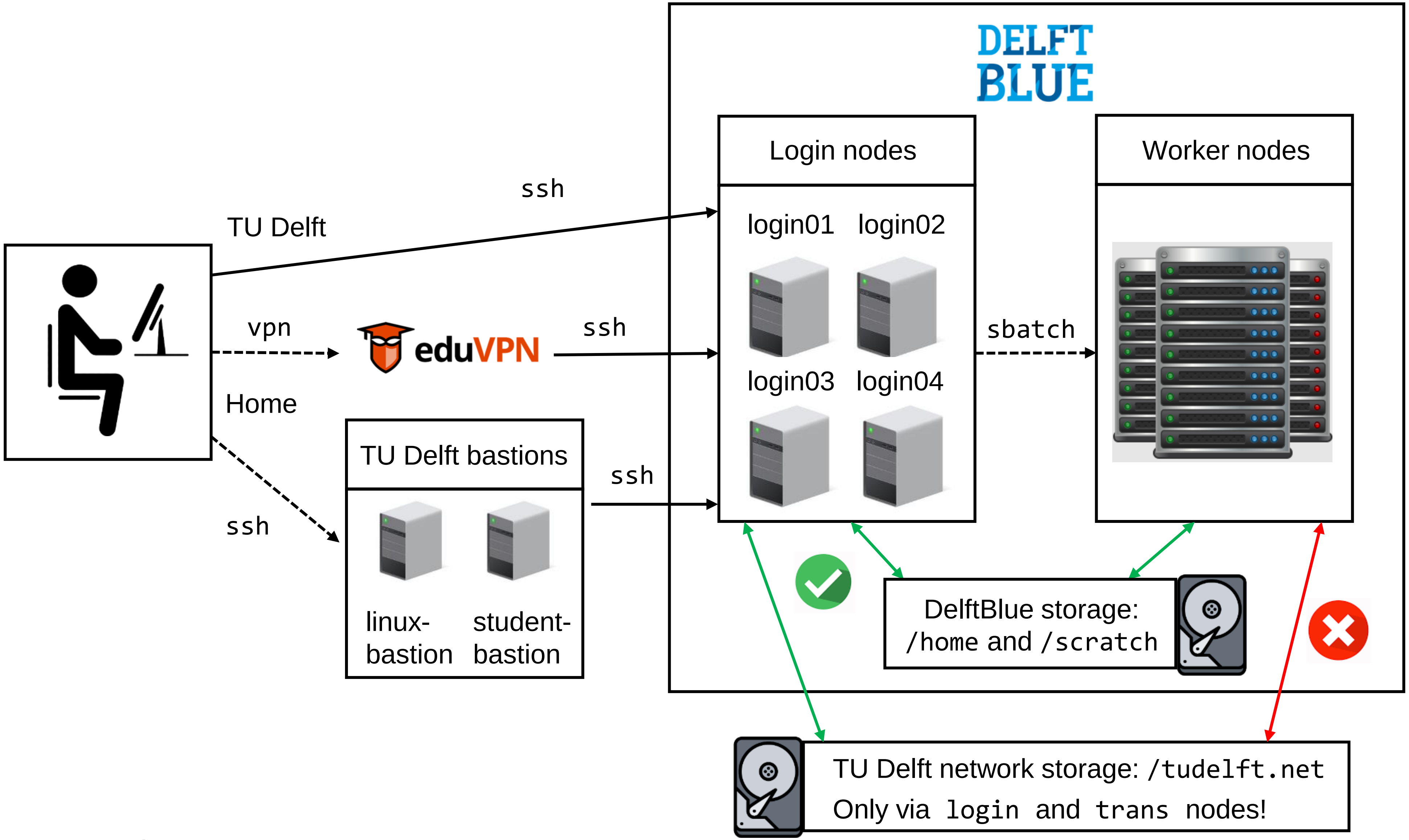


DELFT BLUE

Login nodes

login01 login02

login03 login04

Worker nodes

sbatch

TU Delft

ssh

vpn

eduVPN

ssh

Home

ssh

TU Delft bastions

