RUNNING YOUR FIRST "ARRAY" OF TASKS ON DELLA

1. Get Della access

Write request for access to Princeton Computational Science and Engineering Support:
https://researchcomputing.princeton.edu/systems/della#access

If it is the first time someone from your group is using Della you'll need your PI to write a short "sponsor" statement. If someone from your group has used Della before, your PI just needs to give approval (easiest to just CC them in the email you write to cses@princeton.edu)

Note it may take a couple hours after the admin puts you in the system before all the permissions get set up and you can actually use the cluster as normal.

2. Test that your access works and you can both use the GUI as well as SSH into your account

Either/both of the following will throw an error if your access hasn't been granted yet.
(Replace "mynetid" with your actual princeton net id).

2A. Through Web GUI (which is surprisingly decent):

Navigate to https://mydella.princeton.edu

Then click on Files and select /scratch/gpfs/mynetid

This is the directory where you'll keep your data, scripts, etc.

(Note: this is not the directory where you'll land if you SSH)

2B. Via terminal:

```
# from your local machine
$ ssh mynetid@della.princeton.edu
# then after you enter your password (note: your normal princeton password)
[mynetid@della]$ cd /
[mynetid@della /]$ cd scratch
[mynetid@della scratch]$ cd gpfs
[mynetid@della gpfs]$ cd mynetid
```

3. Write and run a test script on the cluster

Note: this is not an array job yet, AND it is not running on the compute nodes, which is where we will normally run big jobs. This is just to make sure everything works. Specifically, the following will run some simple tasks

on the “head node” AKA “login node”. In general you don’t want to do anything fancy here, but it is useful to test that software is working, etc.

write a script on your local machine

for pythonistas
make a file on your local machine, call it `test_script.py` and add something simple like
`print('hello della')`

for matlaborers
make a file on your local machine, call it `test_script.m` and add
`disp('hello della');`

Now login to the WebGUI mydella.princeton.edu and upload the file to your `/scratch/gpfs/mynetid` folder

now ssh into della
from a terminal on your local machine (you can also click “open terminal” from the Web GUI)
`$ ssh mynetid@della.princeton.edu`
move to the directory where you put your script
`[mynetid@della mynetid]$ cd /scratch/gpfs/mynetid`
list the files in the directory
`[mynetid@della mynetid]$ ls`

In order to run our script we need to have access to python or MATLAB.
On the cluster software is already installed under the hood and summoned via “modules”

E.g. to load anaconda 3 version 2021.5 you would call
`[mynetid@della mynetid]$ module load anaconda3/2021.5`

Or to load MATLAB R2019a
`[mynetid@della mynetid]$ module load matlab/R2019a`

To see all possible modules run
`[mynetid@della mynetid]$ module avail`

Or to see specific modules starting with e.g. “anaconda3” run
`[mynetid@della mynetid]$ module avail anaconda3`

Now that our software is loaded we can run it.
`[mynetid@della mynetid]$ python test_script.py`

or if you used MATLAB (note that the flags that suppress the GUI being loaded, etc.)
`[mynetid@della mynetid]$ matlab -singleCompThread -nodisplay -nosplash -r test_script`

It should print out “hello della”.

More stuff about modules and which ones to load for common software etc here:
https://researchcomputing.princeton.edu/support/knowledge-base/modules

4. Write a test script that can be run as an array job

In the last section we executed a script on the head node. Typically we will want to run several tasks i.e. several copies of the same script on the compute nodes, with each copy e.g. using a different parameter set.

When we run several tasks, each task has an ID, which we need to access from within our script. The task ID is stored in an environment variable and we need to access it from within Python or MATLAB.

Within a script it is useful to store this in a variable called something like “task_id”. We can then use it to access, say, a parameter we want to use for our script.

