C.E.C.I

Consortium des Equipements
de Calcul Intensif
en Fédération Wallonie-Bruxelles

Preparing, submitting and managing jobs with Slurm

damien.francois@uclouvain.be
October 2023

Until now:

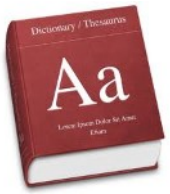
- access the cluster ✔
- copy data to/from the cluster ✔
- choose and activate software ✔
- run software in the command line prompt ✔
- create/write text files ✔
- actually run software on the cluster ❓

tl;dr:

DON'T: run software on the login node

DO: submit a *job* to the *resource manager|job scheduler*

What is a job?



Dictionary

job [1] |jäb|

noun

1 a paid position of regular employment *: jobs are created in the private sector, not in Washington | a part-time job.*

2 a task or piece of work, esp. one that is paid *: she wants to be left alone to get on with the job | you* ***did a good job of*** *explaining.*

- a responsibility or duty *: it's our job to find things out.*
- [in sing.] informal a difficult task *: we thought you'd have a job getting there.*
- [with adj.] informal a procedure to improve the appearance of something, esp. an operation involving plastic surgery *: she's had a nose job | someone had done a skillful paint job.*
- [with adj.] informal a thing of a specified nature *: the car was a blue malevolent-looking job.*
- informal a crime, esp. a robbery *: a series of daring bank jobs.*
- Computing an operation or group of operations treated as a single and distinct unit.

What is a resource manager/scheduler ?

Job scheduler

From Wikipedia, the free encyclopedia

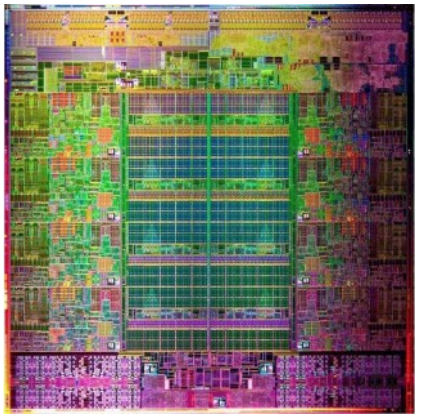
A **job scheduler** is a computer application for controlling unattended background program execution of jobs.[1] This is commonly called **batch scheduling**, as execution of non-interactive jobs is often called batch processing, though traditional *job* and *batch* are distinguished and contrasted; see that page for details. Other synonyms include **batch system**, **distributed resource management system** (**DRMS**), **distributed resource manager** (**DRM**), and, commonly today, **workload automation** (**WLA**). The data structure of jobs to run is known as the job queue.

Resource management (computing)

From Wikipedia, the free encyclopedia

In computer programming, **resource management** refers to techniques for managing resources (components with limited availability).

resources:

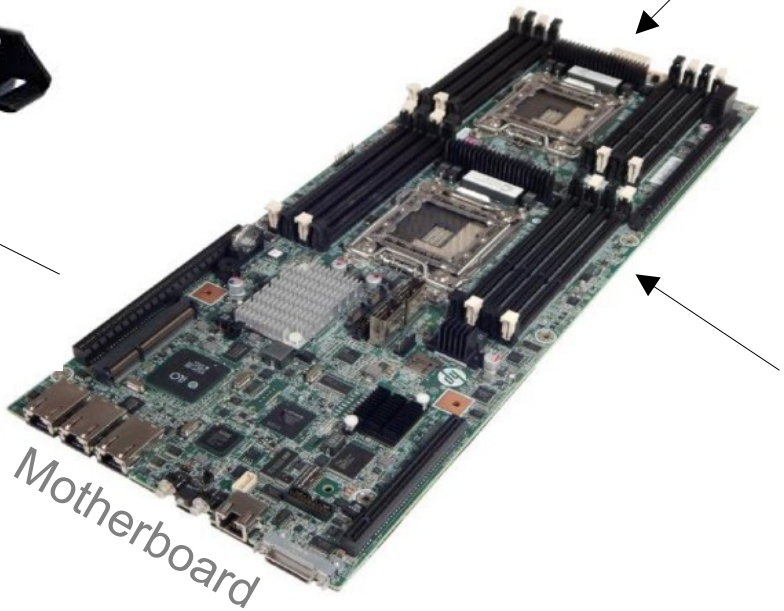


CPU (core)

CPU (socket)

GPU

Compute node

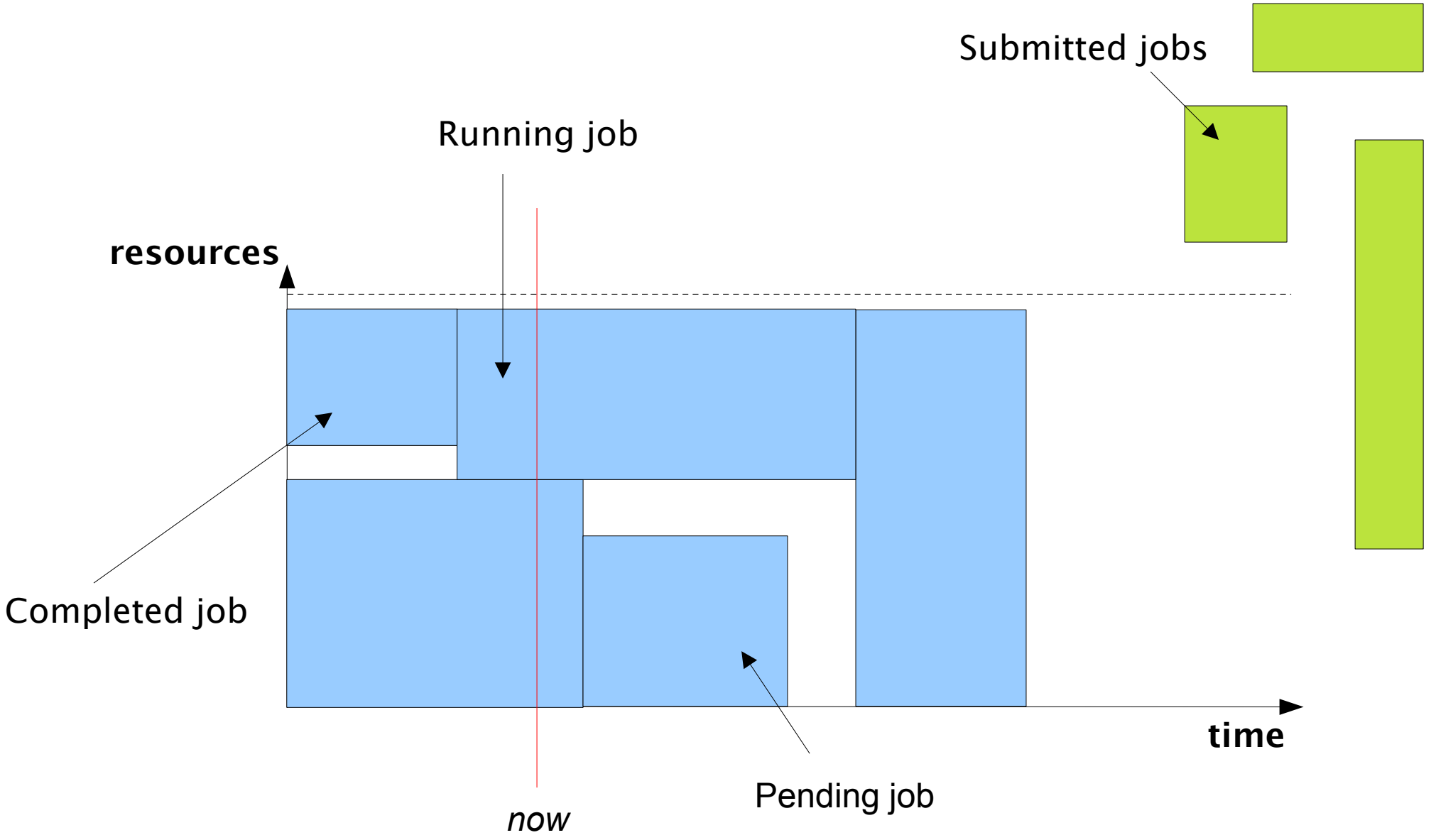


Motherboard

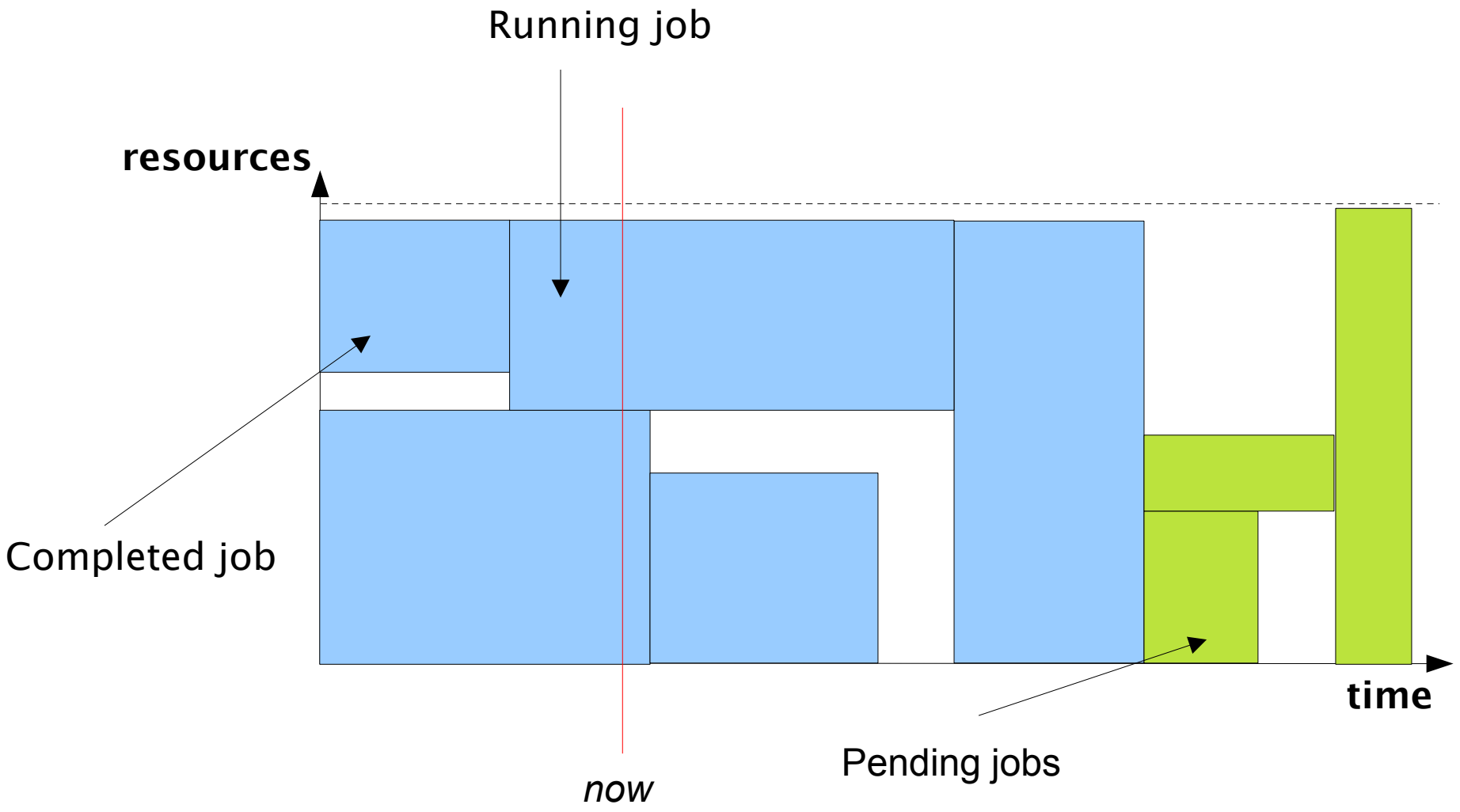
RAM

Disk

scheduling:


Submitted jobs
Running job
resources
Completed job
time
Pending job
now

scheduling:



scheduling:


Running job
resources
exclusive allocation of resources
to jobs over time
Completed job
time
Pending jobs
now


SCORE
3780
LEVEL
3
LINES
32


slurm
workload manager

Slurm

Free and free

Mature (exists since ~2003)

Very active community

Many success stories

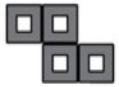
Widely used

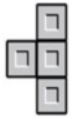
Also an intergalactic soft drink

Futurama (TV Series, creators David X. Cohen, Matt Groening)
Fry and the Slurm Factory (1999)
20th Century Fox Television

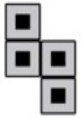
Topics:

 . How to create a job

 . How to choose resources

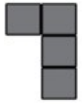
 . Understand priorities

 . Typical workloads

 . Interactive sessions

---

 . Workflows

 . Advanced submission techniques

Part I. You will learn how to:

Create and submit a job

Monitor and inspect jobs

Control (your own) jobs

with

Make up your mind ...

- operations you need to perform
- resources you need for those operations

e.g. launch program 'myprog'

Job steps

e.g. 1 core, 2GB RAM
for 1 hour

Job parameters

... then write a submission script...

It is a shell script (Bash)

Bash sees these as comments

Slurm takes them as parameters

Job step creation

```
#!/bin/bash
# Submission script for demonstrating
# slurm usage.

# Job parameters
#SBATCH --job-name=demo
#SBATCH --output=res.txt
# Needed resources
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=2000
#SBATCH --time=1:00:00

# Operations
echo "Job start at $(date)"
# Job steps
srun ~/bin/myprog < mydata1

echo "Job end at $(date)"
```

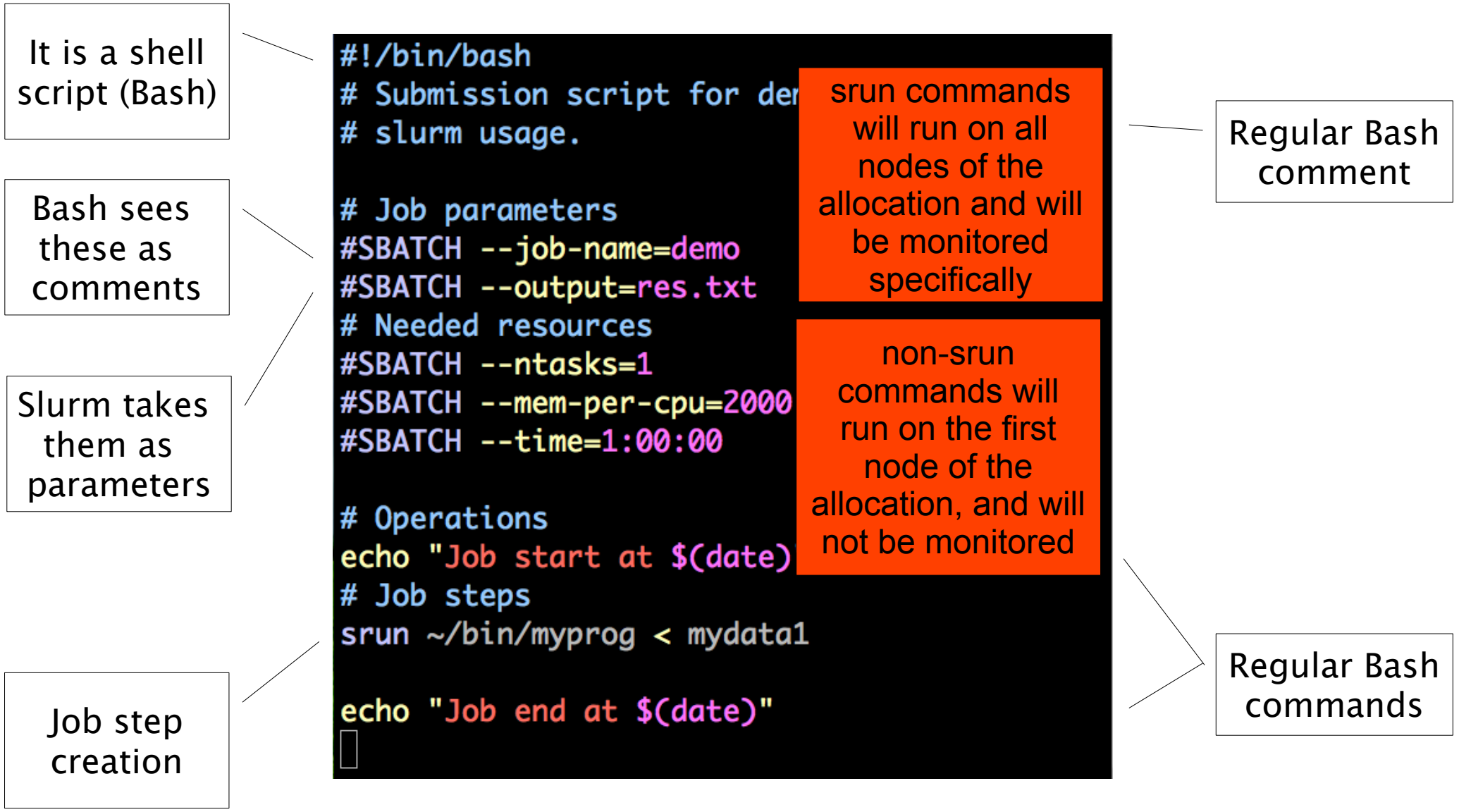
Regular Bash comment

Regular Bash commands

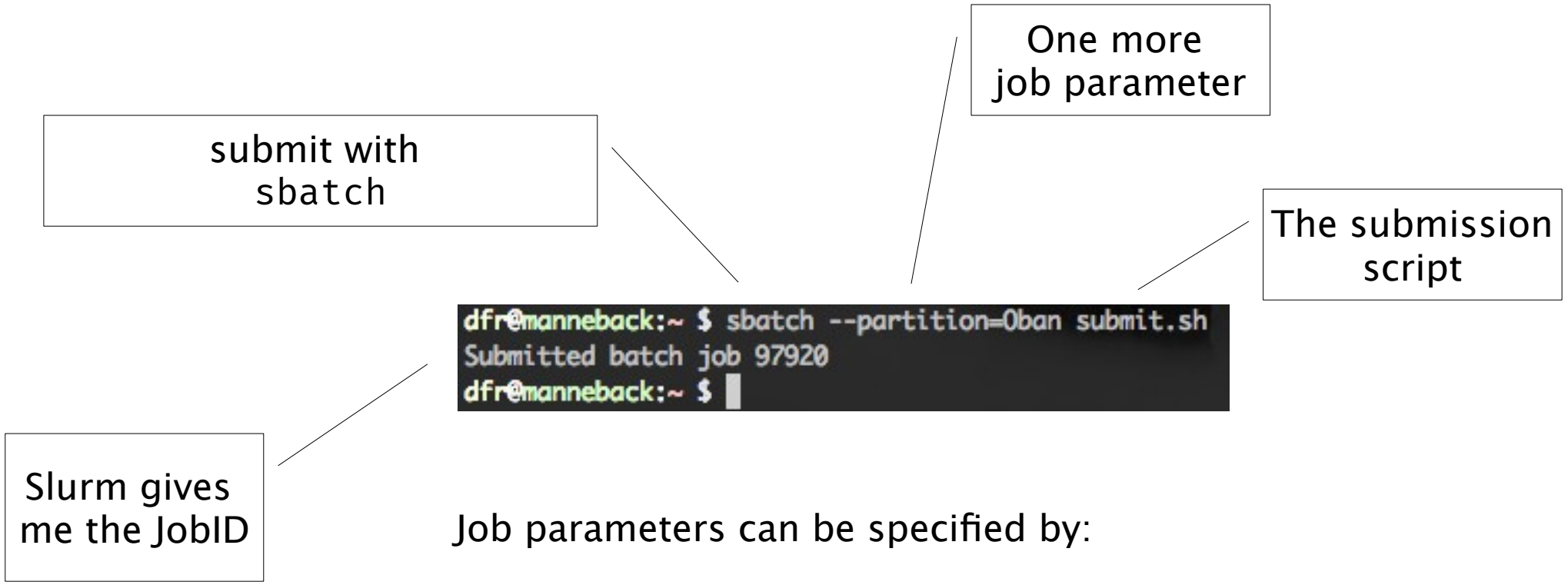
... then write a submission script...

... then write a submission script...

... then write a submission script...



... and submit it with sbatch



Job parameters can be specified by:

- #SBATCH directives in the submission script ;
- environment variables ;
- parameters on the `sbatch` command line.

Most of the parameters have default values and can be omitted.

The job ID is used later on to uniquely identify the job.

Submit your first job!

1. Connect to a cluster
2. Open a text editor and write the script for a job that will run the "`hostname`" command
   - No parameter needed in this simple test
   - `hostname` will write the name of the computer
   - Do not forget the Shebang (`#!...`) line
3. Submit the job
4. Look for files created in the directory

Monitor jobs with squeue command

```
SQUEUE(1)                          Slurm components
                    SQUEUE(1)

NAME
       squeue  -  view  information  about jobs
       located in the SLURM scheduling queue.

SYNOPSIS
       squeue [OPTIONS...]

DESCRIPTION
       squeue is used to view job and job  step
       information for jobs managed by SLURM.

OPTIONS
       -A                              <account_list>,
       --account=<account_list>
              Specify  the accounts of the jobs
              to view. Accepts  a  comma  sepa-
              rated list of account names. This
:
```

man squeue

Monitor jobs with squeue command

```
$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             12324     batch     demo      dfr  R   11:10:02      4 node[001-004]
             12325     batch     demo      dfr PD      00:00      2 (Resources)
             12329     batch   prod_1      bvr PD      00:00      1 (Priority)
             12422     debug   test_2      bvr  R      04:01      1 node005
```

JOBID	the job ID assigned by Slurm
PARTITION	set of nodes the job was submitted to
NAME	name of the job as specified with `--job-name`
USER	username of the user who submitted the job
ST	State of the job: Running, PenDing, ...
TIME	Running time of the job
NODES	Number of nodes requested (`--nodes`)
NODELIST	Nodes assigned to the job by Slurm `node[001-004]` = `node001`, `node002`, `node003`, and `node004`
(REASON)	Reason why the job is pending (`Resources`): your job is next, (`priority`): you need to wait, ...