linux-bastion student-bastion

ssh

DelftBlue storage: `/home` and `/scratch`

TU Delft network storage: `/tudelft.net`

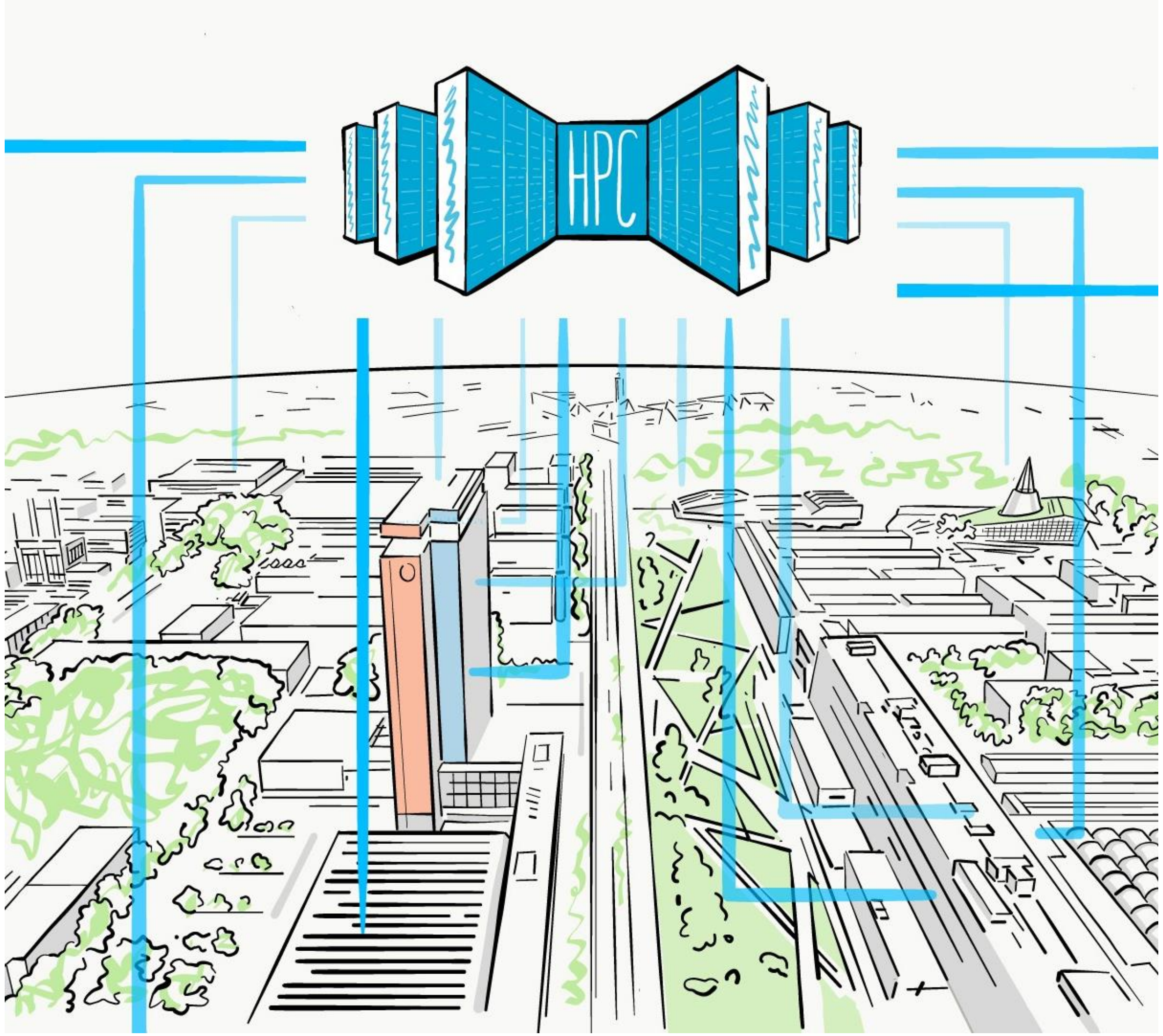
Only via `login` and `trans` nodes!

