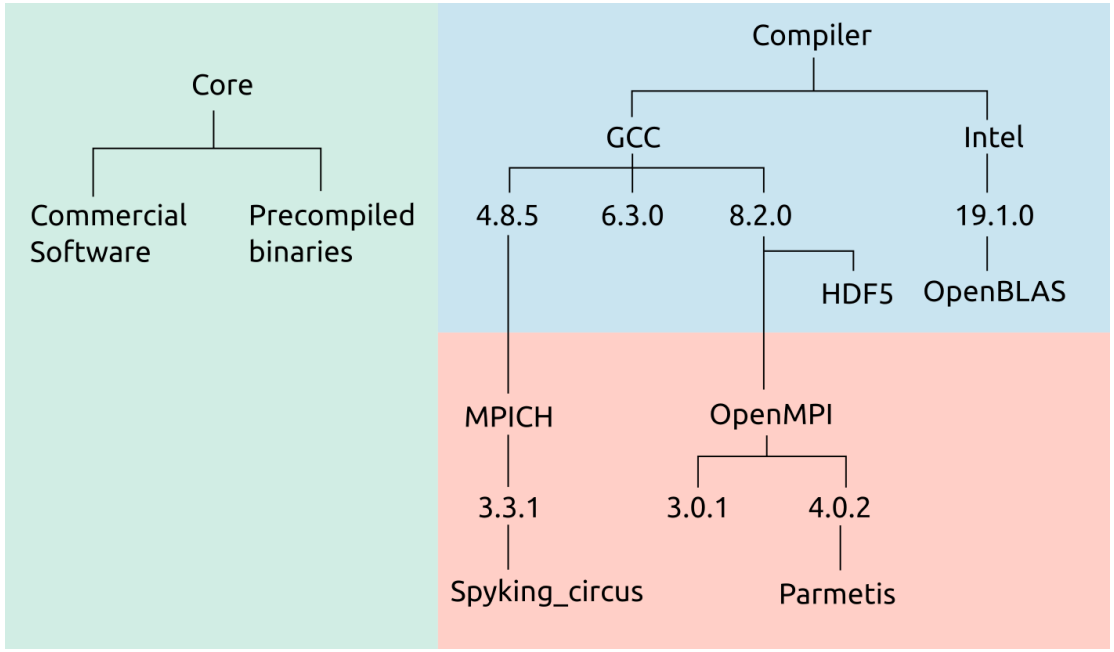
- Compiler layer

```
$ module load gcc/6.3.0 python/3.8.5
```

- MPI layer

```
$ module load gcc/6.3.0 openmpi/4.0.2 openblas
```

Modules > LMOD modules



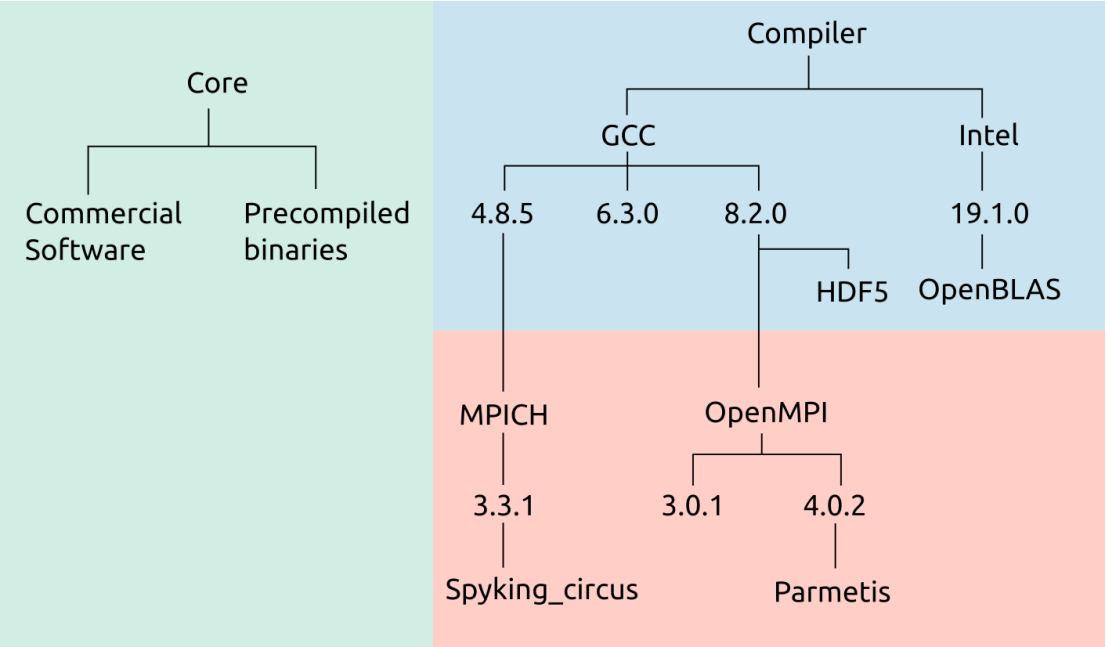
The four main toolchains are

1. GCC 4.8.5 (supports C++11 standard)
2. GCC 6.3.0 (supports C++14 standard)
3. GCC 8.2.0 (supports C++17 standard)
4. GCC 11.4.0 (supports C++20 standard)
5. Intel 19.1.0