Monitor jobs with squeue command

```
$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             12324     batch     demo      dfr  R   11:10:02      4 node[001-004]
             12325     batch     demo      dfr PD      00:00      2 (Resources)
             12329     batch   prod_1      bvr PD      00:00      1 (Priority)
             12422     debug   test_2      bvr  R      04:01      1 node005


$ squeue --me
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             12324     batch     demo      dfr  R   11:10:02      4 node[001-004]
             12325     batch     demo      dfr PD      00:00      2 (Resources)

$ squeue --me --start
             JOBID PARTITION     NAME     USER ST          START_TIME              NODES SCHEDNODES           NODELIST(REASON)
             12325     batch     demo      dfr PD    2025-02-12T09:12                  2 node[001-002]        (resources)

$ squeue --partition=debug
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             12422     debug   test_2      bvr  R      04:01      1 node005

$ squeue --Format=jobid,partition,timeused,timelimit --partition=debug
             JOBID               PARTITION           TIME                TIME_LIMIT
             12422               debug               04:01               20:00
```

Get all information Slurm has about a job with `scontrol show <jobid>`

```
JobId=12324 JobName=demo
   UserId=dfr(3000003) GroupId=dfr(3000003) MCS_label=N/A
   Priority=6936634 Nice=0 Account=ceci QOS=normal
   JobState=RUNNING Reason=None Dependency=(null)
   Requeue=0 Restarts=0 BatchFlag=1 Reboot=0 ExitCode=0:0
   RunTime=00:00:00 TimeLimit=14:00:00 TimeMin=N/A
   SubmitTime=2021-10-06T16:07:57 EligibleTime=2021-10-06T16:07:57
   AccrueTime=2021-10-06T16:07:57
   StartTime=2021-10-07T17:42:35 EndTime=2021-10-07T21:42:35 Deadline=N/A
   SuspendTime=None SecsPreSuspend=0 LastSchedEval=2021-10-06T16:08:38
   Partition=batch AllocNode:Sid=lm3-w078:184117
   ReqNodeList=(null) ExcNodeList=(null)
   NodeList=(null)
   FedOrigin=cluster1 FedViableSiblings=lemaitre3 FedActiveSiblings=cluster1
   NumNodes=4 NumCPUs=4 NumTasks=1 CPUs/Task=4 ReqB:S:C:T=0:0:*:*
   TRES=cpu=4,mem=2400M,node=1,billing=4
   Socks/Node=* NtasksPerN:B:S:C=1:0:*:* CoreSpec=*
   MinCPUsNode=4 MinMemoryCPU=600M MinTmpDiskNode=0
   Features=(null) DelayBoot=00:00:00
   OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
   Command=/home/users/d/f/dfr/test.sh
   WorkDir=/home/users/d/f/dfr/test.sh
   StdErr=home/users/d/f/dfr/res.txt
   StdIn=/dev/null
   StdOut=home/users/d/f/dfr/res.txt
   Power=
   MailUser=damien.francois@uclouvain.be MailType=FAIL
```

`man scontrol`

Cancel jobs with ... scancel

```
$ squeue --me
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             12324     batch     demo      dfr  R   11:10:02      4 node[001-004]
             12325     batch     demo      dfr PD      00:00      2 (Resources)


$ scancel 12324

$ squeue --me
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             12325     batch     demo      dfr PD      00:00      2 (Resources)
```

```
Usage: scancel [-A account] [--batch] [--full] [--interactive] [-n job_name]
               [-p partition] [-Q] [-q qos] [-R reservation][-s signal | integer]
               [-t PENDING | RUNNING | SUSPENDED] [--usage] [-u user_name]
               [--hurry] [-V] [-v] [-w hosts...] [--wckey=wckey]
               [job_id[_array_id][.step_id]]
```

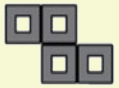
`man scancel`

Modify jobs with `scontrol update jobid=<id> <parameter>=<value>`

```
$ squeue --me
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             12324     batch     demo      dfr  R   11:10:02      4 node[001-004]
             12325     batch     demo      dfr PD      00:00      2 (Resources)

$ scontrol update jobid=12325 numnodes=3

$ squeue --me
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
             12324     batch     demo      dfr  R   11:10:02      4 node[001-004]
             12325     batch     demo      dfr PD      00:00      3 (Resources)
```

Most parameters can only be changed
for *PENDING* jobs

`man scontrol`

Part  . You will learn how to:

discover cluster features (resources),
target specific features and tune your jobs,
choose suitable resource values, and
get job actual resource usage.

in your submission scripts for

Use `sinfo` to find out about the nodes and the partitions

```
$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
batch*       up 2-00:00:00      2   idle node[001-002]
batch*       up 2-00:00:00      1  alloc node003
batch*       up 2-00:00:00      1    mix node004
debug        up      06:00      1   idle node005
```

PARTITION	Partition name
AVAIL	State of the partition (Up, Down, ...)
TIMELIMIT	Maximum run time for jobs submitted to that partition
NODES	Number of nodes in the partition
STATE	State of nodes in partition
NODELIST	List of compute nodes in said state in the partition

How to discover cluster resources

>

Use `sinfo` to find out about the nodes and the partitions

```
$ sinfo --format "%4D %9P  %25f  %.5c  %.8m %G"
NODE PARTITION  AVAIL_FEATURES               CPUS    MEMORY GRES
4    batch*     amd,rome,7542,zenver2          64    257790 gpu:TeslaA100:2
1    debug      amd,rome,7542,zenver2          64   1031900 (null)
```

NODES	Number of nodes with displayed characteristics
PARTITION	Partition in which nodes reside
AVAIL_FEATURES	“Features” of the node, chosen by the admins to characterise them
CPUS	Numer of “compute units” or “slots” offered by the nodes e.g. core
MEMORY	Amount of memory (RAM in MB) offered by the nodes
GRES	“Generic resources” offered by the nodes, e.g. GPUs

Use `sacctmgr` and `scontrol` to find out about QOSes and licences

```
$ sacctmgr list qos
      Name   Priority  GraceTime    Preempt   PreemptExemptTime PreemptMode
---------- ---------- ---------- ---------- ------------------- -----------
    normal          0   00:00:00                                     cluster
  priority      10000   00:00:00                                     cluster

$ scontrol show licenses
LicenseName=abaqus@ucl
    Total=48 Used=0 Free=48 Remote=yes
```

QOS: Quality of Service: used by sysadmin to organize/prioritize jobs
License: used to organise software license distribution to jobs
often used also for other cluster-wise resources

Target resources with #SBATCH parameters

You want	You ask
To choose a specific feature (e.g. a processor type or a network type)	`--constraint`
To use a generic resources (e.g. a GPU)	`--gres` (or `--gpu`)
To access a specific licensed software	`--licenses`
To chose a partition	`--partition`
To use a specific QOS	`--qos`

To choose the CPU distribution on nodes	`--nodes`
	`--ntasks-per-nodes`
	`--cpus-per-tasks`

Tune your jobs with #SBATCH parameters

You want	You ask
To set a job name	`--job-name`
To attach a comment to the job	`--comment="Some comment"`
To get emails	`--mail-type=BEGIN|END|FAILED|ALL|TIME_LIMIT_90` `--mail-user=my@mail.com`
To set the name of the output file	`--output=result-%j.txt` `--error=error-%j.txt`
To enquiry when it would start	`--test-only`
To specify an ordering	`--dependency=after(ok|notok|any):jobids` `--dependency=singleton`

Full list of options in sbatch manpage

Try to get the cluster info!

1. Connect to a cluster
2. run the “`sinfo`” command
3. run the “`sinfo -Nl`” command
4. run the “`sinfo -R`” command

run “`type sinfo`” to know if `sinfo` is
the stock `sinfo` or a modified version

run “`command sinfo`” to get the stock version

A word about limits

- Natural limits: the hardware specifications
- Admin-defined limits: to ensure fair access for everyone

e.g. max job time

View limits with `sacctmgr`

Limits that can be set :

- number of running, or submitted jobs
- size of a job
- duration of a job
- CPU usage of all jobs of a user
- cluster usage of an account
- ...

View limits with `sacctmgr`

Limits can be set :

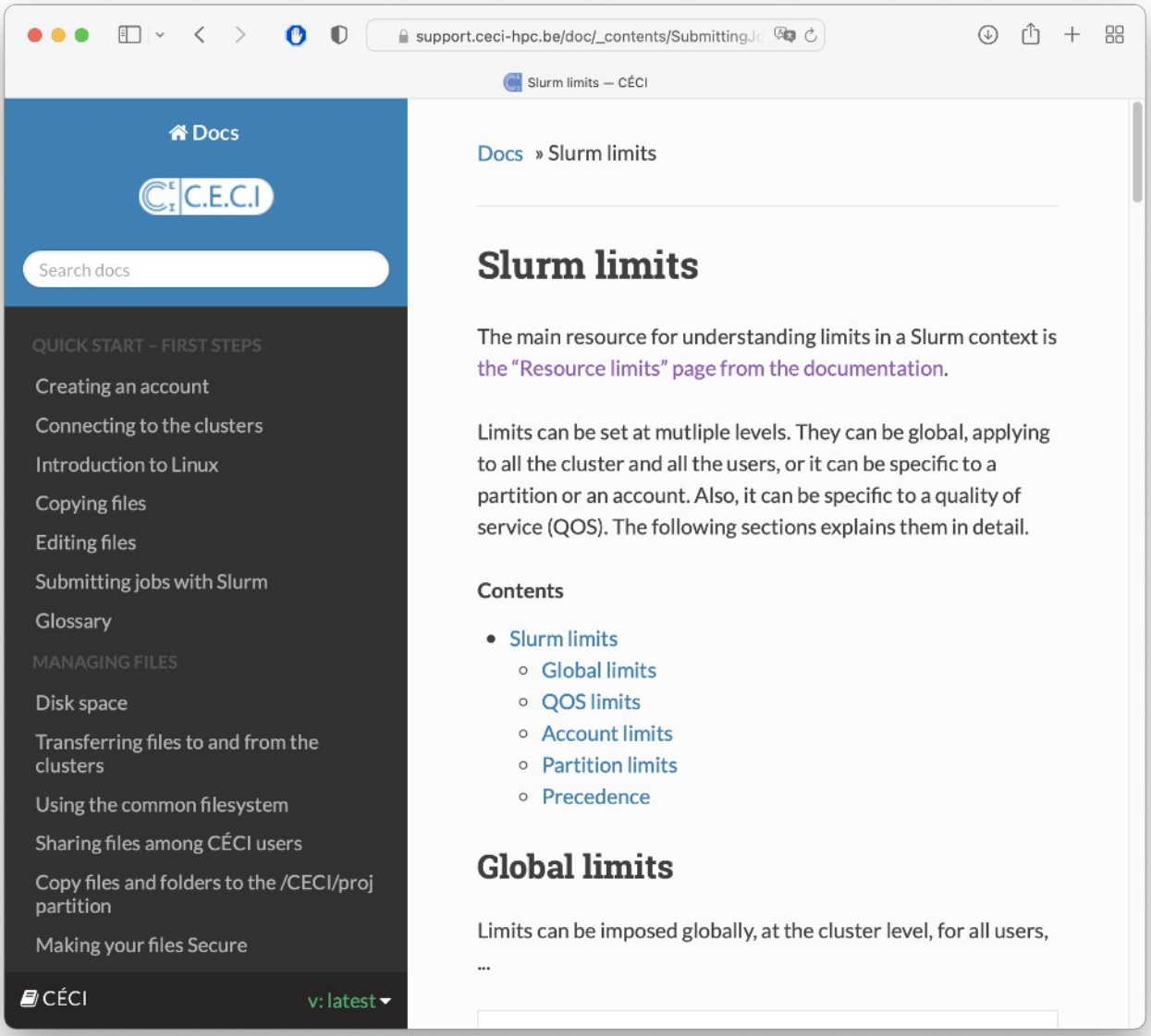
- globally for all users: `sacctmgr show cluster`
- globally for a specific user: `sacctmgr list user $USER withassoc`
- at the QOS level: `sacctmgr list qos`
- at the Account (project) level: `sacctmgr list account MyAccount withassoc where user=$USER`
- on partitions: `scontrol show partitions`

View limits with `sacctmgr`

```
dfr@nic5-login1 ~ $
dfr@nic5-login1 ~ $ sacctmgr  list user  $USER withassoc  format=User,Cluster,QOS,GrpTRES,GrpJobs,GrpSubmit,GrpSubmit,MaxTRES,MaxTRESPerUser,MaxJobsPU
      User    Cluster                 QOS       GrpTRES GrpJobs GrpSubmit       MaxTRES     MaxTRESPU MaxJobsPU
---------- ---------- ------------------- ------------- ------- --------- ------------- ------------ ---------
       dfr       nic5              normal
dfr@nic5-login1 ~ $ sacctmgr list qos format=Name,GrpTRES,GrpJobs,GrpSubmit,GrpSubmit,MaxTRES,MaxTRESPerUser,MaxJobsPU
      Name       GrpTRES GrpJobs GrpSubmit       MaxTRES     MaxTRESPU MaxJobsPU
---------- ------------- ------- --------- ------------- ------------ ---------
    normal                                        cpu=648       512
dfr@nic5-login1 ~ $
```

man sacctmgr

View limits with `sacctmgr`



https://support.ceci-hpc.be/doc/_contents/SubmittingJobs/SlurmLimits.html

View reason for which your job is pending with `squeue -l -j <JOBID>`

```
[dfr@lemaitre3 ~]$ squeue --me -l
Wed Aug 24 11:00:30 2022
CLUSTER: lemaitre3
             JOBID PARTITION     USER    STATE       TIME TIME_LIMI  NODES NODELIST(REASON)
          70786661     batch      dfr  PENDING       0:00      6:00     50 (Resources)
          70786672     batch      dfr  PENDING       0:00      6:00     50 (Priority)
          70786664     batch      dfr  PENDING       0:00      6:00      1 (BeginTime)
          70786673     batch      dfr  PENDING       0:00      6:00      1 (ReqNodeNotAvail)
          70786670     batch      dfr  PENDING       0:00      6:00      1 (Dependency)
          70786657     batch      dfr  PENDING       0:00      6:00      1 (JobHeldUser)
          70786658     debug      dfr  PENDING       0:00      6:00      5 (PartitionNodeLimit)
```

https://slurm.schedmd.com/squeue.html#SECTION_JOB-REASON-CODES

Submit your second job!

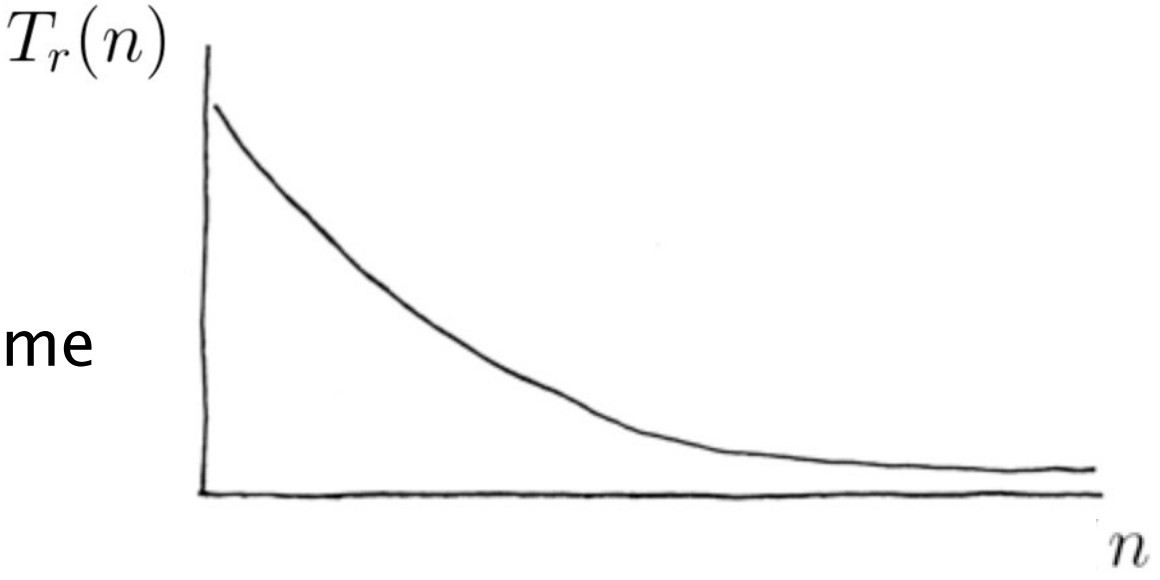
1. Connect to a cluster
2. Open a text editor and write the script for a job that will run the "`sleep 3000`" command and request a 5 minutes run time .
3. Submit the job (on a debug partition)
4. Look for files created in your directory

A word about resource requests.

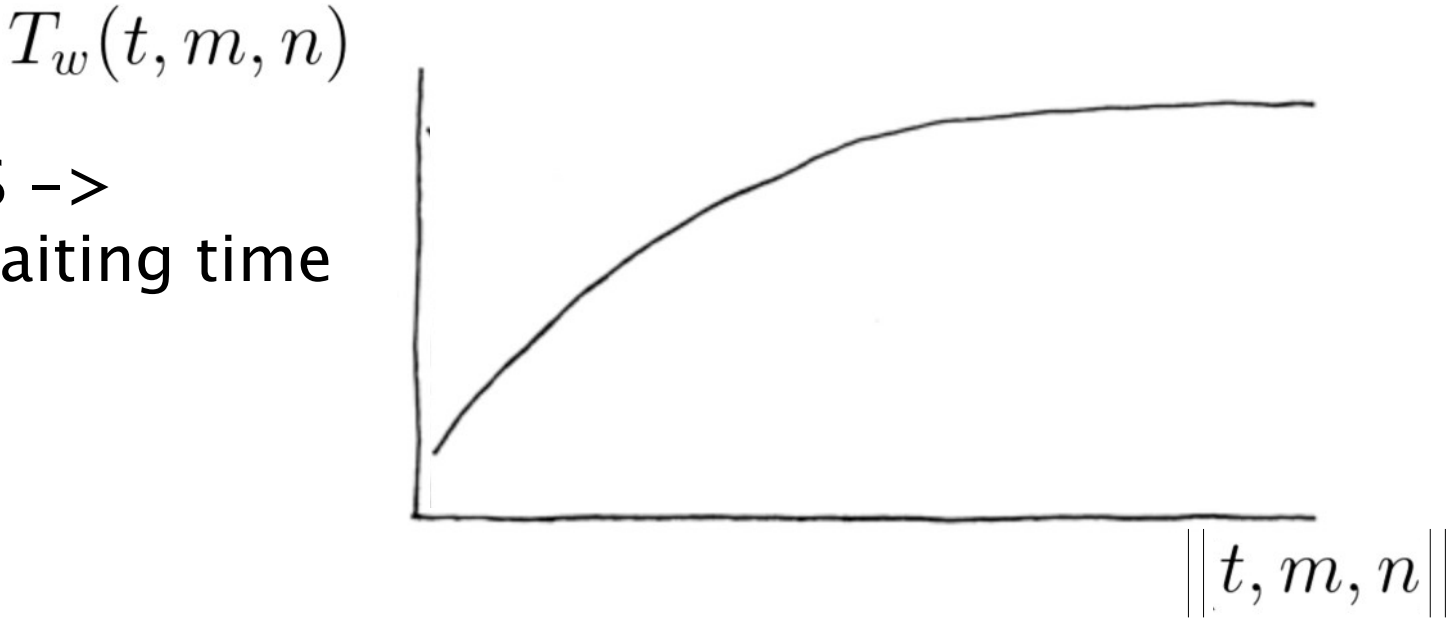
There is not magic solution to finding the optimal resource request for a given job

Too much -> idle resources -> waste of resources
Too few -> job killed -> waste of resources

More CPUs ->
less running time

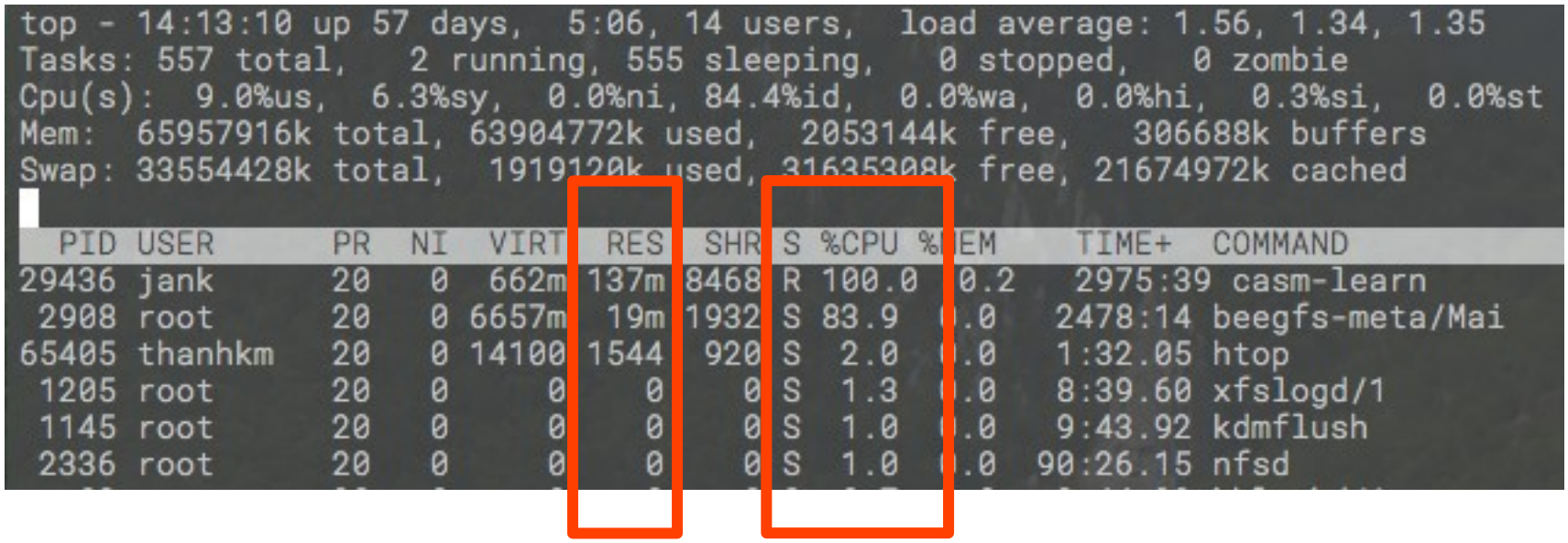


More CPUS ->
more waiting time



Practical approach

Run a sized-down problem on your laptop or the frontend and observe memory usage and CPU usage for several values of the number of CPUs with the `top` command.



