Tutorial on the Mercury Computing Cluster

September 21, 2022

Walter W. Zhang

Outline

I. Connecting to the server

II. Mercury server overview

III. Running common programs

IV. Tips and tricks

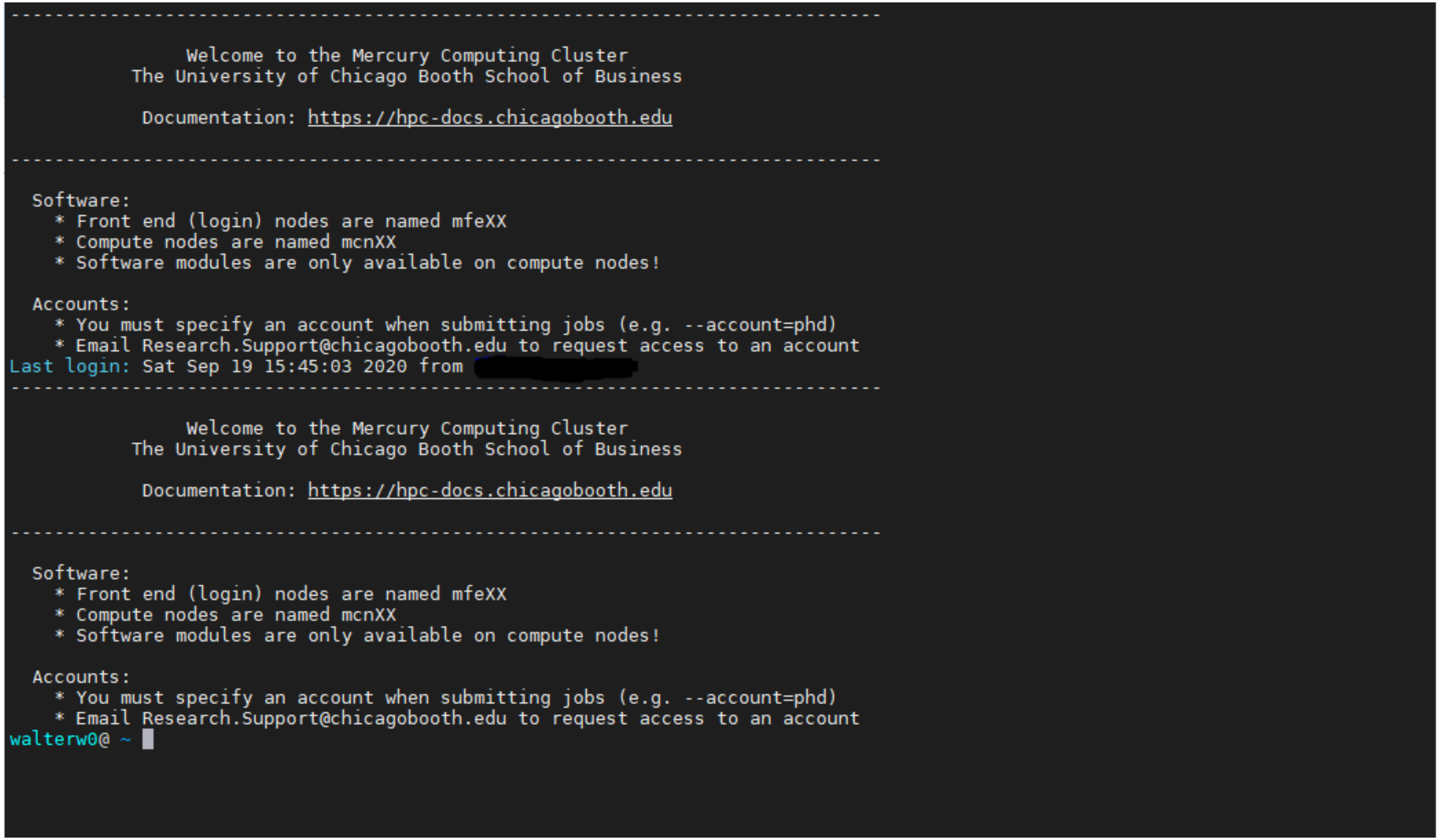
V. Guided lab example

Connecting to the server

- With you Chicago Booth ID and password you can connect to the server from the terminal
  - This gets you on a *login node*

```
ssh -YC <Booth-ID>@mercury.chicagobooth.edu
```

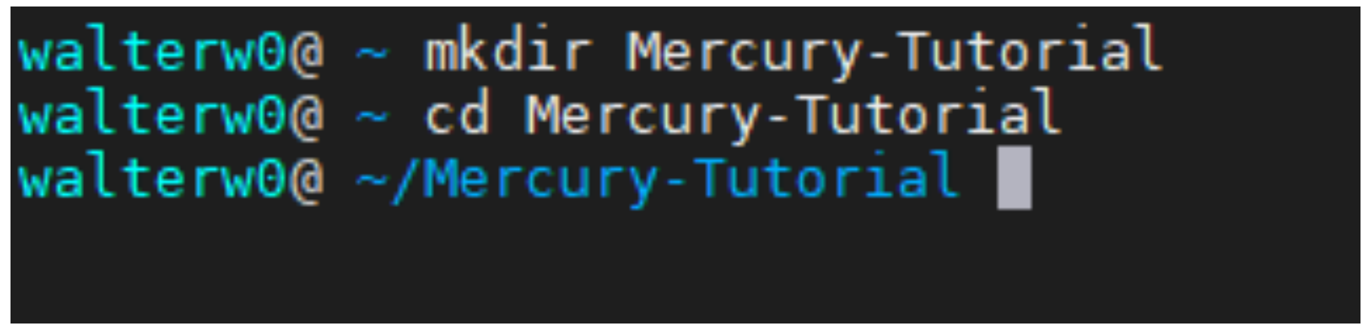
- Alternatively, try rstudio-research.chicagobooth.edu or jupyter.chicagobooth.edu
  - Rstudio server
  - Jupyter Hub (IPython Notebooks)
  - Limited computation power



```
---------------------------------------------------------------------------

               Welcome to the Mercury Computing Cluster
          The University of Chicago Booth School of Business

            Documentation: https://hpc-docs.chicagobooth.edu

---------------------------------------------------------------------------

  Software:
    * Front end (login) nodes are named mfeXX
    * Compute nodes are named mcnXX
    * Software modules are only available on compute nodes!

  Accounts:
    * You must specify an account when submitting jobs (e.g. --account=phd)
    * Email Research.Support@chicagobooth.edu to request access to an account
Last login: Sat Sep 19 15:45:03 2020 from [redacted]
---------------------------------------------------------------------------

               Welcome to the Mercury Computing Cluster
          The University of Chicago Booth School of Business

            Documentation: https://hpc-docs.chicagobooth.edu

---------------------------------------------------------------------------

  Software:
    * Front end (login) nodes are named mfeXX
    * Compute nodes are named mcnXX
    * Software modules are only available on compute nodes!

  Accounts:
    * You must specify an account when submitting jobs (e.g. --account=phd)
    * Email Research.Support@chicagobooth.edu to request access to an account
walterw0@ ~
```

Connecting to the server

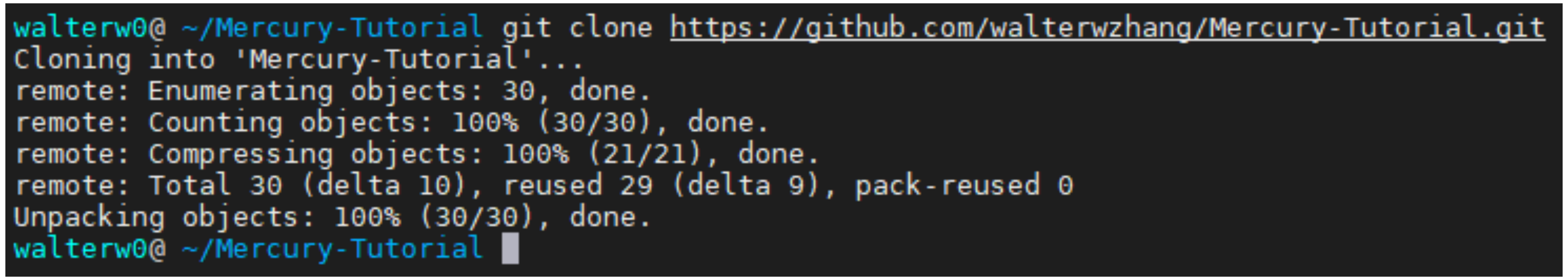
1. Create a folder and open it



```
walterw0@ ~ mkdir Mercury-Tutorial
walterw0@ ~ cd Mercury-Tutorial
walterw0@ ~/Mercury-Tutorial
```

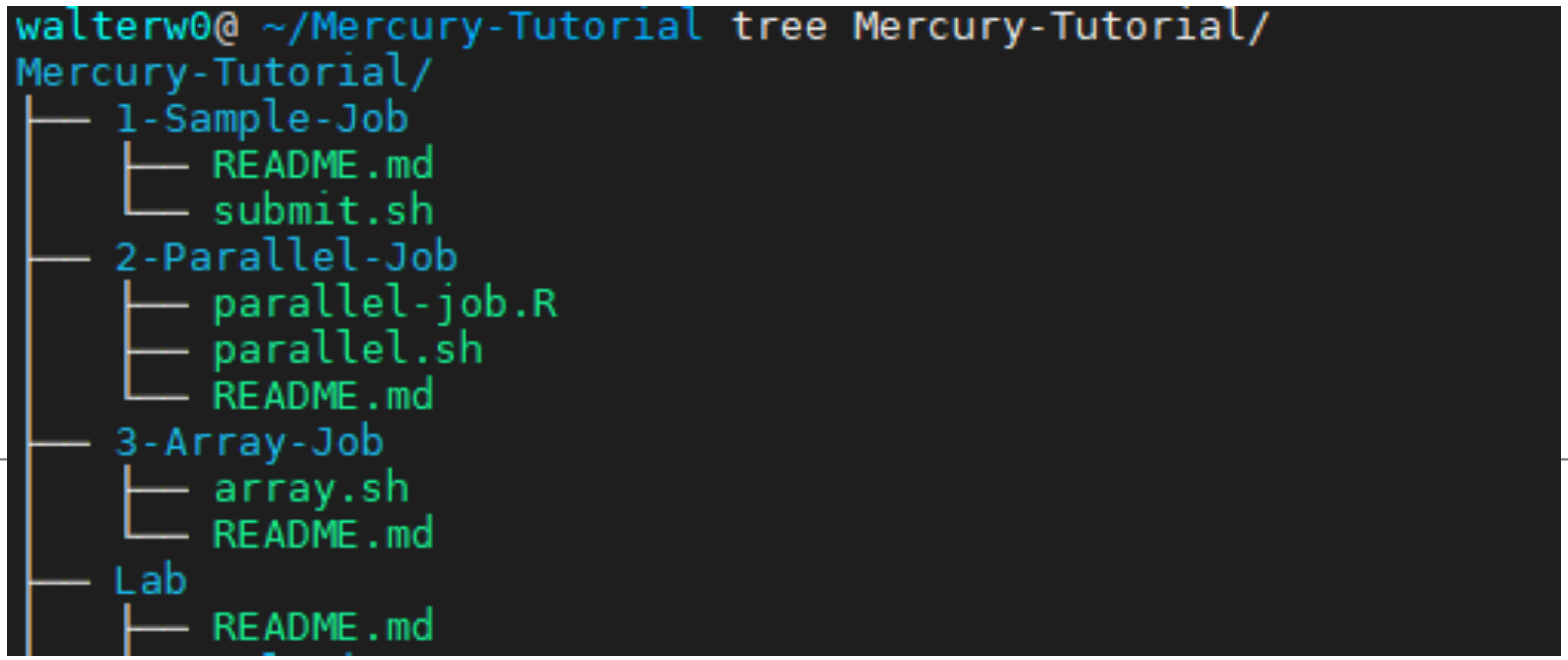
2. Clone the GitHub Repository

```
git clone https://github.com/walterwzhang/Mercury-Tutorial.git
```



```
walterw0@ ~/Mercury-Tutorial git clone https://github.com/walterwzhang/Mercury-Tutorial.git
Cloning into 'Mercury-Tutorial'...
remote: Enumerating objects: 30, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 30 (delta 10), reused 29 (delta 9), pack-reused 0
Unpacking objects: 100% (30/30), done.
walterw0@ ~/Mercury-Tutorial
```

3. See what's inside 👀



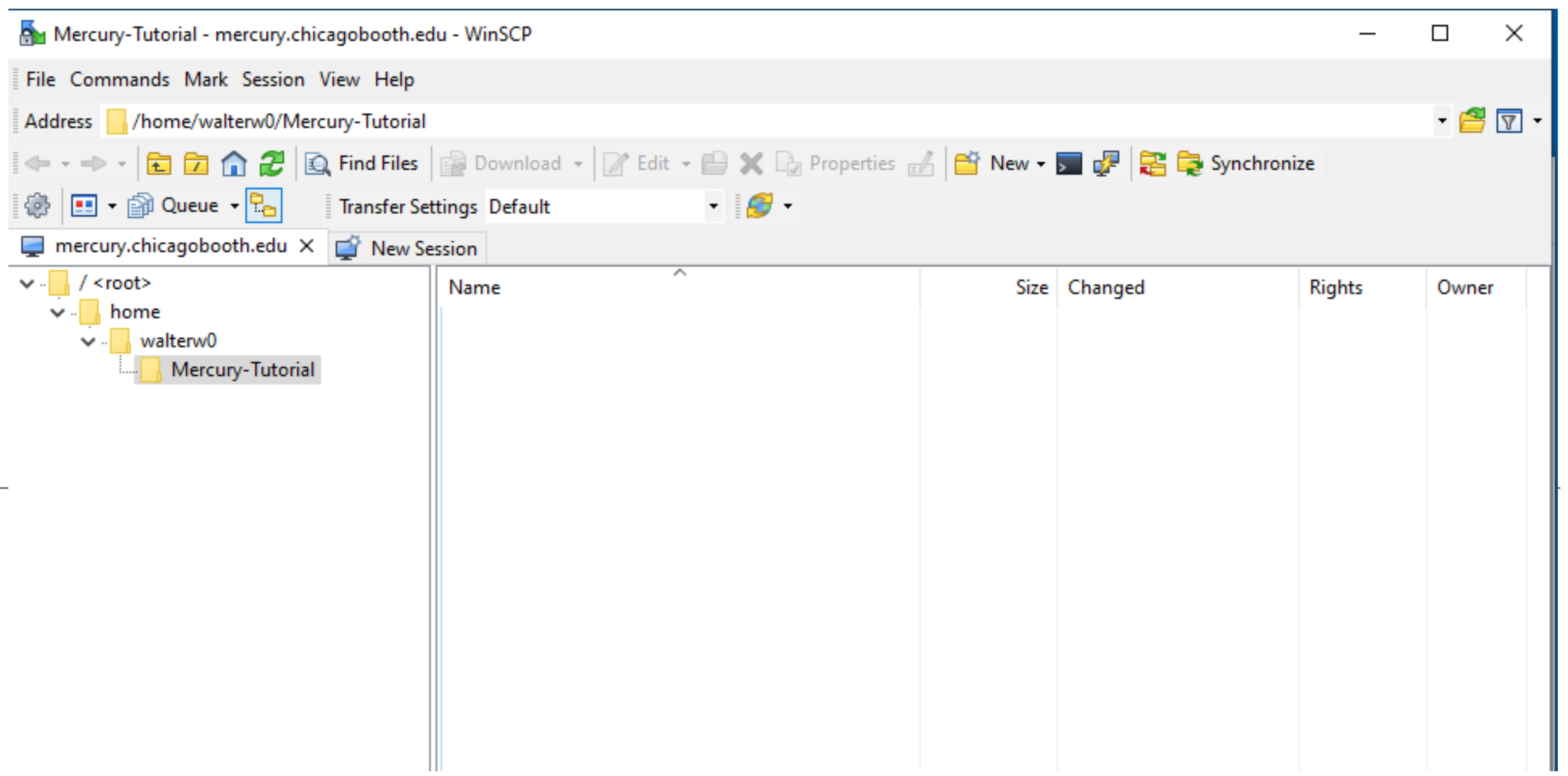
```
walterw0@ ~/Mercury-Tutorial tree Mercury-Tutorial/
Mercury-Tutorial/
├── 1-Sample-Job
│   ├── README.md
│   └── submit.sh
├── 2-Parallel-Job
│   ├── parallel-job.R
│   ├── parallel.sh
│   └── README.md
├── 3-Array-Job
│   ├── array.sh
│   └── README.md
├── Lab
│   ├── README.md
```

Connecting to the server

Use *FileZilla*, *Cyberduck*, or *WinSCP* to transfer files between the server and your local machine

These are SFTP clients that are often easier to work with than the command line for moving files around

- Usually these will be slower than the command line approach
- https://hpc-docs.chicagobooth.edu/accessing.html



Mercury-Tutorial - mercury.chicagobooth.edu - WinSCP

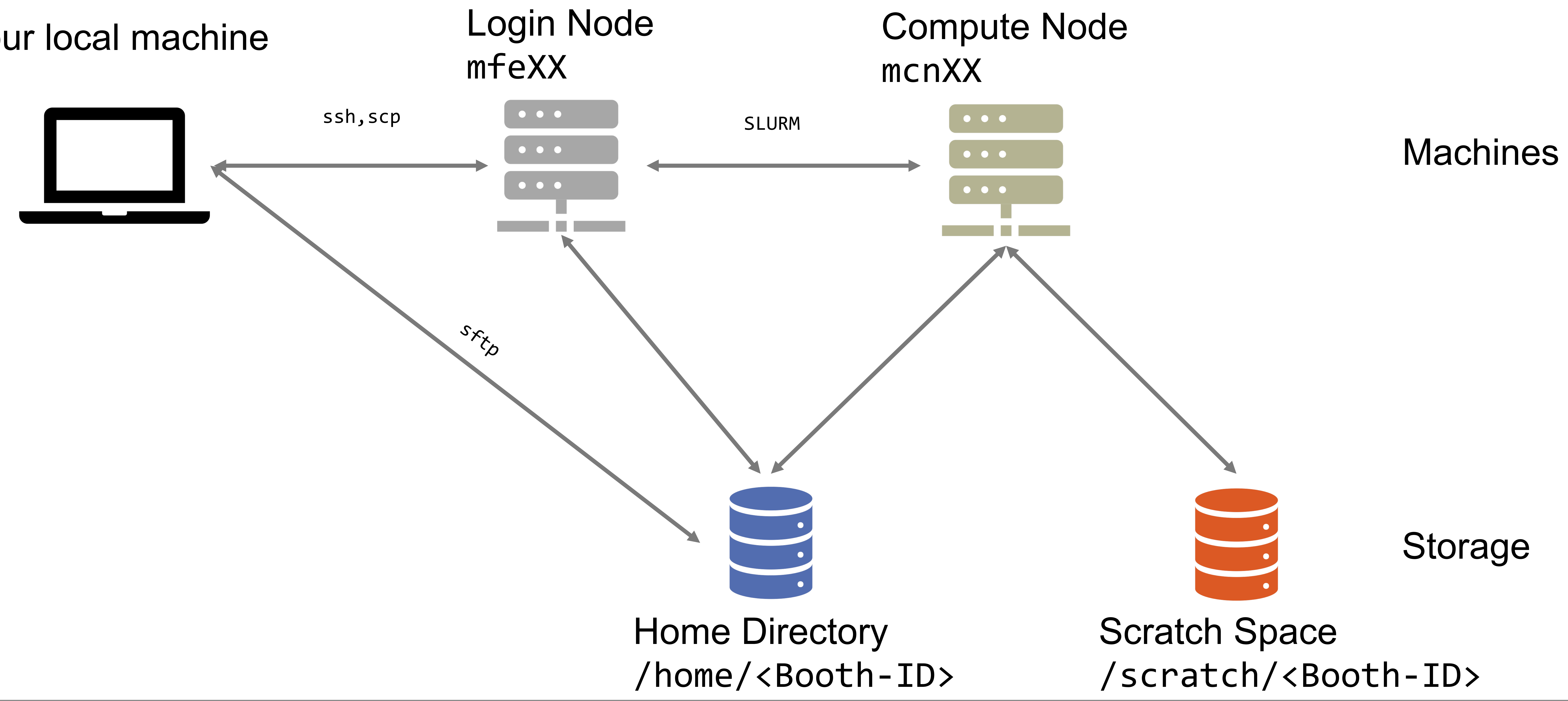
Address /home/walterw0/Mercury-Tutorial

Transfer Settings Default

mercury.chicagobooth.edu

/ <root>
home
walterw0
Mercury-Tutorial

Mercury Overview



CHICAGO BOOTH

Mercury Overview

Some Definitions:

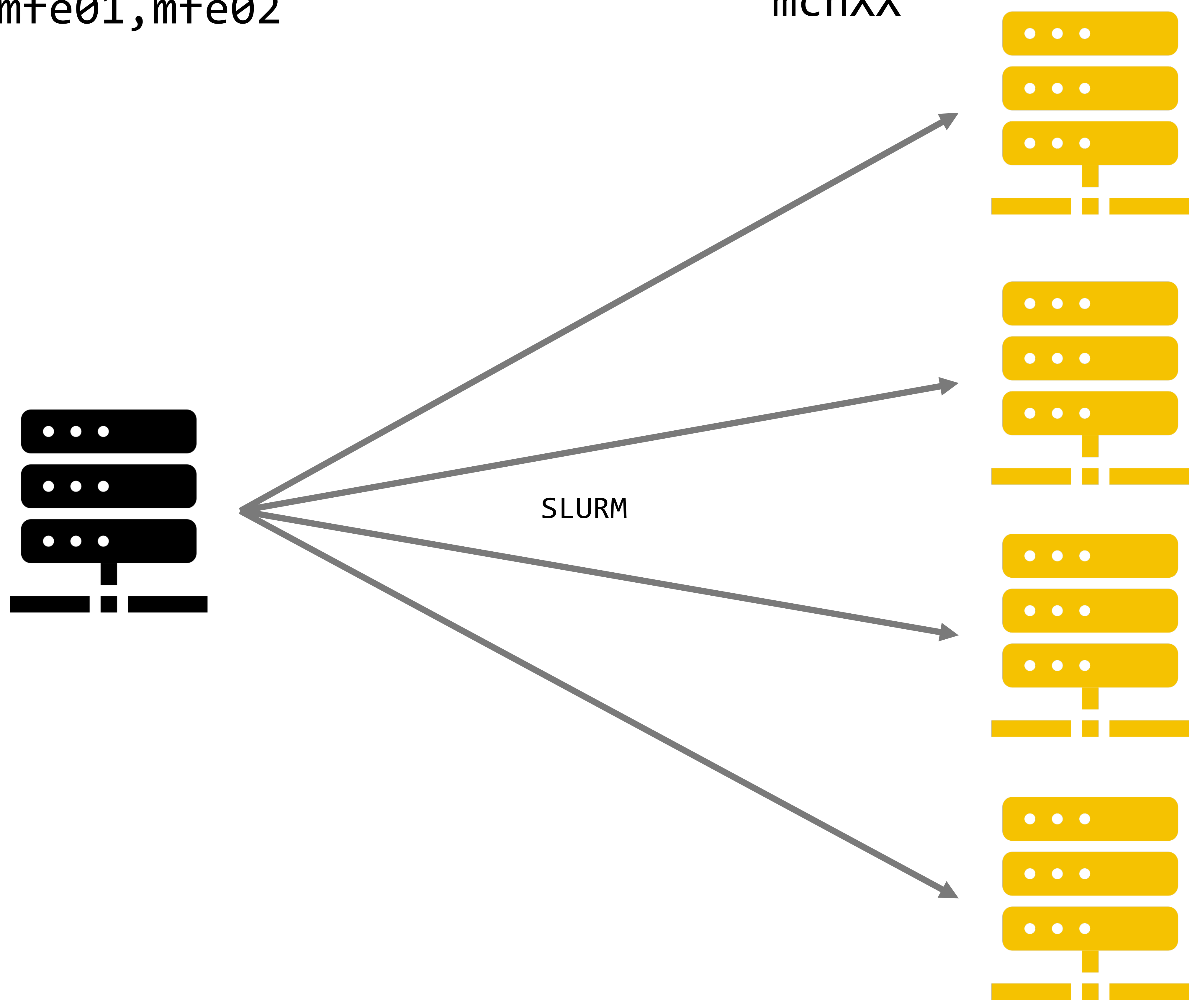
- *Processor/CPU*: Chip that responds to and processes instructions from the computer
- *Core*: Smallest compute unit that can run a program
- *Socket*: A packaged compute unit – can have many cores
- *Node*: Computer system that contains one or more sockets, memory, and storage units, and is connected to other nodes
- *GPU*: Graphics processing unit “repurposed” for matrix operations
- *Partition*: Type of node with different limits (e.g. `standard/long/GPU/highmem`)

Mercury Overview

Interactive Session: `srun --account=phd --pty bash -l`

Login Node
`mfe01,mfe02`

Compute Nodes
`mcnXX`



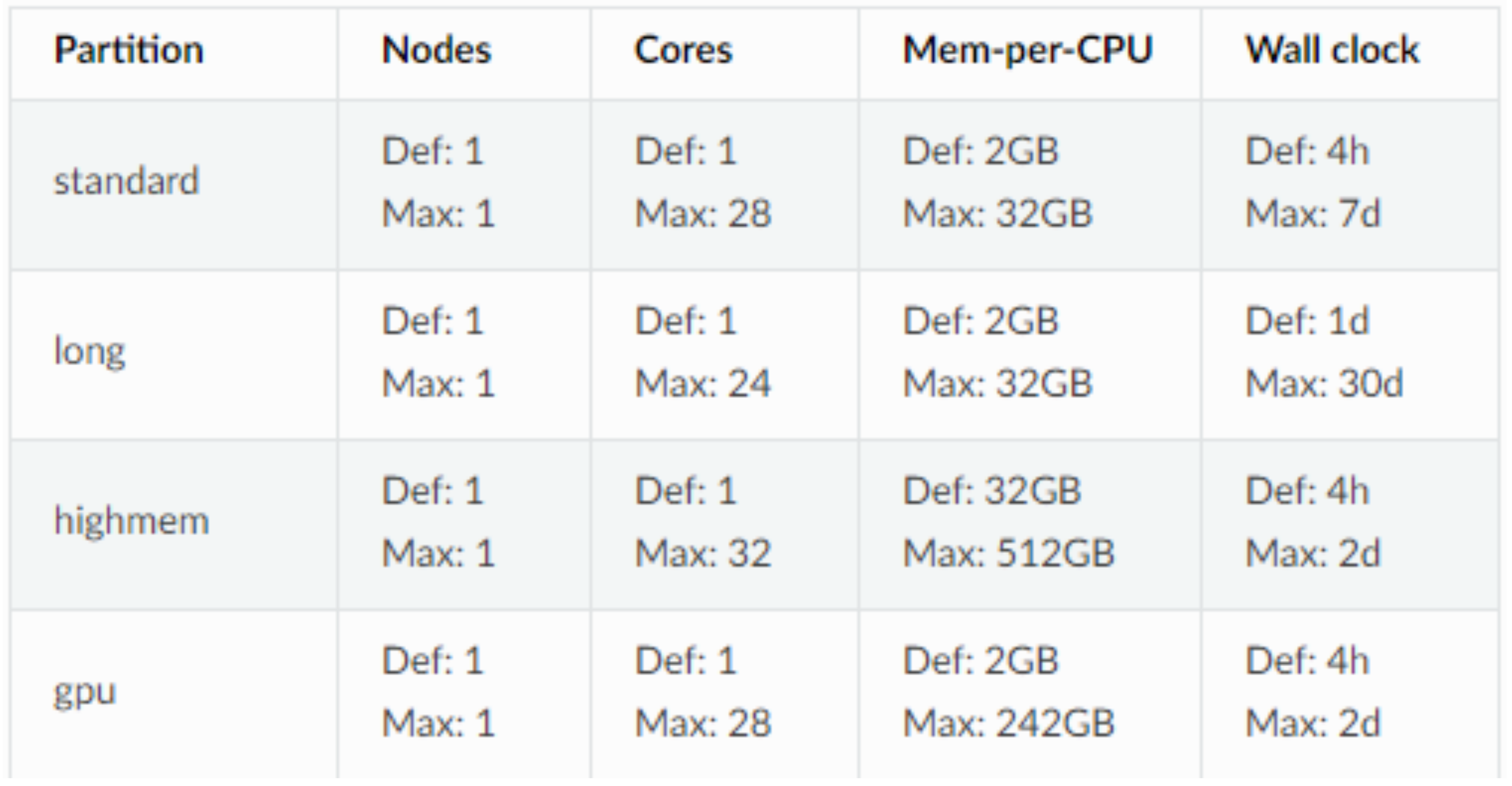
standard
mcn[01,25-27,31-33,52-57,59-61]

long
mcn[29,59-61,63]

highmem
mgpu[01,02]

gpu
mcn[58,62]
NVDIA K80 Cards

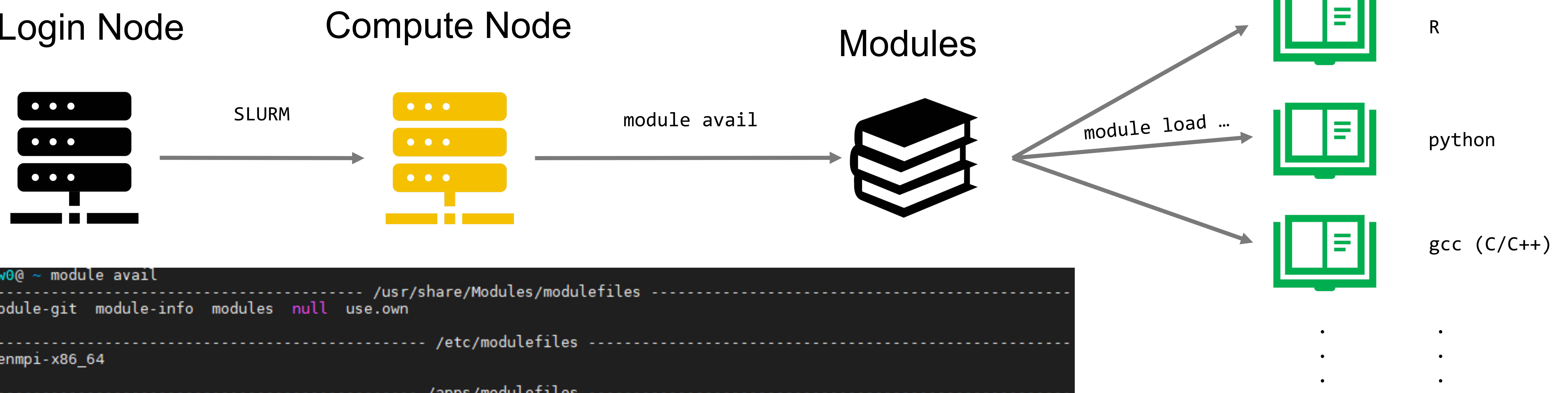
Usage Limits

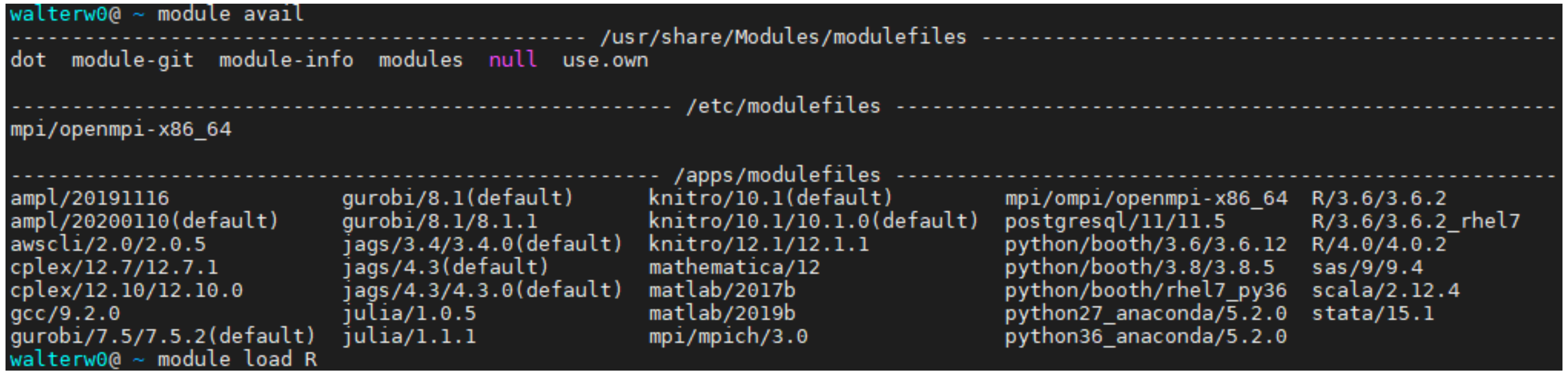


Partition	Nodes	Cores	Mem-per-CPU	Wall clock
standard	Def: 1 Max: 1	Def: 1 Max: 28	Def: 2GB Max: 32GB	Def: 4h Max: 7d
long	Def: 1 Max: 1	Def: 1 Max: 24	Def: 2GB Max: 32GB	Def: 1d Max: 30d
highmem	Def: 1 Max: 1	Def: 1 Max: 32	Def: 32GB Max: 512GB	Def: 4h Max: 2d
gpu	Def: 1 Max: 1	Def: 1 Max: 28	Def: 2GB Max: 242GB	Def: 4h Max: 2d

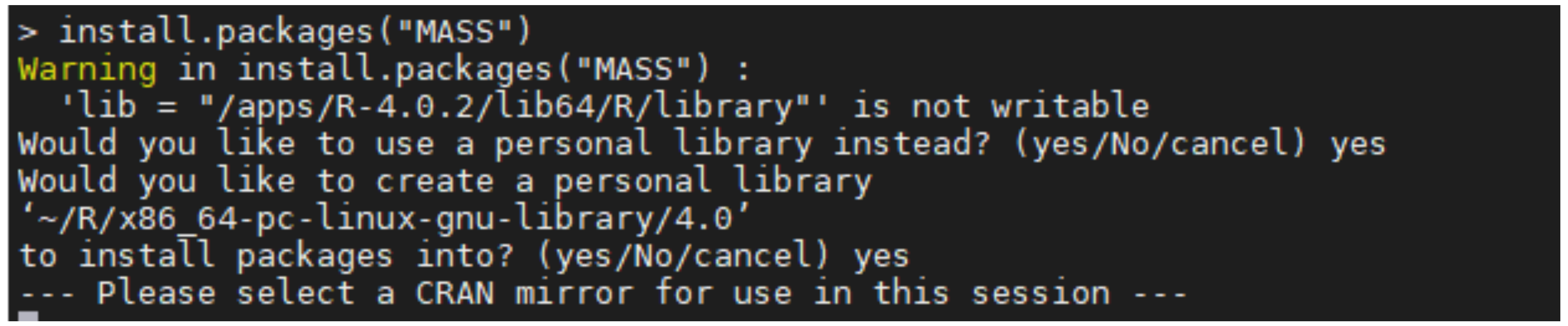
Mercury Overview

Interactive Session: `srun --account=phd --pty bash -l`





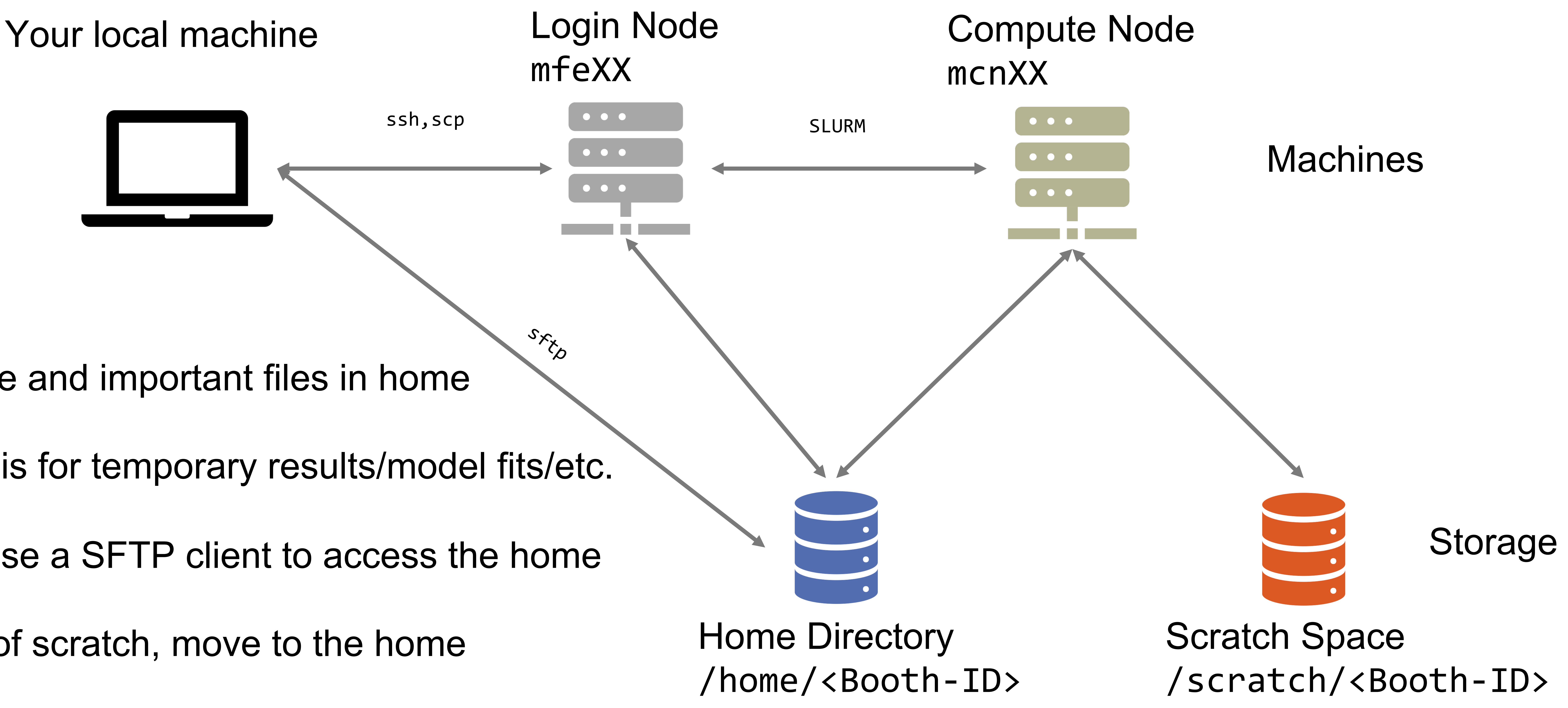
```
walterw0@ ~ module avail
---------------------------------------------- /usr/share/Modules/modulefiles -----------------------------------------------
dot  module-git  module-info  modules  null  use.own

---------------------------------------------------- /etc/modulefiles -------------------------------------------------------
mpi/openmpi-x86_64

--------------------------------------------------- /apps/modulefiles ------------------------------------------------------
ampl/20191116              gurobi/8.1(default)     knitro/10.1(default)         mpi/ompi/openmpi-x86_64  R/3.6/3.6.2
ampl/20200110(default)     gurobi/8.1/8.1.1        knitro/10.1/10.1.0(default)  postgresql/11/11.5       R/3.6/3.6.2_rhel7
awscli/2.0/2.0.5           jags/3.4/3.4.0(default) knitro/12.1/12.1.1           python/booth/3.6/3.6.12  R/4.0/4.0.2
cplex/12.7/12.7.1          jags/4.3(default)       mathematica/12               python/booth/3.8/3.8.5   sas/9/9.4
cplex/12.10/12.10.0        jags/4.3/4.3.0(default) matlab/2017b                 python/booth/rhel7_py36  scala/2.12.4
gcc/9.2.0                  julia/1.0.5             matlab/2019b                 python27_anaconda/5.2.0  stata/15.1
gurobi/7.5/7.5.2(default)  julia/1.1.1             mpi/mpich/3.0                python36_anaconda/5.2.0
walterw0@ ~ module load R
```



```
> install.packages("MASS")
Warning in install.packages("MASS") :
  'lib = "/apps/R-4.0.2/lib64/R/library"' is not writable
Would you like to use a personal library instead? (yes/No/cancel) yes
Would you like to create a personal library
'~/R/x86_64-pc-linux-gnu-library/4.0'
to install packages into? (yes/No/cancel) yes
--- Please select a CRAN mirror for use in this session ---
```

