

Delft Blue Tutorial

For machine learning / scikit with conda

Tom Viering

SSH & FileTransfer tools

- Mac: Core Shell <https://codinn.com/shell/>
- Windows: MobaXTerm <https://mobaxterm.mobatek.net/>
- Linux: everything can be done within your OS already
 - Terminal: use ssh command (first ssh to student-linux, then ssh to Delft Blue)
 - You can use the GUI of your OS to transfer files :

Most advanced Linux file managers are able to use SFTP (the SSH file transfer protocol), along with Ubuntu's default called **Nautilus**

In Nautilus, you can use *File* → *Connect to Server...* in order to connect to remote file systems using various protocols, including SFTP, FTP, FTPS and SMB.

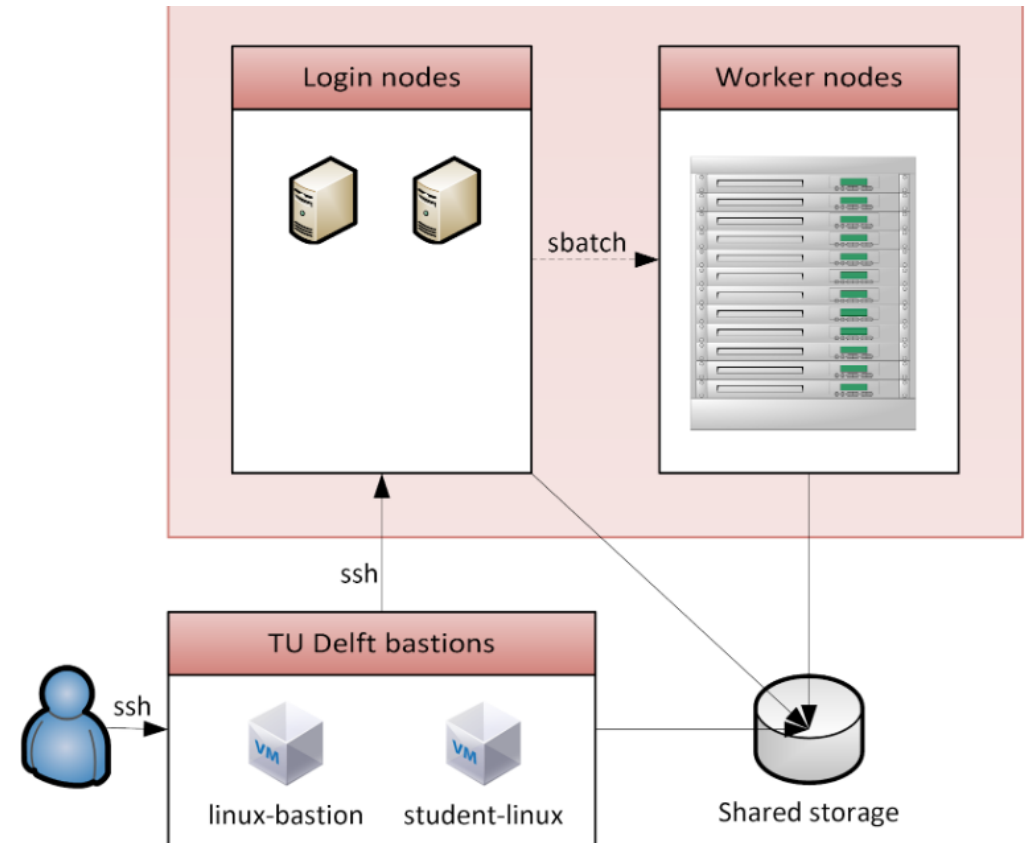
Enter the necessary information (server address, user and password might be asked when connecting) and the remote access will be mounted temporarily in your system, until you unmount it again.

In the following

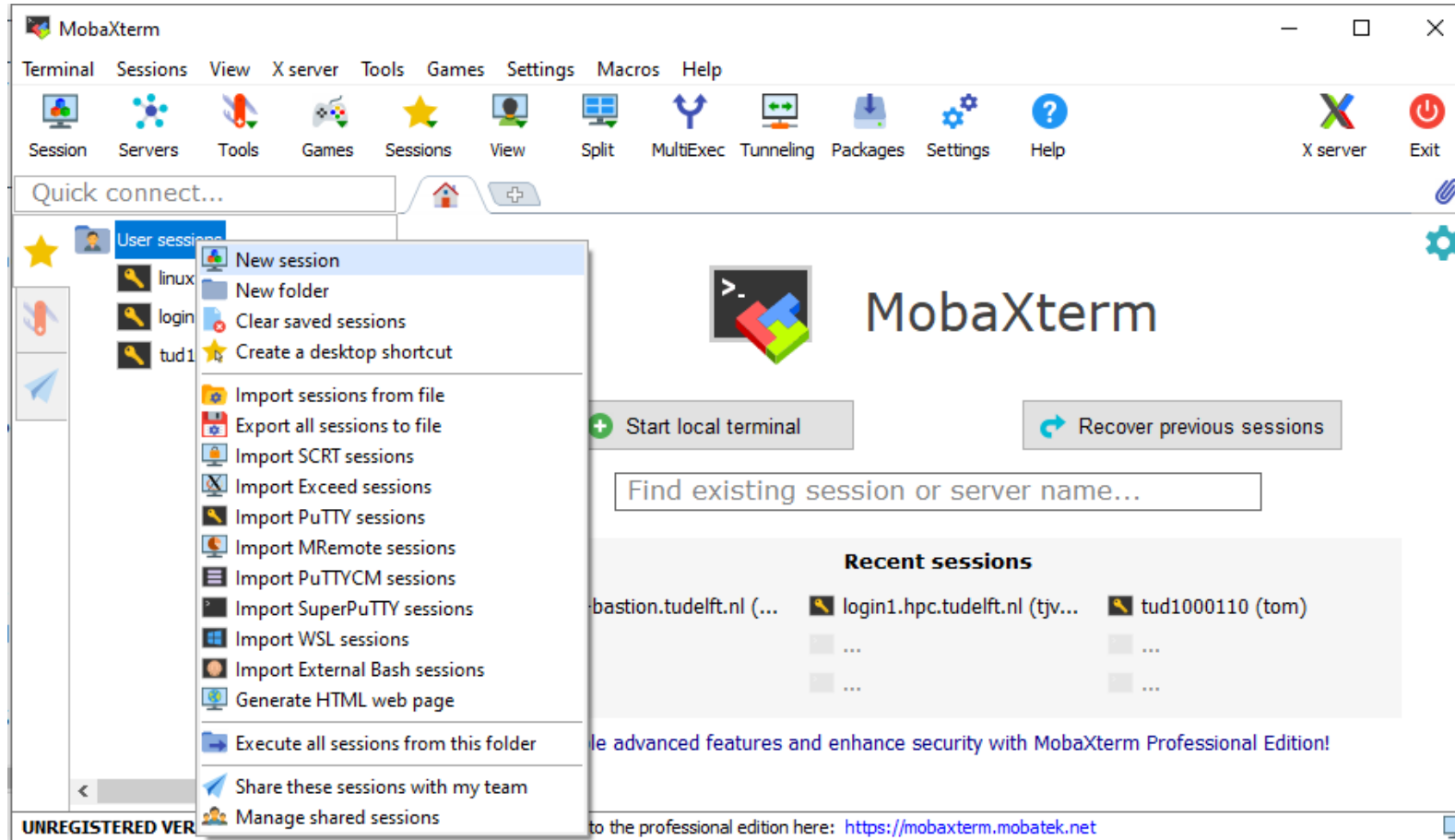
- Sometimes, there will be my netid, you will have to change it to your own!

