



# ilifu Online Training – Advanced slurm

Theophilus Matsepane

System Administrator & User Support, ilifu  
University of the Western Cape, March 2026



UNIVERSITY of the  
WESTERN CAPE

- Interactive Jobs
  - Persistent Terminals (`tmux`)
- Advanced Slurm monitoring:
  - How busy is the cluster? (`sinfo`)
  - Where are my jobs in the queue? (`squeue`)
  - Which jobs have I submitted in the past and what happened? (`sacct`)
- Advanced Slurm job submission:
  - Slurm Arrays
  - Slurm Job Dependencies



# What is slurm?



- **Slurm** (Simple Linux Utility for Resource Management)
  - a highly scalable, open-source workload manager
  - and job scheduler for Linux clusters
- Manages compute nodes
  - `scontrol`
- Allocates resources for jobs
  - `sbatch`, `srun` (`sinteractive`)
- Handles queues
- Widely used in HPC
- It ensures efficient resource sharing and job execution

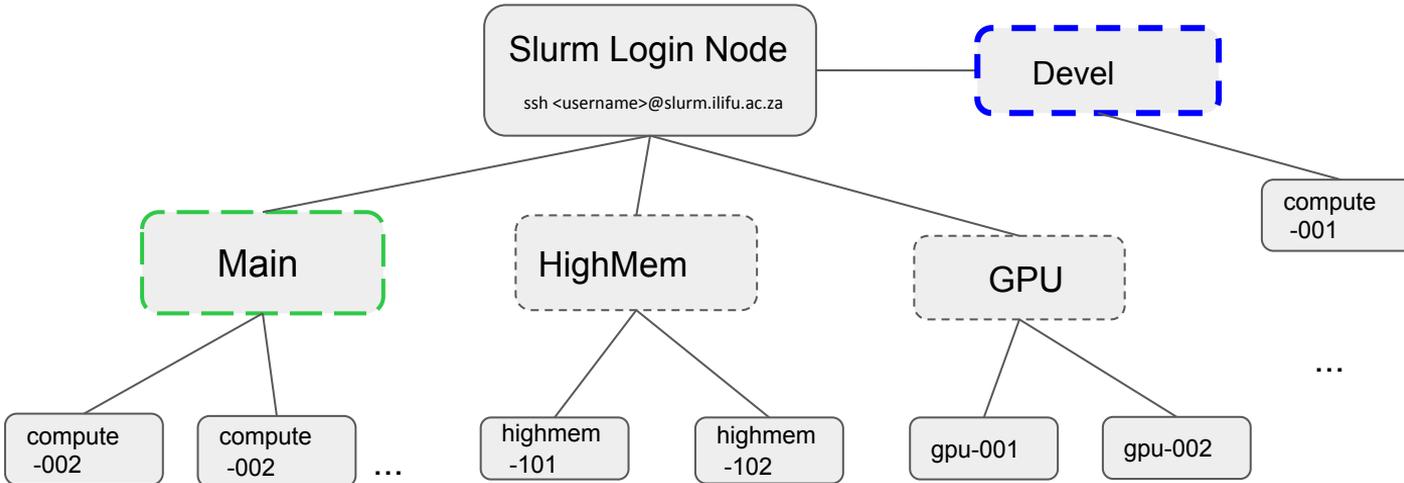


# SLURM

[http://docs.ilifu.ac.za/#/tech\\_docs/running\\_jobs?id=specifying-resources-when-running-jobs-on-slurm](http://docs.ilifu.ac.za/#/tech_docs/running_jobs?id=specifying-resources-when-running-jobs-on-slurm)



Main	HighMem	GPU	Devel
~85 nodes	8 nodes	7 nodes	1 node
32 CPUs / 232 GB RAM	32 CPUs / 480 GB RAM	2 GPUs / 232 GB RAM	32 CPUs / 237 GB RAM



# SLURM - running an interactive job



Login to Slurm Login Node

```
$ ssh <username>@slurm.ilifu.ac.za
```

```
$ sinteractive
```

- Devel partition

```
theo@slurm-login:~$ sinteractive
Starting interactive Slurm session.
srun: job 12963504 queued and waiting for resources
srun: job 12963504 has been allocated resources
theo@compute-001:~$
```