TU Delft

# DelftBlue

- 10.000 CPU cores
- Over 200 compute nodes
- 10 GPU nodes

# DelftBlue community: 3mE, ABE, AE, AS, CEG, EEMCS, IDE, TPM, QuTech



TU Delft

# How do I work with cluster?

1. **Prepare input files for your code** on a personal computer
2. **Upload input files** and required data to the cluster’s storage
3. **Determine** required **resources**
4. **Create job script**
5. **Submit** job(s) to scheduler
6. **Monitor** progress (via output files) and resource use (via statistics)
7. **Download** results to personal computer for further processing
8. **Cleanup** files

# Typical submission script

- Create a job script in a file

```
#!/bin/sh                                    ➔ system info: type of script

#SBATCH --job-name=job_name                  ➔ select job name
#SBATCH --partition=compute                  ➔ select partition to run your job on
#SBATCH --account=research-eemcs-diam        ➔ specify account
#SBATCH --time=01:00:00                      ➔ request run time
#SBATCH --ntasks=4                           ➔ number of tasks (parallel)
#SBATCH --cpus-per-task=1                    ➔ CPUs (threads) per task
#SBATCH --mem-per-cpu=1G                     ➔ RAM per CPU

module load 2023r1                           ➔ Use DelftBlue software collection
module load openmpi                          ➔ load openmpi module
