Submitting a job (deprecated: see Slurm)

Once you have connected to the cluster, you can submit jobs to run on the cluster. A job has two parts: - A script describing what the job does - A description of the resources required to run the job - this describes things like the amount of memory and number of CPUs required by your job

The script should contain the commands required to run your job, and can be as simple or as complicated as you need. In the simplist case your script contains exactly the same commands that you would type on the command.

If you don't give a description of the resources, then the cluster will assume a default set of resources which is suitable for small jobs, but you should provide a more realistic estimate if you think your job requires more power. This is because the cluster assigns jobs to compute nodes based on these descriptions, and if you understate them, then the cluster may allocate more jobs that the node can handle and this will make your job run slower or even fail completely. Resource allocation is covered in more detail here.

Submitting jobs

After you first login to the cluster (as covered by the Connecting page) you are presented with a prompt something like the following:

```
[jbarber@submit ~]$
```

This is the command prompt. It shows you your username (jbarber), the name of the computer you're logged into (submit) and the directory you are currently in (~ this symbol represents your home directory, which is the directory under which all of your personal files are kept). You can list the files in the directory by typing the command ls and then pressing the Enter key. If you do this you should see something like this:

```
[jbarber@submit ~]$ ls
example1.sh
[jbarber@submit ~]$
```

example1.sh is a file on the cluster which is also the script which we will submit to the cluster using the qsub command:

```
[jbarber@submit ~]$ qsub example1.sh
37.maui.grid.fe.up.pt
```

The string 37.maui.grid.fe.up.pt is an unique identifier for this job. Now the job will be submitted to the management node and if possible it will be run.

Job status

After a job is submitted, we can see its status by using the *qstat* command:

```
[jbarber@submit ~]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
37.maui                    example1.sh      jbarber                0 R batch
```

Here we can see that the job is running (represented by the 'R' in the 'S' column). The *qstat* command shows all of your jobs in the system, if your only interested in a single job, you can specify the job id to restrict the output:

```
[jbarber@submit ~]$ qstat 37.maui
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
37.maui                    example1.sh      jbarber                0 R batch
```

After a short time, the job will complete and the display will change to:

```
[jbarber@submit ~]$ qstat 37.maui
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
37.maui                    example1.sh      jbarber         00:00:00 C batch
```

with the 'C' meaning 'complete'. The system only keeps a record of the job completing for a short period, after a while it will disappear. If you run the ls command again, you will see that two new files have appeared:

```
[jbarber@submit ~]$ cat example1.sh.o37
Hello World!
[jbarber@submit ~]$ cat example1.sh.e37
[jbarber@submit ~]$
```

We can see here that the example1.sh.o37 file contains the message 'Hello World!' and that the *example1.sh.e37* file is empty.

Anatomy of a script

We can also use cat to look at the script:

```
[jbarber@submit ~]$ cat example1.sh
#!/bin/bash
sleep 30
echo "Hello World!"
[jbarber@submit ~]$
```

This shows us several things. The first line *#!/bin/bash* is an instruction to the operating system to use the bash program to run the script. The second line sleep 30 is a command which simply does nothing for 30 seconds. The final line echo “Hello World!” is responsible for creating the output we saw in *example1.sh.o37* - echo just prints out its arguments. The bash program is the same program that we

have been running our commands in, so anything you type in your session can be put in a script and run. This is how we tell the cluster what to do - we put the commands we want to run in a script and submit it to the cluster, replacing sleep and echo with programs that do more useful work.

Interactive Jobs

As well as submiting your jobs through scripts, it's possible to run a shell as a job using what's called an interactive job. This gives you a shell on one of the cluster nodes, exactly as you have on the submit host. This is ideal for doing interactive tasks such as compiling new software or doing data analysis. To invoke a interactive shell, simply use the -I argument to qsub:

```
[jbarber@submit ~]$ qsub -I
qsub: waiting for job 63314.maui.grid.fe.up.pt to start
qsub: job 63314.maui.grid.fe.up.pt ready

[jbarber@avafat01 ~]$
```

Note that the hostname has changed in the prompt to *avafat01* (this is one of the cluster nodes). To finish the interactive job, quit the shell by typing exit.

Deleting and changing jobs

If you realize there is a problem after you've submitted a job, you can use the qdel command to stop the job:

```
[jbarber@submit ~]$ qsub  example1.sh
63238.maui.grid.fe.up.pt
[jbarber@submit ~]$ qdel 63238.maui.grid.fe.up.pt
[jbarber@submit ~]$ qstat 63238.maui.grid.fe.up.pt
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
63238.maui                 example1.sh      jbarber         00:00:00 C batch
[jbarber@submit ~]$
```

Alternatively, if you just want to modify the job you can use the *qalter* command. For details of the job properties that you can change, see the *qalter* man page on the submit host (*man qalter*)

PBS Script Generator

For your convenience there is a PBS Script Generator that you can use to build the shell script:

https://grid.fe.up.pt/web/index.php?page=pbs_script_generator

This tool will generate code just like this:

```
#!/bin/bash
###-----Comments can be here as long as they're preceded by (#) hash-----###
```

```
   #######------------and hash (#) is not followed by PBS-----------#######

###-----PBS Directives Start-----###
#PBS -V
#PBS -S /bin/bash
#PBS -N teste
#PBS -l nodes=4:ppn=10
#PBS -l walltime=24:00:00
#PBS -l mem=4GB
#PBS -q std
#PBS -M rmatos@fe.up.pt
#PBS -m e
#PBS -e localhost:${PBS_O_WORKDIR}/${PBS_JOBNAME}.e${PBS_JOBID}
#PBS -o localhost:${PBS_O_WORKDIR}/${PBS_JOBNAME}.o${PBS_JOBID}
###-----PBS Directives End-----###

############### Enter Shell/Execution Statements here ... ###############
echo "Hello World"
exit 0;
```

which you can conveniently copy-paste to the submit node.