How to connect

- Set up an SSH tunnel
- `linux-bastion.tudelft.nl` for staff
- `student-linux.tudelft.nl` for students



Make new session




Fill out the info

Session settings



Basic SSH settings

Remote host * ☒ Specify username  Port

Advanced SSH settings

Terminal settings

Network settings

Bookmark settings



Proxy settings (experimental)

Proxy type: Host: Login: Port:

OK


Cancel


Set up the gateway (tunnel)


MobaXterm jump hosts configuration ×


Define one or several SSH jump hosts
(Jump through one or several SSH servers in order to reach your end-server)

Gateway host	Username	Port	<input type="checkbox"/> Use SSH key
linux-bastion.tudelft.nl	tjviering	22	

 student-linux.tudelft.nl for students



 Add another jump host


 OK

Session settings



Basic SSH settings

Remote host *

☒ Specify username 

Port

Advanced SSH settings

Terminal settings

Network settings

Bookmark settings



Proxy settings (experimental)

Proxy type:

Host:

Login:

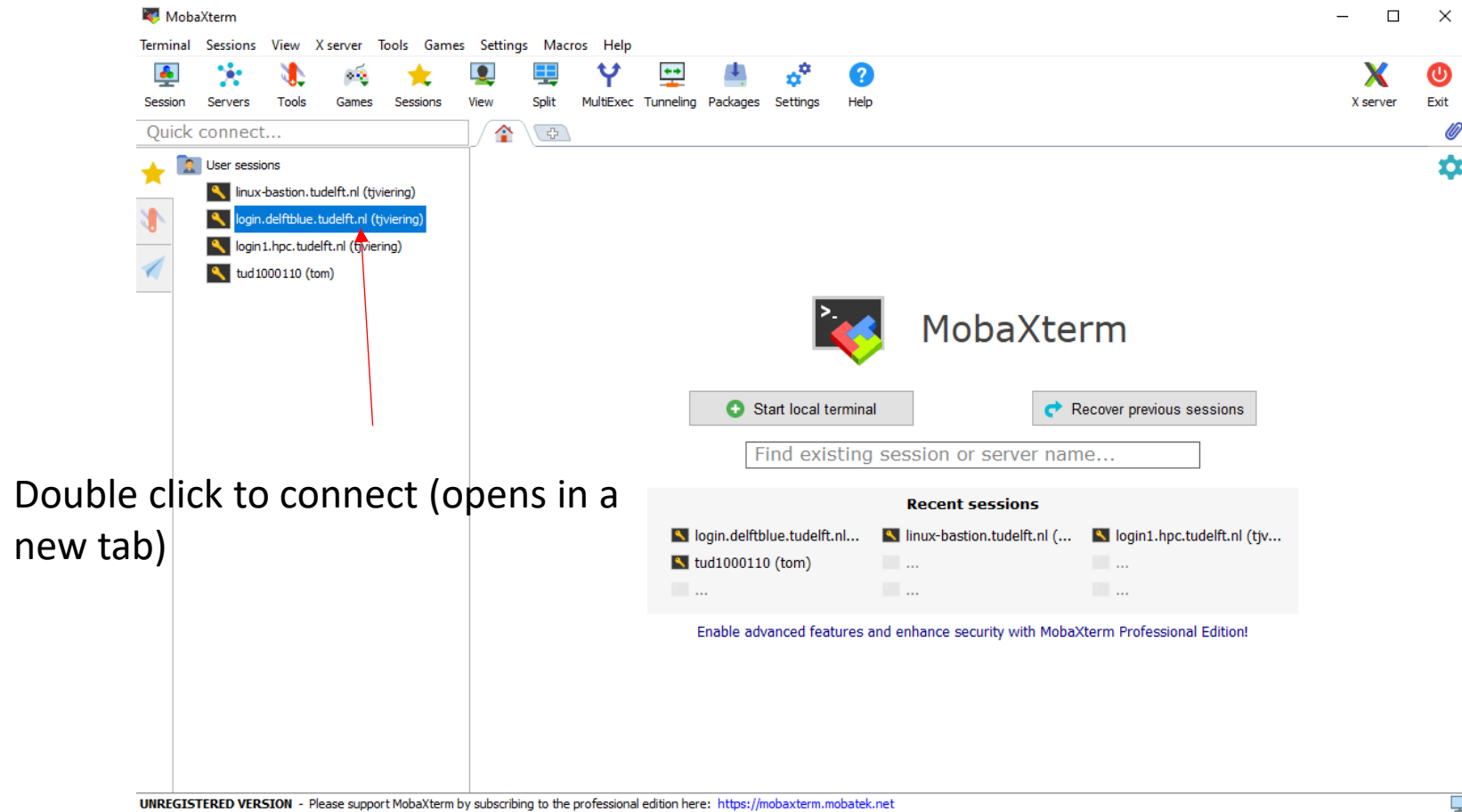
Port:

OK

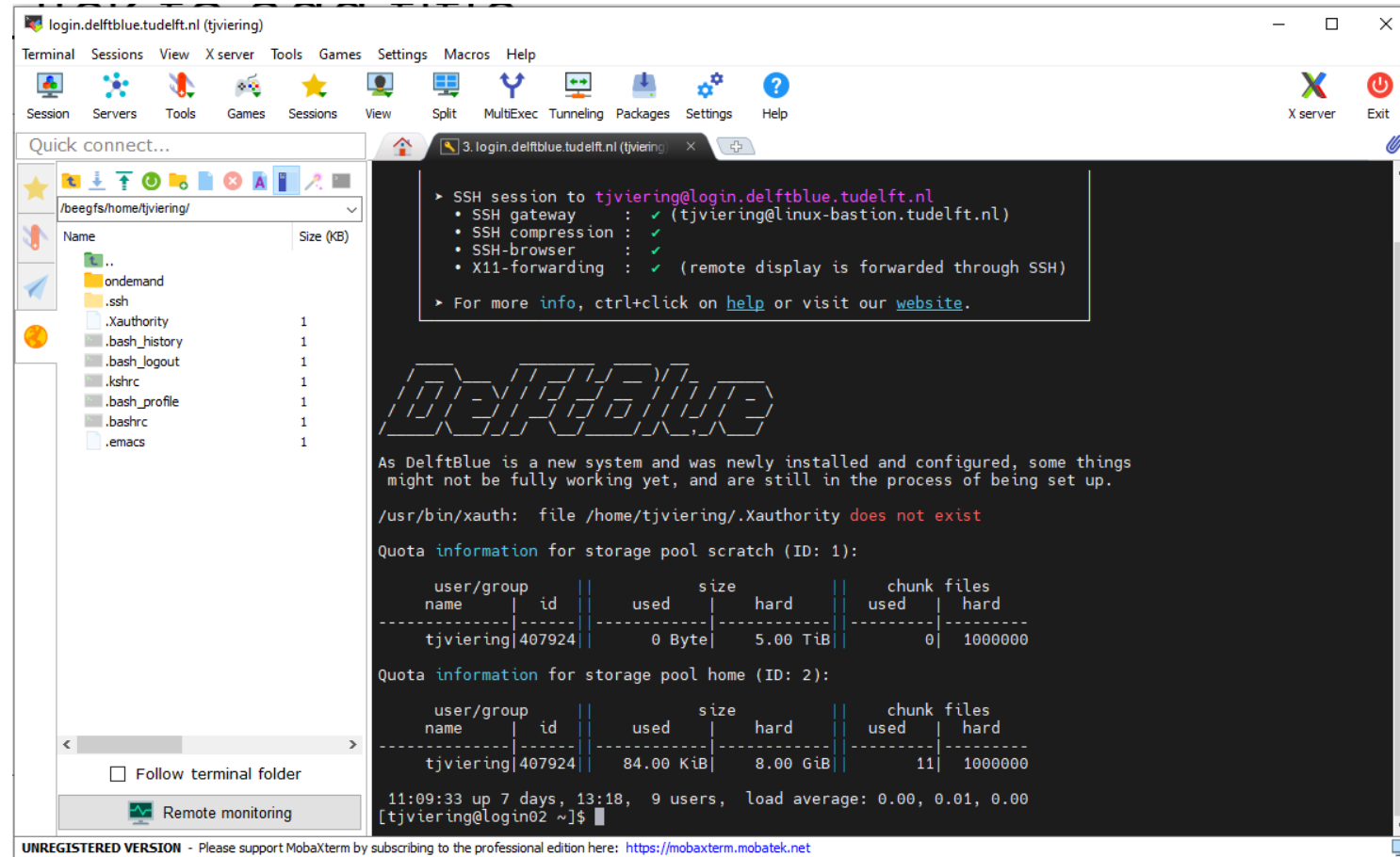
Cancel

This may take a while and may ask for a password (its testing the connection)
For your sanity: I suggest to let MobaXterm remember the passwords

Start new session



Connected, hooray!



Important directories

`/beegfs/home/tjviering/`

home dir (8GB space, backed up)

Use it to store your code, python environment, etc. small files!!!

`/beegfs/scratch/tjviering/`

Extremely fast and large storage (5TB per user)

Use for storing datasets, results, etc.