- For each of the toolchains ~400-500 applications and libraries are available

https://scicomp.ethz.ch/wiki/Euler_applications_and_libraries

Modules > LMOD modules



- Safety rules to avoid misconfiguration
- Only one compiler/MPI combination can be loaded at the same time
- When changing the compiler or MPI, LMOD will try to reload all currently loaded modules with the new compiler/MPI

Modules > Commands (demonstration)

Load Python module in GCC/6.3.0 toolchain

```
[sfux@eu-login-31 ~]$ module load gcc/6.3.0 python/3.8.5
```

List available python module

```
[sfux@eu-login-31 ~]$ module avail python
-------------------- /cluster/apps/lmodules/Compiler/gcc/6.3.0 ---------------------
python/2.7.14    python/3.6.4    python/3.7.4    python/3.8.5 (D)    python_gpu/3.8.5
```

List all currently loaded modules

```
[sfux@eu-login-31 ~]$ module list

Currently Loaded Modules:
  1) StdEnv   2) gcc/6.3.0   3) openblas/0.2.20   4) python/3.8.5
```

Modules > Commands

`module`	get info about module sub-commands
`module avail`	list all modules available on the cluster
`module key` *`keyword`*	list all modules whose description contains *keyword*
`module help` *`name`*	get information about module *name*
`module show` *`name`*	show what module *name* does (without loading it)
`module unload` *`name`*	unload module *name*
`module purge`	unload all modules at once

Modules > How to install applications locally?

Users can install additional applications in their home directory, but only if the quotas (space: 21 GB, files/directories: 200'000) are not exceeded

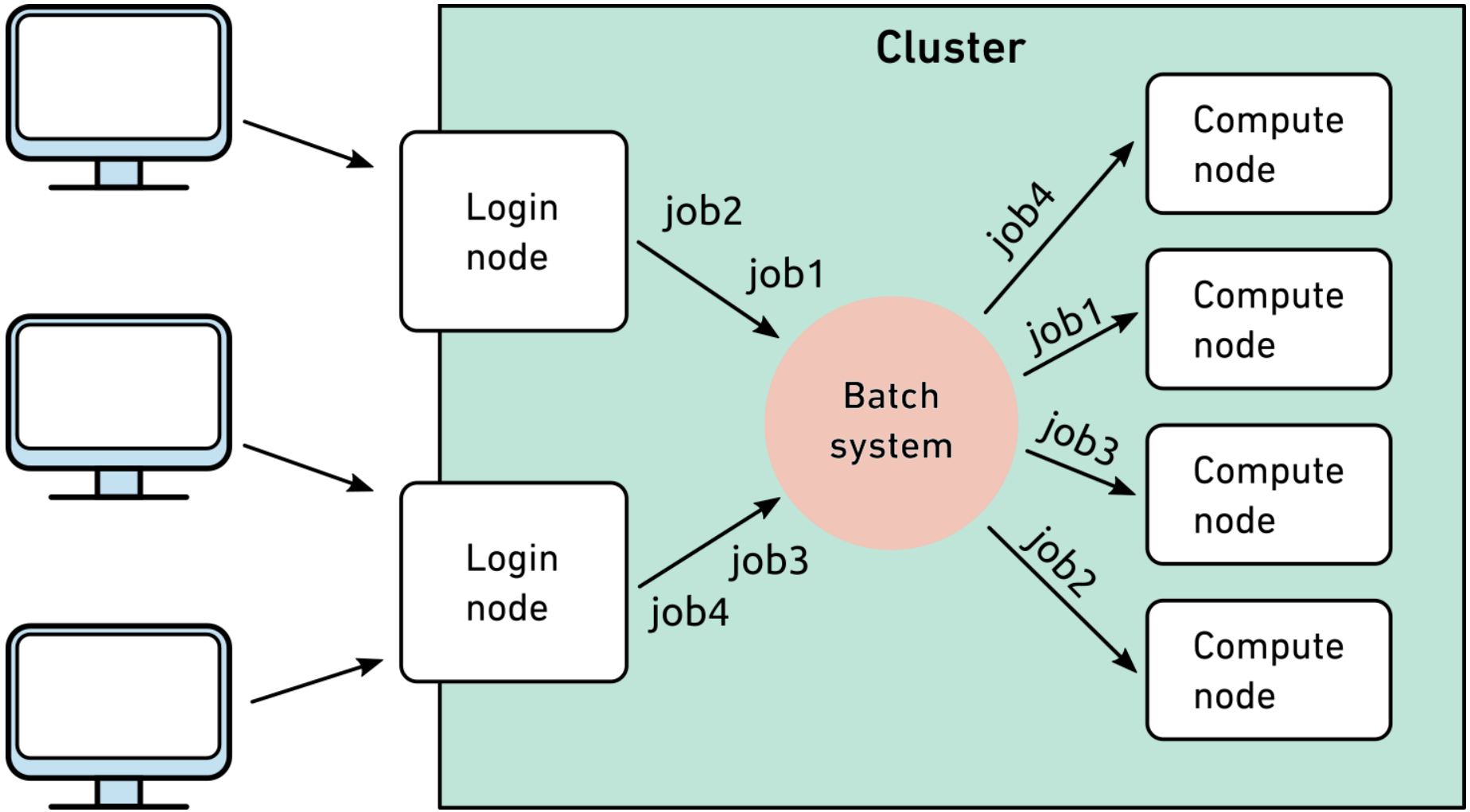
- Avoid anaconda installations as they often conflict with the files/directories quota. Alternatively, you can create a Python virtual environment.
- For Python and R, packages can easily be installed locally

```
$ pip install --user packagename
```