srun ./executable > output.log               ➔ start tasks with srun
```

# Example 1: Hello, World! on 4 CPUs

Our first submission script `helloworld.sh`:

```
#!/bin/bash
#SBATCH --job-name="01_hello"
#SBATCH --time=00:10:00
#SBATCH --partition=compute
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=1GB
#SBATCH --account=education-eemcs-courses-linuxcli
#SBATCH --reservation=delftblueworkshop

echo "Hello, World!" >> helloworld.txt
echo "The following nodes are reporting for duty:" >> helloworld.txt
srun hostname >> helloworld.txt
echo "Have a great day!" >> helloworld.txt
```

# Example 1: Hello, World! on 4 CPUs

```
NetID@login01:~ $ sbatch helloworld.sh
```

```
Hello, World!
The following nodes are reporting for duty:
cmpXXX
cmpXXX
cmpXXX
cmpXXX
Have a great day!
```

# Example 1.2: Hello, World! On 2x2 CPUs

Our submission script `helloworld2nodes.sh`:

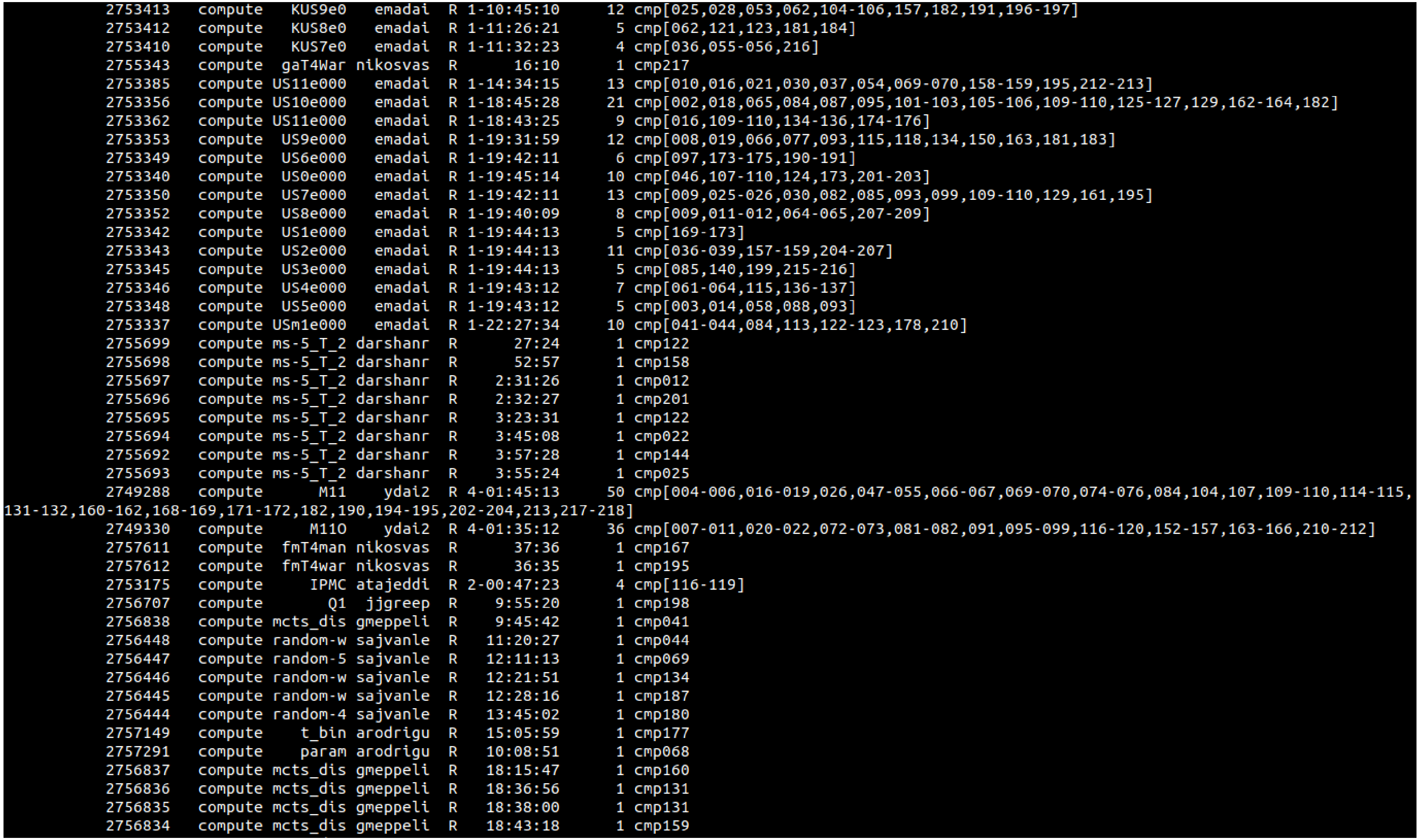
```
#!/bin/bash
#SBATCH --job-name="01_hello"
#SBATCH --time=00:10:00
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=1GB
#SBATCH --account=education-eemcs-courses-linuxcli
#SBATCH --reservation=delftblueworkshop

echo "Hello, World!" >> helloworld.txt
echo "The following nodes are reporting for duty:" >> helloworld.txt
srun hostname >> helloworld.txt
echo "Have a great day!" >> helloworld.txt
```