- Once we are on a compute node, we need to load the modules (or software) that we require
- If you need to install packages (e.g. for python or R), install them to a locally library (which is in your home directory)
- Remove modules with "`module unload …`" or "`module purge`"
- Type "`exit`" quit the compute node instance
- Check the machine name with "`hostname`"

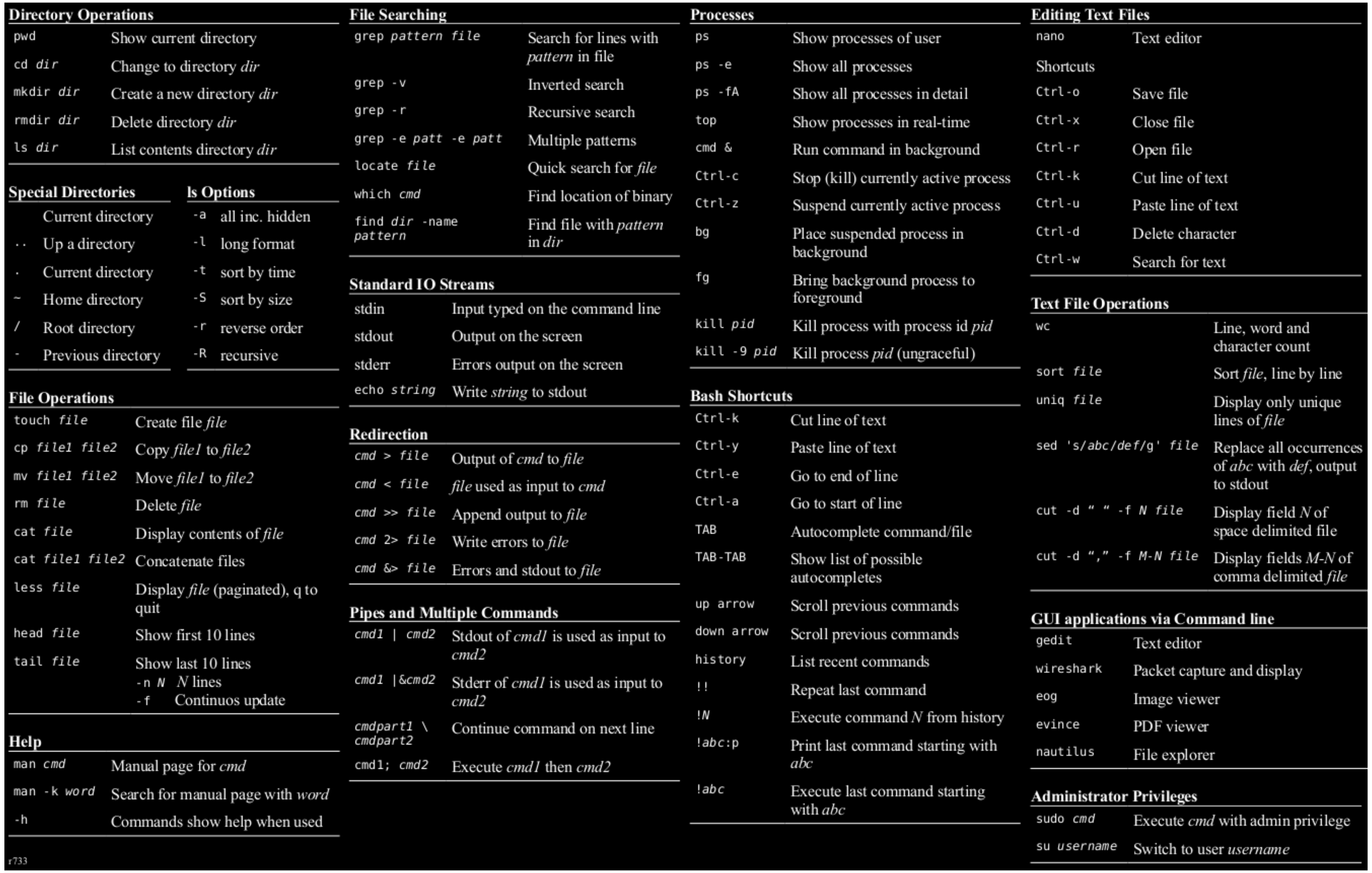
Storage Management



- Store your code and important files in home directory
- Scratch space is for temporary results/model fits/etc.

- You can only use a SFTP client to access the home directory
- To get results of scratch, move to the home directory first

Linux cheat sheet



Directory Operations

`pwd`	Show current directory
`cd dir`	Change to directory *dir*
`mkdir dir`	Create a new directory *dir*
`rmdir dir`	Delete directory *dir*
`ls dir`	List contents directory *dir*

Special Directories

	Current directory
`..`	Up a directory
`.`	Current directory
`~`	Home directory
`/`	Root directory
`-`	Previous directory

ls Options

`-a`	all inc. hidden
`-l`	long format
`-t`	sort by time
`-S`	sort by size
`-r`	reverse order
`-R`	recursive

File Operations

`touch file`	Create file *file*
`cp file1 file2`	Copy *file1* to *file2*
`mv file1 file2`	Move *file1* to *file2*
`rm file`	Delete *file*
`cat file`	Display contents of *file*
`cat file1 file2`	Concatenate files
`less file`	Display *file* (paginated), q to quit
`head file`	Show first 10 lines
`tail file`	Show last 10 lines `-n N` *N* lines `-f` Continuos update

Help

`man cmd`	Manual page for *cmd*
`man -k word`	Search for manual page with *word*
`-h`	Commands show help when used

r733

File Searching

`grep pattern file`	Search for lines with *pattern* in file
`grep -v`	Inverted search
`grep -r`	Recursive search
`grep -e patt -e patt`	Multiple patterns
`locate file`	Quick search for *file*
`which cmd`	Find location of binary
`find dir -name pattern`	Find file with *pattern* in *dir*

Standard IO Streams

stdin	Input typed on the command line
stdout	Output on the screen
stderr	Errors output on the screen
`echo string`	Write *string* to stdout

Redirection

`cmd > file`	Output of *cmd* to *file*
`cmd < file`	*file* used as input to *cmd*
`cmd >> file`	Append output to *file*
`cmd 2> file`	Write errors to *file*
`cmd &> file`	Errors and stdout to *file*

Pipes and Multiple Commands

`cmd1 | cmd2`	Stdout of *cmd1* is used as input to *cmd2*
`cmd1 |&cmd2`	Stderr of *cmd1* is used as input to *cmd2*
`cmdpart1 \ cmdpart2`	Continue command on next line
`cmd1; cmd2`	Execute *cmd1* then *cmd2*

Processes

`ps`	Show processes of user
`ps -e`	Show all processes
`ps -fA`	Show all processes in detail
`top`	Show processes in real-time
`cmd &`	Run command in background
`Ctrl-c`	Stop (kill) currently active process
`Ctrl-z`	Suspend currently active process
`bg`	Place suspended process in background
`fg`	Bring background process to foreground
`kill pid`	Kill process with process id *pid*
`kill -9 pid`	Kill process *pid* (ungraceful)

Bash Shortcuts

`Ctrl-k`	Cut line of text
`Ctrl-y`	Paste line of text
`Ctrl-e`	Go to end of line
`Ctrl-a`	Go to start of line
`TAB`	Autocomplete command/file
`TAB-TAB`	Show list of possible autocompletes
`up arrow`	Scroll previous commands
`down arrow`	Scroll previous commands
`history`	List recent commands
`!!`	Repeat last command
`!N`	Execute command *N* from history
`!abc:p`	Print last command starting with *abc*
`!abc`	Execute last command starting with *abc*

Editing Text Files

`nano`	Text editor
Shortcuts	
`Ctrl-o`	Save file
`Ctrl-x`	Close file
`Ctrl-r`	Open file
`Ctrl-k`	Cut line of text
`Ctrl-u`	Paste line of text
`Ctrl-d`	Delete character
`Ctrl-w`	Search for text

Text File Operations

`wc`	Line, word and character count
`sort file`	Sort *file*, line by line
`uniq file`	Display only unique lines of *file*
`sed 's/abc/def/g' file`	Replace all occurrences of *abc* with *def*, output to stdout
`cut -d " " -f N file`	Display field *N* of space delimited file
`cut -d "," -f M-N file`	Display fields *M-N* of comma delimited *file*

GUI applications via Command line

`gedit`	Text editor
`wireshark`	Packet capture and display
`eog`	Image viewer
`evince`	PDF viewer
`nautilus`	File explorer

Administrator Privileges

`sudo cmd`	Execute *cmd* with admin privilege
`su username`	Switch to user *username*

Reference Link

CHICAGO BOOTH

SLURM (Simple Linux Utility for Resource Management)

- SLURM Is the Job scheduler for the server

- You submit job “requests” to SLURM with instructions
  - (e.g.) I want 1 core with 2 GB for 2 hours. Then on this compute instance, I want to run my R/Python/MATLAB script

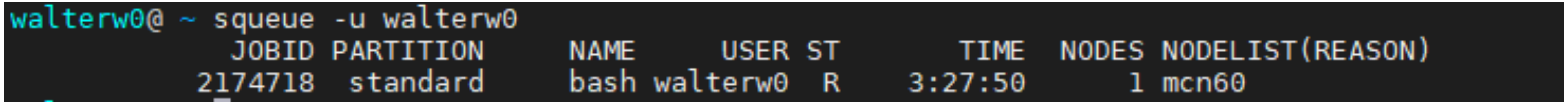
- SLURM then puts the job request in a queue. When a node is available the job is allocated and executed
  - The queue contains all the *running* and *pending* jobs

- There are “batch” and “interactive” jobs
  - Batch jobs require a batch script, or a set of instructions to SLURM

SLURM Commands

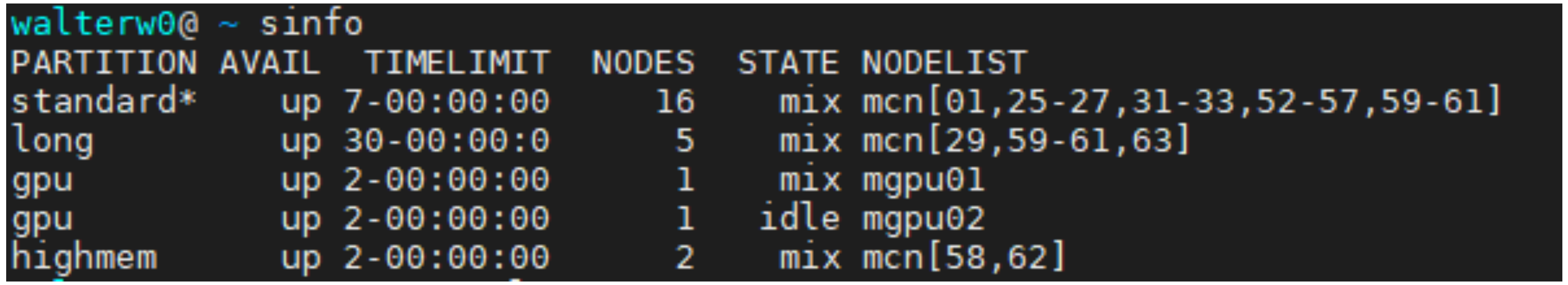
Useful SLURM commands:

- `squeue`: sees all jobs in the queue
  - `squeue -u <Booth-id>`: sees all *your* jobs in the queue



```
walterw0@ ~ squeue -u walterw0
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           2174718  standard     bash walterw0  R    3:27:50      1 mcn60
```

- `sinfo`: sees status of all the nodes on the server



```
walterw0@ ~ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
standard*    up 7-00:00:00     16    mix mcn[01,25-27,31-33,52-57,59-61]
long         up 30-00:00:0      5    mix mcn[29,59-61,63]
gpu          up 2-00:00:00      1    mix mgpu01
gpu          up 2-00:00:00      1   idle mgpu02
highmem      up 2-00:00:00      2    mix mcn[58,62]
```

SLURM Batch Script

Tells the computer to use `/bin/bash` for the script

Comment, everything ignored after # in bash

`#SBATCH` is a directive that is also a comment
- Only is relevant to the SLURM scheduler

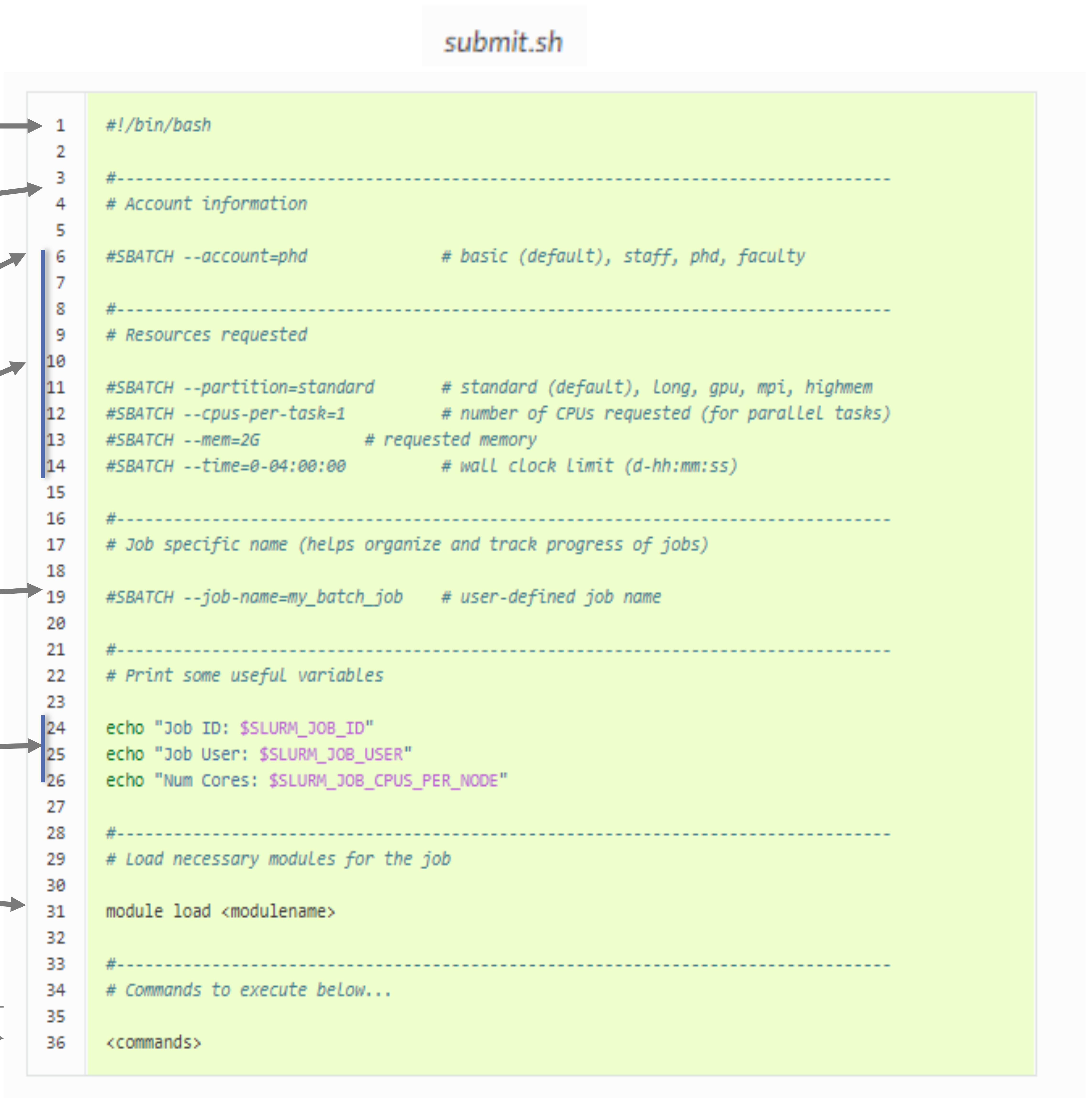
Set account, time, nodes, CPUs & memory

Give your job a name (optional)

Print out job details

Load required modules

Run commands/code


submit.sh

```bash
#!/bin/bash

#-----------------------------------------------------------------------------
# Account information

#SBATCH --account=phd              # basic (default), staff, phd, faculty

#-----------------------------------------------------------------------------
# Resources requested

#SBATCH --partition=standard       # standard (default), long, gpu, mpi, highmem
#SBATCH --cpus-per-task=1          # number of CPUs requested (for parallel tasks)
#SBATCH --mem=2G           # requested memory
#SBATCH --time=0-04:00:00          # wall clock limit (d-hh:mm:ss)

#-----------------------------------------------------------------------------
# Job specific name (helps organize and track progress of jobs)

#SBATCH --job-name=my_batch_job    # user-defined job name

#-----------------------------------------------------------------------------
# Print some useful variables

echo "Job ID: $SLURM_JOB_ID"
echo "Job User: $SLURM_JOB_USER"
echo "Num Cores: $SLURM_JOB_CPUS_PER_NODE"

#-----------------------------------------------------------------------------
# Load necessary modules for the job

module load <modulename>

#-----------------------------------------------------------------------------
# Commands to execute below...

<commands>
```

SLURM Batch Script

- After creating the batch SLURM script, `submit.sh`, we submit it to the scheduler using

```
sbatch submit.sh
```

- See your job in action with `squeue`
  - Each submitted job will be automatically assigned a `<job-id>`

- Cancel the job with `scancel <job-id>`
  - Cancel *all* your jobs with `scancel --user=<Booth-id>`

- See the run statistics of the completed job with `sacct -j <job-id>`
  - Included both successfully completed and failed jobs
  - Once the job is done it will give out a `<job-id>.out` and `<job-id>.err` which are the output and error messages from the job

Example 1: Sample SLURM Job

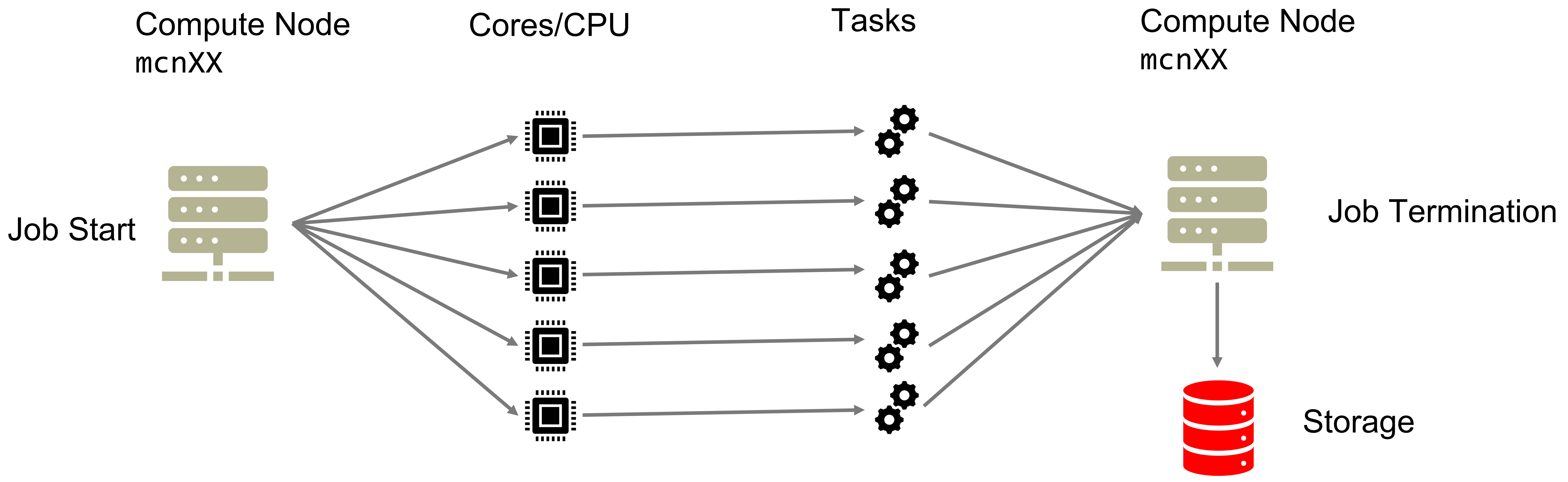
- We will walkthrough the sample SLURM Job in the repository

