# Cheat Sheet

Preprint, compiled March 26, 2025

**immediate**

## 1 Getting help

| Email | Description |
|---|---|
| help@sharcnet.ca | For SHARCNET specific issues |
| accounts@tech.alliancecan.ca | Questions about accounts |
| renewals@tech.alliancecan.ca | Questions about account renewals |
| globus@tech.alliancecan.ca | Questions about Globus file transfer services |
| cloud@tech.alliancecan.ca | Questions about using Cloud resources |
| allocations@tech.alliancecan.ca | Questions about the Resource Allocation Competition |
| support@tech.alliancecan.ca | For any other questions or issues |

## 2 Training courses

| URL | QR code |
|---|---|
| https://training.sharcnet.ca/ | |
| https://training.computeontario.ca/ | |

## 3 Connecting to nibi

| Method | URL/Command |
|---|---|
| From a browser | https://jupyterhub.nibi.sharcnet.ca/ |
| From command line | `ssh username@nibi.sharcnet.ca` |
| To cloud | https://cloud.nibi.alliancecan.ca/ |
| To transfer large data | https://globus.alliancecan.ca/ |

## 4 Using cluster nibi

### *4.1 Using modules*

| Command | Description |
|---|---|
| `module avail` | To list all available modules. Check this link https://docs.alliancecan.ca/wiki/Available_ for a complete list of installed modules |
| `module list` | To list preloaded modules |
| `module spider keyword` | To search for a module by keyword |
| `module load foo[/ver]` | To load module `foo [version ver]` |

### *4.2 Most commonly used Linux commands*

| Command | Description |
|---|---|
| `ls` | List files and directories in the current directory |
| `cd` | Change directory, e.g. |
| `cd DIR` | Go to directory `DIR` |
| `cd` | Go back to home directory |
| `cd ..` | Go back to the previous directory |
| `pwd` | Show current directory |
| `mkdir` | Make directories, e.g. |
| `mkdir dir1[ dir2[ ...]]` | Make directories |
| `mkdir -p path/to/dir` | Make directory recursively |
| `cp source dest` | Copy files |
| `mv source dest` | Move or rename files and directories |
| `find` | Find a file or directory that matches certain criteria |
| `du -sh` | Find the disk usage |
| `man command` | See the manual page of command |
| `quota` | Find the disk quota |

### *4.3 Slurm commands*

The Slurm scheduler has a rich set of commands, one needs to refer to the Slurm documentation for details. The following is a list of commonly used Slurm commands:

To submit a job using job submission script `submit_ajob.sh`

```
sbatch submit_ajob.sh
```

To see the history of my jobs, use command `sacct`, with options (Check the Slurm documentation or the man page of `sacct`)

```
sacct -j jobid
sacct -u $USER starttime t1 endtime t2
sacct -u $USER -o ReqCPUs,ReqMem,NNodes,Starttime,Endtime
```

To cancel a jobs

```
scancel jobid
```

To see the system information

```
sinfo
```

To your queued jobs

```
squeue -u $USER
```

To see the fairshare

```
sshare
```

To see the epilogue of a job

```
seff jobid
```

To allocate cores and/or nodes and use them interactively

```
salloc account=def -my_group_account ntasks=32 time=1:00
salloc account=def -my_group_account mem=0 nnodes=1
```

One needs to create a job submission script per job. It is a Shell script. The following is a sample script, named `submit_ajob.sh`:

### *4.4 Sample script for submitting a serial job*

```
#!/bin/bash
#SBATCH - -time=00 -01:00:00 # DD -HH:MM
#SBATCH - -account=my_group_account
module load python/3.6
python simple_job.py 7 output
```

To submit the job, run the following command

```
sbatch submit_ajob.sh
```

### *4.5 Sample script for submitting multiple jobs*

```
#!/bin/bash
#SBATCH - -time=01:00
#SBATCH - -account=my_group_account
#SBATCH - -array=1 -200
python simple_job.py $SLURM_ARRAY_TASK_ID output
```

### *4.6 Sample script for submitting multicore threaded jobs*

```
#!/bin/bash
#SBATCH - -account=my_group_account
#SBATCH - -time=0 -03:00
#SBATCH - -cpus -per -task=32
#SBATCH - -ntasks=1
#SBATCH - -mem=20G
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
./myprog.exe
```