Practical approach

- You can also use `/usr/bin/time -v`

(use full path not just “time”)

```
$ /usr/bin/time --verbose timeout 5s yes > /dev/null
Command exited with non-zero status 124
    Command being timed: "timeout 5s yes"
    User time (seconds): 4.92
    System time (seconds): 0.06
    Percent of CPU this job got: 99%
    Elapsed (wall clock) time (h:mm:ss or m:ss): 0:05.00
    Average shared text size (kbytes): 0
    Average unshared data size (kbytes): 0
    Average stack size (kbytes): 0
    Average total size (kbytes): 0
    Maximum resident set size (kbytes): 776
    Average resident set size (kbytes): 0
    Major (requiring I/O) page faults: 0
    Minor (reclaiming a frame) page faults: 482
    Voluntary context switches: 4
    Involuntary context switches: 30
    Swaps: 0
    File system inputs: 0
    File system outputs: 0
    Socket messages sent: 0
    Socket messages received: 0
    Signals delivered: 0
    Page size (bytes): 4096
    Exit status: 124
```

Pragmatic approach

- Use guesstimates for the first job
- Then analyze the accounting information
- Extrapolate for next jobs

Use the sstat command for running **steps** (started with srun)

```
sstat(1)                Slurm Commands                sstat(1)

NAME
       sstat  -  Display various status information of a run-
       ning job/step.

SYNOPSIS
       sstat [OPTIONS...]

DESCRIPTION
       Status  information  for  running  jobs  invoked  with
       Slurm.

       The  sstat command displays job status information for
       your analysis.  The sstat command displays information
       pertaining to CPU, Task, Node, Resident Set Size (RSS)
       and Virtual Memory (VM).  You can  tailor  the  output
       with  the  use  of the --fields= option to specify the
       fields to be shown.

       For the root user, the sstat command displays job sta-
       tus data for any job running on the system.

       For  the non-root user, the sstat output is limited to
       the user's jobs.
```

Use the sacct command for **completed** jobs

```
SACCT(1)                        Slurm components
                    SACCT(1)

NAME
       sacct - displays accounting data for all
       jobs and job  steps  in  the  SLURM  job
       accounting log or SLURM database

SYNOPSIS
       sacct [OPTIONS...]

DESCRIPTION
       Accounting  information for jobs invoked
       with SLURM are either logged in the  job
       accounting  log  file  or  saved  to the
       SLURM database.

       The sacct command displays job  account-
       ing  data  stored  in the job accounting
       log file or SLURM database in a  variety
:
```

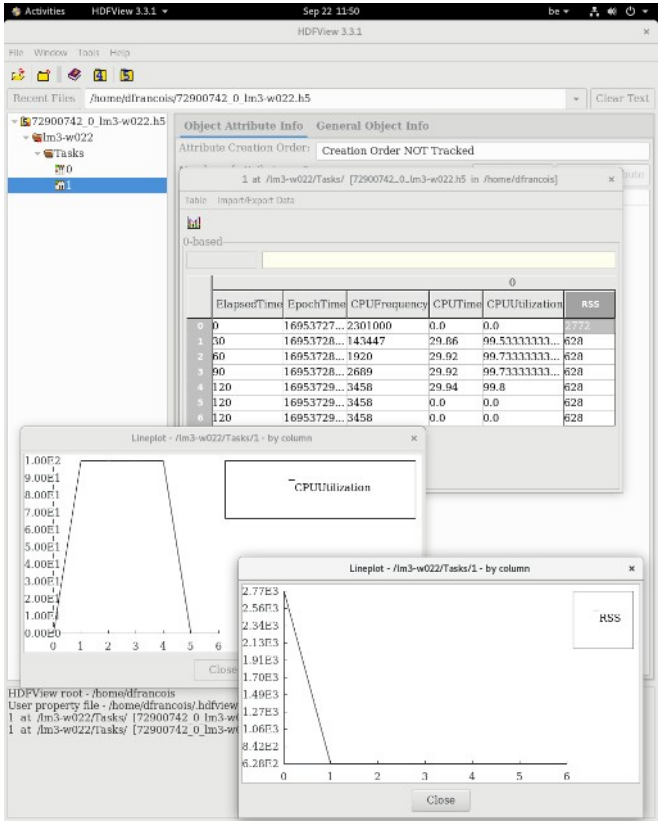
Use the `sacct` command for **completed** jobs

```
$ sacct --format Jobid,ReqMem,MaxRSS,TimeLimit,AllocCPUS,CPUTime,TotalCPU
       JobID     ReqMem     MaxRSS  Timelimit    Elapsed AllocCPUS    CPUTime  TotalCPU
------------ ---------- ---------- ---------- ---------- ---------- ---------- ---------
   12329           1Gc                00:05:00   00:03:22          2   00:06:44 06:20.781
   12329.ba+       1Gc        820K                00:03:22          2   00:06:44 06:20.780
   12329.ex+       1Gc       1044K                00:03:22          2   00:06:44 06:20.780
   12329.0         1Gc       1044K                00:00:00          2   00:06:44 00:00.001
   12329.1         1Gc       1044K                00:03:21          2   00:06:44 06:20.780
```

JobID	Job ID . Step ID of the job step
ReqMem	Requested memory (Gc: **G**igaByte per **c**ore)
MaxRSS	Actually-used memory (**R**esident **S**et **S**ize)
Timelimit	Time limit requested for the job with `--time`
Elapsed	Actual time used by the job
AllocCPUs	Number of allocated CPUs to the job
CPUTime	CPUtime allocated to the job (Elapsed * AllocCPUs)
TotalCPU	Actual CPU time consumed by the job

Use `--profile` for detailed information

```
[dfr@lemaitre3 ~] (StdEnv) $ scontrol show config | grep AcctGatherProfileType
AcctGatherProfileType   = acct_gather_profile/hdf5
[dfr@lemaitre3 ~] (StdEnv) $ scontrol show config | grep ProfileHDF5Dir
ProfileHDF5Dir          = /scratch/acct_gather
[dfr@lemaitre3 ~] (StdEnv) $ sbatch --time=5:00 -n2 --wrap "srun stress -c 2 -t 120" --profile=all
Submitted batch job 72900743 on cluster lemaitre3
[dfr@lemaitre3 ~] (StdEnv) $
[dfr@lemaitre3 dfr] (StdEnv) $ pwd
/scratch/acct_gather/dfr
[dfr@lemaitre3 dfr] (StdEnv) $ ls
72900724_0_lm3-w022.h5        72900742_0_lm3-w022.h5
72900724_batch_lm3-w022.h5    72900742_batch_lm3-w022.h5
```



- Time series for CPU usage, memory, etc.
- Might not be available on all clusters
- Self service alternative : sps

  https://github.com/OxfordCBRG/sps

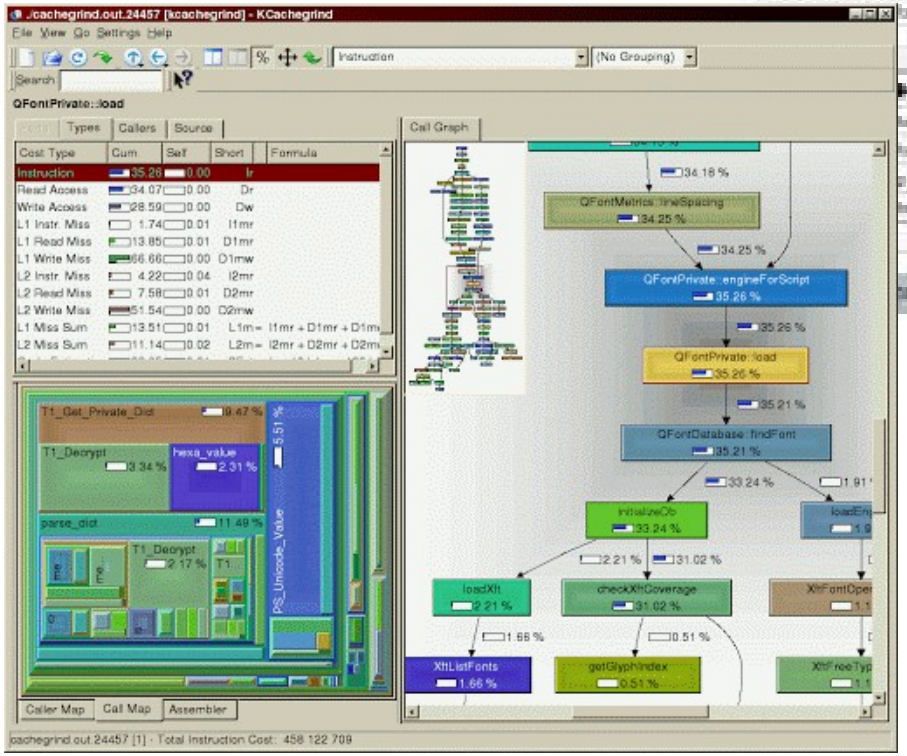
  creates .csv files

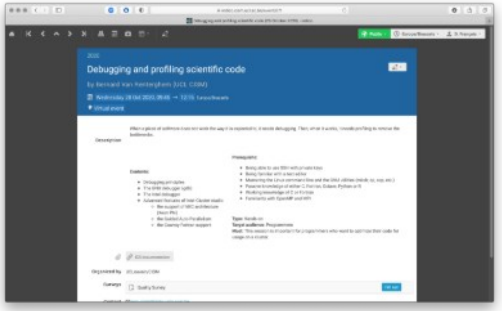
Look at your jobs!

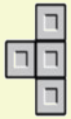
1. Connect to a cluster
2. run the `sacct` command to see your job history

Best approach

Use profiling tools...





Part  . You will learn how to:

understand priorities, fairshare,
and scheduling

in

Priority is weighted sum of multiple job/account caracteristics

```
Job_priority =
        (PriorityWeightAge) * (age_factor) +
        (PriorityWeightFairshare) * (fair-share_factor) +
        (PriorityWeightJobSize) * (job_size_factor) +
        (PriorityWeightPartition) * (partition_factor) +
        (PriorityWeightQOS) * (QOS_factor) +
        SUM(TRES_weight_cpu * TRES_factor_cpu,
            TRES_weight_<type> * TRES_factor_<type>,
            ...)
```

https://slurm.schedmd.com/priority_multifactor.html

Use sprio to get the details

```
SPRIO(1)                          SLURM commands
                    SPRIO(1)

NAME
       sprio - view the factors that comprise a
       job's scheduling priority

SYNOPSIS
       sprio [OPTIONS...]

DESCRIPTION
       sprio is used to view the components  of
       a  job's  scheduling  priority  when the
       multi-factor    priority    plugin    is
       installed.  sprio is a read-only utility
       that  extracts  information   from   the
       multi-factor    priority    plugin.    By
       default, sprio returns  information  for
       all pending jobs.  Options exist to dis-
       play specific jobs by job  ID  and  user
:
```

Check the priority settings!

1. Connect to a cluster
2. Run `sprio -w`
3. Run `scontrol show conf|grep ^Priority`
4. Look for the meaning of the items with `man slurm.conf | grep Priority`

The “faireshare” factor helps everyone getting access to resources

- A share is allocated to you: 1/#users
- If your actual usage is above that share, your fairshare value is decreased towards 0.
- If your actual usage is below that share, your fairshare value is increased towards 1.
- The actual usage taken into account decreases over time; usage two months ago has less impact on the fairshare than usage two days ago.

The Slurm Fairshare Formula

- The Slurm Fairshare formula has been designed to provide fair scheduling to users based on the allocation and usage of **every** account. Now, the usage term is **effective** usage:

$$F = 2^{(-U_E/S)} \quad \text{(Effective Usage Formula)}$$

$$U_E = U_{Achild} + ((U_{Eparent} - U_{Achild}) * S_{child}/S_{all\_siblings})$$

Where:

U_E is the effective usage of the child user or child account

U_{Achild} is the actual usage of the child user or child account

UE_{parent} is the effective usage of the parent account

S_{child} is the shares allocated to the child user or child account

$S_{all\_siblings}$ is the shares allocated to all the children of the parent account

Fairshare-Decay Factor

- Most workload spans multiple time periods. Slurm's fairshare priority calculation places more importance on the most recent resource usage and less importance on usage from way back
- The metric used is based on a half-life formula that favors most recent usage statistics, based on a **decay** factor (D):

$$U_H = U_{current\_period} + (D * U_{last\_period}) + (D * D * U_{period-2}) + \ldots$$

Where:

U_H	is the historical usage subject to the half-life decay
$U_{current\_period}$	is the usage charged over the current measurement period
U_{last_period}	is the usage charged over the last measurement period
$U_{period-2}$	is the usage charged over the second last measurement period
D	is a decay factor between zero and one that delivers the half-life decay based off the PriorityDecayHalfLife setting in the slurm.conf file

Get your current share with sshare

```
SSHARE(1)                       SLURM Commands
                SSHARE(1)

NAME
       sshare  - Tool for listing the shares of
       associations to a cluster.

SYNOPSIS
       sshare [OPTIONS...]

DESCRIPTION
       sshare is used to view SLURM share
       information.  This command is only
       viable when running with the prior-
       ity/multifactor plugin.  The sshare
       information is derived from a database
       with the interface being provided by
       slurmdbd (SLURM Database daemon) which
       is read in from the slurmctld and used
       to process the shares available to a
:
```

Get your current share with sshare

```
[dfr@lemaitre3 ~]$ sshare -a -l | grep -v 0.000000 | head -20
Account                    User  RawShares  NormShares    RawUsage   NormUsage  EffectvUsage  FairShare                    GrpTRESMins                  TRESRunMins
-------------------- ---------- ---------- ----------- ----------- ----------- ------------- ---------- ------------------------------ ------------------------------
root                                          1.000000   823547414                 1.000000   0.870551                                cpu=2585068,mem=8474861656,en+
 ceci                                1000000    0.999998   823547414    1.000000     1.000000   0.870550                                cpu=2585068,mem=8474861656,en+
  ceci                  aishamwe          1    0.000248      672111    0.000816     0.001064   0.551422                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                  alaertsl          1    0.000248       16012    0.000019     0.000267   0.861131                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                  alsteens          1    0.000248       33202    0.000040     0.000288   0.851133                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                     apatil         1    0.000248       41848    0.000051     0.000299   0.846148                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                  asandron          1    0.000248      765941    0.000930     0.001178   0.517367                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                   asasani          1    0.000248       18786    0.000023     0.000271   0.859510                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                     asion          1    0.000248     1063616    0.001292     0.001539   0.422638                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                   astassi          1    0.000248     1569463    0.001906     0.002153   0.299720                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                  astanciu          1    0.000248        7184    0.000009     0.000256   0.866311                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                   benadil          1    0.000248        1882    0.000002     0.000250   0.869437                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                  bmajerus          1    0.000248       39644    0.000048     0.000296   0.847416                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                  cbouquia          1    0.000248     3904922    0.004742     0.004988   0.061323                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                  ccarpent          1    0.000248        4595    0.000006     0.000253   0.867837                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                     chunli         1    0.000248    10552009    0.012815     0.013059   0.000670                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                   cvinnis          1    0.000248       13749    0.000017     0.000264   0.862456                                cpu=0,mem=0,energy=0,node=0,b+
  ceci                   davenet          1    0.000248     3554430    0.004316     0.004563   0.077811                                cpu=0,mem=0,energy=0,node=0,b+
```

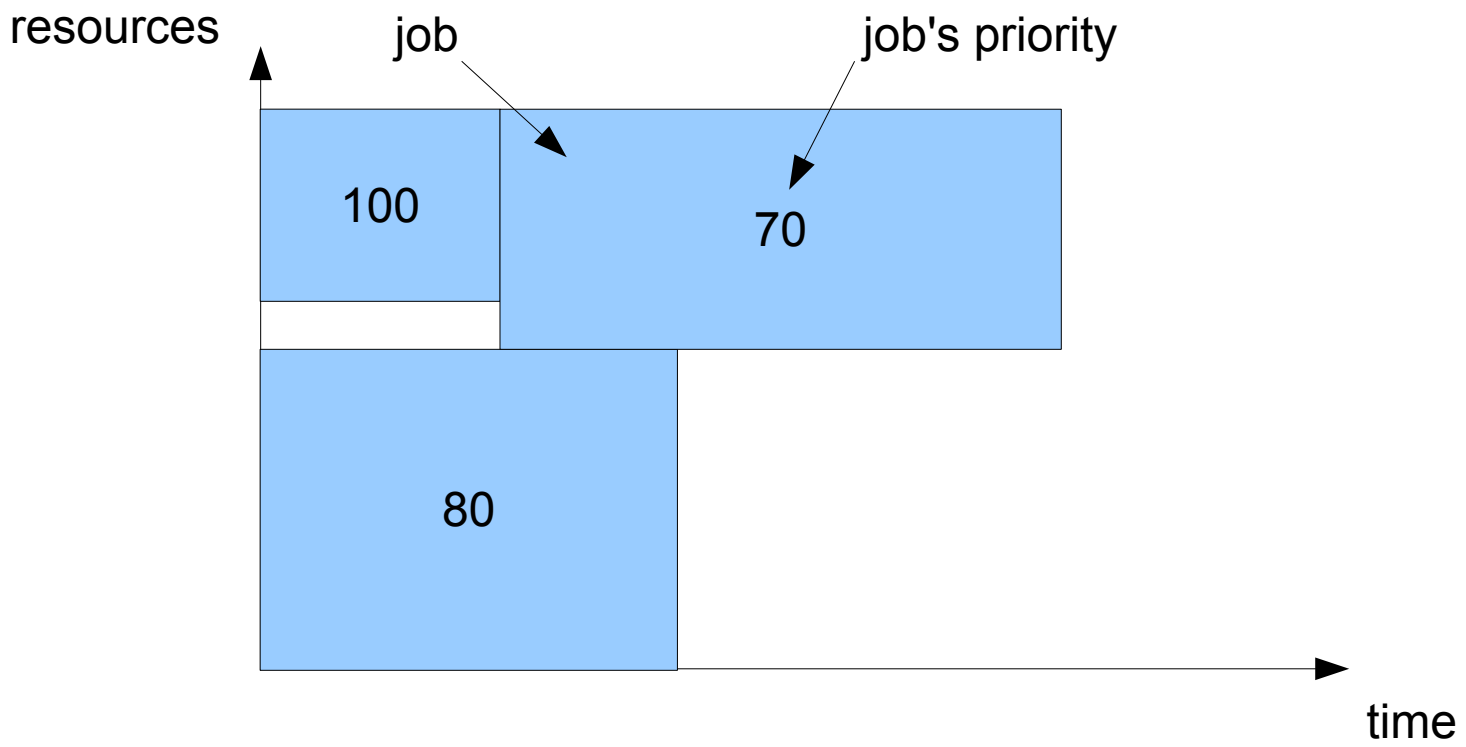
Normalised share for CÉCI 1000000/(1000000+1+1) = 0,999998
Normalised share for a CÉCI user 0,999998 * 1/4037 = 0,0002477081992
RawUsage User1 = 672111
NormalisedUsage User1 = 672111 / 823547414 = 0,0008161169455
EffectiveUsage User1 = 0,0008161169455 + (1,000000 - 0,0008161169455)*0,0002477081992/0,999998 = 0,001063623481
FairShare User1 = 2**(-0,001063623481/ 0,0002477081992 /5) = 0,5514219814

```
[dfr@lemaitre3 ~]$ scontrol show config | grep Damp
FairShareDampeningFactor = 5
```

```
[dfr@lemaitre3 ~]$ sshare -an | grep '  ceci' | wc -l
4037
```

Resources are “reserved” for top job but small jobs can be “backfilled”

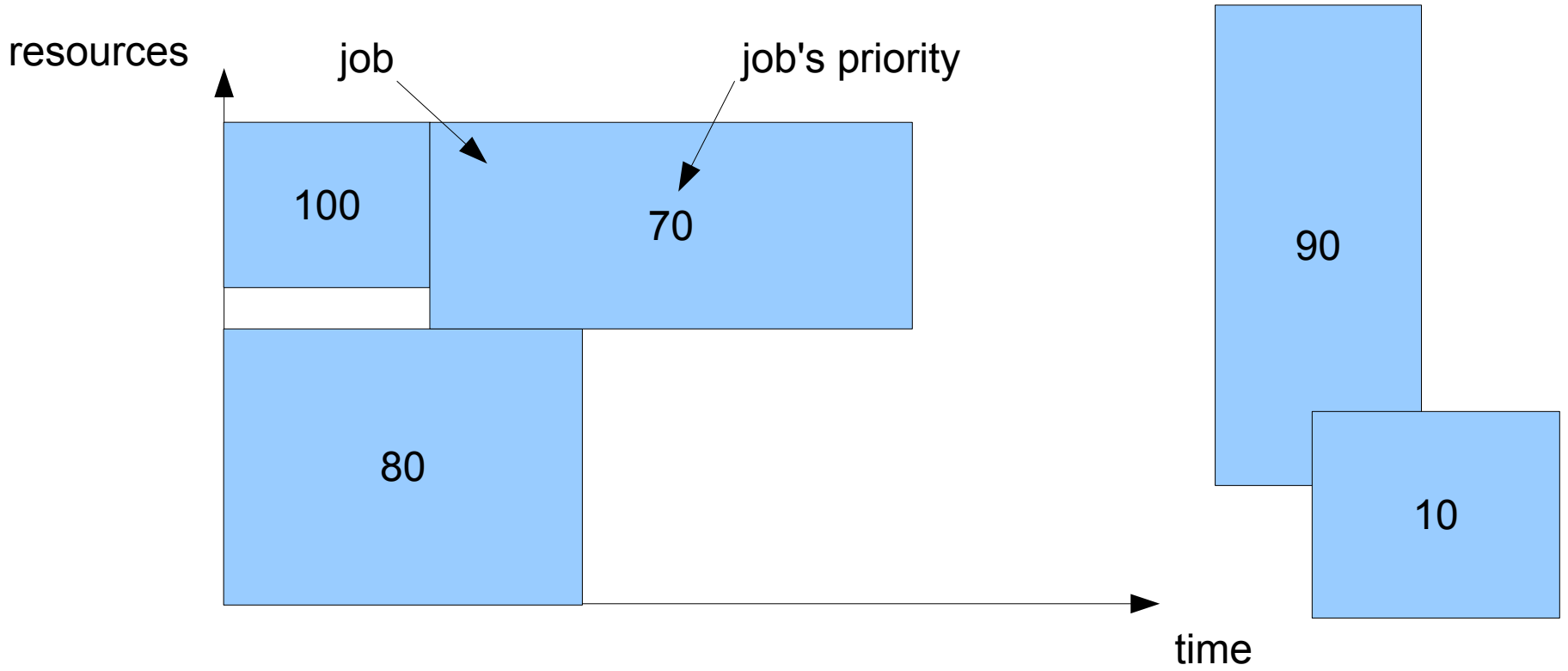
A job with a lower priority can start before a job with a higher priority if it does not delay that job's start time.



