ETH zürich

# Euler: ETH's own HPC

Practical example

# Why using a HPC?

Your personal computer can rarely deal with the amount of computational resources needed for big data.

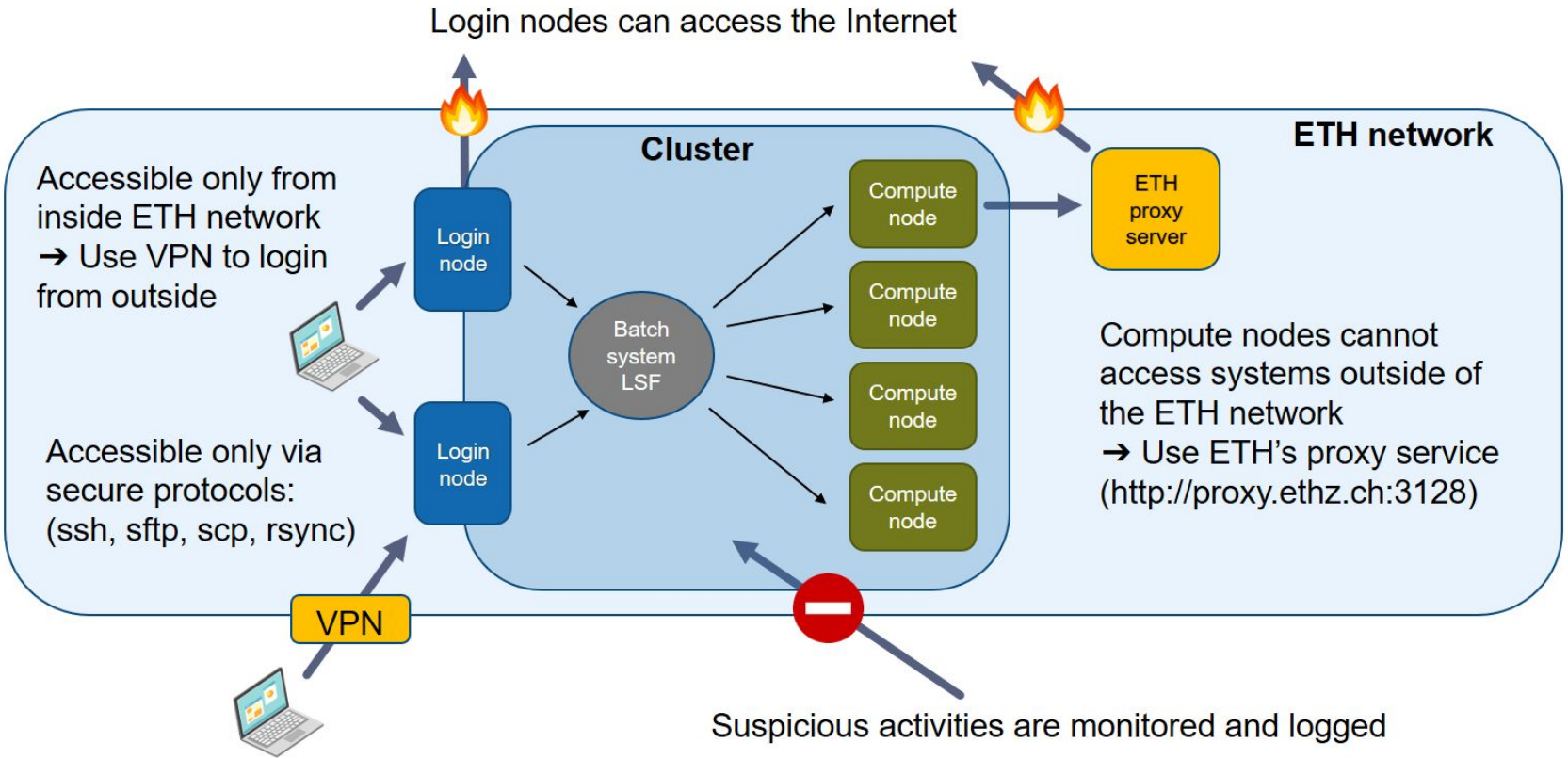
High performance computing (HPC) can leverage distributed compute resources to solve complex problems with large datasets

- Terabytes to petabytes to zetabytes of data
- Results in minutes/hours instead of days or weeks (parallelization)

Each HPC is composed of several compute nodes of variable amount of memory which are interconnected in a network.