Not backed up: once in a while everything is deleted (last time May 5th)!
They will warn you via [Mattermost DHPC channel](#)

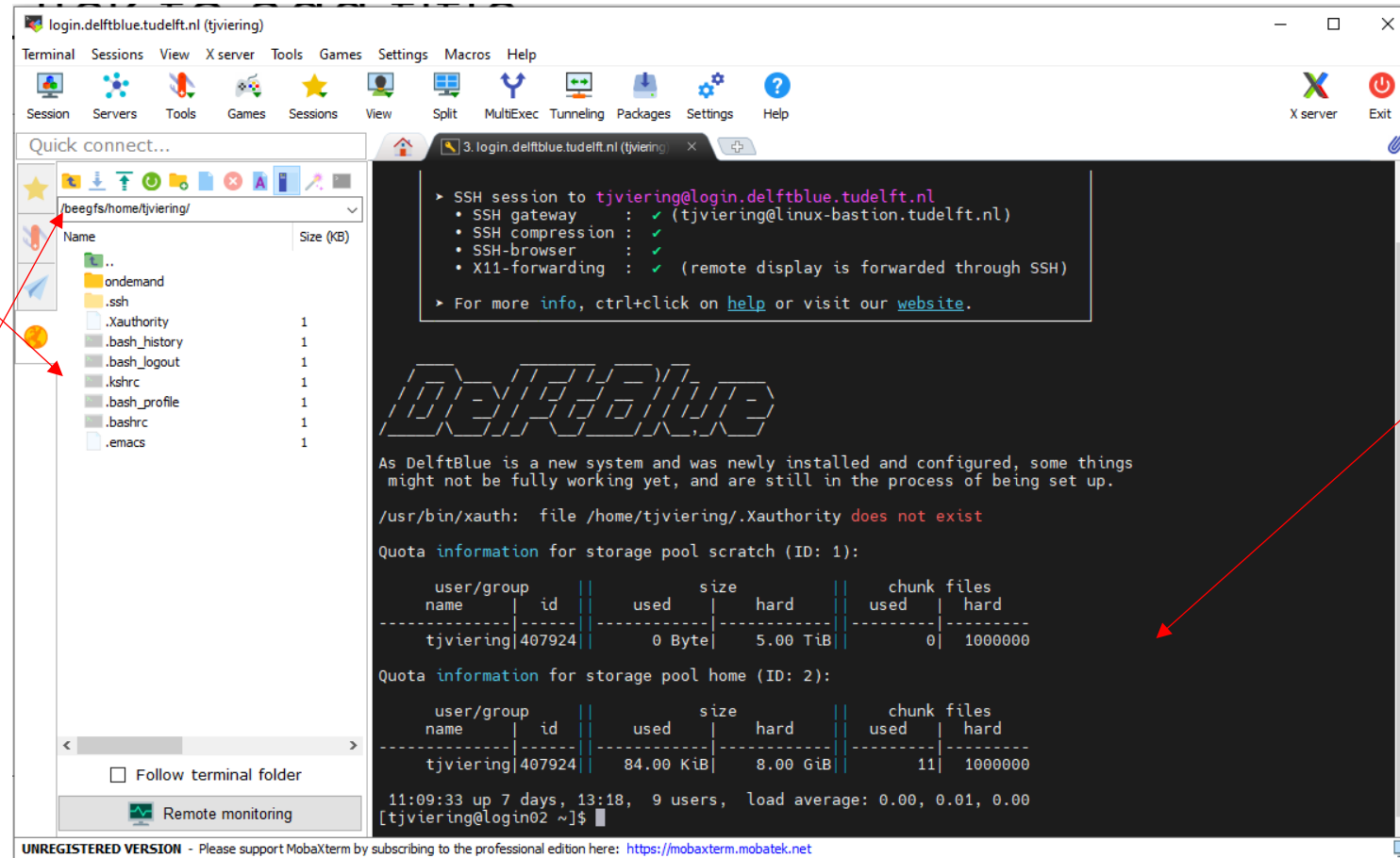
As soon as your experiment is done, transfer the files to your home directory or laptop to make sure you don't lose them.

Xterm interface

File explorer

Use for
uploading,
downloading,
moving, etc. of
files and folders

Current directory



Terminal
Used for running
commands

Navigating using the commandline

- `pwd`: displays the current directory of the shell
- `mkdir`: makes new directory
- `ls`: displays what is in the current directory
- `cd`: navigate to a directory (“`cd ..`” moves to the parent directory)
- `exit`: closes the connection to this server (useful in interactive job)

```
[tjviering@cmp082 ~]$ pwd
/home/tjviering
[tjviering@cmp082 ~]$ mkdir test_directory
[tjviering@cmp082 ~]$ ls
ondemand  test_directory
[tjviering@cmp082 ~]$ cd test_directory
[tjviering@cmp082 test_directory]$ ls
[tjviering@cmp082 test_directory]$ pwd
/home/tjviering/test_directory
[tjviering@cmp082 test_directory]$ cd ..
[tjviering@cmp082 ~]$ ls
ondemand  test_directory
[tjviering@cmp082 ~]$ █
```

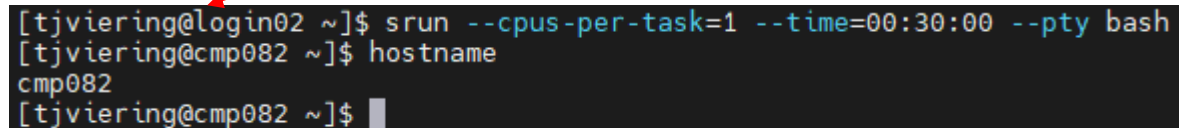
Debugging your code

- Run this to start an interactive session:

```
srun --cpus-per-task=40 --time=01:00:00 --  
mem=100G --pty bash
```

- This will start a job of 30 minutes in the cluster with 2G mem
- We can use it to debug our code on the cluster and make sure everything works before we do a real big experiment!

Before I was in the login02 node (this node is used to submit jobs)



```
[tjviering@login02 ~]$ srun --cpus-per-task=1 --time=00:30:00 --pty bash  
[tjviering@cmp082 ~]$ hostname  
cmp082  
[tjviering@cmp082 ~]$ █
```

A terminal window showing the execution of the `srun` command to start an interactive session. The prompt changes from `[tjviering@login02 ~]` to `[tjviering@cmp082 ~]` after the command is executed. The `hostname` command is then run, outputting `cmp082`. A red arrow points from the text "Before I was in the login02 node" to the `login02` part of the first prompt. Another red arrow points from the text "Now I am in a compute node" to the `cmp082` part of the second prompt.