### *4.7 Sample script for submitting multiprocess parallel jobs*

```
#!/bin/bash
#SBATCH - -account=my_group_account
#SBATCH - -time=5 -00:00
#SBATCH - -ntasks=100
#SBATCH - -mem -per -cpu=4G
srun ./mympiprog.exe
```

### *4.8 Sample script for submitting a GPU job*

```
#!/bin/bash
#SBATCH - -account=my_group_account
#SBATCH - -time=0 -03:00
#SBATCH - -gpus -per -node=h100:2
#SBATCH - -mem=20G
./myprog.exe
```

### *4.9 Sample script for submitting a hybrid MPI-threaded job*

```
#!/bin/bash
#SBATCH - -account=my_group_account
#SBATCH - -time=0 -03:00
#SBATCH ntasks=16       # MPI ranks
#SBATCH cpus -per -task=4 # threads
#SBATCH - -mem=20G
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
srun cpus -per -task=$SLURM_CPUS_PER_TASK ./myprog
```

### *4.10 Requesting scheduling jobs by node*

```
A sample submission script:
#!/bin/bash
#SBATCH - -account=my_group_account
#SBATCH - -time=0 -03:00
#SBATCH ntasks=16      # MPI ranks
XXXXXXXXXXXXXXXXXXXX
```

**Tip**

- Requesting more resources (runtime, CPU cores, memory) than what the job process requires can result in a longer queue times.
- The recent usage of an account is calculated independently on each of the Alliance general purpose systems and the availability of the resources varies across systems.
- More resources are available to full-node jobs. If your job can efficiently use multiples of 32 cpu cores (graham) it gains access to a larger set of nodes if it is submitted as a full-node job.

- Less than 20% of all resources are available via default accounts. If a project needs more than the default level usage, a larger target share of the system can be obtained through the annual Resources Allocation Competition (RAC)

## 5 USING PYTHON

To create a virtual environment with NumPy as example Python package

```
module load python/3.12
virtualenv --no-download ~/ENV
source ~/ENV/bin/activate
pip install --no-index --upgrade pip
pip install --no-index numpy
```

If you need a specific version of a module, then install it like this:

```
pip install --no-index numpy==1.26.4
```

To flag `--no-index` will install a package from our wheelhouse. These are always preferable to installing from the internet as they will be tuned to run on our systems.

To see the available wheels for a particular version of Python, use

```
avail_wheels numpy --all_versions -p 3.12
```

or see https://docs.alliancecan.ca/wiki/Available_Python_wheels for a list. If you don't see a wheel that you need, please submit a ticket and we will install it.

**Note**
Some Python packages are provided as software modules so they are not installed with `pip`. Please see the link above for a list of those modules.

## 6 USING APPTAINER

Some packages are difficult to install in our Linux environment. The alternative is to install them in a container. Here is an example with Anaconda and Numpy.

Create file `image.def` with

```
Bootstrap: docker
From: mambaorg/micromamba:latest

%post
    micromamba install -c conda-forge numpy
```

Build image with

```
module load apptainer
apptainer build image.sif image.def
```

Run python in image with

```
apptainer run image.sif python
```

**Hint**
Add `--nv` flag if you want to use GPUs.

## 7 ADVANCED RESEARCH COMPUTING RESOURCES ACROSS CANADA

| Cluster | Cores | GPUs | Max memory | Storage |
|---|---|---|---|---|
| **fur**<br>**nibi**<br>**trillium**<br>**rorqual**<br>**narval** | 165,120 | 640 | | 49TB<br>25TB |

## 8 SPECIFICATION OF NIBI

| Nodes | Cores | Memory | CPU | GPU |
|---|---|---|---|---|
| 700 | 192 | 768GB DDR5 | 2 x Intel 6972P @ 2.4 GHz, 384MB cache L3 | |
| 10 | 192 | 6TB DDR5 | 2 x Intel 6972P @ 2.4 GHz, 384MB cache L3 | |
| 36 | 192 | 1.5TB | 1 x Intel 8570 @ 2.1 GHz, 300MB cache L3 | 8 x Nvidia H100 SXM (80 GB memory) |
| 6 | 96 | 512GB | 4 x AMD MI300A @ 2.1GHz | 4 x AMD CDNA 3 (128 GB HBM3 memory - unified memory model) |