The **sinteractive** script is a **wrapper script** we created for calling Slurm's **srun** command with useful default parameters



```
$ srun --pty bash
```

- Main partition

```
srun -p Main --pty bash
```

- Main partition

```
theo@slurm-login:~$ srun --pty bash
srun: job 12963497 queued and waiting for resources
srun: job 12963497 has been allocated resources
theo@compute-006:~$
```



# SLURM - running an interactive job



Login to Slurm Login Node

```
$ ssh <username>@slurm.ilifu.ac.za
```

Defaults: 1 CPU (Central Processing Unit) for 3 hours:

```
$ sinteractive -c 1 -time 03:00
```

Example: 5 CPU for 5 days (maximum):

```
$ sinteractive -c 5 -time 5-00:00
```

Note: All resources on the Devel partition are shared. Including CPU and memory.

For interactive jobs requiring dedicated resources, please use: **srun** on the Main partition



# SLURM - running an interactive job (x11 support)



Login to Slurm Login Node

```
$ ssh -Y <username>@slurm.ilifu.ac.za
```

```
$ sinteractive --x11
```

Testing X11 Support:

```
$ xmessage 'hello world'
```

OR

```
$ xeyes
```

Windows users:

Install X11 server:

e.g. Xming or Vcxsrv

```
theo@slurm-login:~$ sinteractive --x11
Starting interactive Slurm session.
srun: job 12829914 queued and waiting for resources
srun: job 12829914 has been allocated resources
theo@compute-001:~$ xmessage 'hello world'
█
```

A screenshot of a terminal window with a dark background. The terminal shows the execution of 'sinteractive --x11' and 'xmessage 'hello world''. Below the terminal output, a standard X11 window titled 'xmessage' is displayed. The window has a white background and a grey border. It contains the text 'hello world' in a monospaced font. At the bottom left of the window is an 'okay' button. The window title bar includes three colored buttons (red, yellow, green) and the text 'xmessage'.

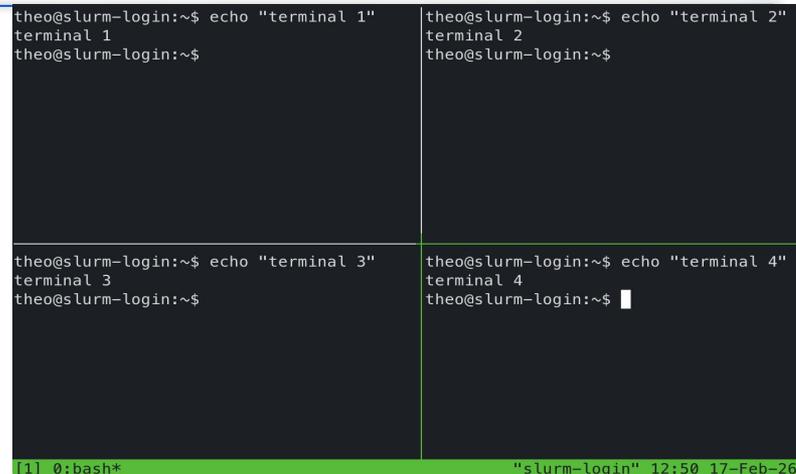
## tmux — terminal multiplexer

- Persistent Connections
- Runs Multiple Terminals

Note: Persistent terminals can be lost if the Slurm-login node had to be restarted

## Commands

- ```
$ tmux # start a new tmux session (usually on slurm-login)
$ tmux attach # Attach to running session
$ tmux ls # List active sessions
$ tmux --help # show help
```



```
theo@slurm-login:~$ echo "terminal 1"
terminal 1
theo@slurm-login:~$

theo@slurm-login:~$ echo "terminal 2"
terminal 2
theo@slurm-login:~$

theo@slurm-login:~$ echo "terminal 3"
terminal 3
theo@slurm-login:~$

theo@slurm-login:~$ echo "terminal 4"
terminal 4
theo@slurm-login:~$

[1] 0: bash* "slurm-login" 12:50 17-Feb-26
```



### Keyboard shortcuts

**Ctrl+b** d # detach session (can reconnect)

**Ctrl+b** x # kill terminal pane

**Ctrl+b** % # split the screen horizontally

**Ctrl+b** " # split the screen vertically

**Ctrl+b** <arrow key> # switch to the pane in arrow direction

Ilifu docs Reference:

[https://docs.ilifu.ac.za/#/tech\\_docs/running\\_jobs?id=persistent-terminals](https://docs.ilifu.ac.za/#/tech_docs/running_jobs?id=persistent-terminals)

Video Tutorial (Jeremy Howard): <https://youtu.be/0pWjZByJ3Lk?t=2474>



DEMO TIME!