Outlook

- Accessing the cluster
- Storage and data transfer
- Modules and applications
- Using the batch system

Batch > Overview



Batch > Overview

- The batch system of Euler is called **SLURM** (Simple Linux Utility for Resource Management)
- SLURM manages all resources available on the cluster and allocates them to user jobs
  - Ensures that resources are used as efficiently as possible
  - Calculates user/job priorities based on a fair share principle
- All computations must be submitted to the batch system
  - There is no other way to access the cluster's compute nodes
- Please do not run computations on the login nodes
  - Login nodes may only be used for file transfer, compilation, code testing and debugging, and quick pre- and post-processing

Batch > Basic job submission

- When you submit a job via `sbatch`, you specify the desired resources for it (in terms of how long the job should run, how many cores, GPUs, etc..)

- The batch system then analyzes the requested resources and dispatches the job to a batch queue. The job will run once the desired resources are available

- If all goes well, `sbatch` :
  - Tells the job's unique identifier ("job ID") – e.g. "1010171"
  - Will automatically create an output stderr/stdout file slurm-jobID.out

- The jobID is important to check, monitor or terminate a job

- If you report a problem with a job (pending, running or done) to cluster support, then **always** provide the corresponding jobID and the slurm-jobID.out file

Batch > Basic job submission

- Use `sbatch` to submit a job to the batch system
- `sbatch [Slurm options] --wrap="job"`
- A *job* can be either …
  - a single Linux command
  - a shell script, passed via “<”
  - a program, with its path
  - a command or program, with its arguments
  - multiple commands, enclosed in quotes

```
pwd
< script
/path/to/program
cmd arg1 arg2
"cmd1 ; cmd2”
```

- We’ll talk about `sbatch`’s options later

Batch > Basic job submission > Examples

```
[sfux@eu-login-03 ~]$ sbatch --wrap="echo hello"
Submitted batch job 1010112

[sfux@eu-login-03 ~]$ sbatch < hello.sh
Submitted batch job 1010113

[sfux@eu-login-03 ~]$ sbatch --wrap="./bin/hello"
Submitted batch job 1010114

[sfux@eu-login-03 ~]$ sbatch --wrap="date; pwd; ls -l"
Submitted batch job 1010115

[sfux@eu-login-03 ~]$ sbatch --wrap="du -sk /scratch > du.out"
Submitted batch job 1010116
```

Batch > Resource requirements

- By default, a job will get 1 core for 1 hour with 1000MB of memory
  - If you need more time and/or processors and/or memory, you must request them
  - Maximum run-time on Euler is 15 days

- These resources are passed to `sbatch` using options :

```
sbatch --time=HH:MM:SS --ntasks=number_of_processors
                                    --mem-per-cpu=2000 --wrap="command”
```

- If you don't specify any unit for the memory request, then the integer value will be interpreted as MB. If you specify values in GB, then you need to add the suffix "g" (in the example above, you would write 2g instead of 2000)

