Run Slurm jobs in a cluster

1. Upload your code using `rsync`:

```
local$ rsync your_folder_name username@clustername.edu
```

or the cool version,

```
  local$ rsync -r -rav -e ssh --include '*/' --include='*'.{m,bash,sh,txt} \
  -p --exclude='*' your_folder_name/ username@clustername.edu
```

2. Connect with the cluster using `ssh`:

```
local$ ssh username@clustername.edu
```

3. Submit your job with `sbatch`:

```
cluster$ sbatch run.bash
```

or the cool version:

```
  cluster$ sbatch --job-name=your_job_name --output=your_job_name.o%j \
  -p partition_name run.bash
```

The extra parameters give you extra control for your job.

4. Check your job in the queue:

```
cluster$ squeue
```

or the cool version,

```
cluster$ squeue -o '%9i %.7P %.20j %.13u\
%.8Q %.2t %.10M %.5D %.4C %42N %32Y %r'
```

The file `run.bash`

All variables needed by Slurm have to be defined either in step 3 or in the file `run.bash`.

Moreover, the main routine of your code `main.[py,m,cpp,etc..]` is called in `run.bash`.

The routine `main.[py,m,cpp,etc..]` is responsible for parallelizing your jobs. It is up to you to neatly code how your processes will be parallelized.

Below is a minimal example of `run.bash` for a Matlab job.

```
#!/bin/bash
#SBATCH -n 1                                              # run one process
#SBATCH --exclusive                                       # dont share node
#SBATCH --cpus-per-task=20                                # how many cpus?
#SBATCH -t 0-11:59:00                                     # run time (d-hh:mm:ss)
#SBATCH --mail-type=END,FAIL
#SBATCH --mail-user=youremail@here.edu
# -- set memory limits
ulimit -v unlimited
ulimit -s unlimited
ulimit -u 10000
# -- load from available modules
module load matlab/r2018b
# -- run job
matlab -nodisplay -nodesktop -r "run ./main.m ; quit"
```

Things to keep in mind

- Keep your code memory-light. Typecast as many variables as you can.
- Keep your code as fast as you can. No unnecessary methods.
- Measure maximum amount of required time and memory for your job. This will specify wall-time and CPU memory in the node.
- Break up your code in space and time as much as you can. Computation speed depends on both time and space.
- Don't be selfish. Everybody likes playing with the cluster. Share.