- Interactive Jobs
- Persistent Terminals (`tmux`)
- Advanced Slurm monitoring:
  - How busy is the cluster? (`sinfo`)
  - Where are my jobs in the queue? (`squeue`)
  - Which jobs have I submitted in the past and what happened? (`sacct`)
- Advanced Slurm job submission:
  - Slurm Arrays
  - Slurm Job Dependencies



| <b>Slurm Commands</b>  | Before Running a Job | Starting a New Job | Monitoring a Running Job | After Jobs have Completed |
|------------------------|----------------------|--------------------|--------------------------|---------------------------|
| <code>sinfo</code>     | X                    |                    |                          |                           |
| <code>squeue</code>    | X                    |                    | X                        |                           |
| <code>sbatch</code>    |                      | X                  |                          |                           |
| <code>srun</code>      |                      | X                  |                          |                           |
| <code>scontrol*</code> |                      |                    | X                        |                           |
| <code>sacct</code>     |                      |                    |                          | X                         |

\* Shows all information about a pending/running job:  
\$ `scontrol show job <jobID>`

# sinfo: How busy is the cluster?



```
$ sinfo
```

```
PARTITION AVAIL TIMELIMIT NODES STATE
Main*      up 14-00:00:0      1 drain*
Main*      up 14-00:00:0     19  mix
Main*      up 14-00:00:0     17  alloc
Main*      up 14-00:00:0     48  idle
```

List partitions and their specs

```
$ sinfo -o "partition,available,cpus,nodes,memory,statecompact"
```

| PARTITION  | AVAIL | CPUS | NODES | MEMORY  | STATE  |
|------------|-------|------|-------|---------|--------|
| Main*      | up    | 32   | 1     | 237568  | drain* |
| Main*      | up    | 32+  | 19    | 237568+ | mix    |
| Main*      | up    | 32   | 17    | 237568+ | alloc  |
| Main*      | up    | 32+  | 48    | 237568+ | idle   |
| Jupyter    | up    | 32   | 1     | 237568  | drng   |
| Jupyter    | up    | 32   | 6     | 237568  | mix    |
| Jupyter    | up    | 32   | 2     | 237568  | alloc  |
| Jupyter    | up    | 32   | 1     | 237568  | idle   |
| JupyterGPU | up    | 32   | 1     | 237568  | mix    |
| JupyterGPU | up    | 32   | 1     | 237568  | alloc  |
| HighMem    | up    | 96   | 1     | 1544192 | mix    |
| HighMem    | up    | 32   | 2     | 515604  | idle   |
| GPU        | up    | 24   | 1     | 237568  | down*  |
| GPU        | up    | 32   | 3     | 237568  | mix    |
| GPU        | up    | 32   | 1     | 237568  | alloc  |
| GPU        | up    | 48   | 2     | 362582  | idle   |
| GPUV100    | up    | 24   | 1     | 237568  | down*  |
| Devel      | up    | 32   | 1     | 237568  | alloc  |

For a list of sinfo columns:

```
$ man sinfo
```



# queue: Where is my job in the queue?



```
$ squeue -u $USER
```

| JOBID    | PARTITION | NAME     | USER | ST | TIME    | NODES | NODELIST(REASON) |
|----------|-----------|----------|------|----|---------|-------|------------------|
| 12848095 | Jupyter   | jupyter- | theo | R  | 1:49:30 | 1     | jupyter-002      |
| 12846866 | Main      | combined | theo | PD | 0:00    | 1     | (Priority)       |
| 12846867 | Main      | combined | theo | PD | 0:00    | 1     | (Dependency)     |
| 12846868 | Main      | astromet | theo | PD | 0:00    | 1     | (Dependency)     |

```
$ squeue -u $USER -o JobID,Partition,State,NumCPUs,MinMemory,NodeList,ReasonList
```