Now I am in a compute node (cmp082 in this case, depends on what is available)
The command “hostname” will also tell you in which node you currently are

Show time left

- Run this to show how much time is left in your interactive job:




```
export PS1="$PS1 \$(squeue -h -j $SLURM_JOBID -o %L) "
```

- If the time is up, your job is killed
- If you exceed the memory, your job may be killed (or the program will be killed that required too much memory)

Loading pre-installed software

- There is already a lot of software available on DelftBlue
- We will only install what is not already available (!)

- Execute this:

- `module load 2022r1`  Load DelftBlue software stack
- `module avail`  Shows what is available
- `module load compute`
- `module load miniconda3`  I will use conda to manage my python environment. You can also use pip or load preinstalled modules directly (e.g. py-numpy), but the 2022r1 don't have pandas....

Finding software

- `module spider conda`
- Searches for the module “conda” and displays the search result

```
[tjviering@cmp082 ~]$ module spider conda
```

```
-----  
miniconda3:
```

```
-----  
  Versions:
```

```
    miniconda3/4.10.3-buki4rd  
    miniconda3/4.10.3-eyq4jvx
```

```
-----  
For detailed information about a specific "miniconda3" package (including how to load the modules) use the module's full name.  
Note that names that have a trailing (E) are extensions provided by other modules.  
For example:
```

```
    $ module spider miniconda3/4.10.3-eyq4jvx
```

```
-----  
[tjviering@cmp082 ~]$ module spider miniconda3/4.10.3-buki4rd
```

```
-----  
miniconda3: miniconda3/4.10.3-buki4rd
```

```
-----  
You will need to load all module(s) on any one of the lines below before the "miniconda3/4.10.3-buki4rd" module is available to load.
```

```
    2022r1  gpu
```

```
Help:
```

```
    The minimalist bootstrap toolset for conda and Python3.
```

Warning: disk space.

- Be careful: by default, Conda installs stuff to your home directory. This directory has a hard limit of 8GB of space.
- If you get errors and don't know why, check how much space is left.
- At login, its displayed how much space there is left in your homedir:

```

DelftBlue
As DelftBlue is a new system and was newly installed and configured, some things might not be fully working yet, and are still in the process of being set up.

Quota information for storage pool scratch (ID: 1):

```

user/group	id	used	size	hard	chunk	files
name					used	hard
tjviering	407924	0 Byte	5.00 TiB		0	1000000

```

Quota information for storage pool home (ID: 2):

```

user/group	id	used	size	hard	chunk	files
name					used	hard
tjviering	407924	3.77 GiB	8.00 GiB		50406	1000000


```

15:08:53 up 7 days, 17:18, 14 users, load average: 0.04, 0.09, 0.08
[tjviering@login03 ~]$
```

Scratch space

Homedir

Create the environment

- You need to load conda first, then you can create your environment:
- **(You can do this on a login node also! Maybe more convenient, because it can be time consuming!)**
- `conda create -n myenv python=3.9 scikit-learn pandas=1.4.1
tqdm numpy scipy`


I use this, because on my laptop I use this pandas version. They need to be the same, otherwise, you cannot load the pickle files from the server.
- Do not interrupt Conda with CTRL+C, otherwise it may corrupt some files...
- Make sure you do not make an environment with the same name that already exists in your Conda installation, this may lead to problems... (`conda info --envs`)
- I don't know if this is because DelftBlue is new, but I had to rerun this command a few times to get stuff working. I got some strange errors about failing to link packages, and it had to redownload packages a few times (got corrupted during download?). So be patient 😊

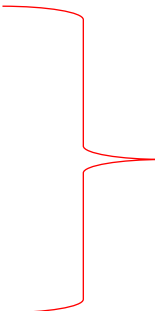
Checking environment 1/2

- First, you need this piece of code to activate conda:
- `unset CONDA_SHLVL`
- `source "$(conda info --base)/etc/profile.d/conda.sh"`
- Now, you can activate the environment:
- `conda activate myenv5`
- After activating, you will see that the environment is active because your commandline starts with (myenv5):
- `(myenv5) [tjviering@cmp081 zh_CN]$`

Checking environment 2/2

- Now let us check if all packages can be imported in Python:

```
[tjviering@cmp081 zh_CN]$ module load compute
[tjviering@cmp081 zh_CN]$ module load miniconda3
[tjviering@cmp081 zh_CN]$ unset CONDA_SHLVL
[tjviering@cmp081 zh_CN]$ source "$(conda info --base)/etc/profile.d/conda.sh"
[tjviering@cmp081 zh_CN]$ conda activate myenv5
(myenv5) [tjviering@cmp081 zh_CN]$ python
Python 3.9.12 (main, Apr 5 2022, 06:56:58)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import scipy as sp
>>> import scipy.optimize
>>> import numpy as np
>>> import pandas as pd
>>> from tqdm import tqdm
>>> import argparse
>>> from hashlib import sha256
>>> exit()
```



We get no errors, it works! 😊

Upload your scripts and data

- You do heavy computations on the cluster via jobfiles.
- **(Never run heavy computations on the login node!!!)**
- First, I put my code in the scratch directory:
`/beegfs/scratch/tjviering/`
- I put the following files there from [our github](#):
`fit_database.py`
`database-accuracy.csv`
- The csv file can be found in:
`lcdb/python/lcdb/database.tar.7z`

Example jobfile

- Upload this example jobfile to “testjob.sh” in scratch:

```
#!/bin/sh
#SBATCH --job-name="lcfit_test"
#SBATCH --partition=compute
#SBATCH --time=01:00:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=512
#SBATCH --output=lcfit0.txt
#SBATCH --error=lcfit0.txt

module load 2022r1
module load compute
module load miniconda3

unset CONDA_SHLVL
source "$(conda info --base)/etc/profile.d/conda.sh"
conda activate myenv5

cd /beegfs/scratch/tjviering/
srun python fit_database.py 0
```

Jobname

This partition only uses CPU (fine for our project!)