Tasks:

1. View the `submit.sh` script, see what it does
2. Submit it to the SLURM job scheduler and find the `job-id`
3. View the SLURM output and error files (if any) from the job

Parallel Jobs

Parallel job is any job that uses parallel processing, which in turn needs more than one core/CPU



Parallel Jobs

- You need to specify the number of cores that you want with

```
#SBATCH --ntasks-per-node=8
```

  - Where we want 8 cores/CPUs here

- You can see the number of CPUs/cores requested the with *environmental* variable:

```
SLURM_JOB_CPUS_PER_NODE
```

  - We can then call this variable in our code to specify how many cores we want to use

- Cannot ask for more cores than what is on a machine (28 for a `standard` partition)

Example 2: Parallel SLURM Job

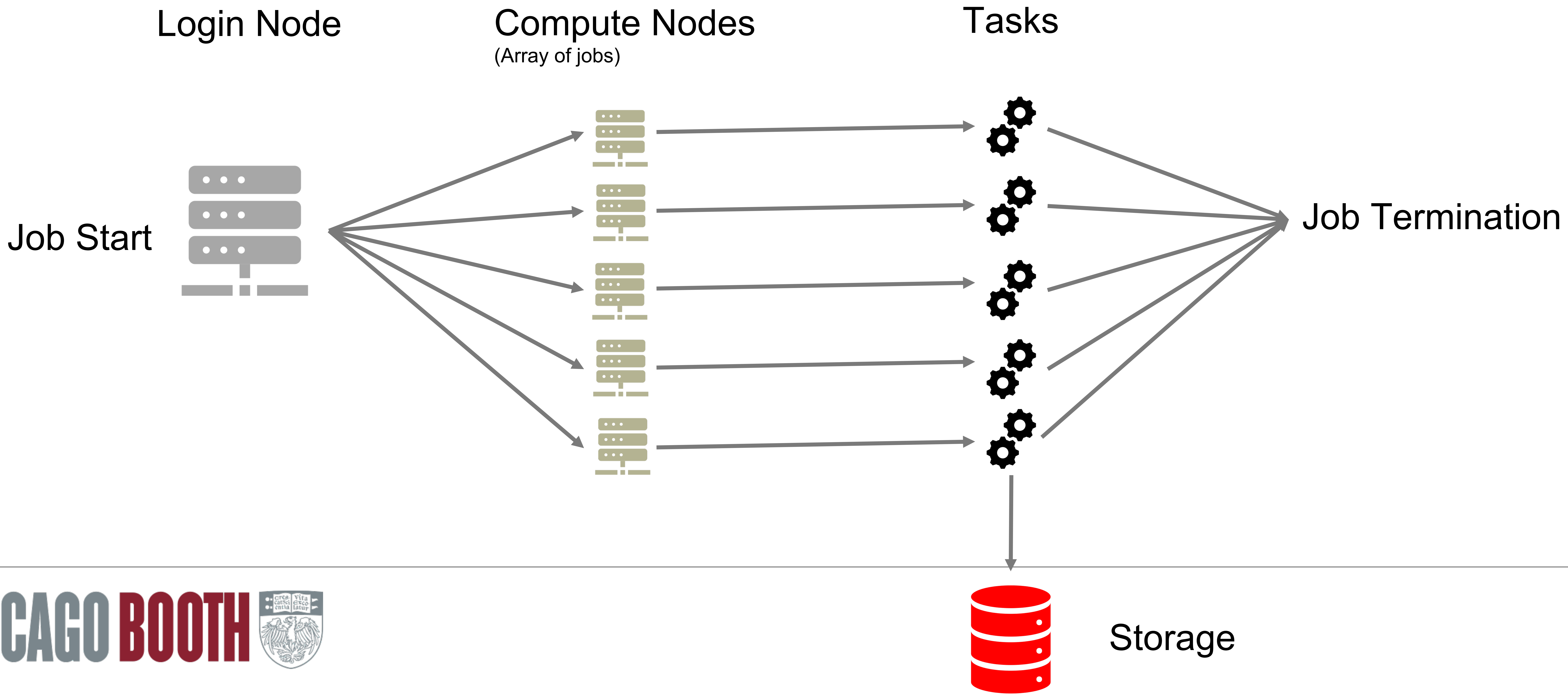
- We will walkthrough submitting a parallel SLURM Job in the repository.

Tasks:

1. View the `parallel.sh` script, see what it does, and find the part that asks for the multi-cores
2. Submit it to the SLURM job scheduler and find the `job-id`
3. View the SLURM output and error files from the job, see if both cores were used

Array Jobs

Arrays jobs submit a collection of similar jobs (e.g. Bootstrap iterations)
- An array of separate jobs



Array Jobs

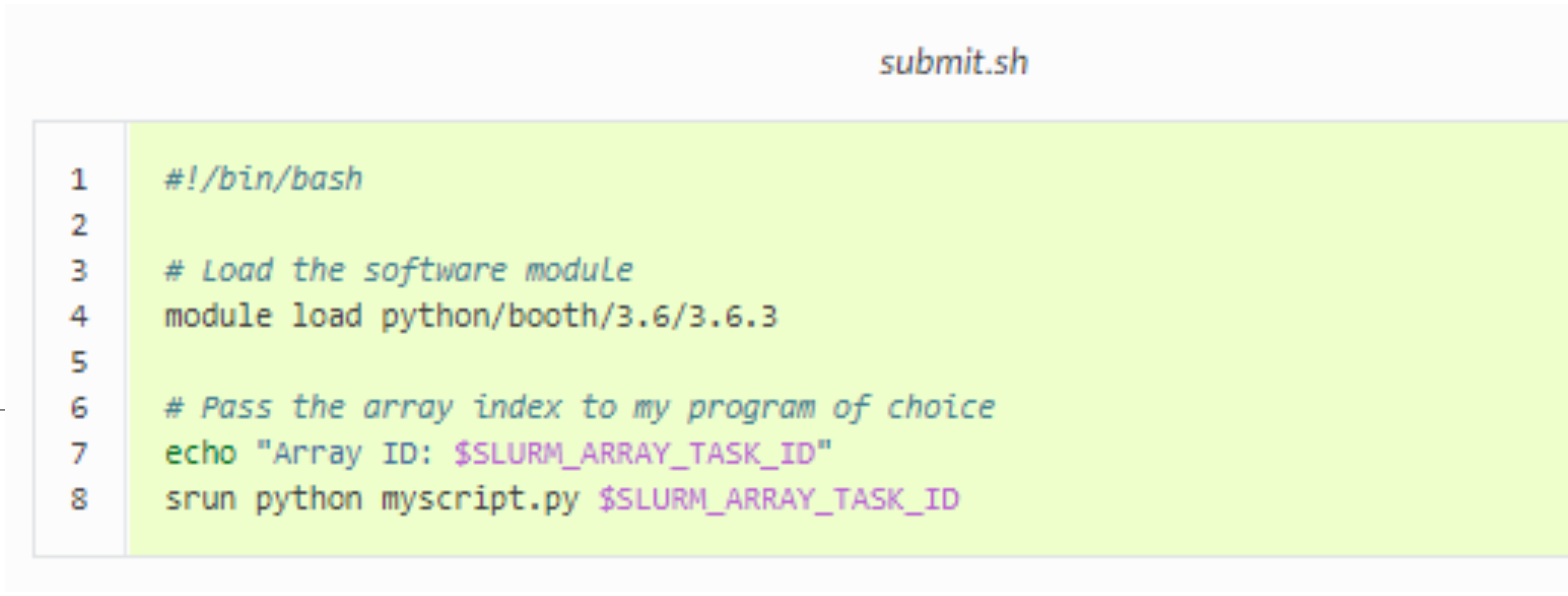
- You need to specify how many array jobs when you submit the job

```
sbatch --array=0,1,5 submit.sh
sbatch --array=[1,500] submit.sh
```

  - First command sends an array with job index {0,1,5} – three jobs in total
  - Second command sends an array with 500 jobs indexed 1 to 500
  - Can also put the `--array` flag in the SLURM batch file

- The array job ID is saved in an *environmental* variable ($SLURM_ARRAY_TASK_ID)
  - You can pass this to your program to use

- Cannot run than more 250 single core jobs at once


submit.sh

```
#!/bin/bash

# Load the software module
module load python/booth/3.6/3.6.3

# Pass the array index to my program of choice
echo "Array ID: $SLURM_ARRAY_TASK_ID"
srun python myscript.py $SLURM_ARRAY_TASK_ID
```

Example 3: Array SLURM Job

- We will walkthrough submitting an array SLURM Job in the repository.

Tasks:

1. View the `array.sh` script, see what it does
2. Submit it to the SLURM job scheduler and find the `job-id`
3. View the SLURM output and error files from the job, see if the expected number of array jobs was run.

Resource Allocation

Cores ⟷ Wall Time ⟷ Memory

Serial Job
1 Core

Parallel Job
>1 Core
Some software may default to using more than one core

Rule of thumb:
Expected run time x 1.3

Trial and Error:
Usually allocate a few GB more than you expect

- Running out of *time* or *memory* will terminate the job
- Running a parallel job with only one core is inefficient

If you over ask for resources, SLURM will take longer to allocate your job (longer queue time)

Resource Allocation

. “Greedy” Allocation Requests

- Ask for all the memory on a node
  - `#SBATCH --mem=0`

- Ask for an exclusive node for your jobs
  - `#SBATCH --exclusive`

- These will lead to significantly longer queue times
  - You need to wait for the whole node to free up

- Tip: Instead of requesting the whole node, leave one GB memory or one core free

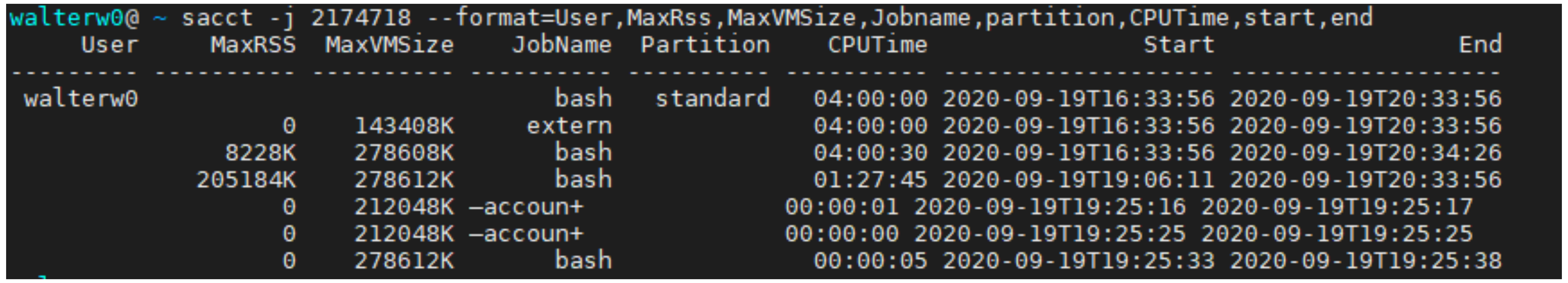
Resource Allocation (Examples)

Some examples:

- Interactive Session on the `gpu` (GPU:1)
  - `srun --account=phd --partition=gpu --gres=gpu:1 --pty bash -l`
- Interactive Session on `highmem`
  - `srun --account=phd --partition=highmem --mem=100G --pty bash -l`
- The resource request format for a SLURM interactive session and a SLURM batch job are similar

Resource Allocation

- What resources did I use?



```
walterw0@ ~ sacct -j 2174718 --format=User,MaxRss,MaxVMSize,Jobname,partition,CPUTime,start,end
     User     MaxRSS  MaxVMSize    JobName  Partition    CPUTime               Start                 End
--------- ---------- ---------- ---------- ---------- ---------- ------------------- -------------------
 walterw0                             bash   standard   04:00:00 2020-09-19T16:33:56 2020-09-19T20:33:56
                   0    143408K     extern              04:00:00 2020-09-19T16:33:56 2020-09-19T20:33:56
               8228K    278608K       bash              04:00:30 2020-09-19T16:33:56 2020-09-19T20:34:26
             205184K    278612K       bash              01:27:45 2020-09-19T19:06:11 2020-09-19T20:33:56
                   0    212048K -accoun+              00:00:01 2020-09-19T19:25:16 2020-09-19T19:25:17
                   0    212048K -accoun+              00:00:00 2020-09-19T19:25:25 2020-09-19T19:25:25
                   0    278612K       bash              00:00:05 2020-09-19T19:25:33 2020-09-19T19:25:38
```

- We can check with

```
sacct -j <jobID> --format=User,MaxRss,MaxVMSize,Jobname,partition,CPUtime,start,end
```

- The `sacct` command lets us view job statistics
  - *MaxRSS* is the memory use
  - *CPUTime* yields the runtime of the job

- Not being overallocated leads to a shorter queuing time and more efficient server usage

Unanticipated Job Termination

Why did my job fail?

1. Ran out of memory
2. Ran out of time
3. SLURM submit script issue (submit.sh)
4. Problem with your code (R/Python code issue)
5. Node failure

- Outside of the node failure you can fix your code or change your requested allocation
  - Email Research Support (rsupport@chicagobooth.edu) if you suspect a node failure

Online Resources

- User guide for Booth Mercury: https://hpc-docs.chicagobooth.edu/index.html
  - FAQ: https://hpc-docs.chicagobooth.edu/faq.html
- SLURM cheat sheet: https://slurm.schedmd.com/pdfs/summary.pdf
- SLURM documentation: https://slurm.schedmd.com/sbatch.html
- Quick Bash Guide: https://github.com/Idnan/bash-guide
- Booth Research Support Email: rsupport@chicagobooth.edu

Questions so far?

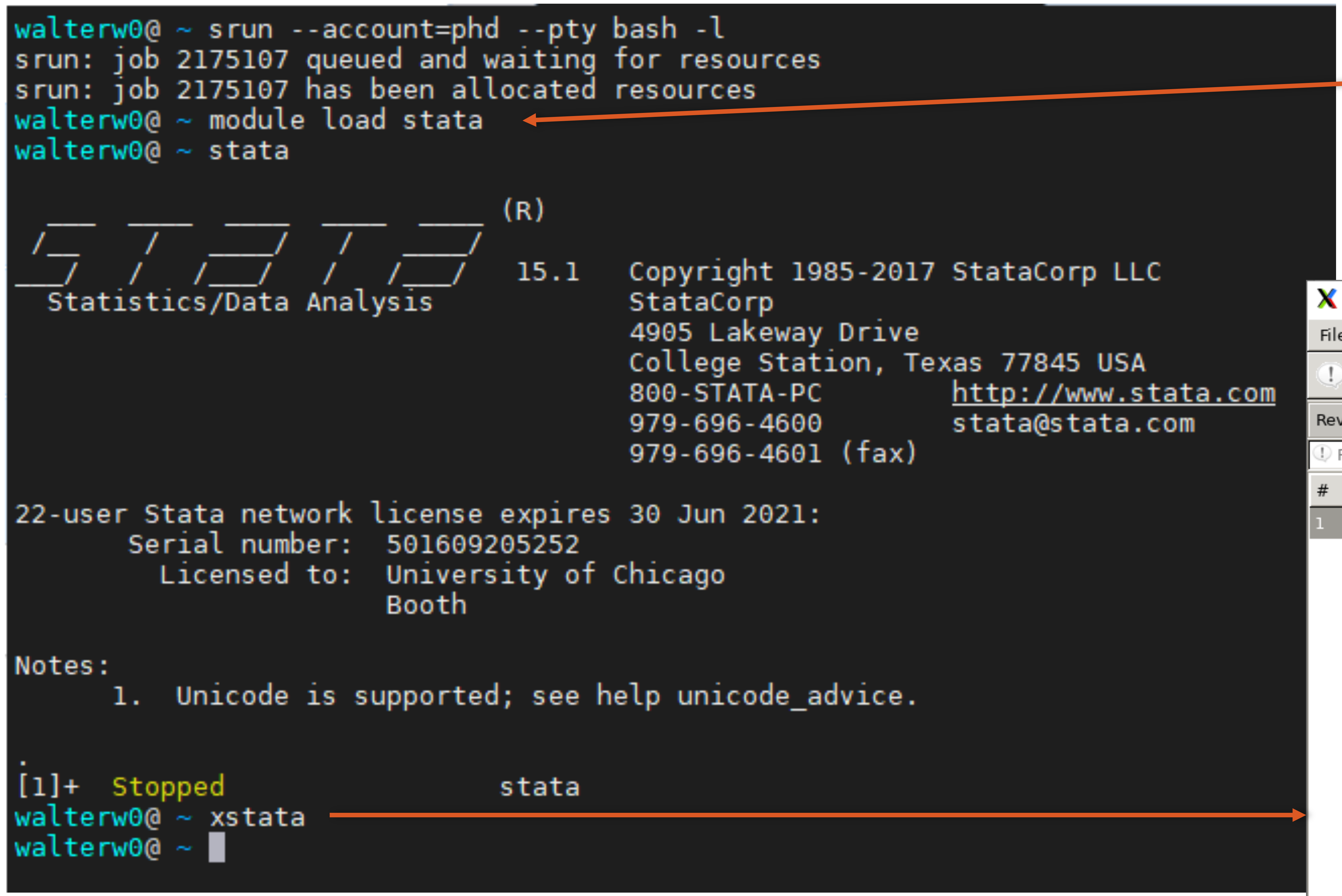
Running Common Programs

- We will walk through setting up some commonly used programs:

1. STATA
2. MATLAB
3. Julia
4. Python
   - Python + Tensorflow
5. R
   - R + C++
   - R + Gurobi
   - R + Knitro

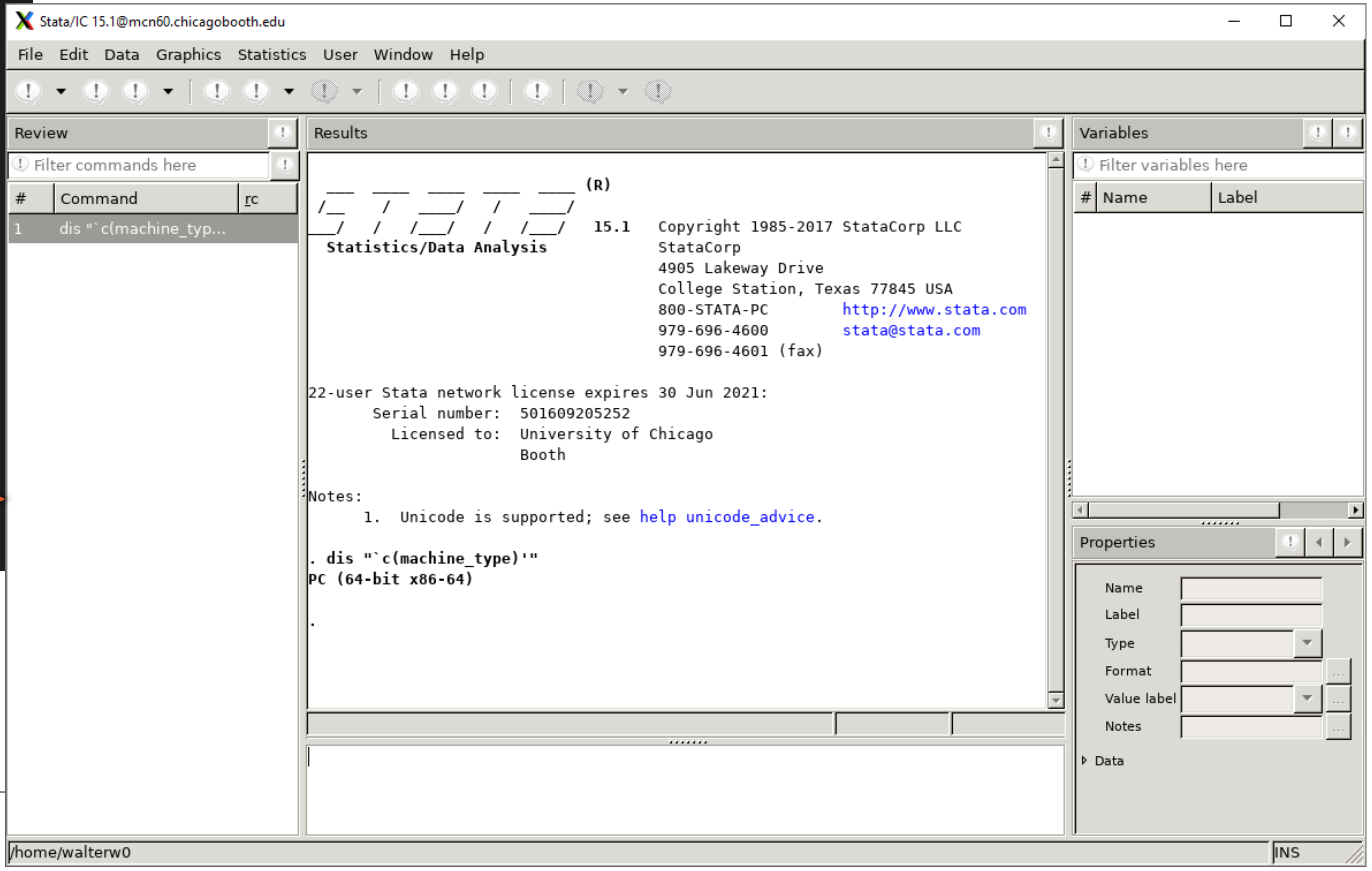
- For the GUI programs, we need to keep the terminal open in our interactive session (also need X11 forwarding set up)
  - Interactive command line plots needs X11 forwarding too
- Running other programs uses the same framework (request job → load module(s) → run code)
  - Packages are always installed to a *local* library
- We always start from the login node in our walkthrough