| JOBID   | PARTITION | STATE   | CPUS | MIN_MEMORY | NODELIST(REASON) |
|---------|-----------|---------|------|------------|------------------|
| 9402788 | Devel     | RUNNING | 1    | 0          | compute-001      |
| 9402721 | Main      | PENDING | 30   | 3096M      | (Resources)      |
| 9402719 | Main      | RUNNING | 30   | 3096M      | compute-260      |

```
$ squeue -u $USER --start
```

| JOBID    | PARTITION | NAME     | USER | ST | START_TIME          | NODES | SCHEDNODES  | NODELIST(REASON) |
|----------|-----------|----------|------|----|---------------------|-------|-------------|------------------|
| 12901511 | Main      | combined | theo | PD | 2026-02-27T03:08:07 | 1     | compute-205 | (Reservation)    |

# sacct: Which jobs have been submitted in the past?



Show jobs submitted by the user in the past 1 day:

```
$ sacct -u $USER
```

Show one line per job (combines job steps):

```
$ sacct -u $USER -X
```

```
theo@slurm-Login:~/demo/casa_interactive$ sacct -u $USER
JobID      JobName    Partition  Account  AllocCPUS      State  ExitCode
-----
12879817   jupyter-1  Jupyter    b11-vlbi-+  1      RUNNING  0:0
12879817.ba+  batch      b11-vlbi-+  1      RUNNING  0:0
12879817.ex+  extern     b11-vlbi-+  1      RUNNING  0:0
12879817.0    batchspaw+ b11-vlbi-+  1      RUNNING  0:0
12900801    astrometry Main        b11-vlbi-+  16     COMPLETED  0:0
12900801.ba+  batch      b11-vlbi-+  16     COMPLETED  0:0
12900801.ex+  extern     b11-vlbi-+  16     COMPLETED  0:0
```

Filter on jobs submitted between a start date and an end date

```
$ sacct -u $USER -X -S 2025-03-03 -E 2026-03-03
```

Show specific columns:

```
$ sacct -u $USER -X -S 2025-03-03 -E 2026-03-03
```

```
-o Jobid,JobName,NCPU,NNodes,Start,End,Elapsed,TimeLimit,State
```

Nicer formatting of columns e.g. %-15 is a column with width of 15 and a left justification

```
$ sacct -u $USER -X
```

```
-o Jobid,JobName%-15,NCPU,NNodes,Start,End,Elapsed,TimeLimit,State%20
```

# sacct: Which jobs have been submitted in the past?



Example for showing how long jobs ran to help set the Wall time (TimeLimit)

```
$ sacct -u $USER -X  
-o Jobid,JobName%-15,NCPU,NNodes,Start,End,Elapsed,TimeLimit,State%20
```

| JobID   | JobName         | NCPUS | NNodes | Start               | End                 | Elapsed  | TimeLimit  | State              |
|---------|-----------------|-------|--------|---------------------|---------------------|----------|------------|--------------------|
| 9402635 | transfer_MS_15+ | 0     | 1      | None                | 2024-03-25T04:39:59 | 00:00:00 | 4-00:00:00 | CANCELLED by 10028 |
| 9402636 | partition       | 23    | 1      | 2024-03-25T04:45:05 | 2024-03-25T04:45:24 | 00:00:19 | 00:30:00   | FAILED             |
| 9402682 | partition       | 23    | 1      | 2024-03-25T06:53:06 | 2024-03-25T07:02:18 | 00:09:12 | 00:30:00   | COMPLETED          |
| 9402683 | validate_input  | 1     | 1      | 2024-03-25T07:02:18 | 2024-03-25T07:02:26 | 00:00:08 | 00:30:00   | COMPLETED          |
| 9402684 | flag_round_1    | 16    | 1      | 2024-03-25T07:02:26 | 2024-03-25T07:17:58 | 00:15:32 | 00:30:00   | OUT_OF_MEMORY      |
| 9403040 | partition       | 23    | 1      | 2024-03-25T08:13:18 | 2024-03-25T08:23:19 | 00:10:01 | 00:30:00   | COMPLETED          |
| 9403041 | validate_input  | 1     | 1      | 2024-03-25T08:23:20 | 2024-03-25T08:23:44 | 00:00:24 | 00:30:00   | COMPLETED          |
| 9403042 | flag_round_1    | 32    | 1      | 2024-03-25T08:23:44 | 2024-03-25T08:52:29 | 00:28:45 | 00:30:00   | COMPLETED          |
| 9403043 | setjy           | 23    | 1      | 2024-03-25T08:52:29 | 2024-03-25T09:20:13 | 00:27:44 | 00:30:00   | COMPLETED          |
| 9403044 | xx_yy_solve     | 1     | 1      | 2024-03-25T09:20:13 | 2024-03-25T09:42:30 | 00:22:17 | 00:30:00   | COMPLETED          |
| 9403045 | xx_yy_apply     | 23    | 1      | 2024-03-25T09:42:30 | 2024-03-25T09:55:40 | 00:13:10 | 00:30:00   | COMPLETED          |
| 9403046 | flag_round_2    | 32    | 1      | 2024-03-25T09:55:40 | 2024-03-25T10:25:45 | 00:30:05 | 00:30:00   | TIMEOUT            |

