Using the Cluster

The cluster is basically a fast comptuer system that is used for running performance-heavy code. This guide is a brief introduction to using the cluster. Throughout the tutorial, wherever you see `<USERNAME>`, replace it with your USC username!

Step One: Logging In

To log into the cluster, you use the `ssh` function:

```
ssh <USERNAME>@discovery.usc.edu
```

To set up your ssh key, you can run the following (note that I haven't tried it yet):

```
ssh-copy-id <USERNAME>@discovery.usc.edu
```

If you don't want to type this line of code all the time, you can run this following line of code once, then restart your terminal so that logging in is faster:

```
echo "alias sshpc='ssh <USERNAME>@discovery.usc.edu'" >> ~/.bashrc
```

When you log in, you'll be put into your home directory, i.e. `/home1/<USERNAME>`. You generally don't want to be putting files here, so you should `cd` to your `project` directory:

```
cd /project/qcb_640/<USERNAME>
```

Note that if you're doing computationally-heavy research with another PI, you'll probably have a separate project directory for their work -- go check in with them! However, for your course, you can just put everything into your `qbc_640` folder.

Step Two: Uploading Scripts

To upload data to the cluster, we can use `scp` in a similar way as we did before:

```
scp python_script.py <USERNAME>@hpc-transfer1.usc.edu:/project/qcb_640/<USERNAME>
```

However, our destination directory is a bit more complicated. To explain:

- `<USERNAME>@hpc-transfer1.usc.edu` is the "computer" that we're accessing. When you're moving files around your own computer, it's assumed that it's all local on your machine. However, in this case, we need to specify.
- `/project/qcb_640/<USERNAME>` is the directory we want to copy to.
- The `:` just separates the "computer" and the path.

Step Three: Running Scripts

To use the high-performance aspect of the cluster, we use the "slurm" commands. The three main commands you need to know are:

- `salloc`: this opens a node in "interactive" mode.
- `sbatch`: this sends a job to be run off to the side.

- `squeue` : this allows you to monitor the jobs you've sbatch'ed.

Make sure that your scripts are in your project directory!

salloc

"Interative mode", from a functional perspective, has the exact same functionality as a regular terminal instance. It just moves you over to another computer. So why do we use `salloc`? The reason is that when we're on discovery (i.e. right after logging in), everyone is using it to handle files and send jobs off to be run. However, if everyone is running code on discovery, then we can actually slow it down or even crash it. **Therefore, we should ALWAYS use `salloc` before running code.**

`salloc` is mostly used for testing code to see if it runs properly; you can see terminal outputs as it runs. To use it, just run `salloc` (the default parameters are fine):

```
salloc
```

Once you run `salloc`, you actually need to load in a few functions before you can actually run scripts. The reason for this behavior is that there's a *lot* of programs that you might want to run, and it would be very slow to load them all. To load them in, use the `module load` commands:

```
module load gcc/11.2.0 openblas/0.3.18 python/3.9.6 r/4.1.2
module load nano
```

From now on, we can run our Python/R scripts as usual. To exit the `salloc` session, just type CTRL+D.

If you don't want to write this line every time you log into the cluster, ask David! It's a bit involved.

sbatch

`sbatch` is used for running large scripts, where we send it off to another computer to crunch on data. This means we can do other work or log off the cluster while the code runs. To use `sbatch`, we have prepared a sample script file to run your scripts (called `run_script`):

```
#!/bin/bash

#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --mem=16GB
#SBATCH --time=1:00:00
#SBATCH --account=qcb_640

module purge
module load gcc/11.2.0 openblas/0.3.18 python/3.9.6 r/4.1.2

python3 script.py
```

The only line you have to pay attention to is the last one, where you can change it to run a different script. To send a job off, just change your `run_script` file and pass it to `sbatch`.

```
sbatch run_script
```

Once you run this line of code, you can use `squeue` to see how the job is running:

```
squeue --user=<USERNAME>
```

You can see how long it's been running for, as well as some other info. Once it's done, it'll disappear and you'll see a log file.

Installing Packages

Installing packages is the one exception to using `salloc`, since these compute nodes don't have internet. In R, you can install packages as usual in R (`install.packages()` and `BiocManager::install()`). In Python, you'll have to load in `conda` and/or `pip` with `module load`. You should install these packages in your home directory (in R, you'll probably see a prompt saying you can't install a directory. Say ok, then the next default file path should be fine).

Wrapping Up

To log off, just type CTRL+D.