A job is a number of cpus times duration

Resources are “reserved” for top job but small jobs can be “backfilled”

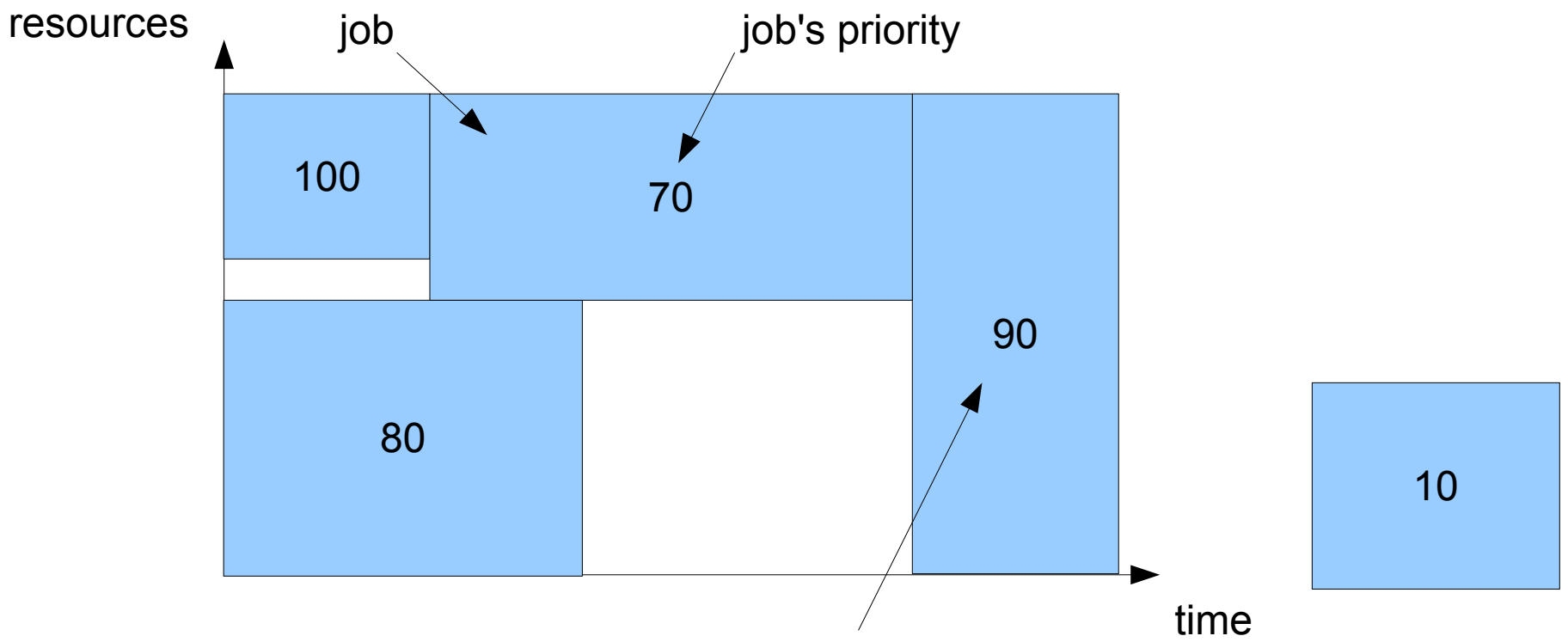
A job with a lower priority can start before a job with a higher priority if it does not delay that job's start time.



Two more jobs to schedule

Resources are “reserved” for top job but small jobs can be “backfilled”

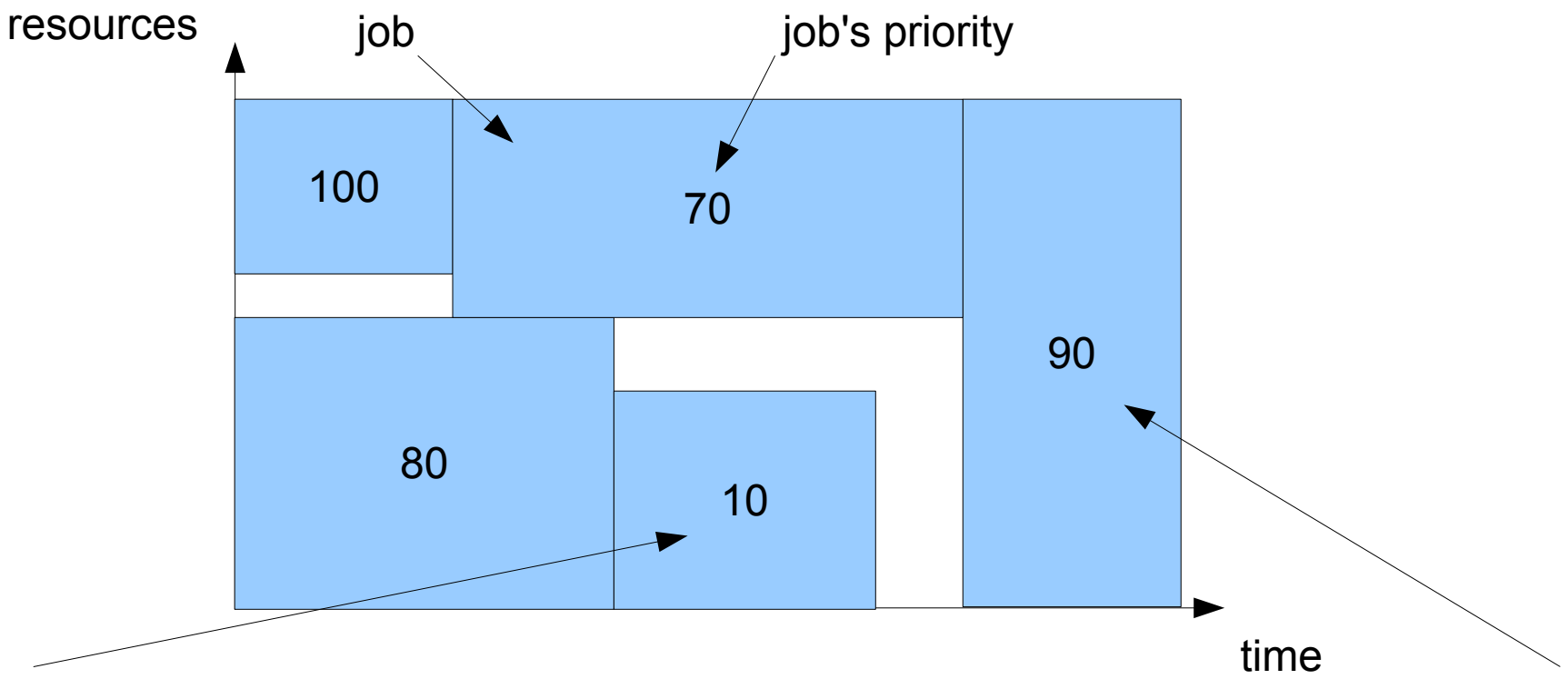
A job with a lower priority can start before a job with a higher priority if it does not delay that job's start time.



This job must wait until job with priority 70 is finished because it needs its resources

Resources are “reserved” for top job but small jobs can be “backfilled”

A job with a lower priority can start before a job with a higher priority if it does not delay that job's start time.



Low priority job has short max run time and less requirements ; it starts before larger priority job

Part ⊞. You will learn how to write submission scripts for :

Multi-node SPMD programs (e.g. MPI)
Single-node shared memory programs (e.g. OpenMP)
Master/slave programs
Embarrassingly parallel workloads
Accelerators (GPUs)

Clusters are *parallel* machines.
They work best with *parallel* jobs.

Types of parallel jobs:

- shared memory, multi-core
- distributed memory, multi-node
- accelerators (GPU)
- embarrasingly parallel

Depends on the sofware !
No magic unfortunately

Example scripts in /CECI/proj/training/slurm

Code (program.c)

Text file

```c
#include <stdio.h>

int main(void)
{
    printf("Hello, World!\n");
}
```

Compiler

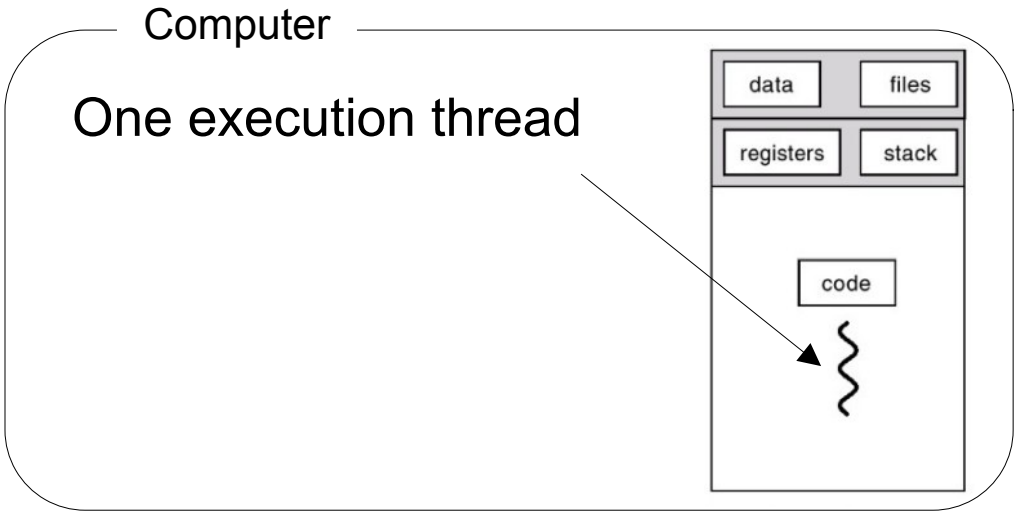
Binary (program.exe)

Executable file

Loader

Process (PID 1235)

Running instance



Code (program.c)

Text file

```c
#include <stdio.h>

int main(void)
{
    printf("Hello, World!\n");
}
```

Compiler

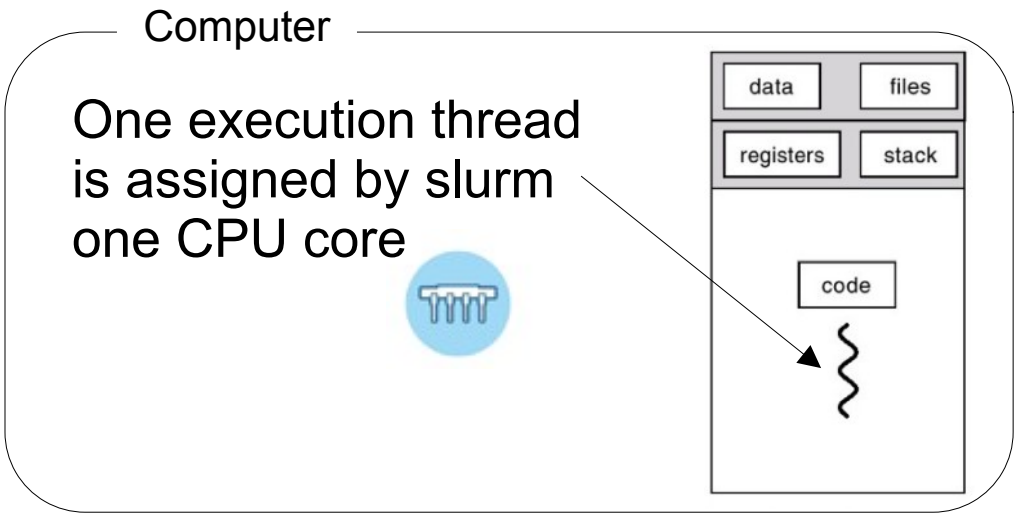
Binary (program.exe)

Executable file

Loader

Process (PID 1235)

Running instance



Code (program.c)

Text file

```c
#include <stdio.h>

int main(void)
{
    printf("Hello, World!\n");
}
```

Compiler

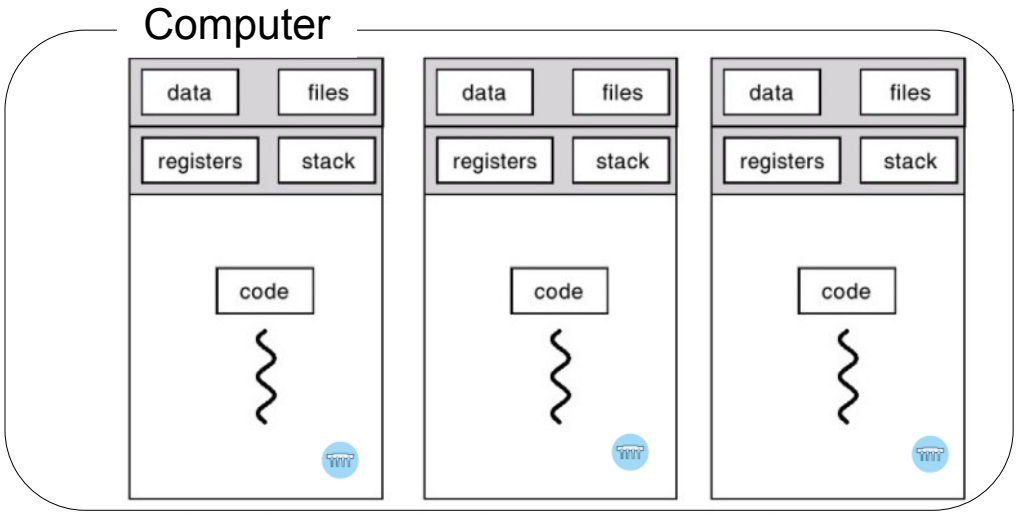
Binary (program.exe)

Executable file

Loader, called multiple times

Multiple Processes

Running instances



Forking Code

Text file

```c
#include <stdio.h>
#include <sys/types.h>;
#include <unistd.h>;
int main()
{

    // make two process which run same
    // program after this instruction
    fork();

    printf("Hello world!\n");
    return 0;
}
```

Compiler

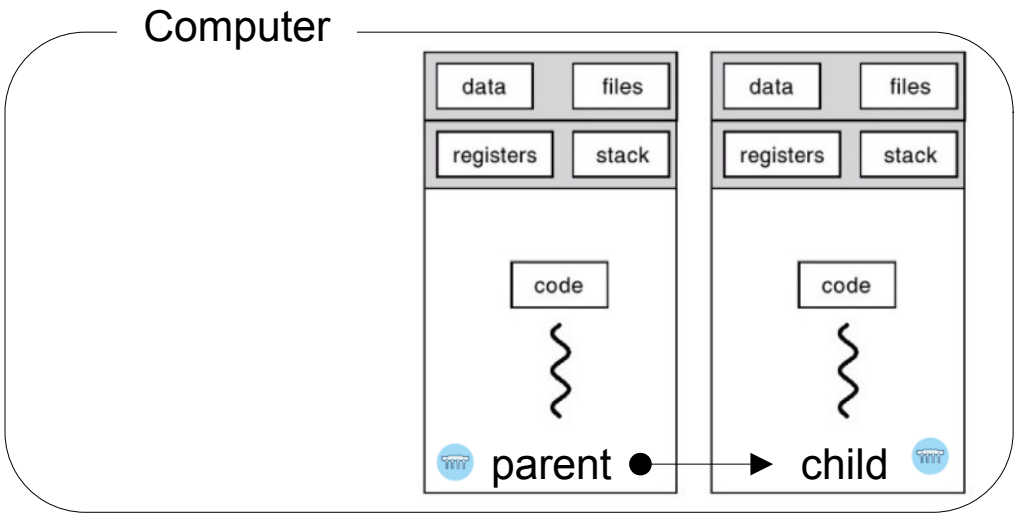
Single binary

Executable file

Loader, called once

Multiple Processes

Running instances



IPC – Inter-process communication

Multithreaded Code

Text file

Compiler

Single binary

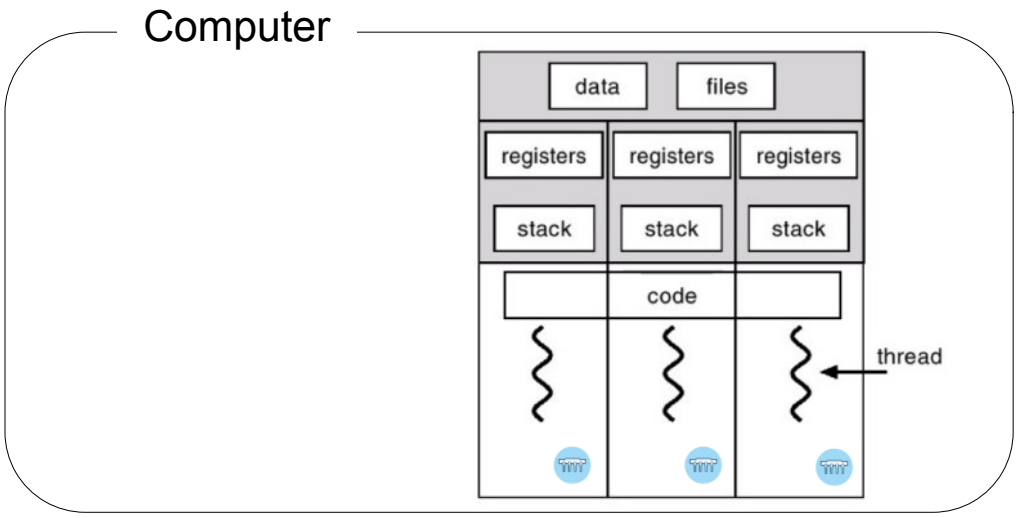
Executable file

Loader, called once

Multithread process

Running instance

```
void print_message_function( void *ptr );

main()
{
   pthread_t thread1, thread2;
   char *message1 = "Hello";
   char *message2 = "World";

   pthread_create( &thread1, pthread_attr_default,
                   (void*)&print_message_function, (void*) message1);
   pthread_create(&thread2, pthread_attr_default,
                   (void*)&print_message_function, (void*) message2);

   exit(0);
}

void print_message_function( void *ptr )
{
   char *message;
   message = (char *) ptr;
   printf("%s ", message);
}
```



Code (program.c)

Text file

```c
#include <stdio.h>

int main(void)
{
    printf("Hello, World!\n");
}
```