1 hour requested (max = 24 hours)

I only use 1 core, for multicore jobs [see here](#)

512 MB requested (for GB type 10G for example)

Where to write the output and errors of this script

Load preinstalled software

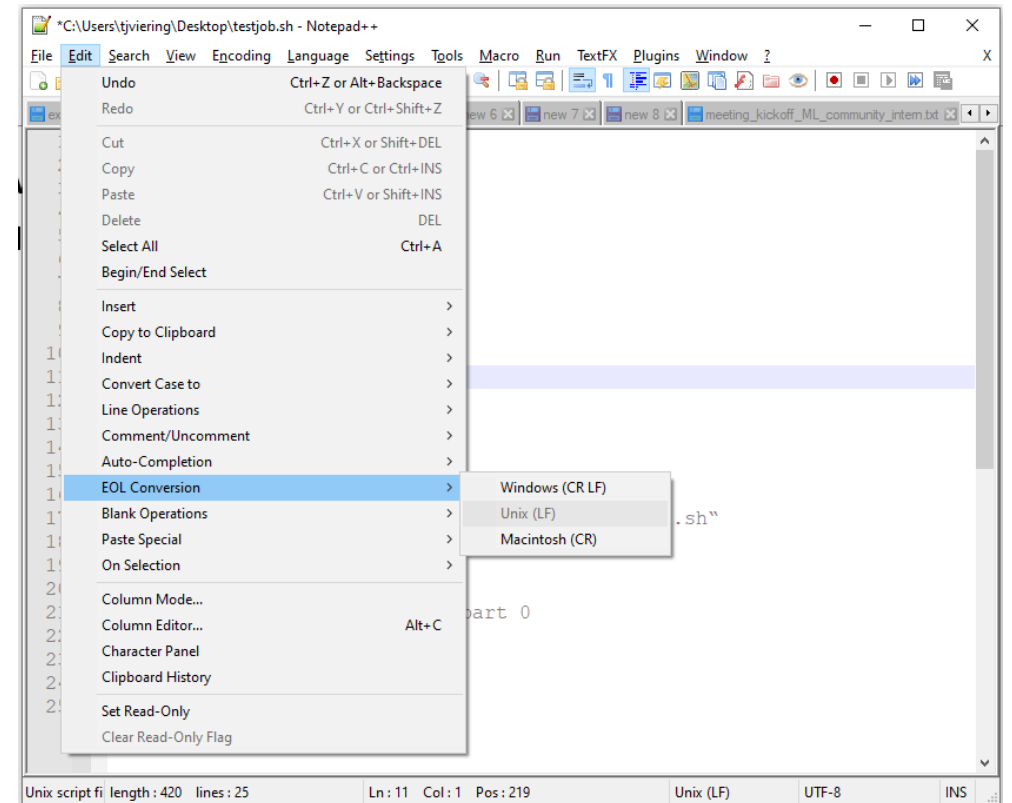
Activate conda environment

Go to the right directory

Run the 0th part of fitting the database
Use srun for a heavy computation (this will collect runtime statistics such as memory used, etc.)