Batch > `sbatch` options

`--ntasks=N`	request *N* cores (*`--nodes=1`* allocates all cores on a single node)
`--time=HH:MM:SS`	request *a runtime of* *`HH:MM:SS`*
`--output="filename"`	redirect job's standard output to *`filename`*
`--error="filename"`	redirect job's error messages to *`filename`*
`--mem-per-cpu=YYY`	request *`YYY`* MB memory per core (or add suffix "g" to specify GBs)
`--tmp=YYY`	request *`YYY`* MB of local scratch space (or add suffix "g" to specify GBs)
`--job-name="jobname"`	assign a *`jobname`* to the job
`--account="share"`	run job under a particular Euler share *`"share"`*
`--mail-type=BEGIN`	send an email when the job begins
`--mail-type=END,FAIL`	send an email when the job ends (finishes successfully or fails)

Batch > `sbatch` GPU options

`--gpus=N`	request *N* gpus
`--gpus=MODEL:N`	request *N* gpus of model *MODEL* (for instance `--gpus=rxt_3090:1`)
`--gres=gpumem:XXg`	request a GPU with at least *XX* GB GPU memory

https://scicomp.ethz.ch/wiki/GPU_job_submission_with_SLURM

Batch > #SBATCH pragmas

- `sbatch` options can be specified either on the command line or inside a job script using the #SBATCH pragma, for example

```
#!/bin/bash
#SBATCH --ntasks=24                    # 24 cores
#SBATCH --time=8:00:00                 # 8-hour run-time
#SBATCH --mem-per-cpu=4000             # 4000 MB per core
cd /path/to/execution/folder
module load gcc/6.3.0 openmpi/4.0.2
mpirun myprogram arg1
```

- In this case, the script can be submitted using the “<” operator

```
$ sbatch < script
```

- `sbatch` options specified on the command line override those inside the script

```
$ sbatch --ntasks=48 < script
```

Batch > Job monitoring

`squeue`	check the state of a job in the queue
`myjobs`	detailed information about a job
`scancel`	kill a job

Advanced commands :

`scontrol`	check resource usage of a job
`sstat`	check information about a running job
`sacct`	detailed information about pending, running and finished jobs

Batch > Job monitoring > `squeue`

- After submitting a job, the job will wait in a queue to be run on a compute node and has the pending status (PD). You can check the job status with the `squeue` command

```
[sfux@eu-login-41 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1433323 normal.4h     wrap     sfux  PD       0:00      1 eu-g1-026-2
           1433322 normal.4h     wrap     sfux  R        0:11      1 eu-a2p-483
```

Batch > Job monitoring > `myjobs`

- Detailed information about a job can be provided by myjobs command:

```
$ myjobs -j 6038307
Job information
 Job ID                          : 6038307
 Status                          : RUNNING
 Running on node                 : eu-g3-022
 User                            : nmarounina
 Shareholder group               : es_cdss
 Slurm partition (queue)         : gpu.24h
 Command                         : script.sbatch
 Working directory               : /cluster/home/nmarounina
Requested resources
 Requested runtime               : 08:00:00
 Requested cores (total)         : 12
 Requested nodes                 : 1
 Requested memory (total)        : 120000 MiB
Job history
 Submitted at                    : 2023-01-09T15:56:09
 Started at                      : 2023-01-09T15:56:38
 Queue waiting time              : 29 s
 Resource usage
 Wall-clock                      : 00:00:36
 Total CPU time                  : 00:00:00
 CPU utilization                 : 0%
 Total resident memory           : 2.94 MiB
 Resident memory utilization     : 0%
```

Batch > Job monitoring > `scancel`

```
[sfux@eu-login-15 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           1525589 normal.24   sbatch     sfux  R       0:11      1 eu-a2p-373
[sfux@eu-login-15 ~]$ scancel 1525589
[sfux@eu-login-15 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
[sfux@eu-login-15 ~]$
```

Options:

job-ID	kill *job-ID*
`--name=`*jobname*	kill all jobs with name *jobname*
`--user=`*username*	kill all jobs from user *username*
`--state=state`	kill all jobs in state *state* (states: PENDING, RUNNING or SUSPENDED)

Dos and don’ts

Dos

- Understand what you are doing
- Ask for help if you don’t understand what you are doing
- Optimize your workflow to make it as efficient as possible
- Keep in mind that our clusters are shared by many users
- Choose the file system you want to use carefully

Don’ts

- Don’t waste CPU time or disk space
- Don’t run applications on the login nodes
- Don’t write large amounts of data to standard output
- Don’t create millions of small files
- Don’t run hundreds of small jobs if the same work can be done in a single job

Getting help

- Wiki: https://scicomp.ethz.ch

- Ticket system
  - https://smartdesk.ethz.ch (ETH account authentication)
  - Please describe your problem as accurately as possible

- E-mail
  - cluster-support@id.ethz.ch
  - Please do not send questions to individual members of the team

- Person-to-person
  - Contact us to set up an appointment at your place
  - Visit us at Binzmühlestrasse 130

Jupyter hub demonstration :

- https://jupyter.euler.hpc.ethz.ch

Questions?