# Example 1.2: Hello, World! On 2x2 CPUs

```
NetID@login01:~ $ sbatch helloworld2nodes.sh
```

```
Hello, World!
The following nodes are reporting for duty:
cmpXXX
cmpXXX
cmpYYY
cmpYYY
Have a great day!
```

# What is a scheduler?



```
           2753413   compute   KUS9e0   emadai  R 1-10:45:10     12 cmp[025,028,053,062,104-106,157,182,191,196-197]
           2753412   compute   KUS8e0   emadai  R 1-11:26:21      5 cmp[062,121,123,181,184]
           2753410   compute   KUS7e0   emadai  R 1-11:32:23      4 cmp[036,055-056,216]
           2755343   compute  gaT4War nikosvas  R      16:10      1 cmp217
           2753385   compute US11e000   emadai  R 1-14:34:15     13 cmp[010,016,021,030,037,054,069-070,158-159,195,212-213]
           2753356   compute US10e000   emadai  R 1-18:45:28     21 cmp[002,018,065,084,087,095,101-103,105-106,109-110,125-127,129,162-164,182]
           2753362   compute US11e000   emadai  R 1-18:43:25      9 cmp[016,109-110,134-136,174-176]
           2753353   compute  US9e000   emadai  R 1-19:31:59     12 cmp[008,019,066,077,093,115,118,134,150,163,181,183]
           2753349   compute  US6e000   emadai  R 1-19:42:11      6 cmp[097,173-175,190-191]
           2753340   compute  US0e000   emadai  R 1-19:45:14     10 cmp[046,107-110,124,173,201-203]
           2753350   compute  US7e000   emadai  R 1-19:42:11     13 cmp[009,025-026,030,082,085,093,099,109-110,129,161,195]
           2753352   compute  US8e000   emadai  R 1-19:40:09      8 cmp[009,011-012,064-065,207-209]
           2753342   compute  US1e000   emadai  R 1-19:44:13      5 cmp[169-173]
           2753343   compute  US2e000   emadai  R 1-19:44:13     11 cmp[036-039,157-159,204-207]
           2753345   compute  US3e000   emadai  R 1-19:44:13      5 cmp[085,140,199,215-216]
           2753346   compute  US4e000   emadai  R 1-19:43:12      7 cmp[061-064,115,136-137]
           2753348   compute  US5e000   emadai  R 1-19:43:12      5 cmp[003,014,058,088,093]
           2753337   compute USm1e000   emadai  R 1-22:27:34     10 cmp[041-044,084,113,122-123,178,210]
           2755699   compute ms-5_T_2 darshanr  R      27:24      1 cmp122
           2755698   compute ms-5_T_2 darshanr  R      52:57      1 cmp158
           2755697   compute ms-5_T_2 darshanr  R    2:31:26      1 cmp012
           2755696   compute ms-5_T_2 darshanr  R    2:32:27      1 cmp201
           2755695   compute ms-5_T_2 darshanr  R    3:23:31      1 cmp122
           2755694   compute ms-5_T_2 darshanr  R    3:45:08      1 cmp022
           2755692   compute ms-5_T_2 darshanr  R    3:57:28      1 cmp144
           2755693   compute ms-5_T_2 darshanr  R    3:55:24      1 cmp025
           2749288   compute      M11    ydai2  R 4-01:45:13     50 cmp[004-006,016-019,026,047-055,066-067,069-070,074-076,084,104,107,109-110,114-115,
131-132,160-162,168-169,171-172,182,190,194-195,202-204,213,217-218]
           2749330   compute     M11O    ydai2  R 4-01:35:12     36 cmp[007-011,020-022,072-073,081-082,091,095-099,116-120,152-157,163-166,210-212]
           2757611   compute  fmT4man nikosvas  R      37:36      1 cmp167
           2757612   compute  fmT4war nikosvas  R      36:35      1 cmp195
           2753175   compute     IPMC atajeddi  R 2-00:47:23      4 cmp[116-119]
           2756707   compute       Q1  jjgreep  R    9:55:20      1 cmp198
           2756838   compute mcts_dis gmeppeli  R    9:45:42      1 cmp041
           2756448   compute random-w sajvanle  R   11:20:27      1 cmp044
           2756447   compute random-5 sajvanle  R   12:11:13      1 cmp069
           2756446   compute random-w sajvanle  R   12:21:51      1 cmp134
           2756445   compute random-w sajvanle  R   12:28:16      1 cmp187
           2756444   compute random-4 sajvanle  R   13:45:02      1 cmp180
           2757149   compute    t_bin arodrigu  R   15:05:59      1 cmp177
           2757291   compute    param arodrigu  R   10:08:51      1 cmp068
           2756837   compute mcts_dis gmeppeli  R   18:15:47      1 cmp160
           2756836   compute mcts_dis gmeppeli  R   18:36:56      1 cmp131
           2756835   compute mcts_dis gmeppeli  R   18:38:00      1 cmp131
           2756834   compute mcts_dis gmeppeli  R   18:43:18      1 cmp159
```

# Typical scheduler commands?

- Submit job using the job script

```
$ sbatch jobscript.sbatch
Submitted batch job 290573
```

- See queue status

```
$ squeue
JOBID PARTITION     NAME          USER ST   TIME

290573 general    jobscrip    somebody  R   0:01
```

- See job output

```
$ cat slurm-290573.out
Hello world!
```

- Cancel job

```
$ scancel 290573
$ squeue
JOBID PARTITION     NAME          USER ST   TIME
NODES NODELIST(REASON)
```

TU Delft

# Module system demo

```
[NetID@login02 ~]$ module avail

[NetID@login02 ~]$ module load

[NetID@login02 ~]$ module list

[NetID@login02 ~]$ module spider {module}
```

- ➔ list available modules
- ➔ load module
- ➔ list loaded modules
- ➔ find module {module}

More info: https://doc.dhpc.tudelft.nl/delftblue/DHPC-modules/

# Example 2: Julia

Let’s prepare a little program to draw a Mandelbrot set:

```
function mandelbrot(a)
    z = 0
    for i=1:50
        z = z^2 + a
    end
    return z
end

for y=1.0:-0.05:-1.0
    for x=-2.0:0.0315:0.5
        abs(mandelbrot(complex(x, y))) < 2 ? print("*") :
print(" ")
    end
    println()
end
```