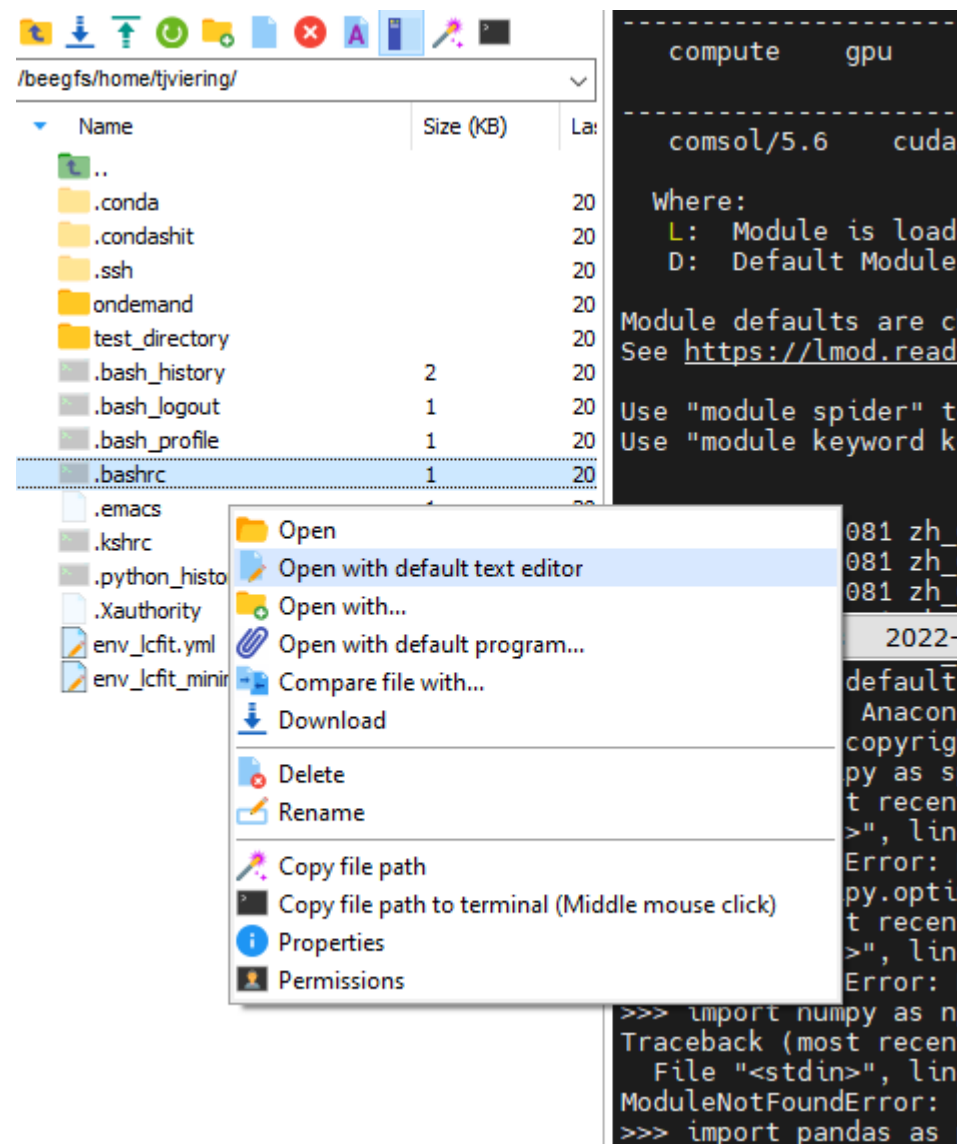
On Windows

- If you use a Windows editor to create jobfiles, this will lead to problems on the cluster, because you may use DOS linebreaks
- Use Notepad++ and fix it:
- Convert line endings before you upload
- Edit -> EOL Conversion -> Unix (LF)
- Save & upload



Useful aliases 1/2

- Open the file
/beegfs/home/**tjviering**/
.bashrc



Useful aliases 2/2

- At the top, add the following lines:

```
alias st='sacct --  
format=JobID,JobName%30,State,Elapsed,TimeLimit,AllocNodes,Pr  
iority,Start,NodeList'  
alias sq='squeue -u tjviering --start'  
alias sq2='squeue -u tjviering --start --format  
=JobName%30'
```

- If you click File->Save, or CTRL-S, the editor will ask you whether to update the file on the cluster, click Yes.
- In the terminal, run `source ~/.bashrc`
- Now you can use `st`, `sq`, `sq2` commands in the terminal

Overview of SLURM commands

`sbatch jobfile.sh`
node (check hostname)

Submits jobfile.sh job to SLURM. You can only do this from the login

SLURM will put the jobs in a queue, until there is a spot for your job to run. You can see why it is in the queue by running:

`sq`

List of jobs in the queue (use `squeue` to see other peoples job also)

`st`

Overview of your jobs (incl. queue, finished, killed, etc.)


`scancel`

Use this to cancel jobs

e.g. `-u tjviering` to cancel all of your jobs (replace netid), or you can cancel a specific job with `scancel jobid`. Find the jobid from `st`. You may need this if you find a bug in your code or jobfile.

st output

The higher your priority, the sooner your jobs will run. The more you use the cluster, the lower your priority.



```
19:00:05 up 7 days, 27:10, 14 users, load average: 0.04, 0.05, 0.00
[tjviering@login03 ~]$ st
```

JobID	JobName	State	Elapsed	Timelimit	AllocNodes	Priority	Start
NodeList							
76464	bash	COMPLETED	00:00:07	00:30:00	1	59677419	2022-05-16T11:19:14
cmp082							
76465	bash	COMPLETED	00:24:30	00:30:00	1	56451612	2022-05-16T11:21:30
cmp082							
76470	bash	OUT_OF_ME+	00:11:36	00:30:00	1	51612903	2022-05-16T11:46:02
cmp082							
76472	bash	COMPLETED	00:25:59	00:30:00	1	50000000	2022-05-16T11:58:03
cmp082							
76473	bash	TIMEOUT	00:30:03	00:30:00	1	45161290	2022-05-16T12:24:21
cmp078							
76481	bash	COMPLETED	00:19:18	00:30:00	1	45088566	2022-05-16T13:05:59
cmp078							
76486	bash	FAILED	00:14:47	00:30:00	1	45088566	2022-05-16T13:25:20
cmp078							
76492	bash	COMPLETED	00:29:01	01:00:00	1	45088566	2022-05-16T13:40:18
cmp078							
76509	lcfit1	FAILED	00:00:02	01:00:00	1	45088566	2022-05-16T14:18:30
cmp081							
76509.batch	batch	FAILED	00:00:02		1		2022-05-16T14:18:30
cmp081							
76511	lcfit1	FAILED	00:00:03	01:00:00	1	45088566	2022-05-16T14:21:09
cmp081							
76511.batch	batch	FAILED	00:00:03		1		2022-05-16T14:21:09
cmp081							
76512	lcfit1	FAILED	00:00:01	01:00:00	1	45088566	2022-05-16T14:22:19
cmp081							
76512.batch	batch	FAILED	00:00:01		1		2022-05-16T14:22:19
cmp081							
76513	lcfit1	FAILED	00:00:04	01:00:00	1	45088566	2022-05-16T14:23:17
cmp081							
76513.batch	batch	FAILED	00:00:04		1		2022-05-16T14:23:17
cmp081							
76513.0	python	FAILED	00:00:02		1		2022-05-16T14:23:19
cmp081							
76515	lcfit1	FAILED	00:00:03	01:00:00	1	45088566	2022-05-16T14:24:11
cmp081							
76515.batch	batch	FAILED	00:00:03		1		2022-05-16T14:24:11
cmp081							
76515.0	python	FAILED	00:00:01		1		2022-05-16T14:24:13
cmp081							
76516	lcfit1	COMPLETED	00:52:15	01:00:00	1	45088566	2022-05-16T14:26:01
cmp081							
76516.batch	batch	COMPLETED	00:52:15		1		2022-05-16T14:26:01
cmp081							
76516.0	python	COMPLETED	00:52:09		1		2022-05-16T14:26:02
cmp081							
76554	bash	FAILED	00:24:41	00:30:00	1	45088566	2022-05-16T15:00:59
cmp081							

```
[tjviering@login03 ~]$
```

Exercise

- Try to get the `testjob.sh` (provided in previous slides) to run correctly. This will fit a part of the learning curve database.
- Make sure everything goes correctly by using the commands `st`, `sq`, and checking the output file
- Use `cat lcfit0.txt` to dump the output of the output file to your terminal.
- Use `tail -f lcfit0.txt` for automatically updating view of the latest changes to the file (CTRL + C to quit)

Debugging code with interactive session

You can also start an interactive session and paste this into it to check if the code is working:

```
module load 2022r1
module load compute
module load miniconda3
unset CONDA_SHLVL
source "$(conda info --base)/etc/profile.d/conda.sh"
conda activate myenv5
cd /beegfs/scratch/tjviering/
python fit_database.py 0
```

However, you can only have 1 interactive session at a time! And maximum 1 hr. So, better to use real jobs for real experiments!