STATA Command Line + GUI



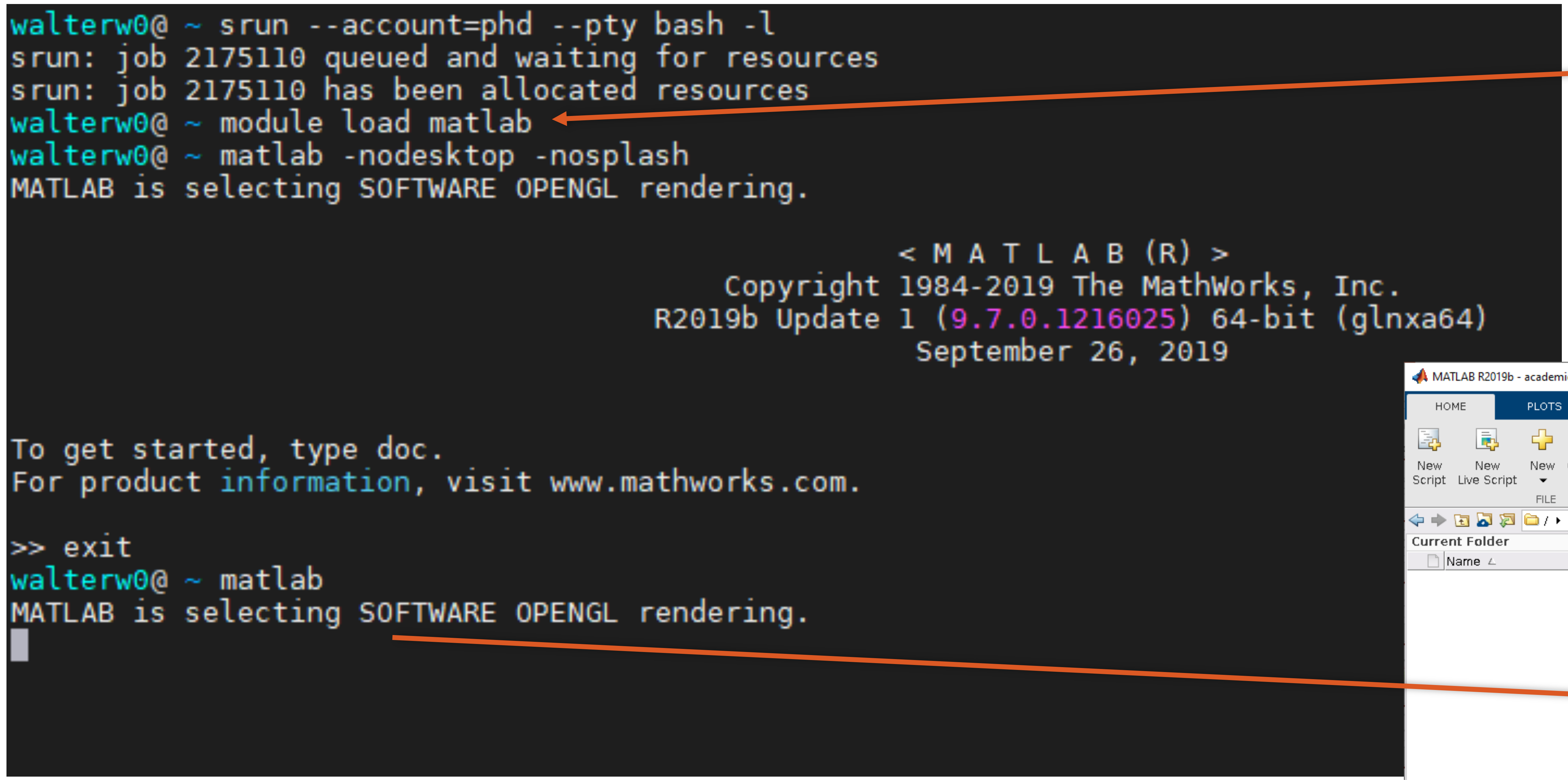
```
walterw0@ ~ srun --account=phd --pty bash -l
srun: job 2175107 queued and waiting for resources
srun: job 2175107 has been allocated resources
walterw0@ ~ module load stata
walterw0@ ~ stata

  ___  ____  ____  ____  ____ (R)
 /__    /   ____/   /   ____/
___/   /   /___/   /   /___/   15.1   Copyright 1985-2017 StataCorp LLC
  Statistics/Data Analysis            StataCorp
                                      4905 Lakeway Drive
                                      College Station, Texas 77845 USA
                                      800-STATA-PC        http://www.stata.com
                                      979-696-4600        stata@stata.com
                                      979-696-4601 (fax)

22-user Stata network license expires 30 Jun 2021:
       Serial number:  501609205252
         Licensed to:  University of Chicago
                       Booth

Notes:
      1.  Unicode is supported; see help unicode_advice.

.
[1]+  Stopped                 stata
walterw0@ ~ xstata
walterw0@ ~ 
```

`module load stata/17.0` if there is more than one version



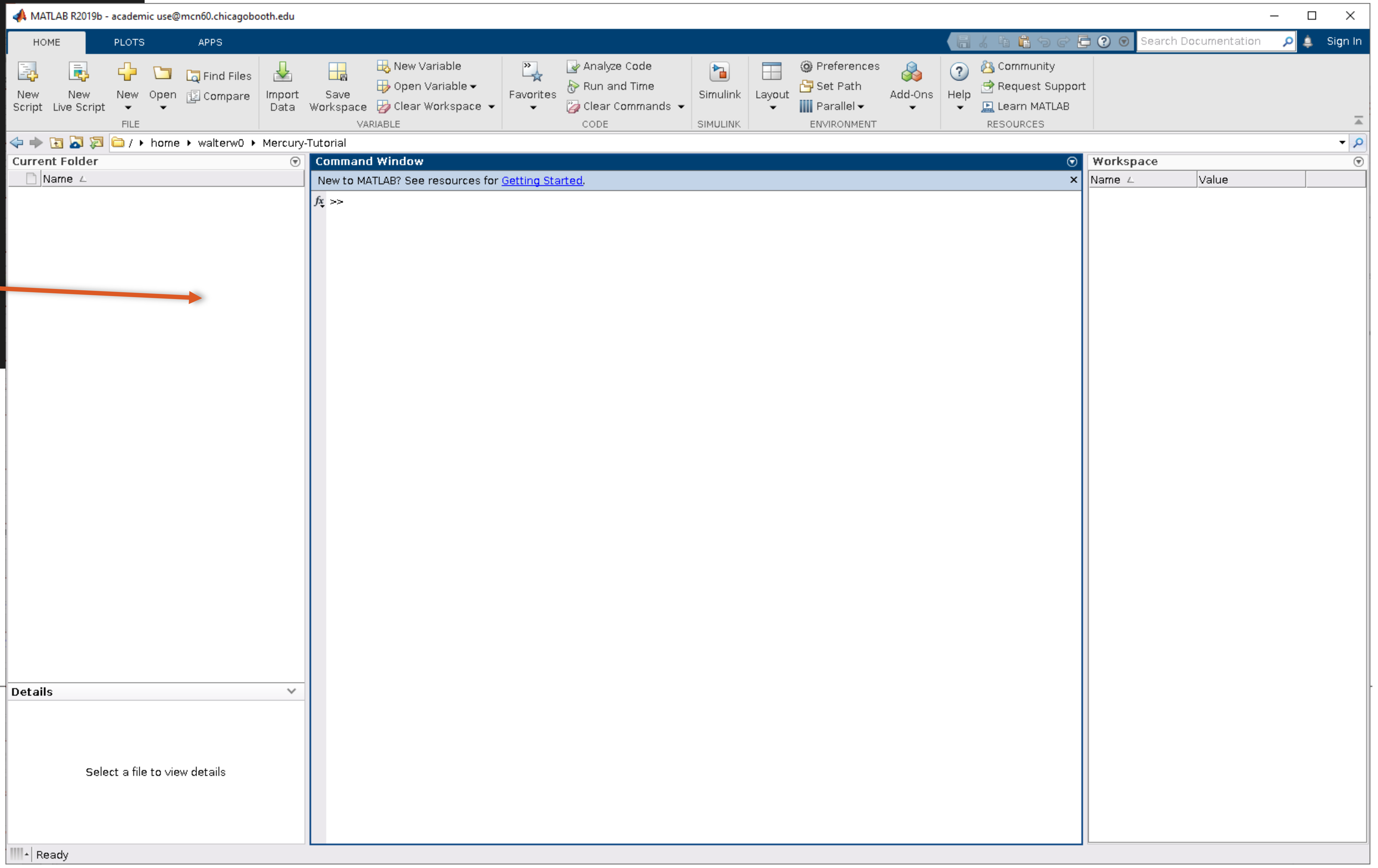
```
  ___  ____  ____  ____  ____ (R)
 /__    /   ____/   /   ____/
___/   /   /___/   /   /___/   15.1   Copyright 1985-2017 StataCorp LLC
  Statistics/Data Analysis            StataCorp
                                      4905 Lakeway Drive
                                      College Station, Texas 77845 USA
                                      800-STATA-PC        http://www.stata.com
                                      979-696-4600        stata@stata.com
                                      979-696-4601 (fax)

22-user Stata network license expires 30 Jun 2021:
       Serial number:  501609205252
         Licensed to:  University of Chicago
                       Booth

Notes:
      1.  Unicode is supported; see help unicode_advice.

. dis "`c(machine_type)'"
PC (64-bit x86-64)

.
```

MATLAB Command Line + GUI



```
walterw0@ ~ srun --account=phd --pty bash -l
srun: job 2175110 queued and waiting for resources
srun: job 2175110 has been allocated resources
walterw0@ ~ module load matlab
walterw0@ ~ matlab -nodesktop -nosplash
MATLAB is selecting SOFTWARE OPENGL rendering.

                                    < M A T L A B (R) >
                          Copyright 1984-2019 The MathWorks, Inc.
                     R2019b Update 1 (9.7.0.1216025) 64-bit (glnxa64)
                                   September 26, 2019


To get started, type doc.
For product information, visit www.mathworks.com.

>> exit
walterw0@ ~ matlab
MATLAB is selecting SOFTWARE OPENGL rendering.
```

`module load matlab/R2021b` (to specify version)

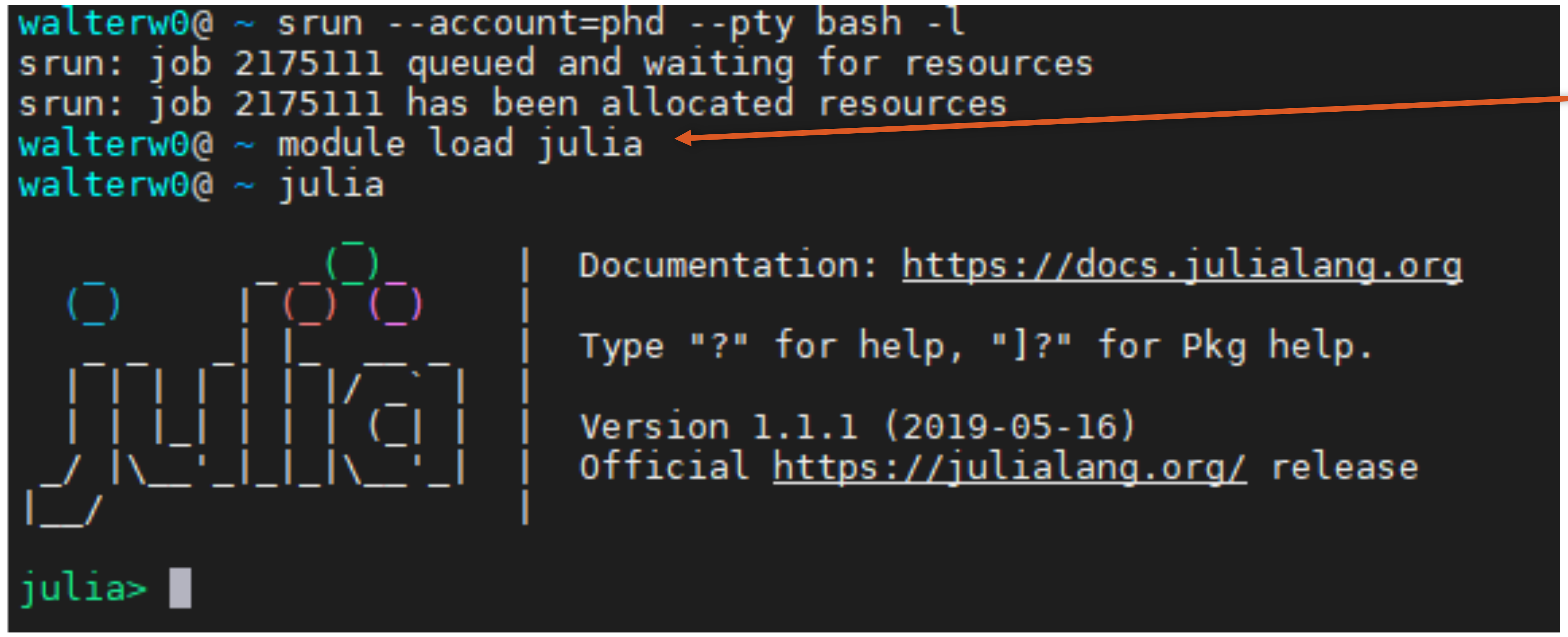


New to MATLAB? See resources for Getting Started.

```
>>
```

Select a file to view details

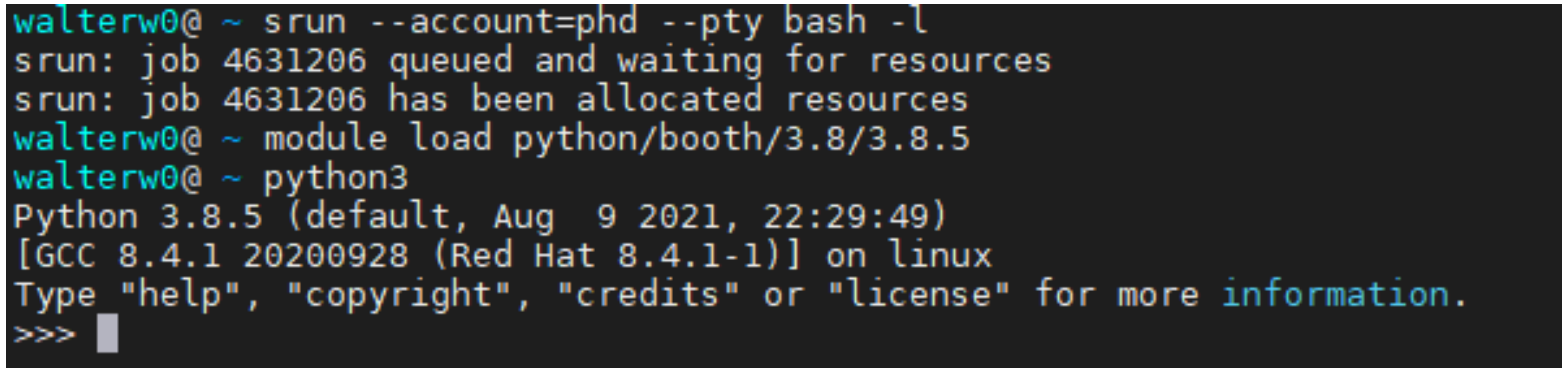
Julia



```
walterw0@ ~ srun --account=phd --pty bash -l
srun: job 2175111 queued and waiting for resources
srun: job 2175111 has been allocated resources
walterw0@ ~ module load julia
walterw0@ ~ julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.1.1 (2019-05-16)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia>
```

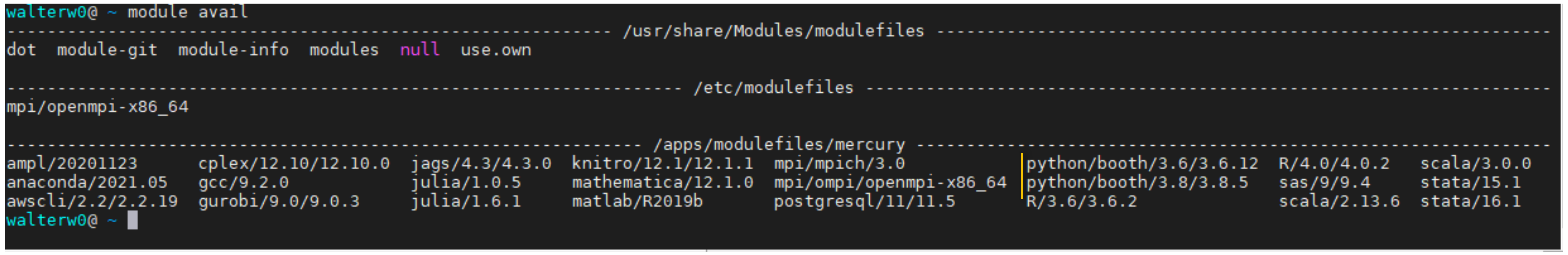
`module load julia/1.6.6` (to specify version)