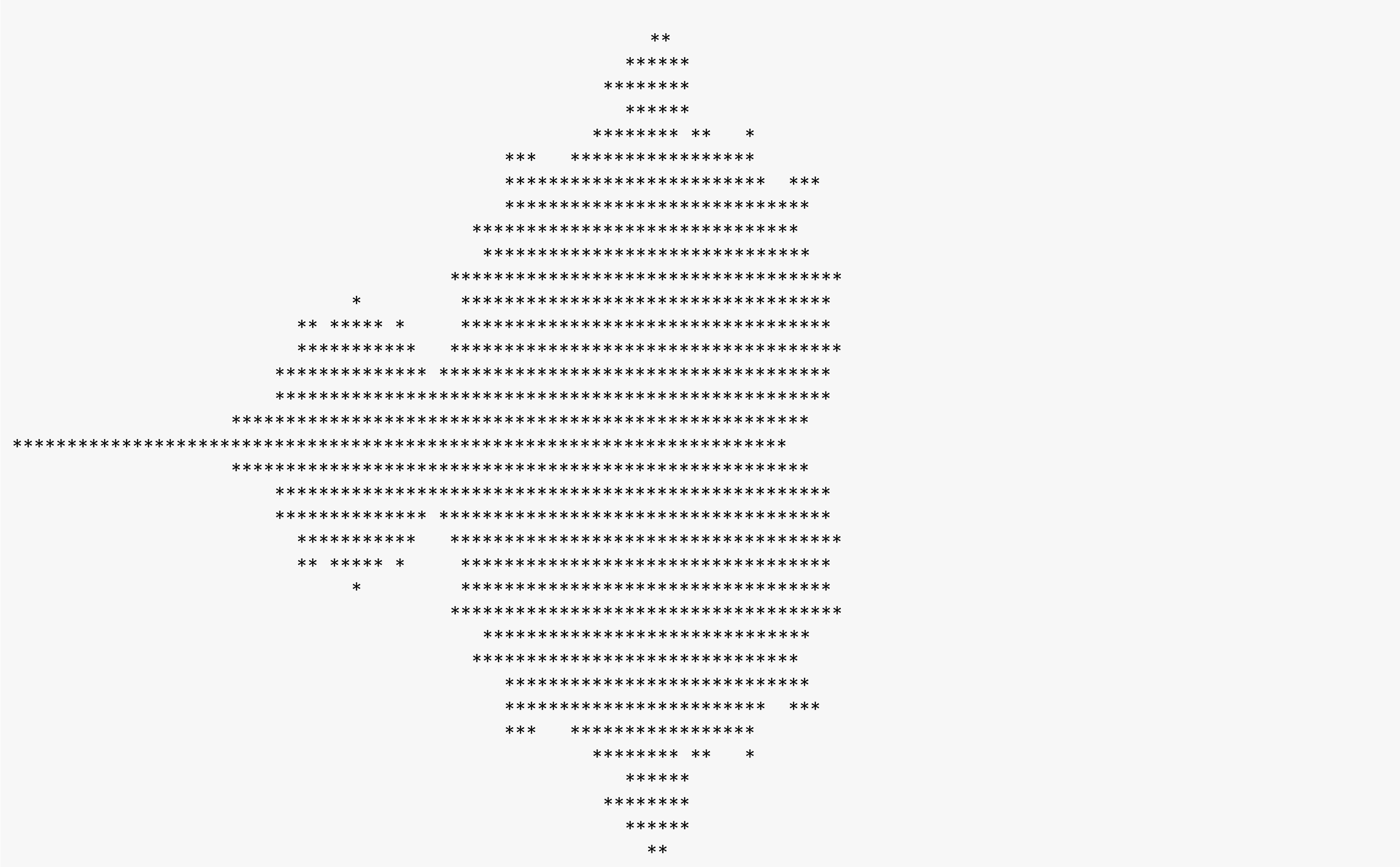
# Example 2: Julia

Submission script `run_julia_mandelbrot.sh`:

```
#!/bin/bash
#
#SBATCH --job-name="julia"
#SBATCH --time=00:10:00
#SBATCH --partition=compute
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=1G
#SBATCH --account=education-eemcs-courses-linuxcli
#SBATCH --reservation=delftblueworkshop

module load 2023r1
module load julia

srun julia mandelbrot.jl
```