The real work

- Note that I split the whole database in 13 parts (see also `fit_database.py main() function`), so that each part approximately takes 1 hr.
- When I want to recompute the whole database, I make 13 jobscripts, each doing a different part. Then I submit all 13 jobscripts at the same time. If I have a high priority, I get all my results in 1-2 hours 😊!
- To make the jobscripts, I have a small piece of code (see next slides)

Generating jobfiles

```
for part in range(0,10):  
    f = open("job%d.sh" % part, "w", newline='\n')  
    f.write("#!/bin/sh\n")  
    f.write("#SBATCH --time=02:00:00\n")  
    f.write("#SBATCH --mincpus=2\n")  
    f.write("#SBATCH --job-name=lc%d\n" % part)  
    f.write("#SBATCH --output=lcfit%d.txt\n" % part)  
    ... (see example jobfile from before)  
    f.write("srun python fit2.py %d\n" % part)  
    f.close()
```

Generates 10 jobfiles

Does it directly right for linux!

Jobname, output, etc. depends on part

Be sure to tell python which part to do

Generating sbatch commands

I use this piece of code to generate the sbatch commands:

```
for part in range(0,10):  
    print('sbatch job%d.sh' % part)
```

I run this locally on my laptop, and copy the results, and paste them in XTerm.

Typical workflow

- For the `fit_database.py` I told you 1 hr + 512 MB is enough. For your own code this may not be the case... You will have to test it before you submit!
- Do like I do:
 1. Split the database in parts
 2. Check how much time is necessary for 1 part (maybe take the largest or most time consuming part!). You can check this easily by using an interactive job.
 3. Use this single part to estimate the time, memory required for all jobs. Be sure to request just a little bit more (20% margin or so), otherwise, if your job may get killed...
 4. Generate the jobfiles and sbatch commands
 5. Submit them all to the cluster at the same time
- Make sure your code on the cluster is up to date before you submit!
- Don't forget to transfer your results file away from scratch, so you don't lose them! For example, transfer to your laptop.

Check efficiency

Slurm Commands (continued)

- See statistics of a running job

```
$ sstat 1
  JobID  AveRSS  AveCPU  NTasks  AveDiskRead  AveDiskWrite
-----
1.0      426K  00:00.0      1      0.52M      0.01M
```

- See accounting information of a finished job (also see --long option)

```
$ sacct -j 1
  JobID  JobName  Partition  Account  AllocCPUS  State  ExitCode
-----
1       jobscript+  general  ewi-insy      2  COMPLETED  0:0
1.batch batch      ewi-insy      2  COMPLETED  0:0
```

- See overall job efficiency of a finished job

```
$ seff 1
Cores per node: 2
CPU Utilized: 01:00:00
CPU Efficiency: 50.00% of 02:00:00 core-walltime
Job Wall-clock time: 01:00:00
Memory Utilized: 0.80 GB
Memory Efficiency: 80.00% of 1.00 GB
```

Tips

- Don't be scared to submit jobs to the cluster 😊
- You can learn via trail and error!
- The DelftBlue is only in beta phase now

How busy is it?

`sinfo -o "%20P%8D%16F%8z%10m%N"`

```
[tjviering@login03 ~]$ sinfo -o "%20P%8D%16F%8z%10m%N"
PARTITION      NODES  NODES(A/I/O/T)  S:C:T  MEMORY  NODELIST
compute*       218    33/181/4/218    2:24:1  192000  cmp[001-218]
gpu             10     1/8/1/10        8:6:2   256000  gpu[001-010]
memory          10     0/10/0/10       2:24:1  768000+ mem[001-010]
trans           2       0/2/0/2        32:1:1    1      file[01-02]
visual          2       0/2/0/2        64:1:1    1      visual[01-02]
```

A: Allocated

I: IDLE (Free for use!!!)

O: Other

T: total

APPENDIX

More information

- <https://gitlab.tudelft.nl/dhpc/docs/-/wikis/Frequently-Asked-Questions>
- <https://gitlab.tudelft.nl/dhpc/docs/-/wikis/home>
- <https://gitlab.tudelft.nl/dhpc/docs/-/wikis/Python>
- <https://gitlab.tudelft.nl/dhpc/docs/-/wikis/Remote-access-to-DelftBlue#dont-forget-either-eduvpn-or-linux-bastion>
- <https://gitlab.tudelft.nl/dhpc/docs/-/wikis/Data-transfer-to-DelftBlue#4-network-drives>
- <https://gitlab.tudelft.nl/dhpc/docs/-/wikis/DHPC-modules>
- <https://gitlab.tudelft.nl/dhpc/docs/-/wikis/Slurm-scheduler#job-submission-and-control-using-slurm>
- You can also ask in the Mattermost channel: [**Mattermost DHPC channel**](#)

Conda space

- `conda clean -all`
- This cleans up your conda cache freeing up some space

If you run out of space with Conda....

- You can store Conda stuff on the scratch directory. But only do this if really necessary!
- Scratch will get removed now and then... So this is why I use the home directory.

- Do the following steps if you want to store your Conda stuff on scratch:

```
mkdir -p /scratch/${USER}/.conda
```

- Move all files from ~/.conda to /scratch/\${USER}/.conda using the mv or rsync command
- Link the folders:

```
ln -s /scratch/${USER}/.conda $HOME/.conda
```

- Now when you open ~/.conda, actually, the scratch directory is used