Python



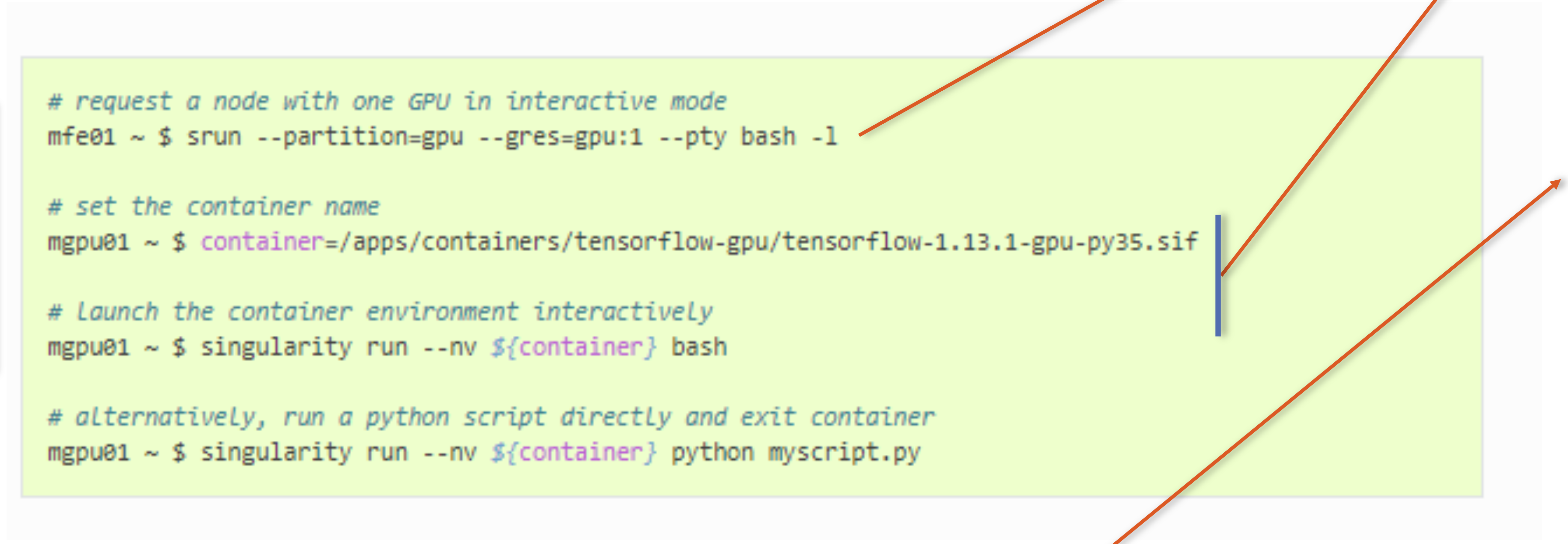
```
walterw0@ ~ srun --account=phd --pty bash -l
srun: job 4631206 queued and waiting for resources
srun: job 4631206 has been allocated resources
walterw0@ ~ module load python/booth/3.8/3.8.5
walterw0@ ~ python3
Python 3.8.5 (default, Aug  9 2021, 22:29:49)
[GCC 8.4.1 20200928 (Red Hat 8.4.1-1)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

- Different versions of Python



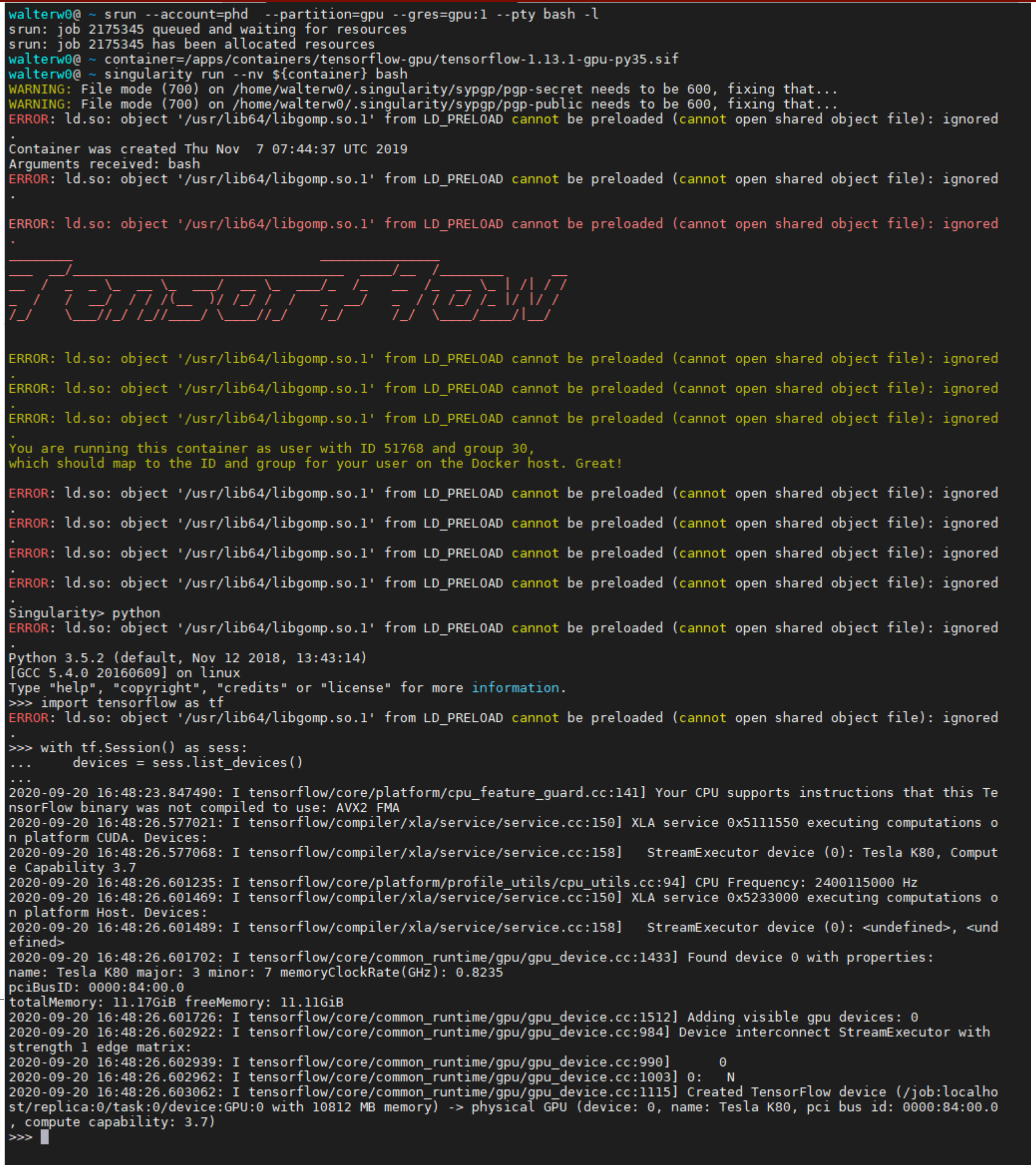
```
walterw0@ ~ module avail
------------------------------------------------------------- /usr/share/Modules/modulefiles --------------------------------------------------------------
dot  module-git  module-info  modules  null  use.own

------------------------------------------------------------------- /etc/modulefiles -----------------------------------------------------------------------
mpi/openmpi-x86_64

--------------------------------------------------------------- /apps/modulefiles/mercury ------------------------------------------------------------------
ampl/20201123      cplex/12.10/12.10.0  jags/4.3/4.3.0  knitro/12.1/12.1.1  mpi/mpich/3.0             python/booth/3.6/3.6.12  R/4.0/4.0.2   scala/3.0.0
anaconda/2021.05   gcc/9.2.0            julia/1.0.5     mathematica/12.1.0  mpi/ompi/openmpi-x86_64   python/booth/3.8/3.8.5   sas/9/9.4     stata/15.1
awscli/2.2/2.2.19  gurobi/9.0/9.0.3     julia/1.6.1     matlab/R2019b       postgresql/11/11.5        R/3.6/3.6.2              scala/2.13.6  stata/16.1
walterw0@ ~ 
```

Python + Tensorflow



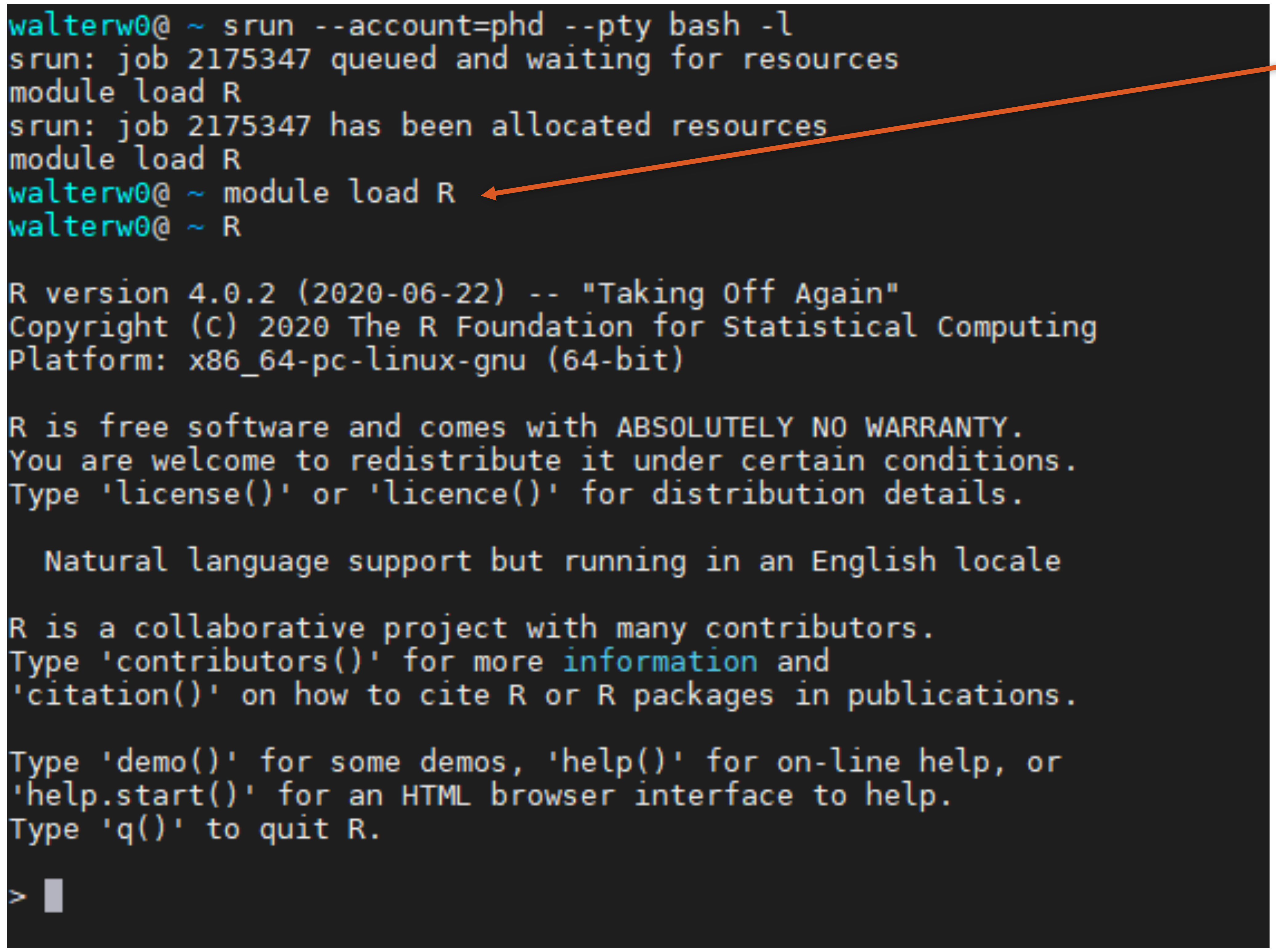
```
# request a node with one GPU in interactive mode
mfe01 ~ $ srun --partition=gpu --gres=gpu:1 --pty bash -l

# set the container name
mgpu01 ~ $ container=/apps/containers/tensorflow-gpu/tensorflow-1.13.1-gpu-py35.sif

# launch the container environment interactively
mgpu01 ~ $ singularity run --nv ${container} bash

# alternatively, run a python script directly and exit container
mgpu01 ~ $ singularity run --nv ${container} python myscript.py
```

- Uses *singularity* as a container
- Also exists a *virtualenv* approach
- Add `--account=phd` to the first line



```
walterw0@ ~ srun --account=phd  --partition=gpu --gres=gpu:1 --pty bash -l
srun: job 2175345 queued and waiting for resources
srun: job 2175345 has been allocated resources
walterw0@ ~ container=/apps/containers/tensorflow-gpu/tensorflow-1.13.1-gpu-py35.sif
walterw0@ ~ singularity run --nv ${container} bash
WARNING: File mode (700) on /home/walterw0/.singularity/sypgp/pgp-secret needs to be 600, fixing that...
WARNING: File mode (700) on /home/walterw0/.singularity/sypgp/pgp-public needs to be 600, fixing that...
ERROR: ld.so: object '/usr/lib64/libgomp.so.1' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored
.
Container was created Thu Nov  7 07:44:37 UTC 2019
Arguments received: bash
ERROR: ld.so: object '/usr/lib64/libgomp.so.1' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored
.

ERROR: ld.so: object '/usr/lib64/libgomp.so.1' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored
.
________                               _______________
___  __/__________________________________  ____/__  /________      __
__  /  _  _ \_  __ \_  ___/  __ \_  ___/_  /_   __  /_  __ \_ | /| / /
_  /   /  __/  / / /(__  )/ /_/ /  /   _  __/   _  / / /_/ /_ |/ |/ /
/_/    \___//_/ /_//____/ \____//_/    /_/      /_/  \____/____/|__/


ERROR: ld.so: object '/usr/lib64/libgomp.so.1' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored
.
ERROR: ld.so: object '/usr/lib64/libgomp.so.1' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored
.
ERROR: ld.so: object '/usr/lib64/libgomp.so.1' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored
.
You are running this container as user with ID 51768 and group 30,
which should map to the ID and group for your user on the Docker host. Great!

ERROR: ld.so: object '/usr/lib64/libgomp.so.1' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored
.
ERROR: ld.so: object '/usr/lib64/libgomp.so.1' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored
.
ERROR: ld.so: object '/usr/lib64/libgomp.so.1' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored
.
ERROR: ld.so: object '/usr/lib64/libgomp.so.1' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored
.
Singularity> python
ERROR: ld.so: object '/usr/lib64/libgomp.so.1' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored
.
Python 3.5.2 (default, Nov 12 2018, 13:43:14)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
ERROR: ld.so: object '/usr/lib64/libgomp.so.1' from LD_PRELOAD cannot be preloaded (cannot open shared object file): ignored
.
>>> with tf.Session() as sess:
...     devices = sess.list_devices()
...
2020-09-20 16:48:23.847490: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this Te
nsorFlow binary was not compiled to use: AVX2 FMA
2020-09-20 16:48:26.577021: I tensorflow/compiler/xla/service/service.cc:150] XLA service 0x5111550 executing computations o
n platform CUDA. Devices:
2020-09-20 16:48:26.577068: I tensorflow/compiler/xla/service/service.cc:158]   StreamExecutor device (0): Tesla K80, Comput
e Capability 3.7
2020-09-20 16:48:26.601235: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2400115000 Hz
2020-09-20 16:48:26.601469: I tensorflow/compiler/xla/service/service.cc:150] XLA service 0x5233000 executing computations o
n platform Host. Devices:
2020-09-20 16:48:26.601489: I tensorflow/compiler/xla/service/service.cc:158]   StreamExecutor device (0): <undefined>, <und
efined>
2020-09-20 16:48:26.601702: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1433] Found device 0 with properties:
name: Tesla K80 major: 3 minor: 7 memoryClockRate(GHz): 0.8235
pciBusID: 0000:84:00.0
totalMemory: 11.17GiB freeMemory: 11.11GiB
2020-09-20 16:48:26.601726: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1512] Adding visible gpu devices: 0
2020-09-20 16:48:26.602922: I tensorflow/core/common_runtime/gpu/gpu_device.cc:984] Device interconnect StreamExecutor with
strength 1 edge matrix:
2020-09-20 16:48:26.602939: I tensorflow/core/common_runtime/gpu/gpu_device.cc:990]      0
2020-09-20 16:48:26.602962: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1003] 0:   N
2020-09-20 16:48:26.603062: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localho
st/replica:0/task:0/device:GPU:0 with 10812 MB memory) -> physical GPU (device: 0, name: Tesla K80, pci bus id: 0000:84:00.0
, compute capability: 3.7)
>>> 
```

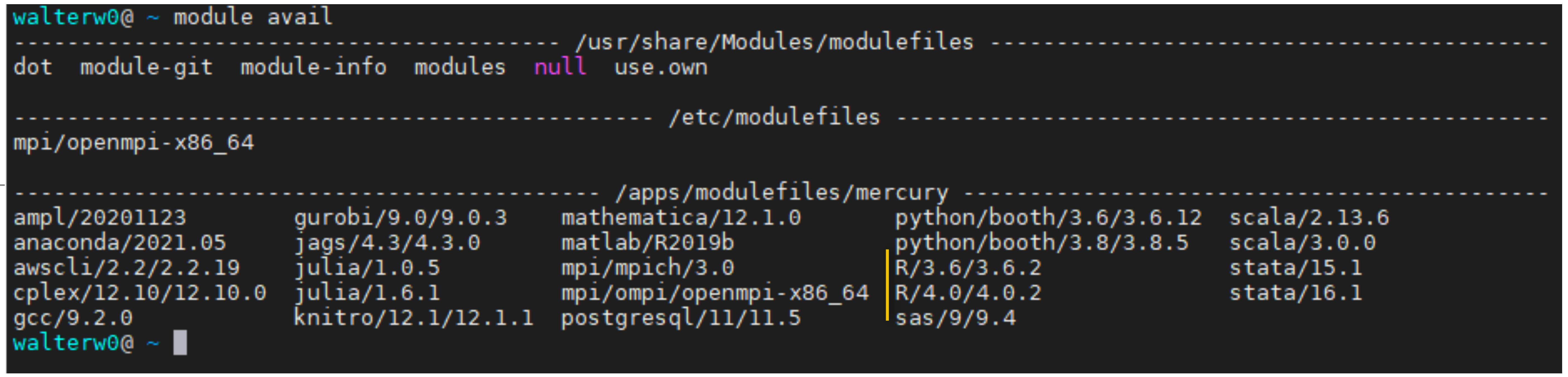
R



```
walterw0@ ~ srun --account=phd --pty bash -l
srun: job 2175347 queued and waiting for resources
module load R
srun: job 2175347 has been allocated resources
module load R
walterw0@ ~ module load R
walterw0@ ~ R

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

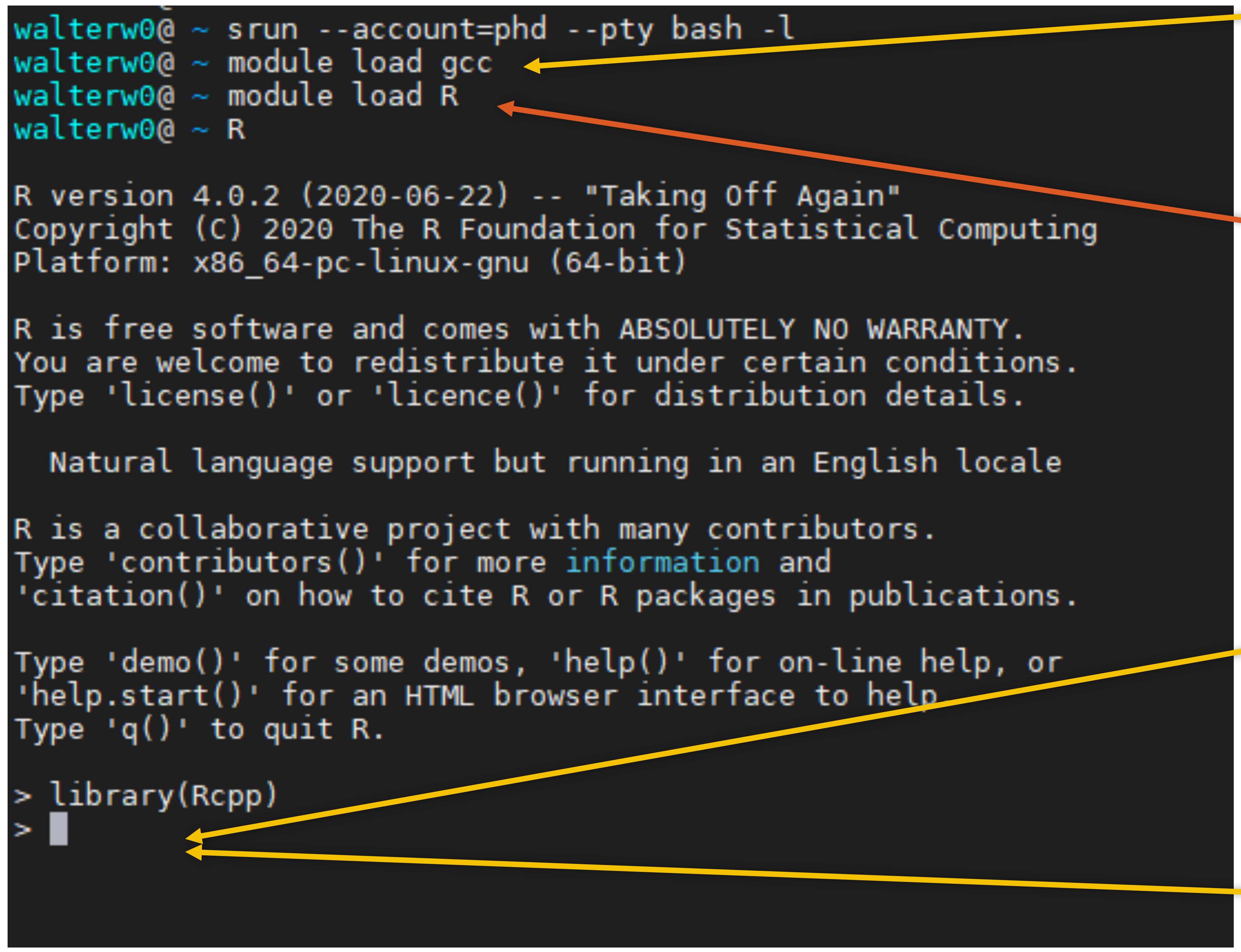
`module load R/4.0/4.0.2` if there is more than one version