# Example 2: Julia

Let’s submit the job and check out the result:



```
                                                          **
                                                        ******
                                                      ********
                                                        ******
                                                     ******** **   *
                                             ***   *****************
                                             ************************  ***
                                             ****************************
                                          ******************************
                                           ******************************
                                        ************************************
                               *         **********************************
                          ** ***** *     **********************************
                          ***********   ************************************
                        ************** ************************************
                        ***************************************************
                    *****************************************************
***********************************************************************
                    *****************************************************
                        ***************************************************
                        ************** ************************************
                          ***********   ************************************
                          ** ***** *     **********************************
                               *         **********************************
                                        ************************************
                                           ******************************
                                          ******************************
                                             ****************************
                                             ************************  ***
                                             ***   *****************
                                                     ******** **   *
                                                        ******
                                                      ********
                                                        ******
                                                          **
```

# Example 3: ASE molecules generator

Let’s install a new python module, called ASE:

```
#!/bin/bash

# Install ASE:
module load 2023r1
module load python
module load py-pip
module load py-numpy
module load py-scipy
module load py-matplotlib

python -m pip install --user ase
```

```
[NetID@login02 ~]$ chmod +x install_ase.sh

[NetID@login02 ~]$ ./install_ase.sh
```

# Example 3: ASE molecules generator

Submission script `sub_to_queue.sh`:

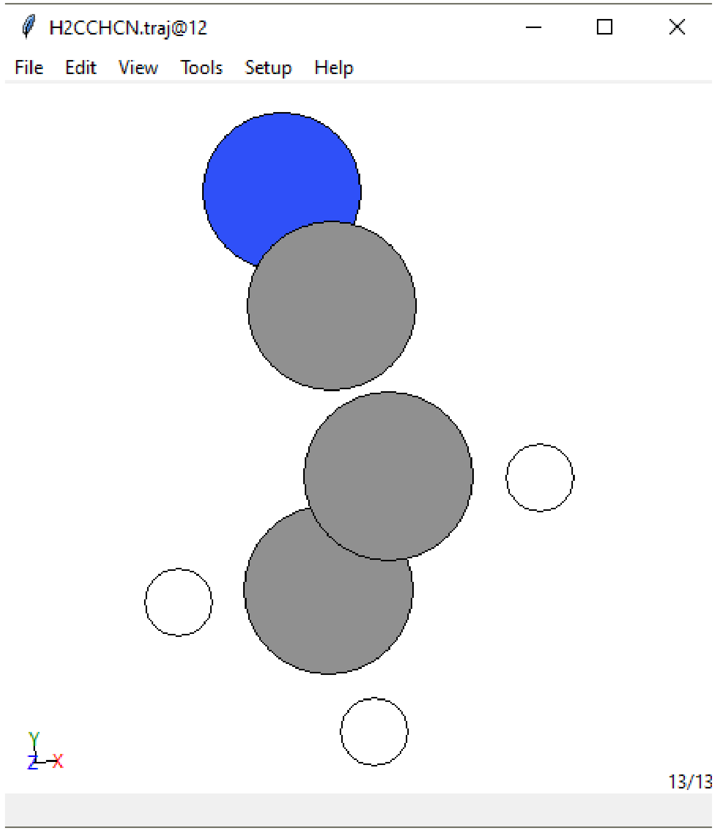
```
#!/bin/bash#
#SBATCH --job-name="gen_mol"
#SBATCH --time=00:10:00
#SBATCH --partition=compute
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=1GB
#SBATCH --account=education-eemcs-courses-linuxcli
#SBATCH --reservation=delftblueworkshop

module load 2023r1
module load python
module load py-numpy
module load py-scipy
module load py-matplotlib

srun python gen_mol_folders.py
```

# Example 3: ASE molecules generator

```
[NetID@login02 ~]$ sbatch sub_to_queue.sh

[NetID@login02 ~]$ cat slurm-XXX.out
```

Exercise:

Install ASE. Submit the job. Inspect the output file and generated folders.