# sacct: Which jobs have been submitted in the past?



Example for showing how much resources jobs used

e.g. MaxRSS = Maximum RAM usage in bytes

Note: Only on job-step level

```
$ sacct -u $USER -S 2024-03-25 -E 2025-03-25 -o  
Jobid%-15, JobName%-15, NCPU, NNodes, Start, MaxRSS, MaxDiskRead, MaxDiskWrite, State%20
```

| JobID          | JobName      | NCPUS | NNodes | Start               | MaxRSS     | MaxDiskRead | MaxDiskWrite | State     |
|----------------|--------------|-------|--------|---------------------|------------|-------------|--------------|-----------|
| 9403045        | xx_yy_apply  | 23    | 1      | 2024-03-25T09:42:30 |            |             |              | COMPLETED |
| 9403045.batch  | batch        | 23    | 1      | 2024-03-25T09:42:30 | 171434320K | 799330.48M  | 458872.78M   | COMPLETED |
| 9403045.extern | extern       | 23    | 1      | 2024-03-25T09:42:30 | 172K       | 0.01M       | 0.00M        | COMPLETED |
| 9403046        | flag_round_2 | 32    | 1      | 2024-03-25T09:55:40 |            |             |              | TIMEOUT   |
| 9403046.batch  | batch        | 32    | 1      | 2024-03-25T09:55:40 | 452567280K | 276836.01M  | 15385.23M    | CANCELLED |
| 9403046.extern | extern       | 32    | 1      | 2024-03-25T09:55:40 | 200K       | 0.01M       | 0.00M        | COMPLETED |

- After / during running jobs :
  - jobId is given from sbatch output / squeue

Shows info about job running including working directory

```
$ scontrol show job <jobID>
```

Shows squeue with start time (%S)

```
$ squeue --start -u $USER
```

Shows info for multi-CPU jobs

```
$ sacct -o JobID%-15,JobName%-15,Partition,Account,  
Elapsed,NNodes%6,NTASK%6,NCPUS%5,MaxDiskRead,MaxDiskWrite,  
NodeList%20,MaxRSS,CPUTime,State,ExitCode
```

Shows jobs started and completed between these dates

```
$ sacct -S 2021-09-01-09:00 -E 2021-09-14-10:00 -X -o  
Jobid,JobName,Start,End,State
```



- Email parameters

```
$ sbatch --mail-user=<address> --mail-type=<event_types>  
- Events : BEGIN,END,FAIL,TIME_LIMIT_80
```

- Exclude nodes
  - e.g. problematic nodes (report to ilifu support)

```
$ srun/sbatch --exclude=compute-[101,103-105]
```

- Other input parameters

```
$ srun/sbatch --help
```



- Specify lower wall-time (default 3 hours) and less memory (default ~7GB) increases chance of job launching immediately
- In steps:

```
$ srun --pty --time=10 --mem=1GB bash
$ singularity shell /idia/software/containers/python-3.6.img
$ python3 job_script.py
```

- In single call:

```
$ srun --pty --time=10 --mem=1GB singularity exec
/idia/software/containers/python-3.6.img python3 job_script.py
```