- Many different versions of R available



```
walterw0@ ~ module avail
----------------------------------------- /usr/share/Modules/modulefiles ------------------------------------------
dot  module-git  module-info  modules  null  use.own

-------------------------------------------------- /etc/modulefiles -------------------------------------------------
mpi/openmpi-x86_64

--------------------------------------------- /apps/modulefiles/mercury ---------------------------------------------
ampl/20201123          gurobi/9.0/9.0.3    mathematica/12.1.0       python/booth/3.6/3.6.12  scala/2.13.6
anaconda/2021.05       jags/4.3/4.3.0      matlab/R2019b            python/booth/3.8/3.8.5   scala/3.0.0
awscli/2.2/2.2.19      julia/1.0.5         mpi/mpich/3.0            R/3.6/3.6.2              stata/15.1
cplex/12.10/12.10.0    julia/1.6.1         mpi/ompi/openmpi-x86_64  R/4.0/4.0.2              stata/16.1
gcc/9.2.0              knitro/12.1/12.1.1  postgresql/11/11.5       sas/9/9.4
walterw0@ ~
```

R + Rcpp

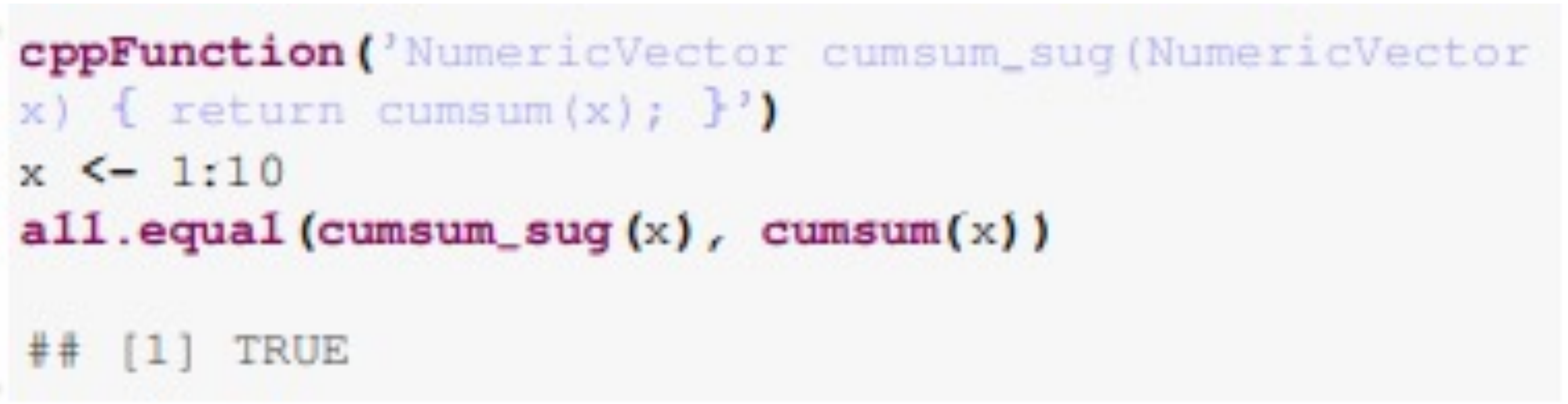


```
walterw0@ ~ srun --account=phd --pty bash -l
walterw0@ ~ module load gcc
walterw0@ ~ module load R
walterw0@ ~ R

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(Rcpp)
> 
```

Generally, we want to load the backend first – Here it's the C++ complier `gcc`

```
module load gcc/9.2.0
module load R/4.0/4.0.2
```

Sample Rcpp Code to run in R
(Cumulative Sum Function)

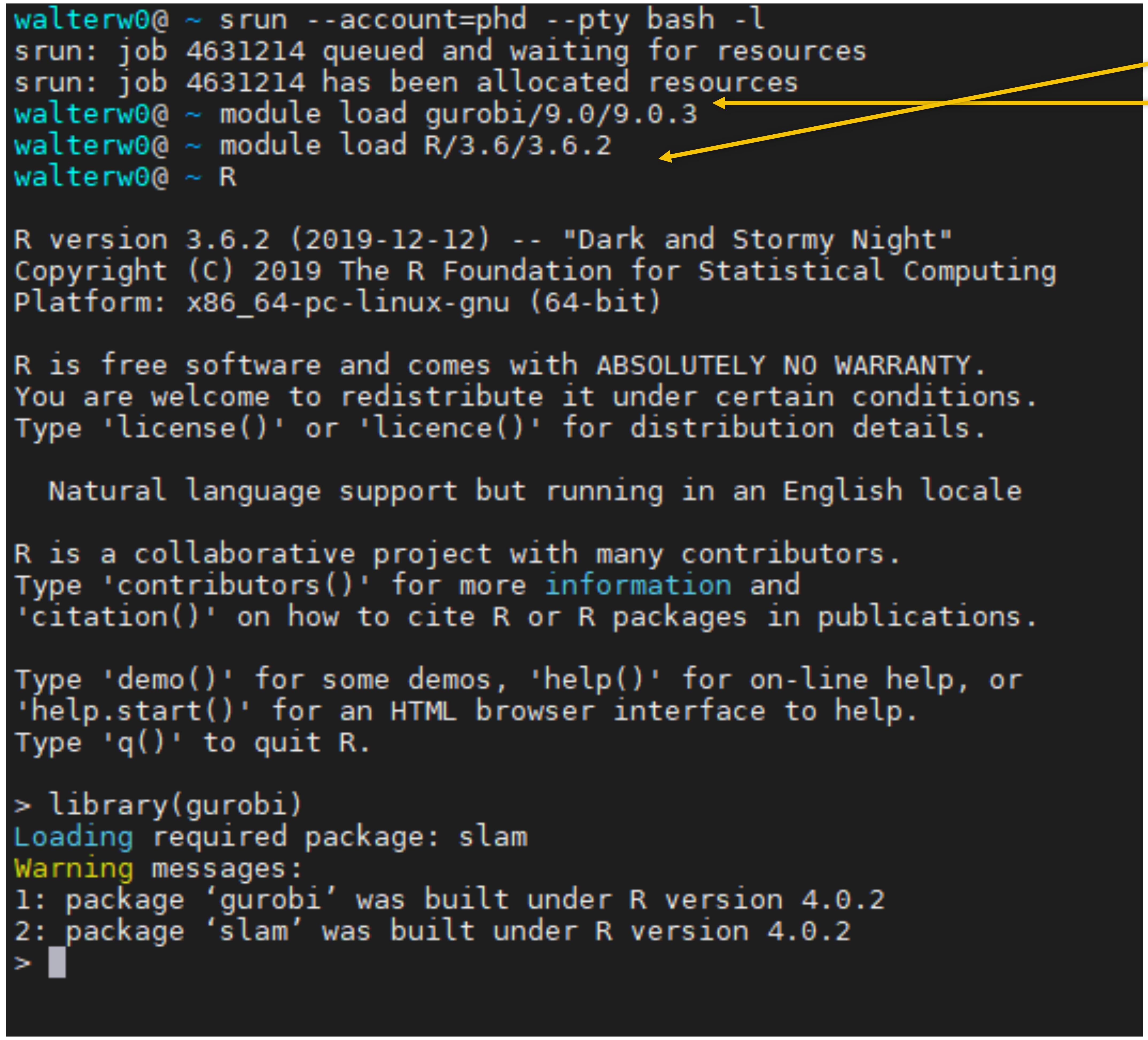


```
cppFunction('NumericVector cumsum_sug(NumericVector
x) { return cumsum(x); }')
x <- 1:10
all.equal(cumsum_sug(x), cumsum(x))

## [1] TRUE
```

R + gurobi

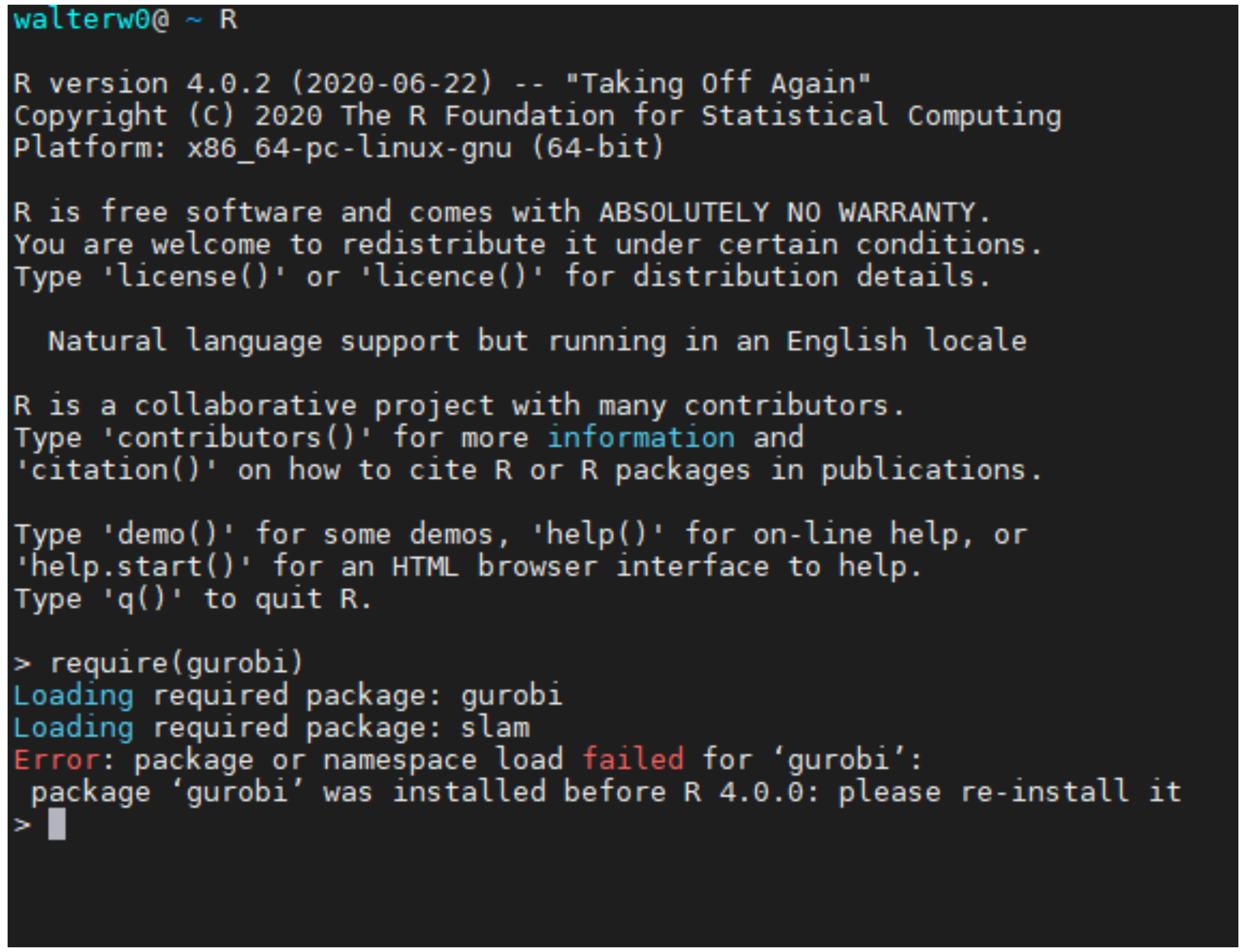
Different versions are important for software dependence

- gurobi needs R v3.6.2 *specifically* to work
- Always load gurobi *first* before R



```
walterw0@ ~ srun --account=phd --pty bash -l
srun: job 4631214 queued and waiting for resources
srun: job 4631214 has been allocated resources
walterw0@ ~ module load gurobi/9.0/9.0.3
walterw0@ ~ module load R/3.6/3.6.2
walterw0@ ~ R

R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(gurobi)
Loading required package: slam
Warning messages:
1: package ‘gurobi’ was built under R version 4.0.2
2: package ‘slam’ was built under R version 4.0.2
> 
```

Wrong R version (v4.0.2)

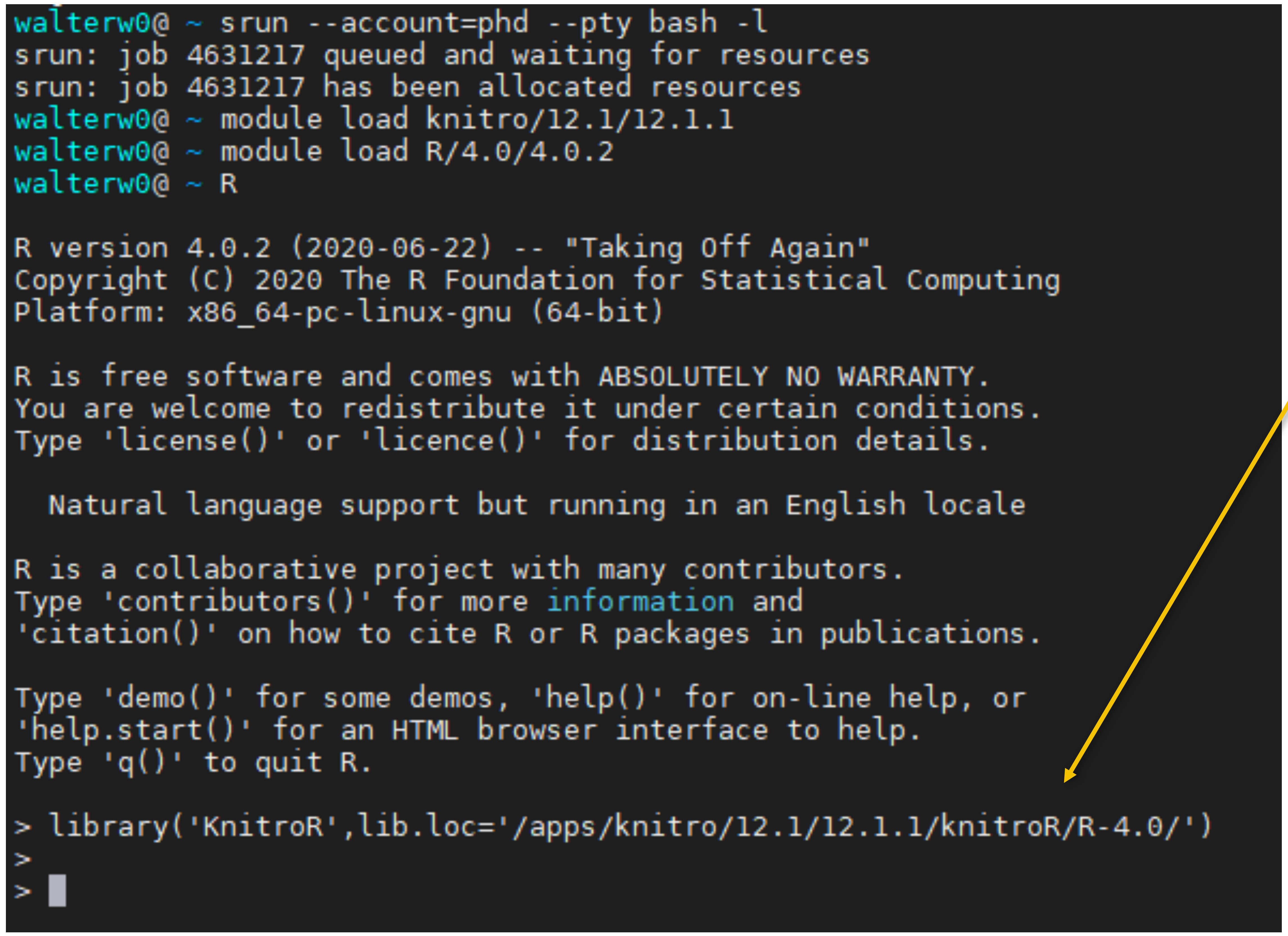


```
walterw0@ ~ R

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> require(gurobi)
Loading required package: gurobi
Loading required package: slam
Error: package or namespace load failed for ‘gurobi’:
 package ‘gurobi’ was installed before R 4.0.0: please re-install it
> 
```

CHICAGO BOOTH

As of Autumn 2021, both R versions (3.6.2/4.0.2) work with gurobi

R + Knitro



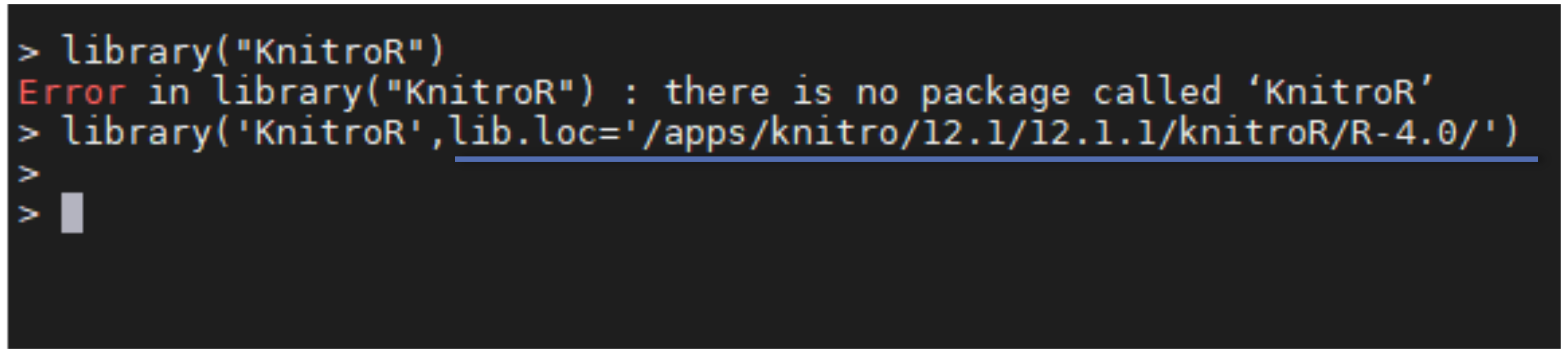
```
walterw0@ ~ srun --account=phd --pty bash -l
srun: job 4631217 queued and waiting for resources
srun: job 4631217 has been allocated resources
walterw0@ ~ module load knitro/12.1/12.1.1
walterw0@ ~ module load R/4.0/4.0.2
walterw0@ ~ R

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library('KnitroR',lib.loc='/apps/knitro/12.1/12.1.1/knitroR/R-4.0/')
>
>
```

Different versions are important for software dependence

- Knitro 12.1 and R v4.0 is needed
- The `KnitroR` package needs to be loaded by specifying its library location

```
library('KnitroR’, lib.loc='/apps/knitro/12.1/12.1.1/knitroR/R-4.0/')
```

Without specifying the library location, it doesn’t work



```
> library("KnitroR")
Error in library("KnitroR") : there is no package called ‘KnitroR’
> library('KnitroR',lib.loc='/apps/knitro/12.1/12.1.1/knitroR/R-4.0/')
>
>
```