Jobs are submitted with a management system and processed with the requested resources.

# Euler, ETH's own HPC



# Connect to Euler

## Access ssh from your Windows computer

Enable ssh like here

Then use ssh in PowerShell window or a Command Prompt window

## Access ssh from Mac

Open a Terminal

```
ssh accountname@euler.ethz.ch
(your LDAP password)
```

# Euler structure

## Personal storage

**Home** (safe long term storage of important and critical data - maximum of 16 GB and a maximum of 100'000 files and directories)

`/cluster/home/`*`username`*

**Scratch** (used for short-term storage of larger amounts of data, data removed after 2 weeks - maximum of 2.5 TB and a maximum of 1'000'000 files and directories)

`/cluster/scratch/`*`username`*

Type `lquota` to see your own resources

# Very brief intro to linux/bash

**ls -ltra:** Display list of files and folders sorted by time in reversed order (newest files at the bottom), including hidden files.

**cd:** Change to your home directory.

**pwd:** Print working directory.

**echo:** Print a string to the standard output.

**less, cat, more:** Display the content of a file.

**cp:** Copy a file.

**mv:** Rename/move a file.

**rm:** Delete a file.

**mkdir:** Create a directory.

## Resources

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

https://gitlab.ethz.ch/thealternative/courses/-/tree/master/bash_course

# Prepare the notebook in the script

Change all the file paths (input and output) to the appropriate paths in Euler

Google colab: File → Download → .py

Jupyter Notebook: File → Download as → .py

# Move the python script to the appropriate directory

## From terminal (not in Euler) - Windows, use putty

```
scp /file/on/your/computer username@euler.ethz.ch://location/on/euler/
```

## Or open

```
wget location_of_script.py
```

# Move the output files back on your computer to inspect

## From terminal (not in Euler) - Windows, use putty

```
scp username@euler.ethz.ch ://location/of/files/on/euler/filenames /location/on/your/computer
```

## To export all the files in the directory

```
scp username@euler.ethz.ch ://location/of/files/on/euler/* /location/on/your/computer
```

# Edit a file in the terminal using vim

```
vim name_of_file.py
```

## Edit the file

Press `'Esc' + i` → insert the text

## Save the file

Press `'Esc' + :wq` (write and quit)

**Wonder how to get started with `vim`?**

Type in `vimtutor` in your terminal

# Submit the job

## Submit the script

```
sbatch --wrap="python /location/of/script/script.py"
```

## Submit the script with specific resources (limited by your cluster privileges)

```
sbatch --ntasks=4 --time=24:00:00 --mem-per-cpu=10000 --mail-type=END --wrap="python test.py"
```

## Check the status of your job

`squeue` or `scontrol show jobid -dd JOBID`

## Kill the job

`scancel`

**Resources**

LSF to Slurm quick reference guide

ETH Scientific computing Wiki

# Submit a job

## Load python (if you are not in your virtual environment)

```
module load python
```

## Create a new folder and move there

```
mkdir test
cd test
```

## Create a python sample script

```
echo “print(‘Hello World’)” >> test.py
```

## Submit the script

```
sbatch --wrap="python test.py"
sbatch --ntasks=4 --time=24:00:00  --mem-per-cpu=10000 --mail-type=END --wrap="python test.py"
squeue
scancel -j JOBID
```

# How to prepare a script to submit to Euler

- Import all the libraries needed
- Specify the exact pathway of your files (if on the cluster)
- Specify the output file/files and directory
- Run a test script first, then request more computational resources when debugged
- Give enough resources and +20% more than expected for the submission