Compiler

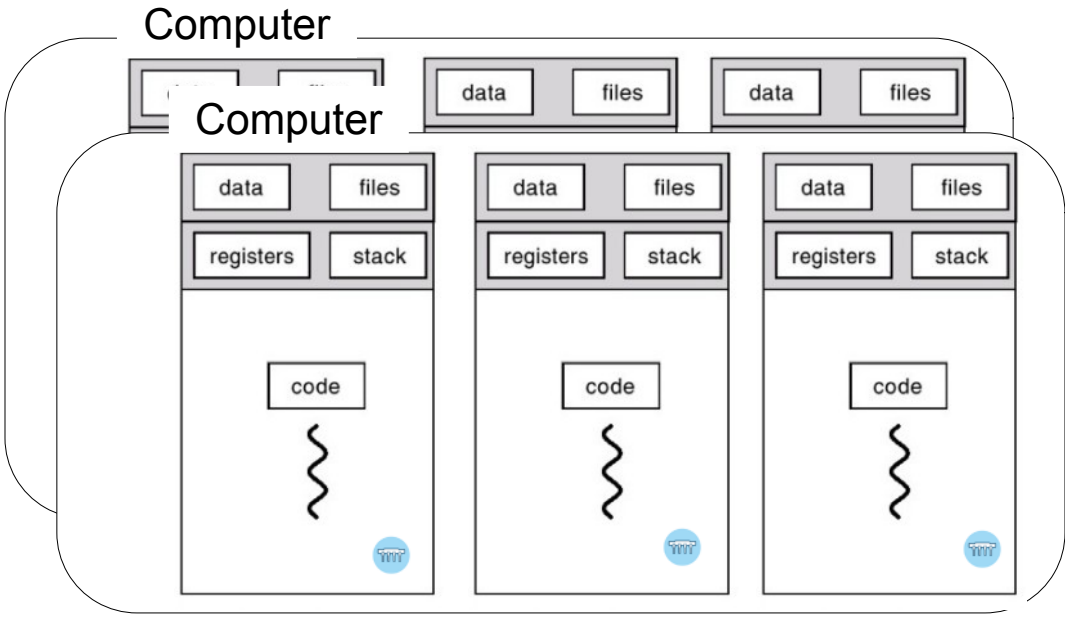
Binary (program.exe)

Executable file

srun, called once

Multiple Processes possibly on multiple nodes

Running instances



A multi-node job is possible **only** if

– all processes are independent ; or

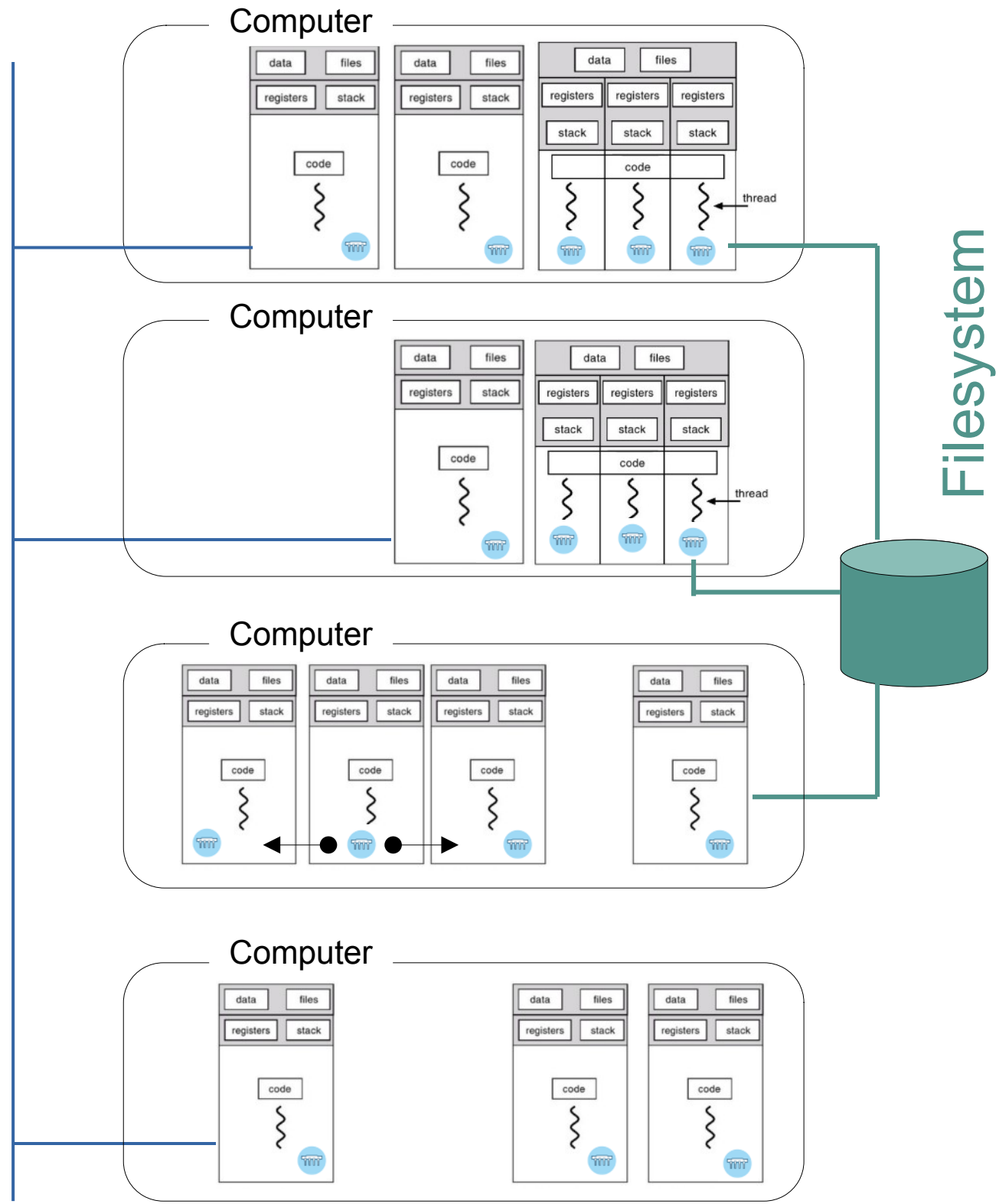
Embarrasingly parallel

– processes communicate through files on a common filesystem/DB ; or

e.g. Master/slave setup

– processes communicate through the network thanks to a dedicated library

e.g. SPMD setup with MPI



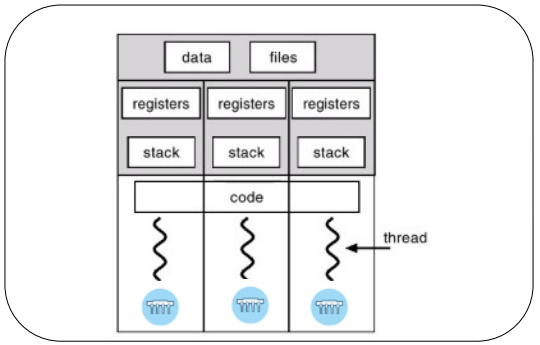
`srun, --ntasks, --ncpus-per-task`

A parallel job typically comprises a sequence of *steps,*
each made of multiple *parallel tasks.*

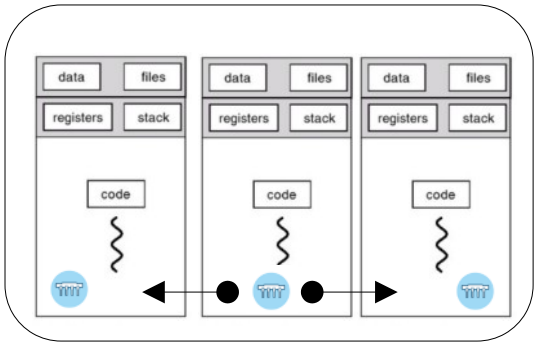
A step is a single invocation of `srun`
A task is a process started by `srun`

A single task can be assigned multiple CPUs
A single task cannot be spread across multiple nodes
Multiple steps can run in parallel if they use a subset of the allocation

Single-node job: Specify a number of “CPUs”



or



You want	You ask
N CPUs to launch *N* threads or processes on the same node (=single task)	`--cpus-per-task=`*N*

submit-omp.sh

```
#! /bin/bash
#SBATCH --cpus-per-task=3

module load GCC
gcc -fopenmp /CECI/proj/training/slurm/omp_hello_world.c -o omp_hello_world
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

./omp_hello_world
```

Or request a full node

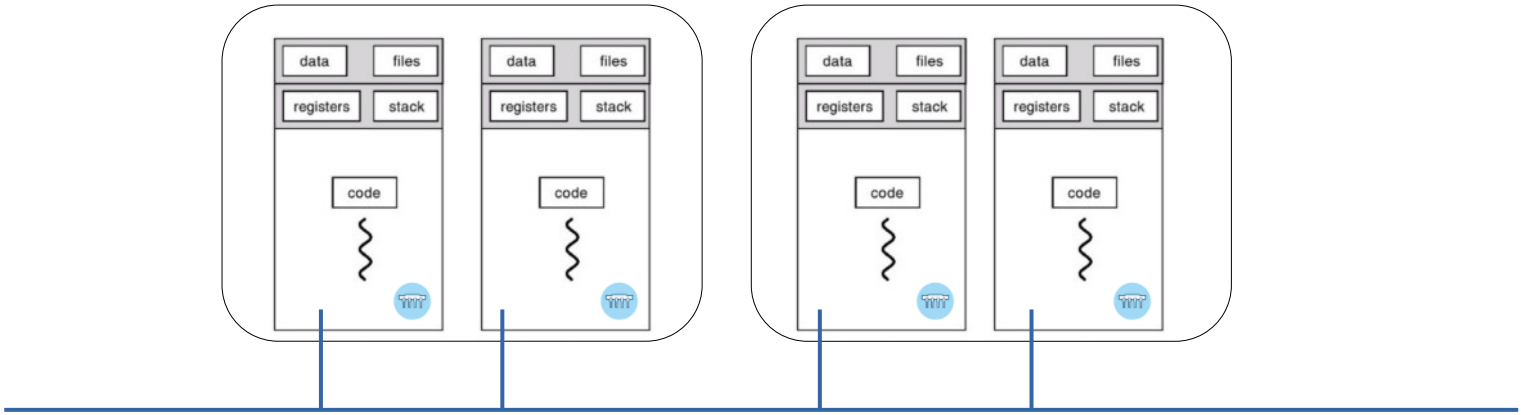
You want	You ask

All the CPUs on the node and all the memory	`--nodes=1`
	`--exclusive`
	`--mem=0`

submit-omp.sh

```
#! /bin/bash
#SBATCH --nodes=1
#SBATCH --exclusive
#SBATCH --mem=0

module load GCC
gcc -fopenmp /CECI/proj/training/slurm/omp_hello_world.c -o omp_hello_world
export OMP_NUM_THREADS=$SLURM_CPUS_ON_NODE

./omp_hello_world
```

Multi-node job: Specify a number of “tasks”



You want	You ask
N CPUs, to launch *N* MPI processes	`--ntasks=`*N*

submit-mpi.sh

```
#! /bin/bash
#SBATCH --ntasks=4

module load OpenMPI
mpicc /CECI/proj/training/slurm/mpi_hello_world.c -o mpi_hello_world

#mpirun ./mpi_hello_world
srun ./mpi_hello_world
```

Specify a number of “tasks” and optionally a number of “nodes”

You want	You ask
N CPUs	`--ntasks=`*N*
N CPUs spread across distinct nodes	`--ntasks=`*N* `--nodes=`*N* *or* *--ntasks=N --ntasks-per-node=1*
N CPUs spread across distinct nodes and nobody else around	`--nodes=`*N* `--exclusive`
N CPUs spread across *N/2* nodes	`--ntasks=`*N* `--ntasks-per-node=2`
N CPUs on the same node	`--ntasks=`*N* `--ntasks-per-node=`*N* *or* *--ntasks=N --nodes=1*
N CPUS spread accross as many nodes as possible	`--ntasks=`*N* `--spread-job`
Between *8* and *16* CPUs based on what is available	`--nodes=`*4*`-8 --ntasks-per-node=2`

Use srun --multi-prog

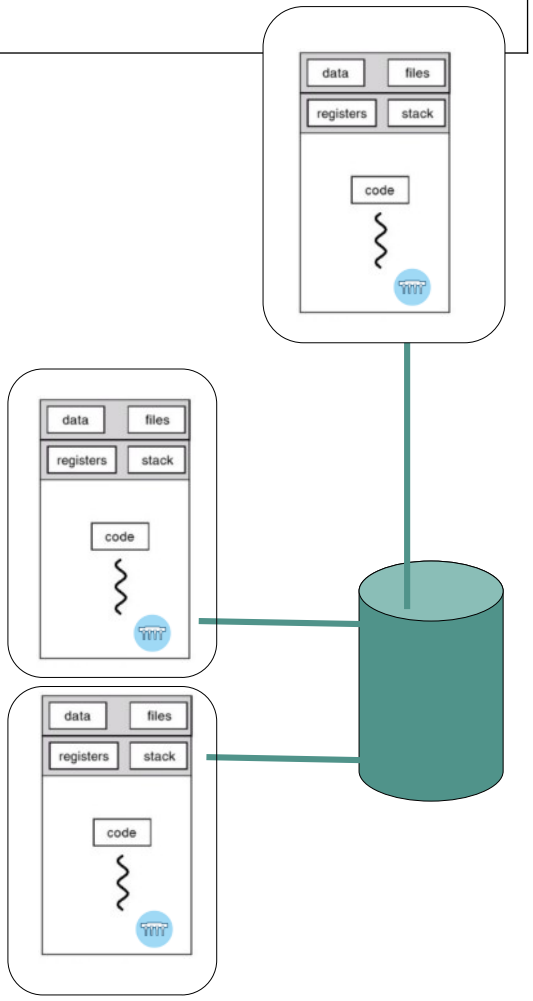
You want	You ask
N CPUs to launch *N* processes	`--ntasks=`*N*

submit-masterslave.sh

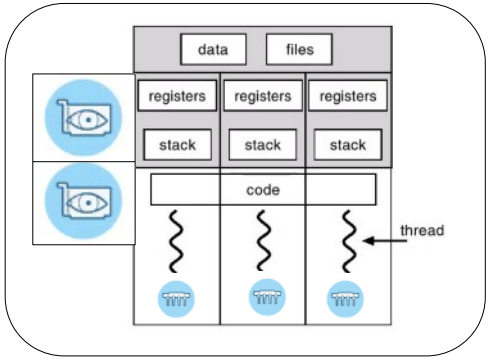
```
#! /usr/bin/env bash
#SBATCH --ntasks=3

cp /CECI/proj/training/slurm/coordinator.sh .
cp /CECI/proj/training/slurm/worker.sh .
cp /CECI/proj/training/slurm/multi.conf .

srun --multi-prog multi.conf
```

multi.conf

```
# multi.conf for --multi-prog
0    ./coordinator.sh
1-2  ./worker.sh
```



Request a GPU with `--gres` or `--gpu`



You want	You ask
N GPUs *N* GPUs per node	`--gpus=N` `--gres=gpu:N`
1 specific GPU (e.g. TeslaV100)	`--gpus=TeslaV100:1` `--gres=gpu:TeslaV100:1`

submit.sh

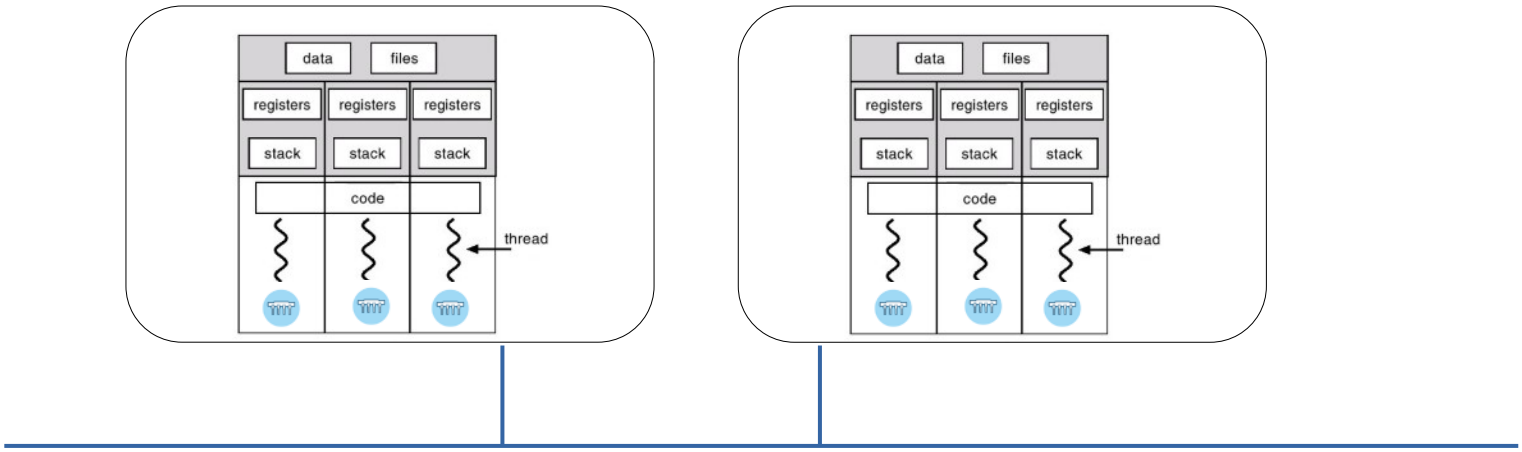
```
#! /bin/bash

#SBATCH --cpus-per-task=3
#SBATCH --mem-per-cpu=1g
#SBATCH --gres=gpu:2

module load CUDA # or cuda on some clusters
nvidia-smi
```

Hybrid jobs

with for instance MPI and OpenMP

submit.sh

```
#! /bin/bash
#
#SBATCH --ntasks=2
#SBATCH --ncpus-per-task=3

module load OpenMPI
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

srun ./hello_world_mpi
```



Create job arrays with `--array`

Using `--array=1-4`, one submission of the script will generate 4 jobs managed as a whole by Slurm.

```
[dfr@lemaitre3 ~] $ sbatch /CECI/proj/training/slurm/submit-array.sh
[dfr@lemaitre3 ~] $ squeue --me
CLUSTER: lemaitre3
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
    72772281_[1-4]     batch array_he      dfr PD       0:00      1 (Priority)
```

Each job will "see" a different value for `$SLURM_ARRAY_TASK_ID`

You want	You ask
N CPUs to launch *N* completely independent jobs	`--array=1-`*N*

submit-array.sh

```
#! /usr/bin/env bash
#SBATCH --array=1-4

[ ! -f ./array_hello.sh ] && \
     cp /CECI/proj/training/slurm/array_hello.sh .

./array_hello.sh $SLURM_ARRAY_TASK_ID
```

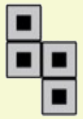
Set job dependencies with --dependency

Using --dependecy=afterok:12345, the submitted job will only start after job 12345 successfully completed

```
[dfr@lemaitre3 ~] $ sbatch /CECI/proj/training/slurm/job-dependee.sh
Submitted batch job 72772285 on cluster lemaitre3
[dfr@lemaitre3 ~] $ sbatch --dependency=afterok:72772285 /CECI/proj/training/slurm/job-
dependent.sh
Submitted batch job 72772286 on cluster lemaitre3
[dfr@lemaitre3 ~] $ squeue  --me
CLUSTER: lemaitre3
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          72772286     batch dependen      dfr PD       0:00      1 (Dependency)
          72772285     batch dependee      dfr PD       0:00      1 (Priority)
```

Dependent jobs will wait for dependee.