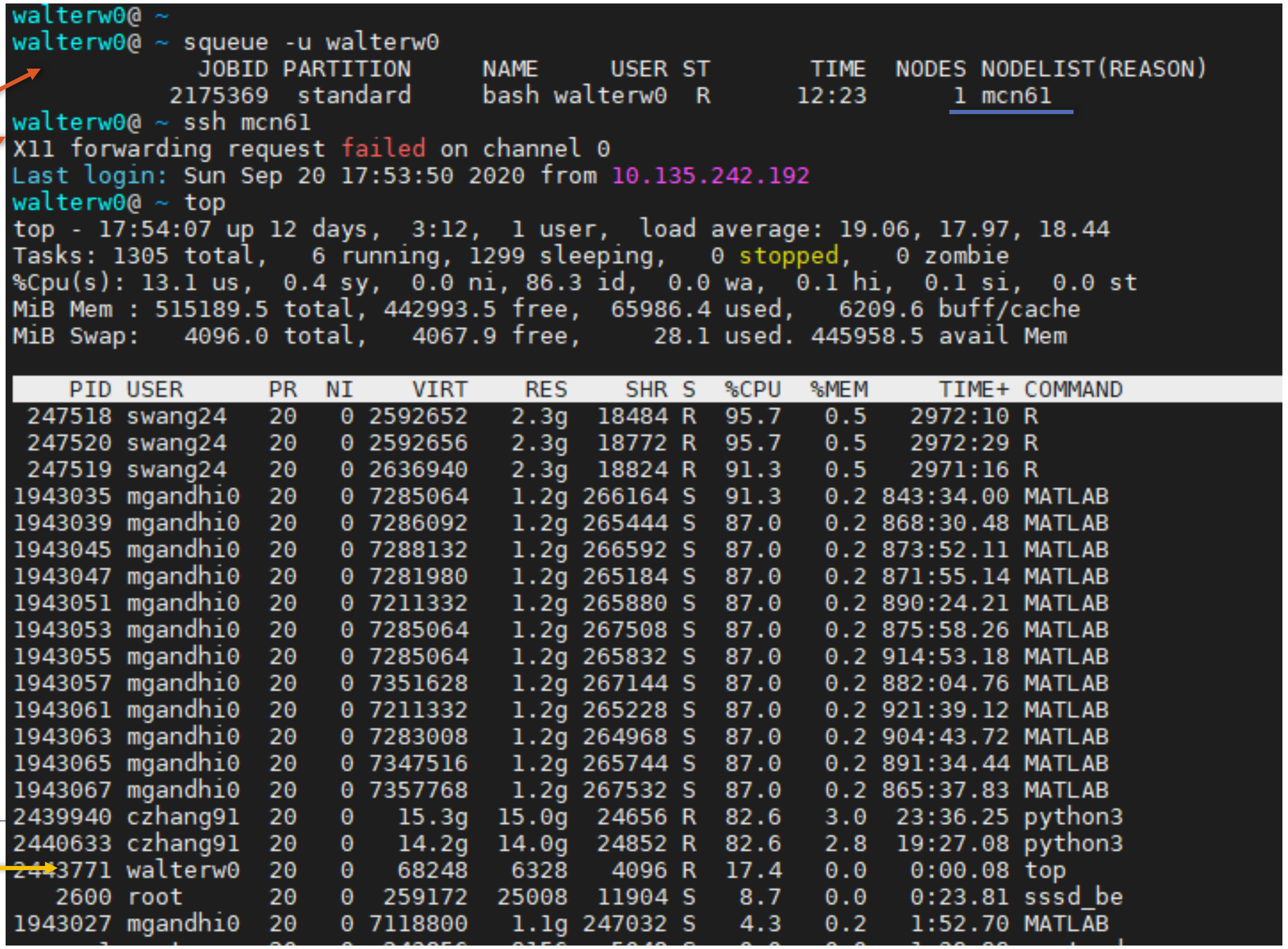
As of Autumn 2021, not specifying the library location for Knitro works

Tip 1: Checking on Jobs

When we have a job running, we can `ssh` *directly* to compute node to see the job

- Without the extant job you cannot `ssh` directly
- `ssh` session terminated when job ends

1. Find the job's node with `squeue`
2. `ssh` directly to the node
3. View the job(s) using `top` or `htop` 👀



```
walterw0@ ~
walterw0@ ~ squeue -u walterw0
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           2175369  standard     bash walterw0  R      12:23      1 mcn61
walterw0@ ~ ssh mcn61
X11 forwarding request failed on channel 0
Last login: Sun Sep 20 17:53:50 2020 from 10.135.242.192
walterw0@ ~ top
top - 17:54:07 up 12 days,  3:12,  1 user,  load average: 19.06, 17.97, 18.44
Tasks: 1305 total,   6 running, 1299 sleeping,   0 stopped,   0 zombie
%Cpu(s): 13.1 us,  0.4 sy,  0.0 ni, 86.3 id,  0.0 wa,  0.1 hi,  0.1 si,  0.0 st
MiB Mem : 515189.5 total, 442993.5 free,  65986.4 used,   6209.6 buff/cache
MiB Swap:   4096.0 total,   4067.9 free,     28.1 used. 445958.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 247518 swang24   20   0 2592652   2.3g  18484 R  95.7   0.5   2972:10 R
 247520 swang24   20   0 2592656   2.3g  18772 R  95.7   0.5   2972:29 R
 247519 swang24   20   0 2636940   2.3g  18824 R  91.3   0.5   2971:16 R
1943035 mgandhi0  20   0 7285064   1.2g 266164 S  91.3   0.2 843:34.00 MATLAB
1943039 mgandhi0  20   0 7286092   1.2g 265444 S  87.0   0.2 868:30.48 MATLAB
1943045 mgandhi0  20   0 7288132   1.2g 266592 S  87.0   0.2 873:52.11 MATLAB
1943047 mgandhi0  20   0 7281980   1.2g 265184 S  87.0   0.2 871:55.14 MATLAB
1943051 mgandhi0  20   0 7211332   1.2g 265880 S  87.0   0.2 890:24.21 MATLAB
1943053 mgandhi0  20   0 7285064   1.2g 267508 S  87.0   0.2 875:58.26 MATLAB
1943055 mgandhi0  20   0 7285064   1.2g 265832 S  87.0   0.2 914:53.18 MATLAB
1943057 mgandhi0  20   0 7351628   1.2g 267144 S  87.0   0.2 882:04.76 MATLAB
1943061 mgandhi0  20   0 7211332   1.2g 265228 S  87.0   0.2 921:39.12 MATLAB
1943063 mgandhi0  20   0 7283008   1.2g 264968 S  87.0   0.2 904:43.72 MATLAB
1943065 mgandhi0  20   0 7347516   1.2g 265744 S  87.0   0.2 891:34.44 MATLAB
1943067 mgandhi0  20   0 7357768   1.2g 267532 S  87.0   0.2 865:37.83 MATLAB
2439940 czhang91  20   0   15.3g  15.0g  24656 R  82.6   3.0  23:36.25 python3
2440633 czhang91  20   0   14.2g  14.0g  24852 R  82.6   2.8  19:27.08 python3
2443771 walterw0  20   0   68248   6328   4096 R  17.4   0.0   0:00.08 top
   2600 root      20   0  259172  25008  11904 S   8.7   0.0   0:23.81 sssd_be
1943027 mgandhi0  20   0 7118800   1.1g 247032 S   4.3   0.2   1:52.70 MATLAB
```

Tip 2: Alias repeated commands

Every time we want an interactive session, we've run:

```
srun --account=phd --pty bash –l
```

- We can make a shortcut or alias for this by defining a new command `sinteractive`

```
alias sinteractive='srun --account=phd --pty bash –l’
```

  - Use the single quotation mark: `

- We can make this alias permeant by putting it in your `~/.bash_profile` file

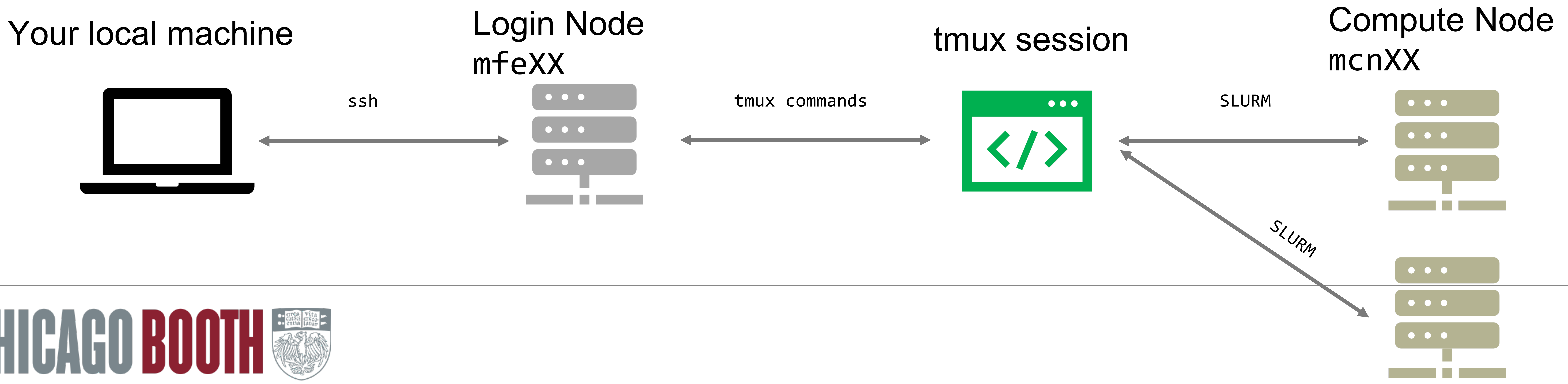
```
echo "alias sinteractive='srun --account=phd --pty bash -l'" >> ~/.bash_profile
```

  - This line *permanently* saves the shortcut to your `~/.bash_profile` file

- Can also do this with other commands

Tip 3: Keep your interactive session running

- `tmux` and `screen` are two “window managers” for your terminal
  - Using them on the login node can let you keep an interactive session running even after exiting the terminal (saves your workplace)
  - Requires some start-up cost to learn all the keyboard commands
  - `tmux` will be on login node mfe01 or mfe02
    - You can `ssh` between the login nodes, so you will always use the same one



Questions so far?

CHICAGO BOOTH

Lab: Bootstrap Standard Errors for OLS

We want to code up the following program using:

1. Serial job (1 job using 1 core)
2. Parallel job (1 job using 2 cores)
3. Array job (2 jobs using 1 core each)

- Suggested languages for the procedure (R/Python/Julia/MATLAB)
- Use the "guided-lab" folder in the Tutorial as your workspace
- Solutions in R are in the solutions folder

- Use your favorite text editor (vim/emacs/nano) to create/edit the files on the server
  - Alternatively use: rstudio.chicagobooth.edu or jupyter.chicagobooth.edu

Lab: Bootstrap Standard Errors for OLS

- **Goal**: We want to estimate non-parametric Bootstrap standard errors for OLS
  - We can then compare them the standard OLS standard errors

- **Setting**:
  1. Simulate 50 covariates (X1, …, X50) ~ N(0,1) and error term ϵ ~ N(0,1) iid with 100,000 observations
  2. Define true coefficients (β1, …, β50) = (1, …, 50)
  3. Construct Y = X’β + ϵ

$$Y = X'\beta + \epsilon$$

$$Y = \begin{bmatrix} X_1 & \dots & X_{50} \end{bmatrix}' \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_{50} \end{bmatrix} + \epsilon$$

$$Y = \begin{bmatrix} X_1 & \dots & X_{50} \end{bmatrix}' \begin{bmatrix} 1 \\ \vdots \\ 50 \end{bmatrix} + \epsilon$$

Lab: Bootstrap Standard Errors for OLS

Procedure: We have (Y, X1, …, X50) from the set up as our data with P = 50 covariates and N = 100,000 observations

1. We choose B = 1,000 total bootstrap iterations to run
2. For *b* from 1 to B:
    1. Sample with replacement (Y, X1, …, X50) to get (Y^b, $X1^b$, …, $X50^b$)
    2. Run OLS of Y^b on ($X1^b$, …, $X50^b$) to get estimates ($\beta1^b$, …, $\beta50^b$)
    3. Save ($\beta1^b$, …, $\beta50^b$)
3. Compute means and standard errors across bootstrap iterations for our bootstrap estimates

$$\hat{\beta}_p^{boot} = \frac{1}{B}\sum_{b=1}^{B}\beta_p^b, \qquad \forall p \in \{1, \ldots, P\}$$

$$se(\hat{\beta}_p^{boot}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\beta_p^b - \hat{\beta}_p^{boot})}$$

CHICAGO BOOTH

Lab: Bootstrap Standard Errors for OLS

Algorithm Pseudocode

```
Setup:
  1. Set the random seed and the true $\beta = (1, \ldots, 50)$ values
  2. Simulate the data $(X_1 \ldots, X_P, \epsilon)$
  3. Construct $Y = X'\beta + \epsilon$ and our data is $(Y, X_1, \ldots, X_P)$
Procedure:
  1. For bootstrap iteration $b$ in $1 : B$
      (a) Construct bootstrapped data $(Y^b, X_1^b \ldots, X_P^b)$ by sampling with replacement
      (b) Run OLS of $Y^b$ on $(X_1^b \ldots, X_P^b)$
      (c) Save the coefficients $(\beta_1^b, \ldots, \beta_P^b)$
  2. Estimate the bootstrap estimates and standard errors for each $p \in \{1, \ldots, P\}$
```

$$\hat{\beta}_p^{boot} = \frac{1}{B} \sum_{b=1}^{B} \beta_p^b,$$

$$se(\hat{\beta}_p^{boot}) = \sqrt{\frac{1}{B-1} \sum_{b=1}^{B} (\beta_p^b - \hat{\beta}_p^{boot})}$$

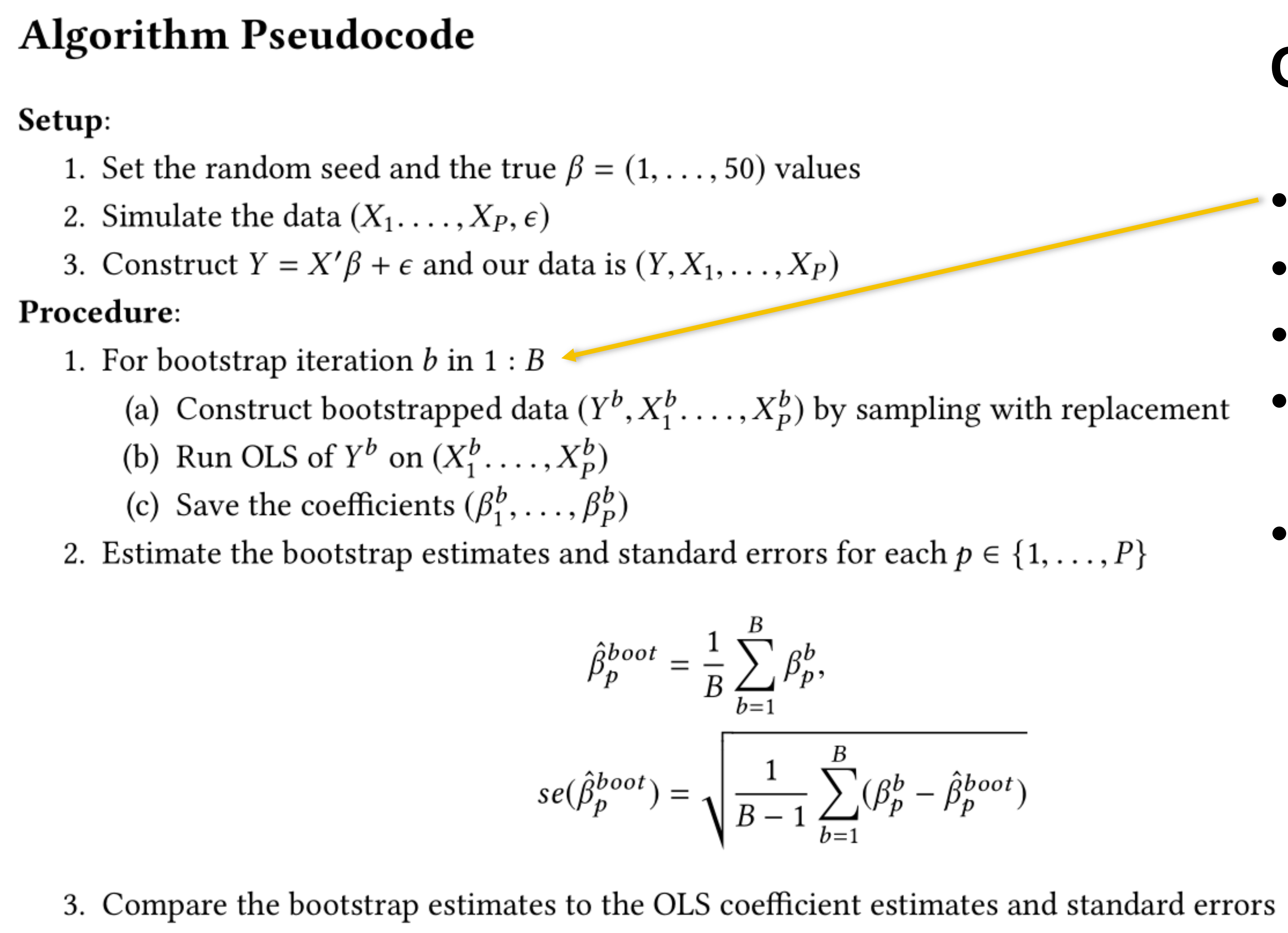
```
  3. Compare the bootstrap estimates to the OLS coefficient estimates and standard errors
```

Questions to think about:

- Which steps can be parallelized?
- How many cores do I need?
- How much memory do I need?
- How much wall time should I request?
- Do I need to store my results, or can I just print them?

CHICAGO BOOTH

Lab: Bootstrap Standard Errors for OLS



Algorithm Pseudocode

```
Setup:
  1. Set the random seed and the true $\beta = (1, \ldots, 50)$ values
  2. Simulate the data $(X_1 \ldots, X_P, \epsilon)$
  3. Construct $Y = X'\beta + \epsilon$ and our data is $(Y, X_1, \ldots, X_P)$
Procedure:
  1. For bootstrap iteration $b$ in $1 : B$
      (a) Construct bootstrapped data $(Y^b, X_1^b \ldots, X_P^b)$ by sampling with replacement
      (b) Run OLS of $Y^b$ on $(X_1^b \ldots, X_P^b)$
      (c) Save the coefficients $(\beta_1^b, \ldots, \beta_P^b)$
  2. Estimate the bootstrap estimates and standard errors for each $p \in \{1, \ldots, P\}$

        $\hat{\beta}_p^{boot} = \frac{1}{B} \sum_{b=1}^{B} \beta_p^b,$

        $se(\hat{\beta}_p^{boot}) = \sqrt{\frac{1}{B-1} \sum_{b=1}^{B} (\beta_p^b - \hat{\beta}_p^{boot})}$

  3. Compare the bootstrap estimates to the OLS coefficient estimates and standard errors
```

Questions to think about:

- Which steps can be parallelized?
- How many cores do I need?
- How much memory do I need?
- How much wall time should I request?
- Do I need to store my results, or can I just print them?

Bootstrap loop

Try coding it up on your local machine and make a guess

Printing the coefficients and standard errors to the console or saving them to a csv file is fine

Lab: Bootstrap Standard Errors for OLS

Assignments:

Code these in your favorite language and submit a batch SLURM job

- Write (1) a SLURM batch script and (2) an estimation script for each assignment

1. **Serial job** (1 job using 1 core)
   - Just follow the pseudocode
2. **Parallel job** (1 job using 2 cores)
   - Bootstrap loop in the estimation script is parallelized (`for b in 1:B`)
3. **Array job** (2 jobs using 1 core each)
   - Split up the serial job to two sub-jobs (first sub-job for the first 500 bootstrap iterations)
   - Use the environment variable (`$SLURM_ARRAY_TASK_ID`)

- **Hints**:
  - I. Use 1 GB of memory and 5 minutes of wall time on your batch jobs
  - II. Recycle your estimation script from the serial job for the parallel and the array jobs
  - III. If you get stuck see the `solutions` folder (solutions are in R)

Algorithm Pseudocode

```
Setup:
  1. Set the random seed and the true $\beta = (1, \ldots, 50)$ values
  2. Simulate the data $(X_1 \ldots, X_P, \epsilon)$
  3. Construct $Y = X'\beta + \epsilon$ and our data is $(Y, X_1, \ldots, X_P)$
Procedure:
  1. For bootstrap iteration $b$ in $1 : B$
      (a) Construct bootstrapped data $(Y^b, X_1^b \ldots, X_P^b)$ by sampling with replacement
      (b) Run OLS of $Y^b$ on $(X_1^b \ldots, X_P^b)$
      (c) Save the coefficients $(\beta_1^b, \ldots, \beta_P^b)$
  2. Estimate the bootstrap estimates and standard errors for each $p \in \{1, \ldots, P\}$
```

$$\hat{\beta}_p^{boot} = \frac{1}{B}\sum_{b=1}^{B}\beta_p^b,$$

$$se(\hat{\beta}_p^{boot}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\beta_p^b - \hat{\beta}_p^{boot})}$$

```
  3. Compare the bootstrap estimates to the OLS coefficient estimates and standard errors
```

Last thoughts or questions?