If using Python, make a script called `test_array_job.py` which contains the lines

```
task_id = int(os.environ['SLURM_ARRAY_TASK_ID'])

all_params = [0, 1, 10, 100]
param = all_params[task_id]

print('Hello! The ID of this task is')
print(task_id)
print('The parameter value is')
print(param)
```

The equivalent MATLAB file is (remember btw that MATLAB indexes from 1)

```
task_id = str2num(getenv('SLURM_ARRAY_TASK_ID'));

all_params = [0, 1, 10, 100];
param = all_params(task_id+1);

disp('Hello! The ID of this task is');
disp(task_id);
disp('The parameter value is');
disp(param);
```

Then upload the script file to your `/scratch/gpfs/mynetid` directory on Della.

5. Write the slurm file that will start up an array of tasks

This is the file that specifies how many tasks to run, how much memory to allocate per task, the max run time, where the output and error messages get written to, etc.

Make a file called `test_array_job.slurm` and fill in the following.

```
#!/bin/bash
#SBATCH --job-name=test_array_job        # create a short name for your job
#SBATCH --nodes=1                # node count
#SBATCH --ntasks=1               # total number of tasks across all nodes
#SBATCH --cpus-per-task=1     # cpu-cores per task (>1 if multi-threaded tasks)
#SBATCH --mem-per-cpu=1G         # memory per cpu-core (4G is default)
#SBATCH --time=00:10:00          # total run time limit (HH:MM:SS)
#SBATCH --array=0-3              # job array with index values 0, 1, 2, 3
#SBATCH -o my_outputs%A.%a.out          # stdout is redirected to that file
#SBATCH -e my_errors%A.%a.err          # stderr is redirected to that file
#SBATCH --mail-type=all
#SBATCH --mail-user=mynetid@princeton.edu

module purge
module load anaconda3/2021.5

python test_array_job.py
```

The equivalent MATLAB file is

```
#!/bin/bash
#SBATCH --job-name=test_array_job        # create a short name for your job
#SBATCH --nodes=1                # node count
#SBATCH --ntasks=1               # total number of tasks across all nodes
#SBATCH --cpus-per-task=1     # cpu-cores per task (>1 if multi-threaded tasks)
#SBATCH --mem-per-cpu=1G         # memory per cpu-core (4G is default)
#SBATCH --time=00:10:00          # total run time limit (HH:MM:SS)
#SBATCH --array=0-3              # job array with index values 0, 1, 2, 3
#SBATCH -o my_outputs%A.%a.out          # stdout is redirected to that file
#SBATCH -e my_errors%A.%a.err          # stderr is redirected to that file
#SBATCH --mail-type=all
#SBATCH --mail-user=mynetid@princeton.edu

module purge
module load matlab/R2019a

matlab -singleCompThread -nodisplay -nosplash -r test_array_job
```

Then upload the slurm file to your `/scratch/gpfs/mynetid` directory on Della.

(You can also make edits to your files directly on Della using the Web GUI.)

6. Test and monitor the slurm file

This is pretty easy

```
[mynetid@della mynetid]$ sbatch test_array_job.slurm
```

We can monitor the status with

```
[mynetid@della mynetid]$ squeue -u mynetid
```

If PD it means the job is pending (often it takes several minutes before it starts running). If the status is PD the content under “Reason” will tell you why (e.g. Resources means it is waiting for resources, Priority means it is waiting for higher priority jobs to finish.)

If R it is running

If CG it is ending.

You can cancel a job by finding its id (using squeue, where the ID is everything before _ if it is an array job).
```
[mynetid@della mynetid]$ scancel job_id
```

7. Check whether the job ran successfully

If everything worked, there should be 4 *.out and 4 *.err files (since we specified `--array=0-3`) in your /scratch/gpfs/mynetid directory.

The *.out files should contain the print/disp statements.
The *.err files will show any errors that arose, so check here if the scripts did not run correctly to see whether you made any syntax errors, etc.

In general you will probably want to have your scripts save results files in some directory in your scratch area (eg `/scratch/gpfs/mynetid/my_simulation_results`), then you can just download them from the Web GUI.