# References

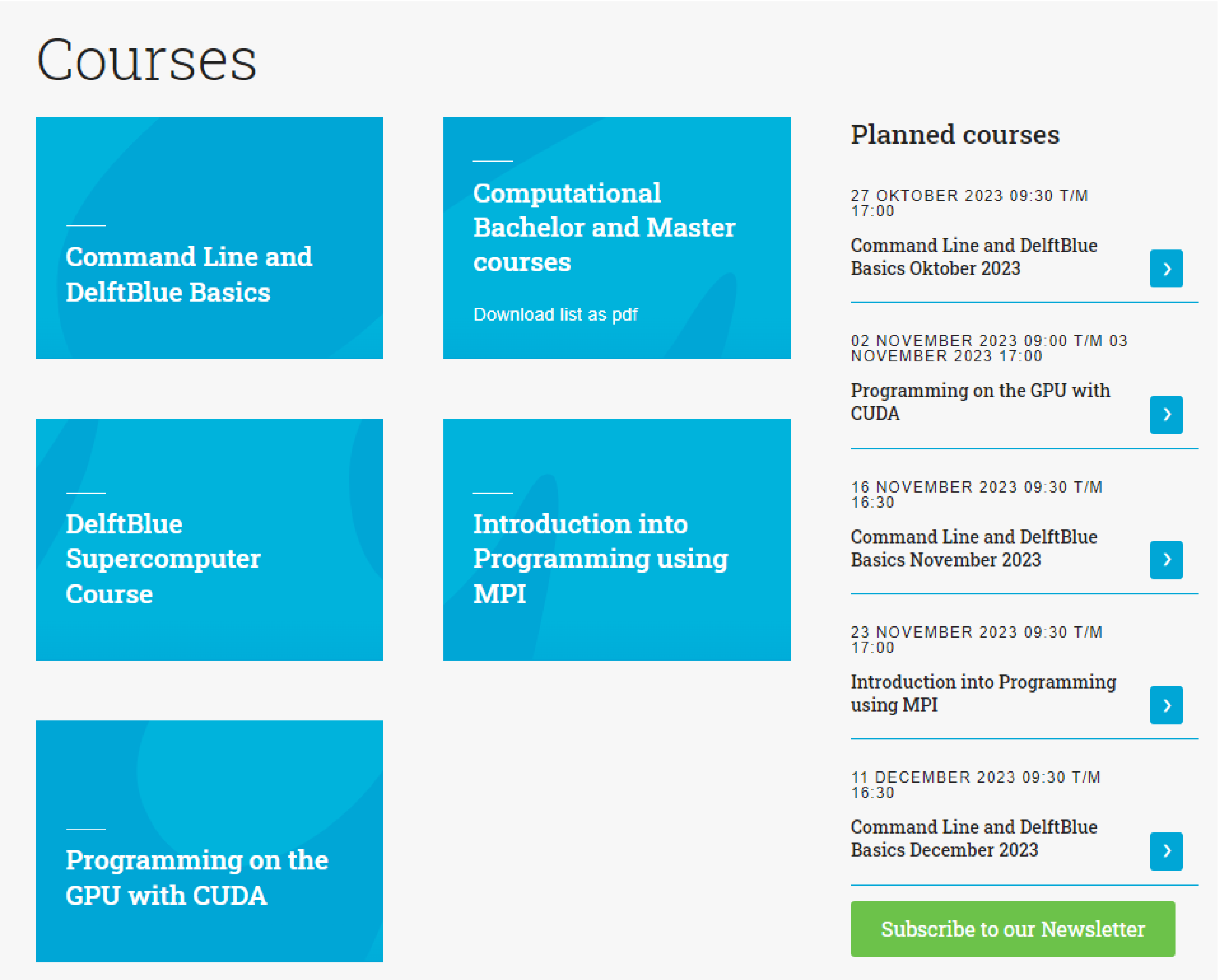
- Documentation:

  https://www.tudelft.nl/dhpc/documentation
- Mattermost:

  https://mattermost.tudelft.nl/dhpc/
- Self Service Portal (TopDesk):

  https://tudelft.topdesk.net
- DCSE Courses:

  https://www.tudelft.nl/cse/education/courses



## Courses

Command Line and DelftBlue Basics

Computational Bachelor and Master courses

Download list as pdf

DelftBlue Supercomputer Course

Introduction into Programming using MPI

Programming on the GPU with CUDA

### Planned courses

27 OKTOBER 2023 09:30 T/M 17:00

Command Line and DelftBlue Basics Oktober 2023 ›

02 NOVEMBER 2023 09:00 T/M 03 NOVEMBER 2023 17:00

Programming on the GPU with CUDA ›

16 NOVEMBER 2023 09:30 T/M 16:30

Command Line and DelftBlue Basics November 2023 ›

23 NOVEMBER 2023 09:30 T/M 17:00

Introduction into Programming using MPI ›

11 DECEMBER 2023 09:30 T/M 16:30

Command Line and DelftBlue Basics December 2023 ›

Subscribe to our Newsletter