Using Cedar

Simple Terminal Commands on Cedar

Command	Result
`ls`	List out all items in current directory
`cd <directory path>`	Change directory to `<directory path>`
`mkdir <new directory path>`	Makes a directory at `<new directory path>`
`rm <file path>`	Removes file at `<file path>`
`rm -r <directory path>`	Removes directory at `<directory path>` and all its contents
`cp <file path> <new file path>`	Copies a file from `<file path>` to `<new file path>`
`cp -R <directory path> <new directory path>`	Copies a directory from `<directory path>` to `<new directory path>`
`mv <file path> <new file path>`	Moves a file from `<file path>` to `<new file path>`
`mv -R <directory path> <new directory path>`	Moves a directory from `< directory path>` to `<new directory path>`
`nano <file path>`	Edit file at `<file path>` if it exists, or create and edit file at `<file path>` if it does not
`module avail python`	Check available python versions on Cedar
`module load python/<version>`	Activate python with the version `<version>`
`python -m venv <new directory path>`	Create a python virtual environment with the currently active version of python in a directory at `<new directory path>`
`source <venv directory>/bin activate`	Activate the virtual environment stored in `<venv directory>`
`deactivate`	Deactivate current virtual environment
`pip install <module name>`	Install the python module `<module name>` in the current virtual environment
`python <file path>`	Runs the file at `<file path>` in current virtual environment if that file is a python file
`clear`	Clear the terminal
`exit`	Exit Cedar

Logging in:

1. Open terminal on Mac or Linux. There's probably a Windows alternative, but I don't know what.
2. Type in "`ssh <your username>@cedar.computecanada.ca`".
3. Type in your Compute Canada password when prompted.
4. Hit ENTER.

Navigating:

You will log into your home directory. In the home directory you should have at least a `projects` directory and a `scratch` directory.

`scratch` is a directory with huge storage capability, but it gets deleted automatically every once in a while, so use it only to store temporary very large files.

In `projects`, you should see a directory called `def-<sponsor username>` (so for Jess, `def-jlmciver`), then inside of this directory, you should find a directory with your username. It can take a few minutes to hours after you get added to the group for this directory to show up.

The directory with your username has a reasonably large memory and doesn't get deleted.

I would recommend never transferring directories directly from your computer to Cedar, since directories will not automatically adjust their owner, meaning they'll use up your personal (very small) storage rather than the memory of your username directory. Instead, only transfer files to Cedar.

Copying Files to/from Cedar:

To copy a file from your computer to Cedar, type in
`scp <path to copy from> <username>@cedar.computecanada.ca:<path destination>`
Usually `<path destination>` will be something like `~/projects/<sponsor username>/<your username>`.
Then type your password when prompted, and wait until the transfer is complete.

To copy a file from Cedar to your computer, type in
`scp <username>@cedar.computecanada.ca:<path to copy from> <path destination>`
This time, `<path destination>` should be a directory you want to copy to on your computer.
Then type your password when prompted, and wait until the transfer is complete.

Running Jobs on Cedar:

Typically, to run a job on Cedar, you make a job script. The job script is a text file with a ".sh" extension. It starts with the line `#!/bin/bash`, then has several directives of the job requested at the start, with `#SBATCH` in front, and finally a series of regular linux commands.

For example, a simple job script, `simple_job.sh`, is given here:

```
#!/bin/bash

#SBATCH --account=def-jlmciver
#SBATCH --time=1:00

echo "Hello World!"
```

This script requests 1 minute on any Cedar computation node with the account `def-jlmciver`. Once it starts, it prints `"Hello World!"` to a file in the same directory as the job script. By default, the file will be called `slurm-<job number>.out`, although you can change this with the job directive `--output`.

Here's a list of some of the directives I found most useful. A full list can be found here, and some Cedar-specific information can be found here.

Directive	Result
`--account`	***Required to run a job.*** The account with which to run this job.
`--time`	***Required to run a job.*** The amount of time this job will be allowed to run before being killed. Jobs with smaller times will be started sooner.
`--mail-type`	Lets you choose what kinds of email updates to receive, from NONE, BEGIN, END, FAIL, REQUEUE, ALL. Default is no mail.
`--mail-user`	The email address to which Cedar will send email updates.
`--mem`	The amount of memory required for the job in MB. Default is 256 MB. Can request memory amounts with other SI prefixes, for instance, if you want 100 GB, you can enter `--mem=100G`
`--cpus-per-task`	The number of CPUs to allocate to the job. Default of 1. If a task is parallelizable, I highly recommend parallelizing it and requesting 32 or the maximum of 48 CPUs, since jobs that use their entire node get priority on Cedar (each node has either 32 or 48 CPUs). It is a violation of the Cedar code of conduct to request large CPU counts without using them though, so make sure your program will use every CPU.
`--output`	Path to a file (that doesn't need to exist yet) that will get all the command line output from the job (std out and std err).

Here's an example of a more complete job script, which prints out a message that it's starting, activates a python virtual environment, then runs a python script to search over a 3-dimensional, 200000000-point grid for maxima of a function, using 48 CPUs running in parallel over the course of 3 days. It also send me email updates whenever the job starts, stops, or runs into an error.

```
#!/bin/bash
#SBATCH --account=def-jlmciver
#SBATCH --time=70:00:00
#SBATCH --cpus-per-task=48
#SBATCH --mail-user=my_email@cool_person.ca
#SBATCH --mail-type=ALL
#SBATCH --mem=100G

echo "Starting Job"
source SpritzEnv/bin/activate
python maximal_gridsearch.py
```

Once you have a job script, you can submit it to Cedar by typing "`sbatch <job script path>`" into the Cedar terminal.

To see jobs you have either queued or running on Cedar, you can run the "`sq`" command.
To cancel all jobs you have submitted, you can use "`scancel -u $USER`", or if you want to cancel only jobs that haven't started yet, you can use "`scancel -t PENDING -u $USER`".
Finally, to cancel a specific job, you can find its job id with "`sq`", and then use "`scancel <job id>`".