You want	You ask
Job B to start after Job A	`--dependency=afterok:<JOBID>`

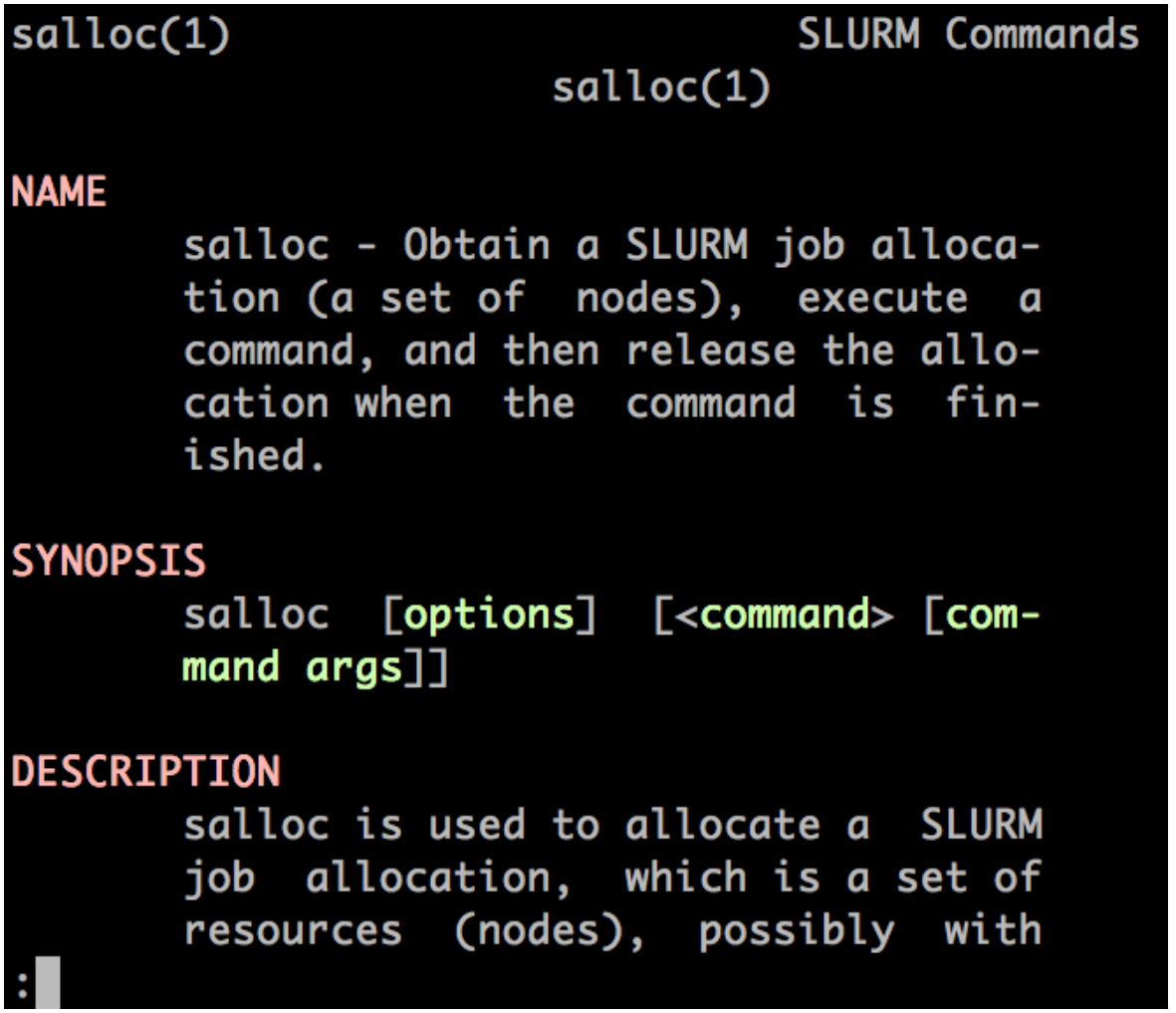
Part  . You will learn how to:

create an interactive Bash session

launch JupyterLab or Rstudio

with

Use salloc to test multi-node setups



e.g. salloc –-ntasks=4 --nodes=2

Use `salloc` to test multi-node setups

```
[dfr@lemaitre3 ~]$ salloc --partition debug --ntasks 2 --nodes 2
salloc: Pending job allocation 70299307
salloc: job 70299307 queued and waiting for resources
salloc: job 70299307 has been allocated resources
salloc: Granted job allocation 70299307
salloc: Waiting for resource configuration
salloc: Nodes lm3-w[091-092] are ready for job

CÉCI clusters: Lemaitre3 - Dragon1 - Dragon2 - Hercules2 - NIC4 - NIC5


   The new NIC5 cluster is now available: give it a try!
   More info on http://www.ceci-hpc.be/clusters.html#nic5

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
289/1984 CPUs available (load 85%) - 116 jobs running, 299 pending.

   You currently have 1 job running, 0 pending.
   You are using 21.1G ( out of 100G ) in $HOME.
   You have 0G of data on $GLOBALSCRATCH.

[dfr@lemaitre3 ~]$ ml OpenMPI
[dfr@lemaitre3 ~]$ mpirun mpi_hello_world
Hello world from processor lm3-w091.cluster, rank 0 out of 2 processors
Hello world from processor lm3-w092.cluster, rank 1 out of 2 processors
[dfr@lemaitre3 ~]$ exit
exit
salloc: Relinquishing job allocation 70299307
salloc: Job allocation 70299307 has been revoked.
[dfr@lemaitre3 ~]$
```

Use srun for a shell on a compute node

```
 Contact, support: https://support.ceci-hpc.be/cecihelp/
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Last login: Wed Nov  3 10:01:33 2021 from 130.104.1.234

CÉCI clusters: Lemaitre3 - Dragon1 - Dragon2 - Hercules2 - NIC4 - NIC5


   The new NIC5 cluster is now available: give it a try!
   More info on http://www.ceci-hpc.be/clusters.html#nic5

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
300/4928 CPUs available (load 93%) - 199 jobs running, 263 pending.

   You currently have 0 job running, 0 pending.
   You are using 0GB (out of 110GB) in $HOME and 271 files (out of 110000).
   You are using 988K (out of 5.0T) in $GLOBALSCRATCH and 6 files (out of 500000).

   Don't know where to start?
            --> http://www.ceci-hpc.be/install_software.html
            --> http://www.ceci-hpc.be/slurm_tutorial.html

dfr@nic5-login1 ~ $ srun --pty bash -l
dfr@nic5-w034 ~ $ hostname
nic5-w034
dfr@nic5-w034 ~ $
```

```
srun --pty bash -l
```

Get an interactive job!

1. Connect to a cluster
2. Run `srun --pty bash -l`
3. Use `hostname` to find out which node was allocated to your job

Get an interactive job!

1. Connect to a cluster
2. Run `srun --pty bash -l`
3. Use `hostname` to find out which node was allocated to your job

On more recent versions of Slurm the `salloc` command can be configured to start the shell on the first node of the allocation automatically.

How to start a Jupyterlab instance >

1. Use `srun` for shell on compute node

```
dfr@nic5-login1 ~ $ srun --pty -c 4 bash -l
srun: job 1824417 queued and waiting for resources
srun: job 1824417 has been allocated resources
dfr@nic5-w022 ~ $ ml releases/2020b JupyterLab

The following have been reloaded with a version change:
  1) releases/2019b => releases/2020b

dfr@nic5-w022 ~ $ jupyter notebook --ip $(hostname -i)
[I 09:36:24.669 NotebookApp] Serving notebooks from local directory: /home/users/d/f/dfr
[I 09:36:24.670 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 09:36:24.670 NotebookApp] http://10.252.2.22:8888/?token=2b8a7237a778e8e3e5a2be95f6c697edee288457d0e09ff4
[I 09:36:24.670 NotebookApp]  or http://127.0.0.1:8888/?token=2b8a7237a778e8e3e5a2be95f6c697edee288457d0e09ff4
[I 09:36:24.670 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 09:36:24.687 NotebookApp] No web browser found: could not locate runnable browser.
[C 09:36:24.688 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/users/d/f/dfr/.local/share/jupyter/runtime/nbserver-3732674-open.html
    Or copy and paste one of these URLs:
        http://10.252.2.22:8888/?token=2b8a7237a778e8e3e5a2be95f6c697edee288457d0e09ff4
     or http://127.0.0.1:8888/?token=2b8a7237a778e8e3e5a2be95f6c697edee288457d0e09ff4
```

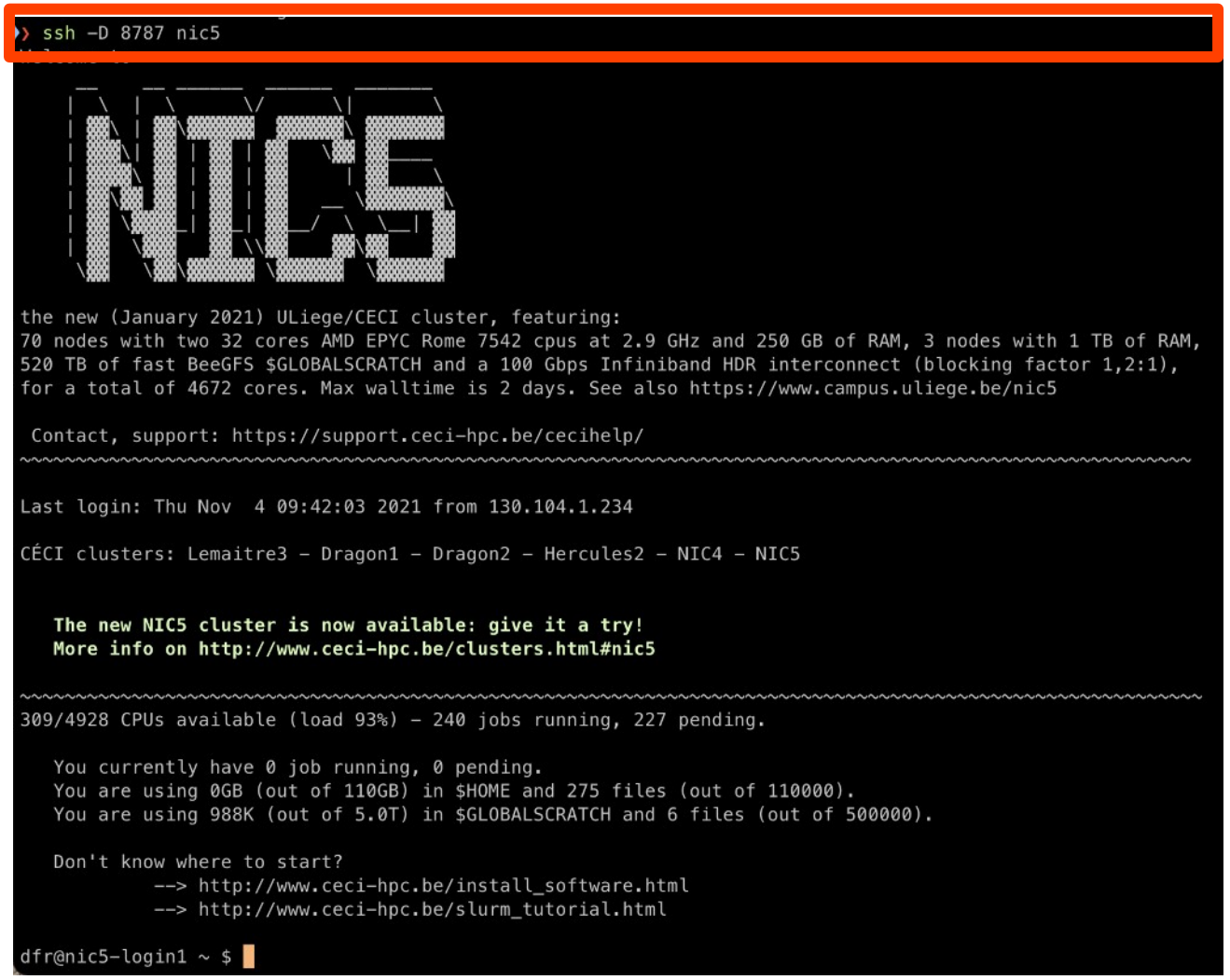
How to start a Jupyterlab instance >

2. Load module and start service

```
dfr@nic5-w055 ~ $ jupyter-lab --ip=$(hostname -i)
[W 09:49:01.604 LabApp] JupyterLab server extension not enabled, manually loading...
[I 09:49:01.609 LabApp] JupyterLab extension loaded from /opt/cecisw/arch/easybuild/2020b/software/JupyterLab/2.2.8-GCCc
ore-10.2.0/lib/python3.8/site-packages/jupyterlab
[I 09:49:01.609 LabApp] JupyterLab application directory is /opt/cecisw/arch/easybuild/2020b/software/JupyterLab/2.2.8-G
CCcore-10.2.0/share/jupyter/lab
[I 09:49:01.611 LabApp] Serving notebooks from local directory: /home/users/d/f/dfr
[I 09:49:01.611 LabApp] Jupyter Notebook 6.1.4 is running at:
[I 09:49:01.612 LabApp] http://10.252.2.55:8888/?token=c8515c45ec8066710aa9ba1c2d15b897a27514157780a3cf
[I 09:49:01.612 LabApp]  or http://127.0.0.1:8888/?token=c8515c45ec8066710aa9ba1c2d15b897a27514157780a3cf
[I 09:49:01.612 LabApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 09:49:01.617 LabApp] No web browser found: could not locate runnable browser.
[C 09:49:01.618 LabApp]

    To access the notebook, open this file in a browser:
        file:///home/users/d/f/dfr/.local/share/jupyter/runtime/nbserver-2312329-open.html
    Or copy and paste one of these URLs:
        http://10.252.2.55:8888/?token=c8515c45ec8066710aa9ba1c2d15b897a27514157780a3cf
     or http://127.0.0.1:8888/?token=c8515c45ec8066710aa9ba1c2d15b897a27514157780a3cf
[I 09:49:09.626 LabApp] 302 GET /?token=c8515c45ec8066710aa9ba1c2d15b897a27514157780a3cf (10.252.1.2) 0.46ms
[W 09:49:11.567 LabApp] Could not determine jupyterlab build status without nodejs
```

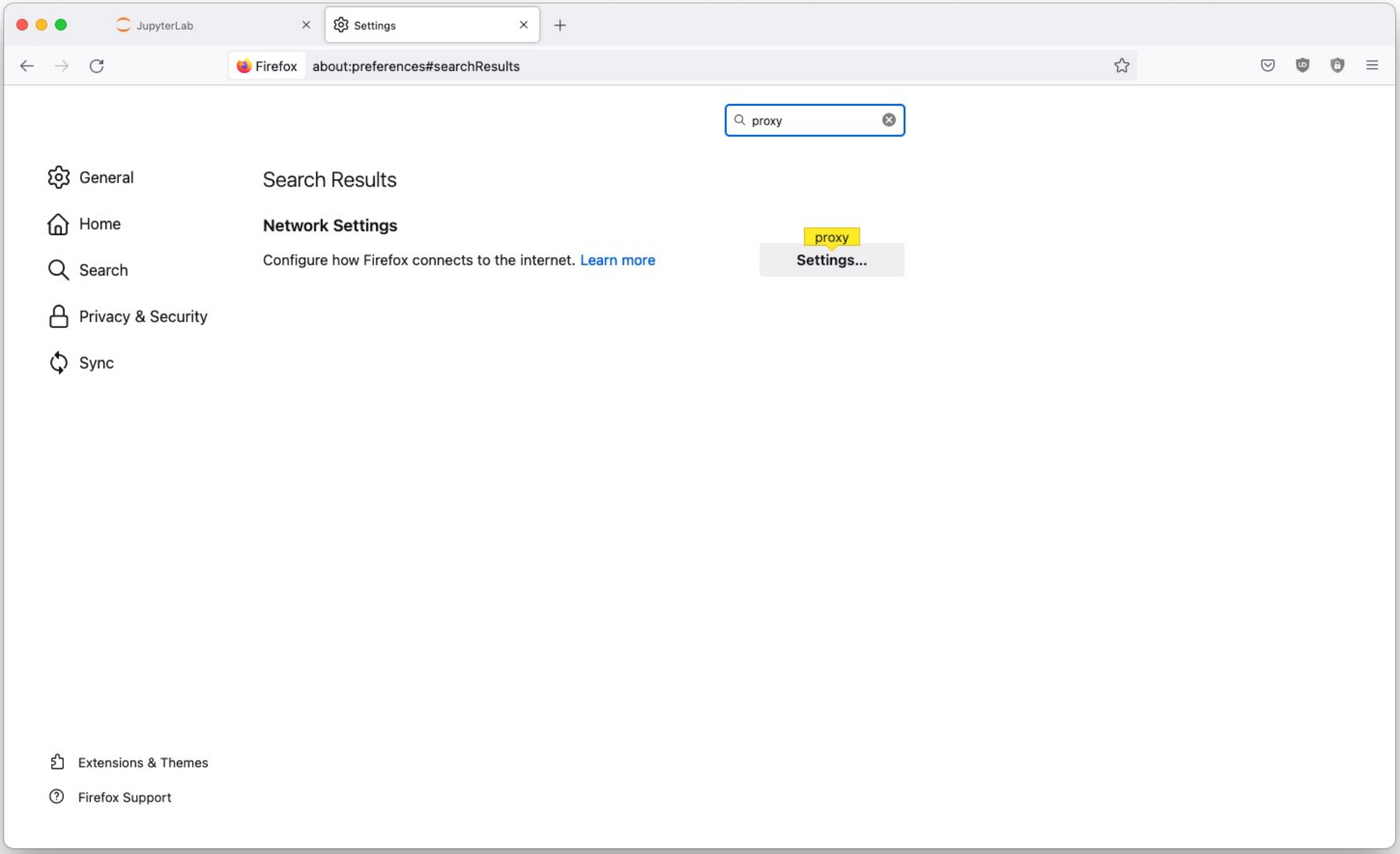
Use the `--ip` option to get the right URL

3. Create SSH tunnel (SOCK proxy)

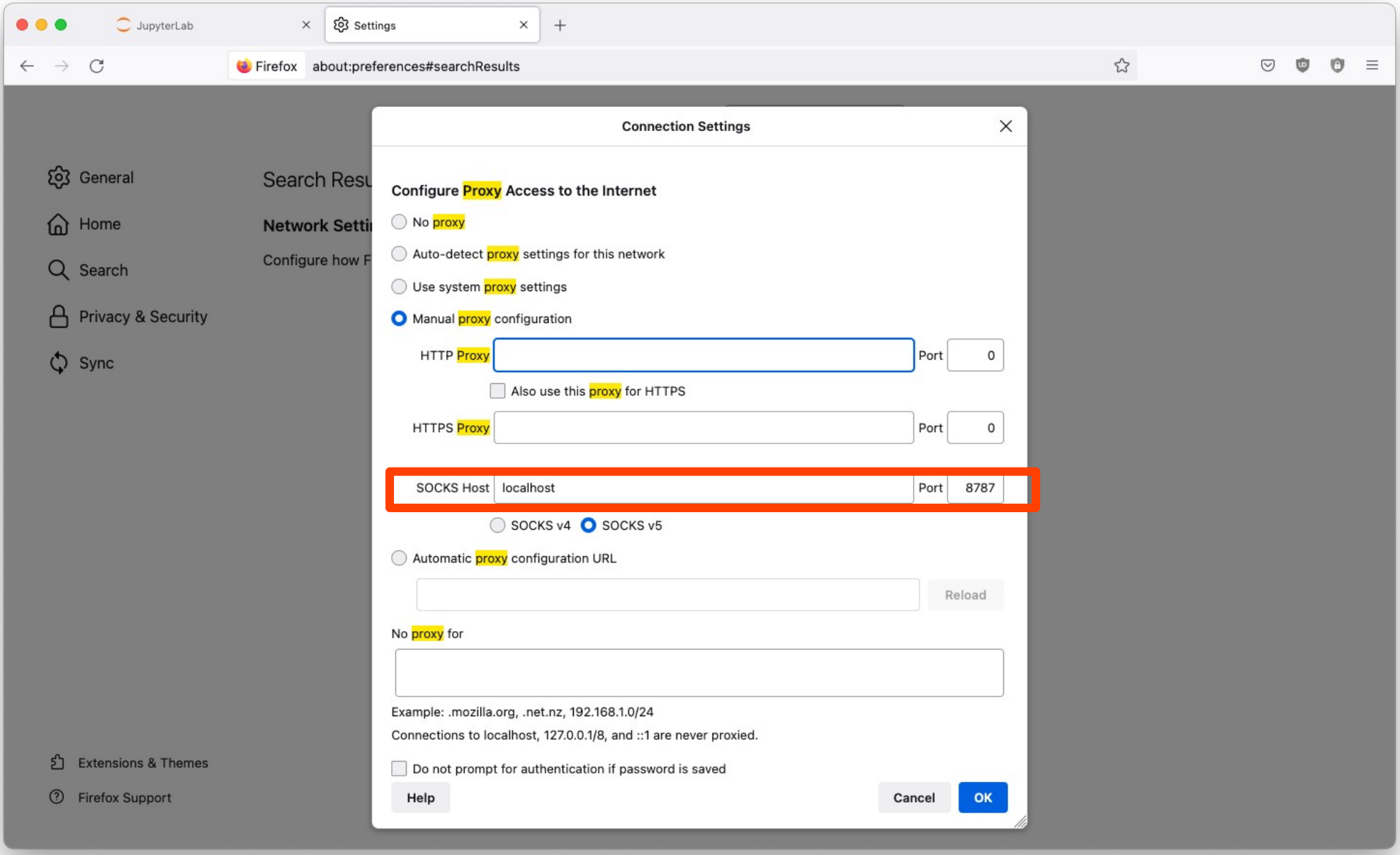


Run `ssh -D` in a new terminal and leave it open

4. Configure browser

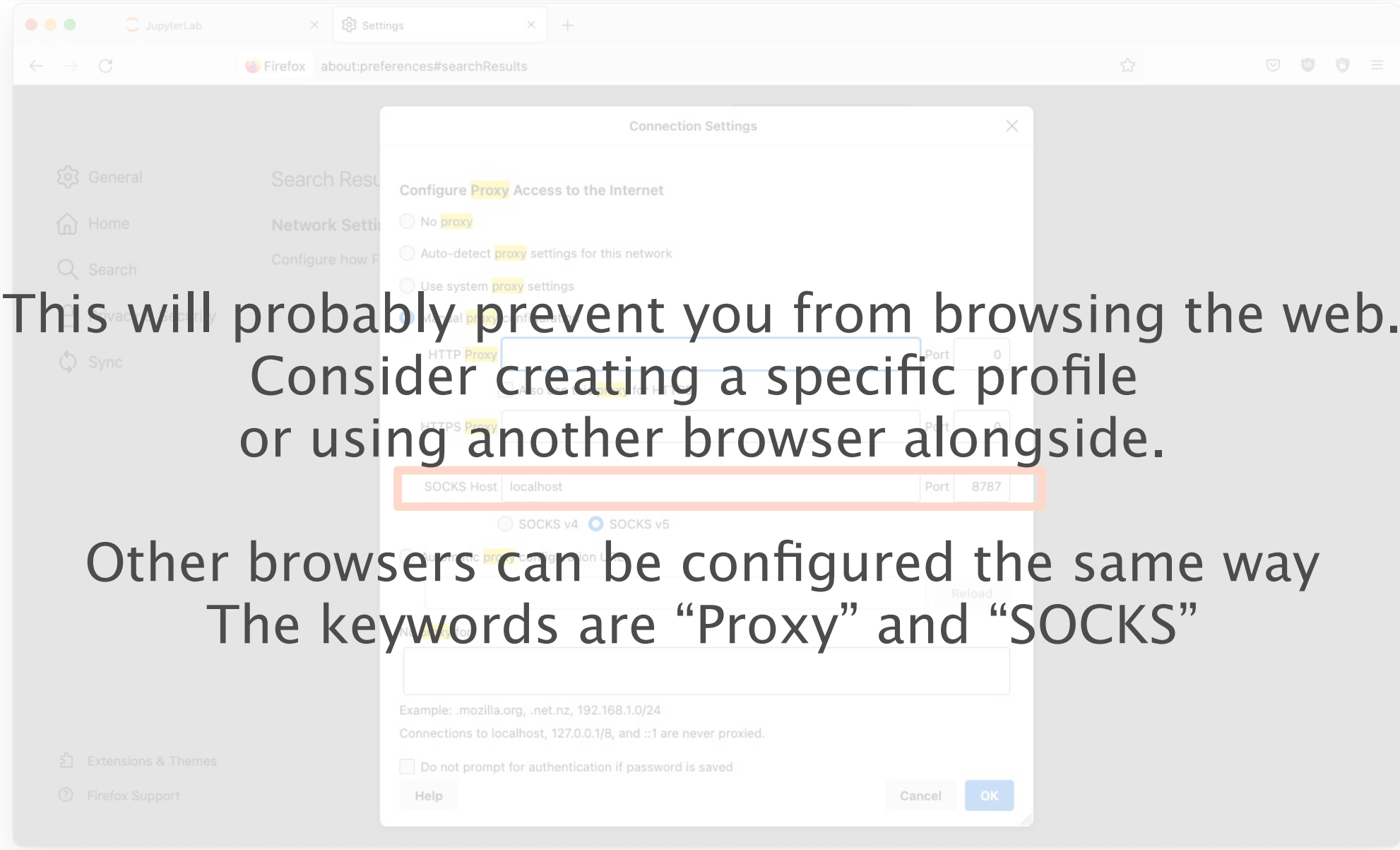


4. Configure browser



Setup same port you chose in Step 3.