- Must manually process after this

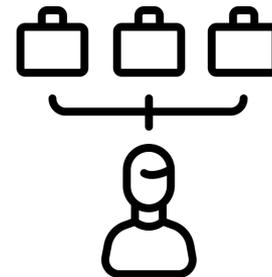


# Topics

- Interactive Jobs
- Persistent Terminals (`tmux`)
- Advanced Slurm monitoring:
  - How busy is the cluster? (`sinfo`)
  - Where are my jobs in the queue? (`squeue`)
  - Which jobs have I submitted in the past and what happened? (`sacct`)
- Advanced Slurm job submission:
  - Slurm Arrays
  - Slurm Job Dependencies



- Array jobs allow quick submission of many similar jobs, each using the same resources, without any manual launch
- Passes array task ID into script, which changes behavior of each job each time i.e different inputs
- Can be used to run many related steps simultaneously in a serial process
- Example job array Running, 20 jobs, with 5 run concurrently.



```
#!/bin/bash
#SBATCH --array=1-20%5
#SBATCH --job-name=myarrayjob
#SBATCH --output=logs/%x-%A_%a.out
#SBATCH --error=logs/%x-%A_%a.err
```

```
module load python/3.11.0
```

```
python myscript.py --input $SLURM_ARRAY_TASK_ID
```



| Parameter                                | Substitution / filename pattern | Environment Variables |
|------------------------------------------|---------------------------------|-----------------------|
| jobID of running job                     | %j                              | SLURM_JOB_ID          |
| Job name                                 | %x                              | SLURM_JOB_NAME        |
| Job array's master job allocation number | %A                              | SLURM_ARRAY_JOB_ID    |
| Job array task ID (index) number         | %a                              | SLURM_ARRAY_TASK_ID   |

For example:

```
#SBATCH --array=1-20%5  
#SBATCH --job-name=myarrayjob  
#SBATCH --output=logs/%x-%A_%a.out
```

With submitted first job id: 1000

%j = 1000,1001...,1019

%x = myarrayjob

%A = 1000

%a = 1,2,3,...,19,20 (order not guaranteed)



# SLURM Job dependencies



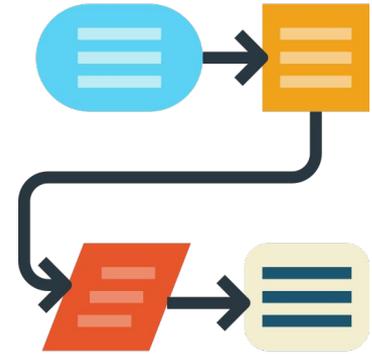
- Allow jobs to be scheduled for running, based on the status of a previous job
  - e.g only begin a particular job once previous one successfully completes

Submit `another_job.sh` to SLURM queue, to begin after jobID 1234 successfully completes , or cancel the job if jobID 1234 fails

```
$ sbatch -d afterok:1234 --kill-on-invalid-dep=yes another_job.sh
```

Submit `another_job.sh` to SLURM queue, to begin after jobID 1234 & 5678 completes

```
$ sbatch -d afterany:1234:5678 another_job.sh
```



| Dependency | Meaning              | When New Jobs starts                     |
|------------|----------------------|------------------------------------------|
| after      | After started        | Once job begins running                  |
| afterany   | After any completion | After job finishes (success/fail/cancel) |
| afterok    | After successful     | Only if job succeeds                     |
| afternotok | After failed         | Only if job fails                        |



- **Cancel pending or running jobs**
  - `scancel <jobid>`, e.g. `scancel 12956381` to cancel a single or an array job  
**12956381**
  - `scancel -u <username>`: **Cancels all jobs belonging to a user**



DEMO TIME!



## Do's :

- Run jobs using sbatch rather than interactive jobs
- Identify job resources requirements:
  - No. of nodes and CPUs, amount of RAM and wall-time.
- Remove files that aren't needed
  - /scratch3 folder after data processing is complete
  - Old raw data, temporary products , etc.
- Use Singularity ( cannot install software on nodes)
- Use **username@transfer.ilifu.ac.za** for data transfers

## Don't:

- Don't run software/heavy processes on login node
- Don't place large files in your home directory (/users)
- Don't transfer using scp/rsync on the login node



# Thank you!

**Remember our support channels!**

[support@ilifu.ac.za](mailto:support@ilifu.ac.za)  
<https://docs.ilifu.ac.za>