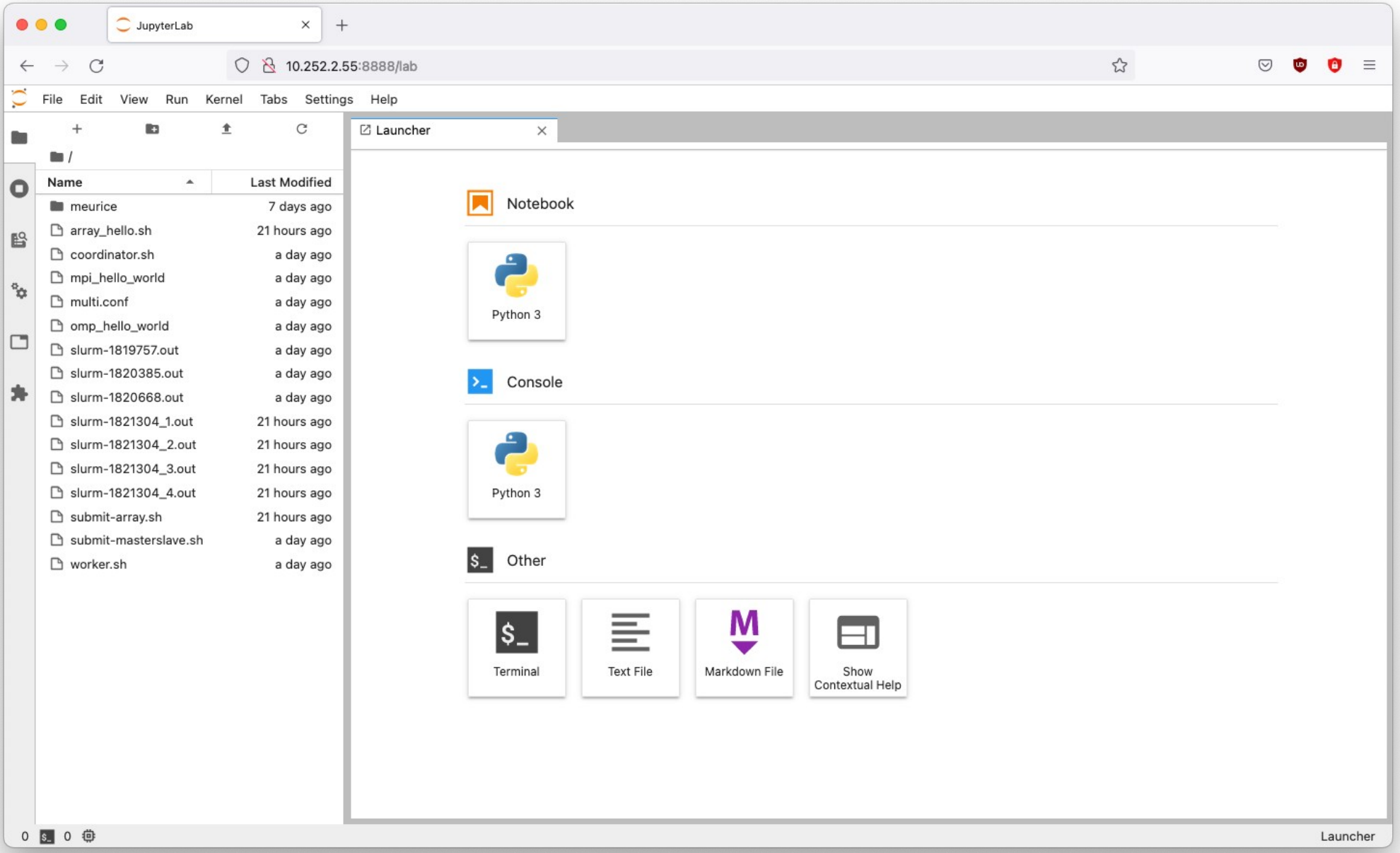
4. Configure browser



This will probably prevent you from browsing the web.
Consider creating a specific profile
or using another browser alongside.

Other browsers can be configured the same way
The keywords are “Proxy” and “SOCKS”

5. Connect to URL



Paste URL you got in Step 2 in address bar

0. Install helper script

```
[dfr@lemaitre3 ~]$ wget https://raw.githubusercontent.com/nickjer/singularity-rstudio/master/rstudio_auth.sh
--2021-11-04 10:16:08--  https://raw.githubusercontent.com/nickjer/singularity-rstudio/master/rstudio_auth.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.111.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 569 [text/plain]
Saving to: 'rstudio_auth.sh.1'

100%[==========================================================================================================================================>] 569         --.-K/s   in 0s

2021-11-04 10:16:08 (16.7 MB/s) - 'rstudio_auth.sh.1' saved [569/569]

[dfr@lemaitre3 ~]$
```

https://raw.githubusercontent.com/nickjer/singularity-rstudio/master/rstudio_auth.sh

```
chmod +x rstudio_auth.sh
```

How to start an Rstudio instance >

1. Use `srun` for shell on compute node

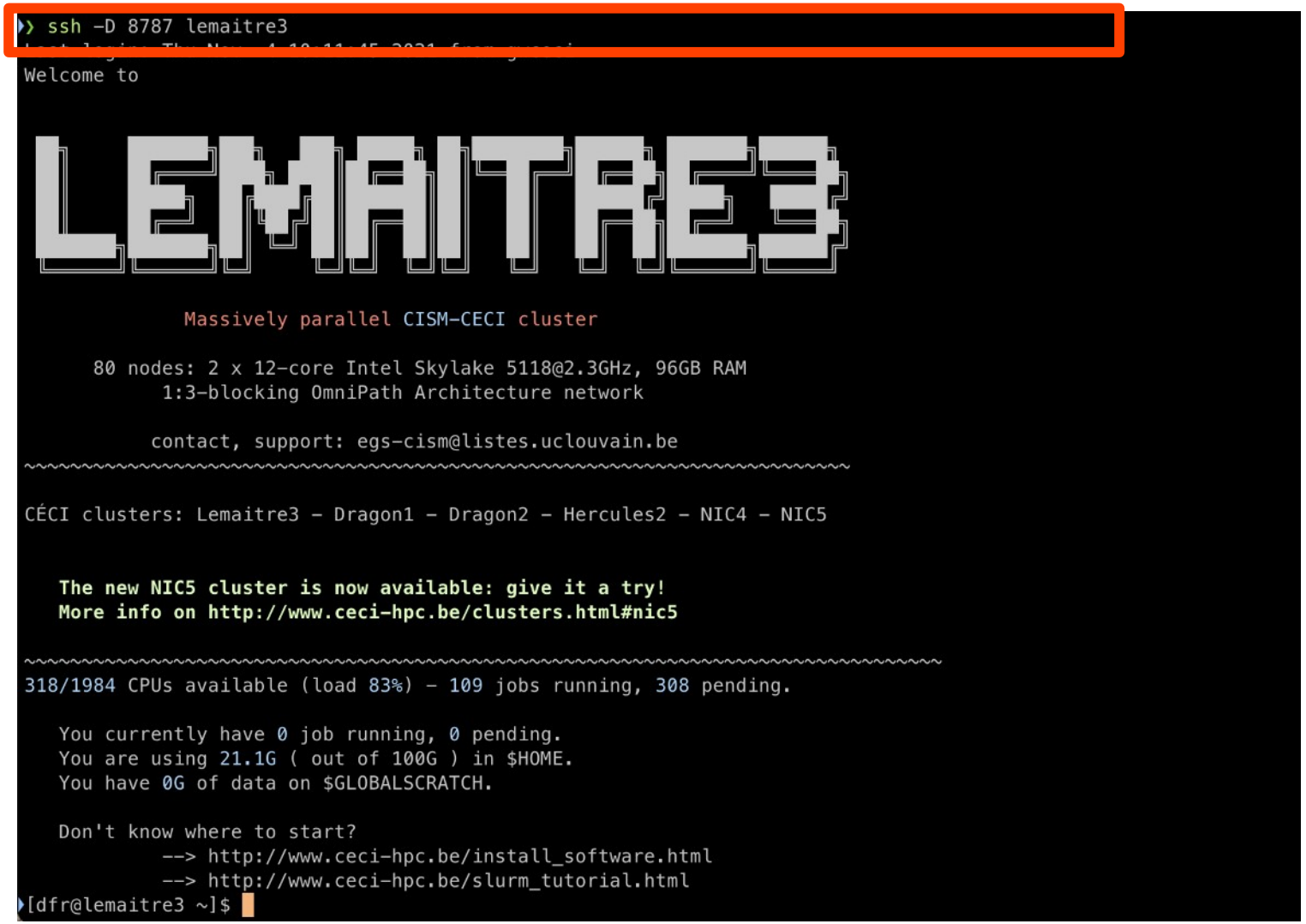
```
[dfr@lemaitre3 ~]$ srun --partition debug --pty bash -l
srun: job 70299666 queued and waiting for resources
srun: job 70299666 has been allocated resources
[dfr@lemaitre3 ~]$  ml releases/2019b RStudio-Server/1.2.5042-foss-2019b-Java-11
[dfr@lemaitre3 ~]$ export IP=$(hostname -i)
[dfr@lemaitre3 ~]$ export PORT=8787
[dfr@lemaitre3 ~]$ export RSTUDIO_PASSWORD="kmGaLbPLOE/uulb2"
[dfr@lemaitre3 ~]$ echo "http://$IP:$PORT"
http://10.7.1.94:8787
[dfr@lemaitre3 ~]$ rserver --server-daemonize=0 --www-port $PORT --rsession-which-r=$(which R) --auth-none 0  --auth-pam-helper $PWD/rstudio_auth.sh
```

2. Load module and start service

```
[dfr@lemaitre3 ~]$ srun --partition debug --pty bash -l
srun: job 70299666 queued and waiting for resources
srun: job 70299666 has been allocated resources
[dfr@lemaitre3 ~]$  ml releases/2019b RStudio-Server/1.2.5042-foss-2019b-Java-11
[dfr@lemaitre3 ~]$ export IP=$(hostname -i)
[dfr@lemaitre3 ~]$ export PORT=8787
[dfr@lemaitre3 ~]$ export RSTUDIO_PASSWORD="kmGaLbPLOE/uulb2"
[dfr@lemaitre3 ~]$ echo "http://$IP:$PORT"
http://10.7.1.94:8787
[dfr@lemaitre3 ~]$ rserver --server-daemonize=0 --www-port $PORT --rsession-which-r=$(which R) --auth-none 0  --auth-pam-helper $PWD/rstudio_auth.sh
```

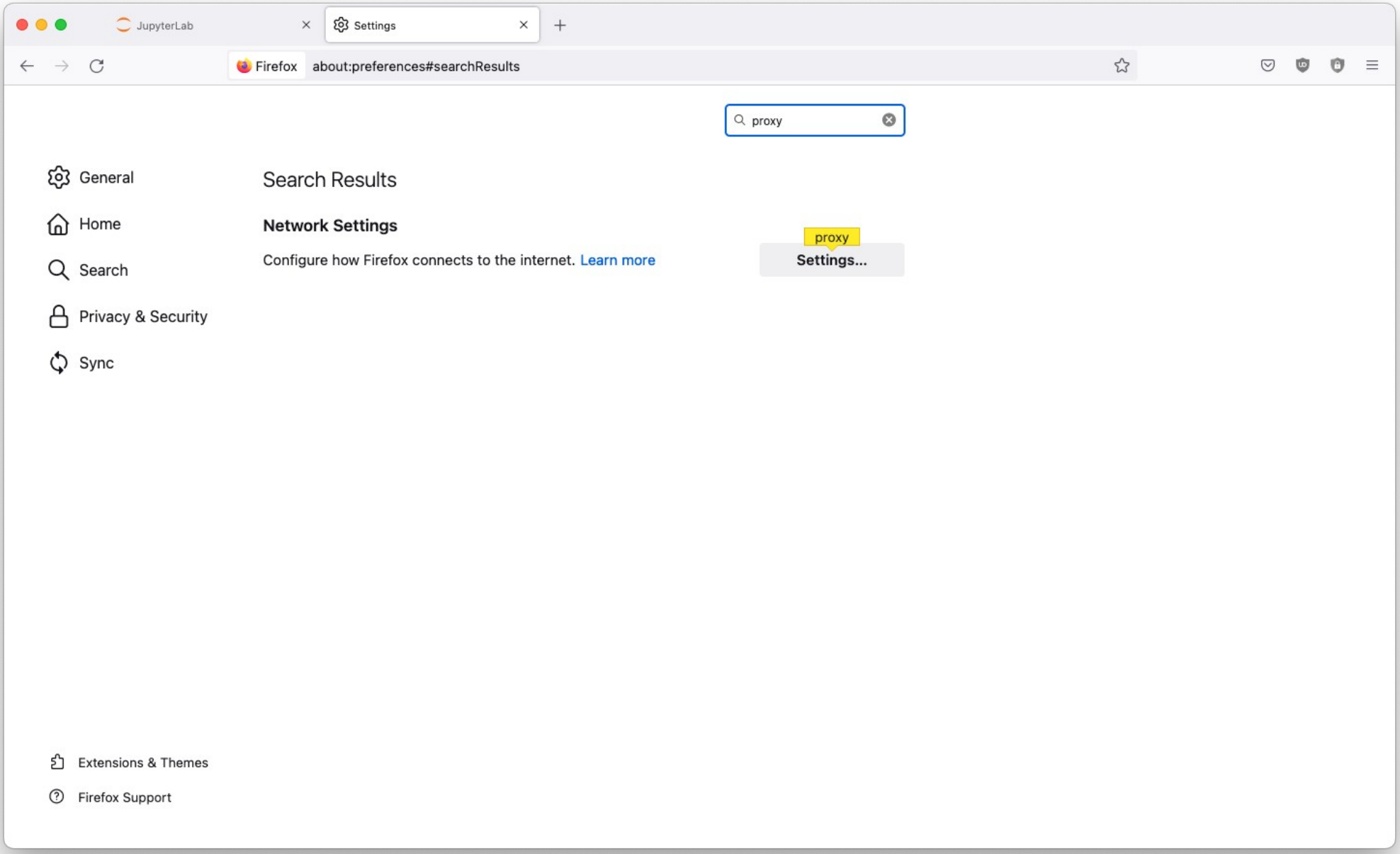
1. Run `hostname` to get the IP address
2. Choose a pasword and a port
3. Run the server

3. Create SSH tunnel (SOCK proxy)

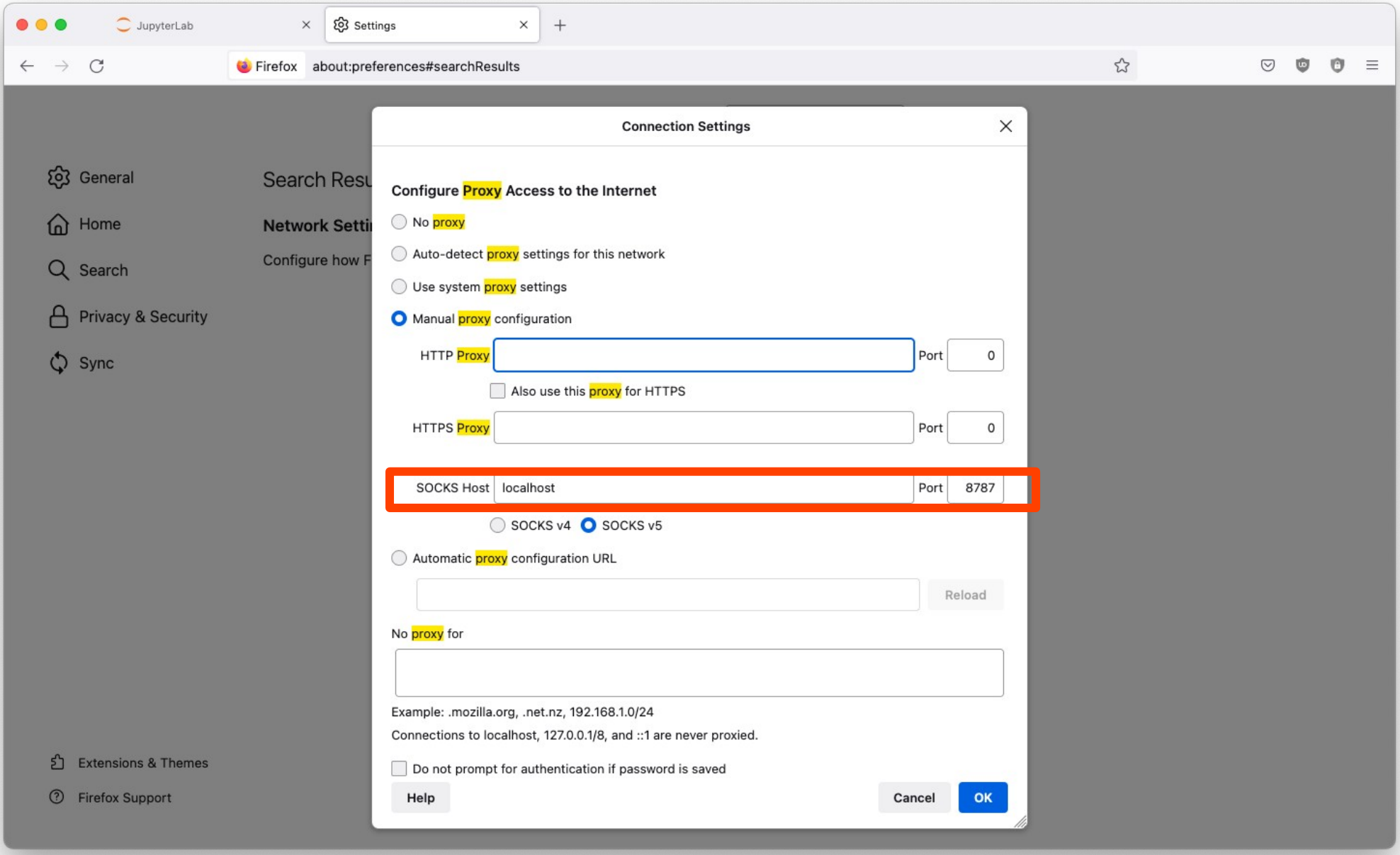


Run `ssh -D` in a new terminal and leave it open

4. Configure browser

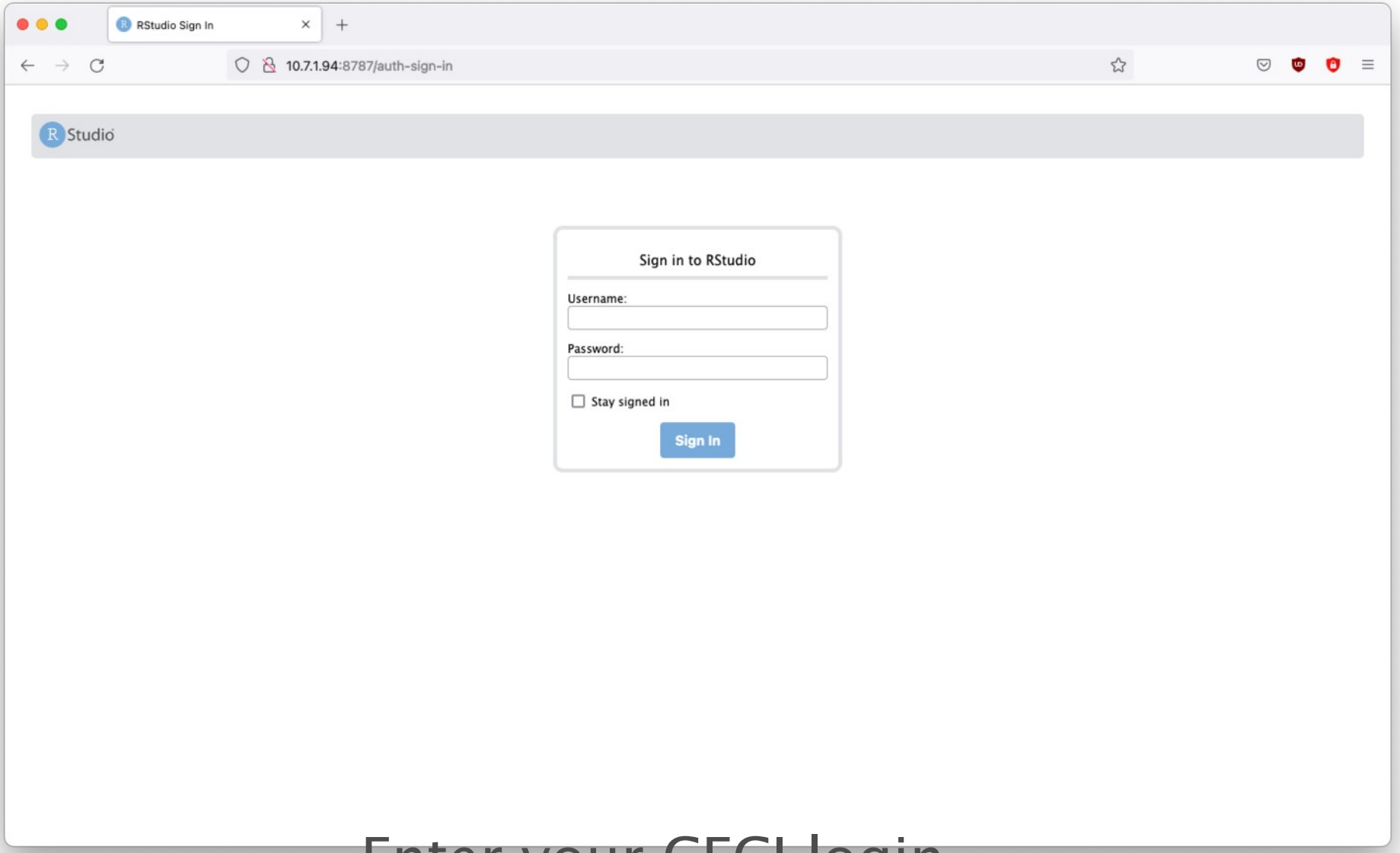


4. Configure browser



Setup same port you chose in Step 3.

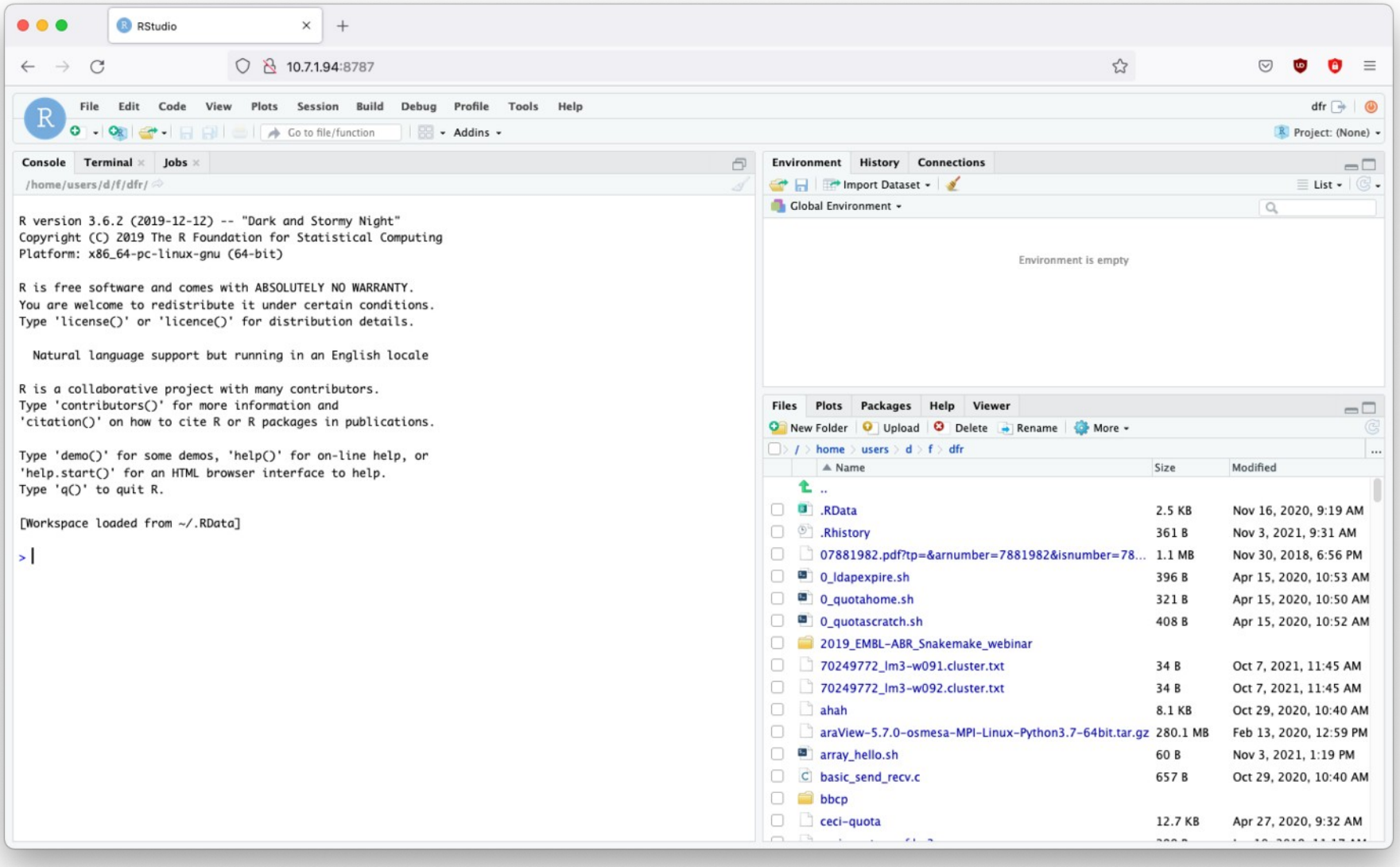
5. Connect to URL built at Step 2



Enter your CECI login
and the password you chose in Step 2.

How to start an Rstudio instance >

5. Connect to URL

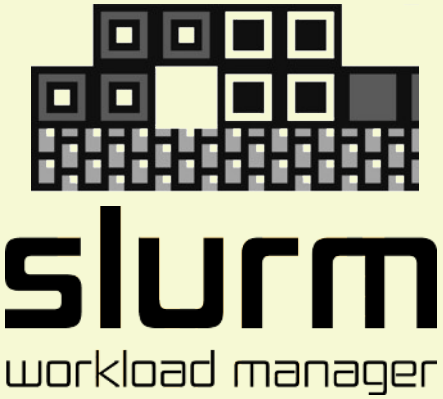


Get an interactive job!

1. Connect to a cluster
2. Start a web service
3. Connect to it

Final words...

before you go...



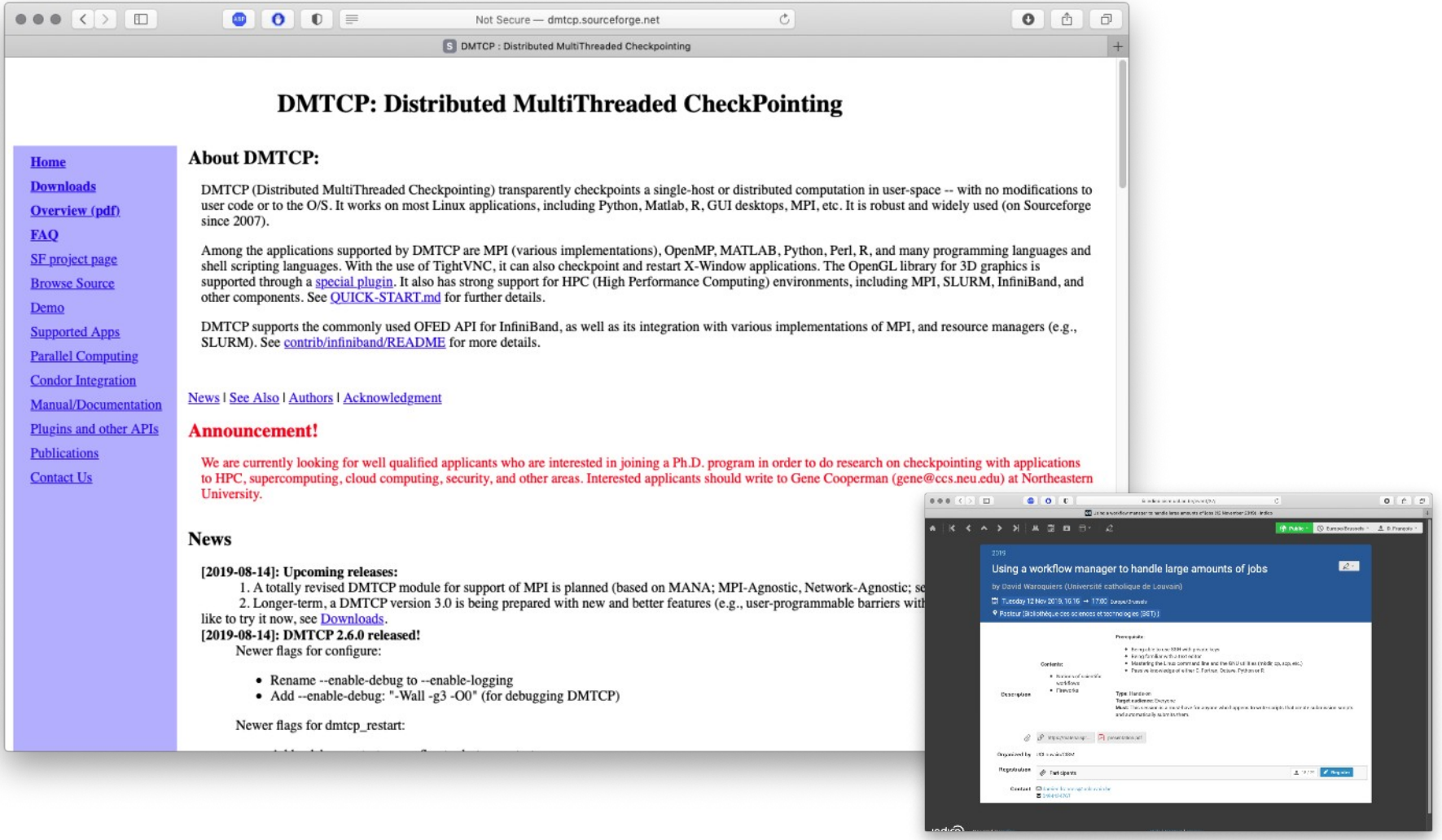
Good practice ; some advice...

- Choose the cluster wisely
- Understand the levels of parallelism required by your job
- Understand the I/O patterns of your jobs ; choose storage
- Do not compute on the login node
- Do not leave interactive sessions idle
- Tests arrays with 2 tasks before running the full array
- Double check the email options
- Do not waste resources ; split job if necessary
- Do not submit micro (<1 minute) jobs ; pack jobs
- Do not run `squeue` every second
- Do not wait for the cluster load to decrease to submit jobs

Checkpointing

when your jobs are toooooo looooooong

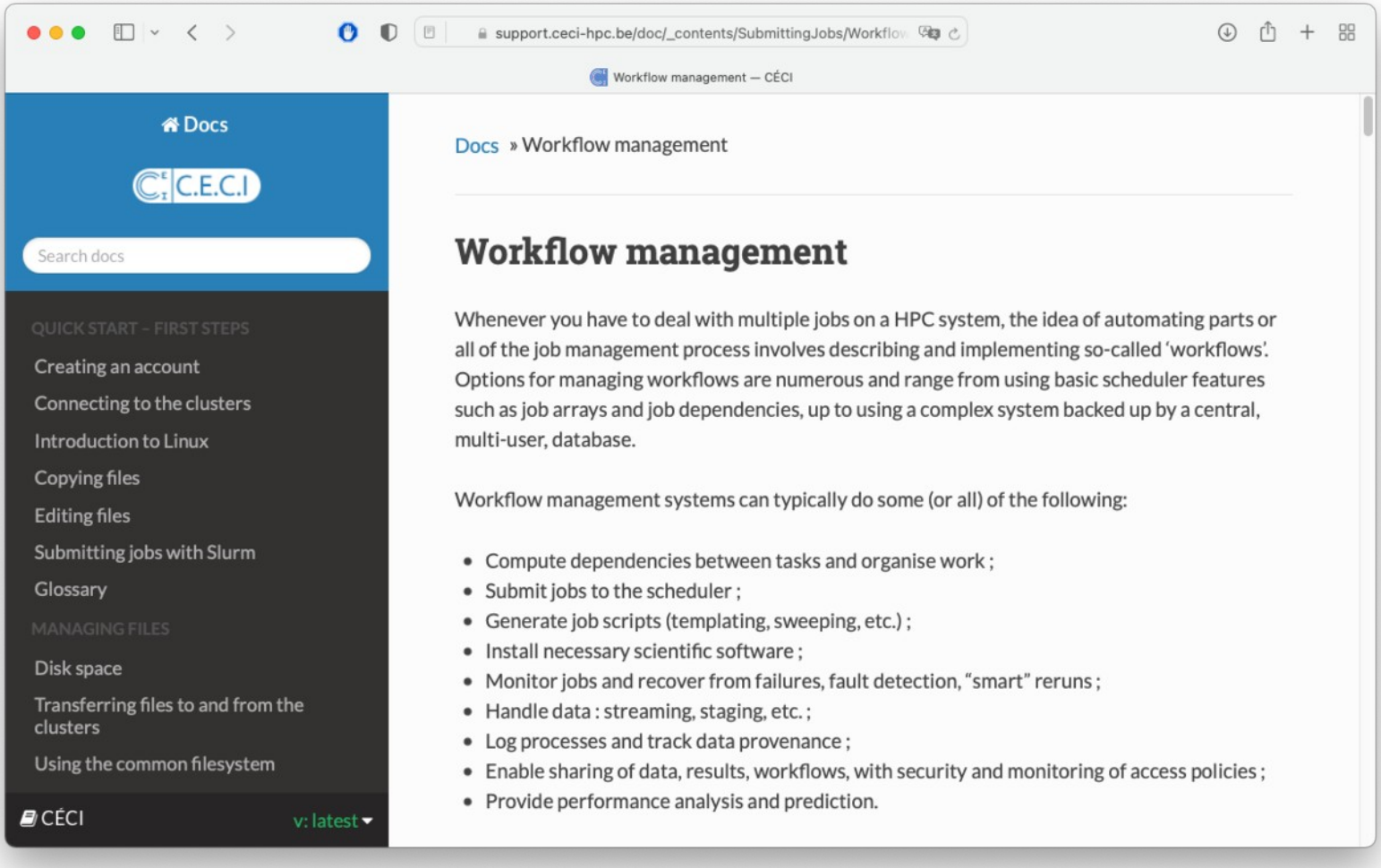
compared with the cluster maximum walltimes



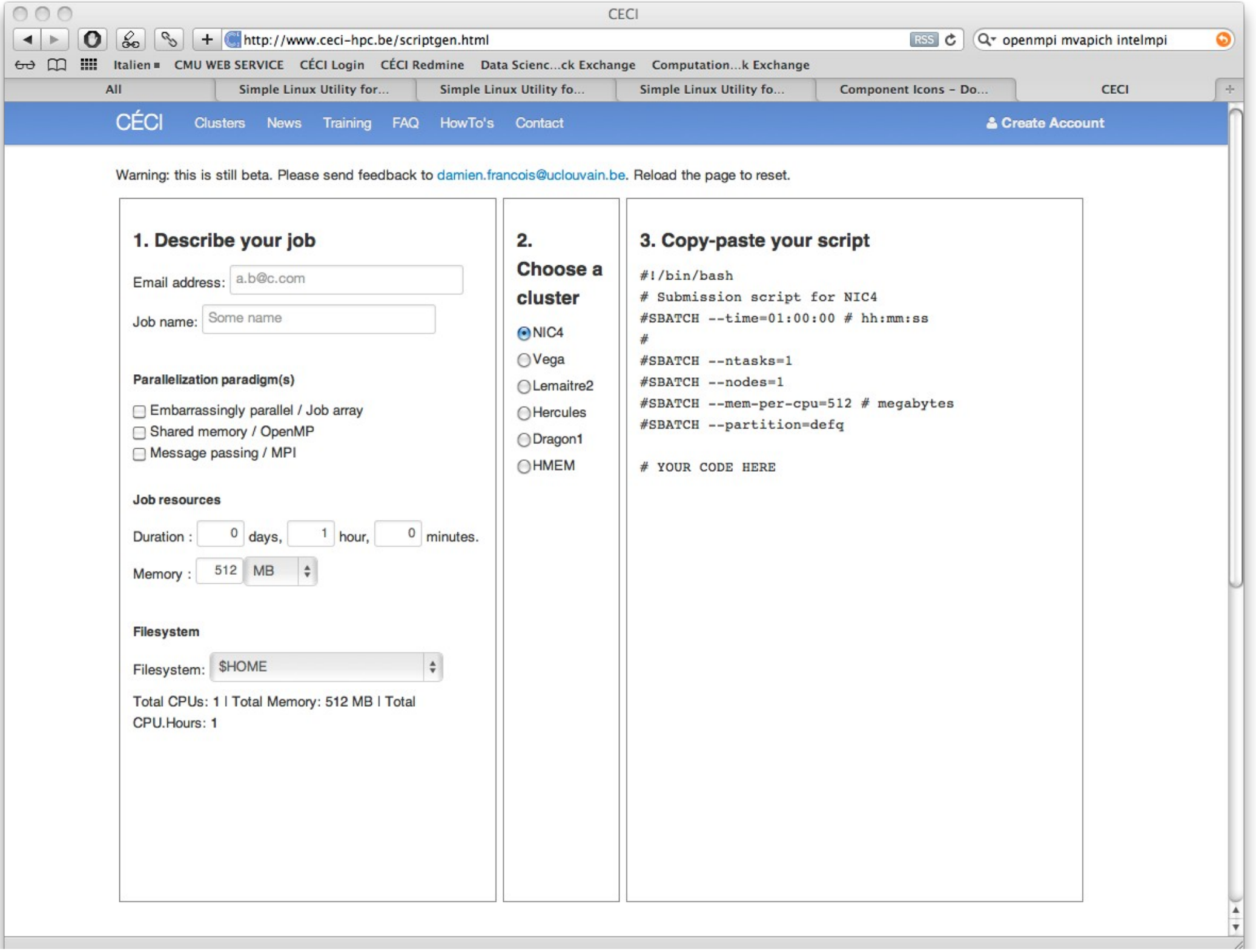
Worfklow management systems

when your job dependencies and parameters are too complex

to handle by hand



https://support.ceci-hpc.be/doc/_contents/SubmittingJobs/WorkflowManagement.html


CECI
http://www.ceci-hpc.be/scriptgen.html
openmpi mvapich intelmpi
Italien
CMU WEB SERVICE
CÉCI Login
CÉCI Redmine
Data Scienc...ck Exchange
Computation...k Exchange
All
Simple Linux Utility for...
Simple Linux Utility fo...
Simple Linux Utility fo...
Component Icons – Do...
CECI
CÉCI
Clusters
News
Training
FAQ
HowTo's
Contact
Create Account
Warning: this is still beta. Please send feedback to damien.francois@uclouvain.be. Reload the page to reset.
1. Describe your job
Email address:
a.b@c.com
Job name:
Some name
Parallelization paradigm(s)
Embarrassingly parallel / Job array
Shared memory / OpenMP
Message passing / MPI
Job resources
Duration :
0
days,
1
hour,
0
minutes.
Memory :
512
MB
Filesystem
Filesystem:
$HOME
Total CPUs: 1 | Total Memory: 512 MB | Total CPU.Hours: 1
2. Choose a cluster
NIC4
Vega
Lemaitre2
Hercules
Dragon1
HMEM
3. Copy-paste your script
#!/bin/bash
Submission script for NIC4
#SBATCH --time=01:00:00 # hh:mm:ss
#
#SBATCH --ntasks=1
#SBATCH --nodes=1
#SBATCH --mem-per-cpu=512 # megabytes
#SBATCH --partition=defq
YOUR CODE HERE

Write the submission script for your use case

- Which program will you use?
  - What type of parallelism? Is the program able
    - to use GPUs?
    - to use multiple nodes?
    - to use multiple cores?
      - How many at the same time?
  - What module(s) to load?
- What data will the job consume or produce?
  - Where is the input data located?
  - Where will the output data be located?
    - How much disk does the job need?
    - How much memory does the job need?

Write the submission script for your use case

- For how long should the job run?
- What should the output file be named?
- Do you want email notifications?
- Do you want to refer to the job by some name rather than ID?
- Which cluster is the most appropriate?
- Which partition should you target?
- Are there specific hardware types you want to avoid?
- What are the limits in place?

Typical skeleton

```
 1 #!/bin/bash
 2
 3 # Requested resources
 4 #SBATCH --ntasks=
 5 #SBATCH --cpus-per-task=
 6 #SBATCH --mem-per-cpu=
 7 #SBATCH --time=
 8
 9 # GPUs?
10 #SBATCH --gres=gpu:
11
12 # Partition, QOS, Licence?
13 #SBATCH --partition=
14 #SBATCH --qos=
15 #SBATCH --licences=
16
17 # Job parameters
18 #SBATCH --output=
19 #SBATCH --mail-type=
20 #SBATCH --mail-user=
21 #SBATCH --job-name=
22
23
24 ### Setup the environment
25
26 module load ...
27 export ...
28
29 ### Prepare data
30
31 mkdir -p ...
32
33 ### Compute
34
35 srun ...
36
37 ### Cleanup results
38
39 cp -r ...
40 rm -r ...
```

- Resources
- Targets
- Parameters
- Environment
- Data in
- Compute
- Data out

Final words...

Write and submit submission scripts

Explore the clusters

Read the man pages of Slurm commands

Use the resources you request

Beware of limits

Build workflows

Submit jobs !