Introduction to HPC Tutorials

Research Applications Team, IT Services, QMUL

Useful Links

- https://docs.hpc.qmul.ac.uk
- https://learn.hpc.qmul.ac.uk
- https://blog.hpc.qmul.ac.uk
- https://github.qmul.ac.uk

Published: October 04, 2023

Contact *its-research-support@qmul.ac.uk* for HPC support.

Tutorial 1 - Logging into the HPC

Objective:	To log into the HPC cluster using an ssh client.
Prerequisites:	A QMUL username.

Instructions:

1. Click on and follow the relevant instructions below for your installed operating system to launch a terminal within an ssh client:
   - Windows
   - Linux
   - MacOSX
2. Log into the HPC by executing:

   ```
   ssh abc123@login.hpc.qmul.ac.uk
   ```

This will attempt a login connection to the QMUL HPC cluster at address *login.hpc.qmul.ac.uk* for username *abc123*.

Note: Replace *abc123* with your QMUL username

Note: Your password will not be displayed on the screen as it is typed.

3. Once logged into the HPC, you should see the welcome message and a prompt similar to one of the following:

   ```
   [abc123@frontend8 ~]$
   ```

   or

   ```
   [abc123@frontend11 ~]$
   ```

   Verify you see the prompt with your username.

Windows Users

If you want to use the Windows 10 command prompt or PowerShell, you may skip this step because these programs contain a native SSH client.

- Download a suitable SSH client, for example MobaXterm (https://mobaxterm.mobatek.net/download-home-edition.html). Download the portable edition to get started but if you have rights to install software on your machine, you may also select the installer edition.
- Open the MobaXterm application and click the Start local terminal button.

Linux Users

Launch the in-built terminal application:

- Centos7: Applications > System Tools > Terminal.
- Ubuntu: Open the dash menu (Windows key) and type terminal.

MacOSX Users

- Launch the in-built terminal application by clicking the Finder icon in the dock > Go > Utilities > Terminal.

Tutorial 2 - Application Modules

Objective:	To view, load, unload and list available application modules.
Prerequisite:	An active connection to the HPC cluster.

Instructions:

1. List all the available software packages by executing:

   ```
   module avail
   ```

   Should the output of this command be paginated (a colon `:` will appear in the bottom left corner of the terminal instead of the prompt), simply press the `space` bar a few times, or press `q` to "quit".

2. List the available versions of *R* by executing:

   ```
   module avail R/
   ```

   If an application is not installed, you should see no output - try printing the available versions for the application "*not installed*".

3. Load the default version of the *R* application into your environment by running:

   ```
   module load R
   ```

   Note: To load a specific version of an application, use the full module name (application name and version).

4. List the currently loaded modules by running:

   ```
   module list
   ```

   The output will now show that R is loaded in your environment. Observe that some applications such as *R*, or *samtools* also load other modules as dependencies. These are also unloaded automatically during a `module unload` of the parent application.

5. Run `R --version` to confirm which version of R has been loaded. This is a built-in command within the R application, rather than a feature of modules. Other applications often display the version number in the output from the `-h` or `--help` options.

6. Now attempt to load the *R/4.2.0* module - note that a module conflict error appears. This is by design - you cannot have two versions of the same application loaded at once, and one needs to be unloaded first.

7. Let's unload the existing *R* module and then load version *R/4.2.0*:

   ```
   module unload R
   module load R/4.2.0
   ```

Note: You may instead use the `module switch R/4.2.0` command to achieve the same outcome in one step.

Run the `R --version` command to confirm that you are now running R version 4.2.0.

8. Unload all modules at once by executing:

   `module purge`

9. Confirm you have no modules loaded.

When requesting a new version or software package, you may be asked to test a development version before it is released. These test installations will be made available through development modules.

After executing `module load use.dev`, the development modules appear in the top of the `module avail` output and can be loaded as usual using the `module load` command.

Tutorial 3 - Batch Job Submission

Objective:	To create and submit a batch job to the HPC cluster.
Prerequisite:	A text editor (e.g. *vim*).

Many Linux users decide to use Vi or ViM (Vi IMproved) as a text editor because of its efficiency to insert, delete, find and replace text only using the keyboard. On the HPC cluster, `vim` is readily available on the frontend and cluster nodes. For more information on how to use Vi, click here.

There are other editors available such as Nano, which is available via a module on the HPC cluster. To use Nano, run `module load nano` before starting this tutorial. For more information on how to use Nano, click here.

Instructions:

1. Open a text editor to create a job script.
2. Add the following text and save the file as *example.sh*:

```
#!/bin/bash
#$ -cwd
#$ -j y
#$ -pe smp 1
#$ -l h_rt=1:0:0
#$ -l h_vmem=1G
#$ -m bea

hostname
date
sleep 2m
date
```

3. Submit the job script to the HPC cluster for processing by executing:

   `qsub example.sh`

4. Make a note of the job number shown in the output after submitting the job. This is a unique identifier which can be used to check the job status and consumed resources.
5. Check the status of your queued / running jobs by executing:

   `qstat`

   Note: Running `qstat -j JOBID` will provide a high level of detail about job JOBID - this should be replaced with the actual job number.
6. Once the job has completed running, confirm that it is no longer visible in the output of the `qstat` command. Examine the job output file, named `example.sh.oJOBID`, replacing JOBID with the actual job number.
7. View the contents of the job completion email to find helpful information such as exit status and resources used.
8. Confirm your job has completed by running `qstat` again - you should not see the job in the output.
9. Modify the resources in the *example.sh* script, increasing the core request to 4 and the **total** memory to 8GB. Add a job name with the `#$ -N NAME` parameter and repeat steps 3-5 and observe the change of output from `qstat`.

   Hint: The `h_vmem` setting is per core.

   If your job is queued, you may see the position in the whole cluster queue with our `showqueue` utility script.

10. Check your list of queued jobs with `qstat` and cancel any jobs submitted in this section that you no longer require.

    Note: you can cancel a queued / running job by running `qdel JOBID`.

Tutorial 4 - Interactive Job Submission

Objective:	To submit interactive jobs to the HPC cluster.
Prerequisite:	An active connection to the HPC cluster.

Jobs that request up to 1 hour will become eligible for the short queue and will often start running immediately. All other jobs may request up to 10 days runtime with the `h_rt=240:0:0` parameter.

Jobs are killed automatically when the runtime has been exceeded. Therefore, we recommend setting the runtime to the maximum of 10 days when running jobs over 1 hour. For example, if your job will take 24 hours, you should still request 10 days to cover any unexpected overheads.

Instructions:

1. Submit an interactive job to the HPC cluster by executing:

   ```
   qlogin
   ```

   Note: By default, all jobs request 1 core, 1GB RAM and 1 hour unless otherwise specified. For example, `qlogin -pe smp 2 -l h_vmem=2G` will request 2 cores and 2GB per core, while keeping the default 1 hour maximum runtime.

2. If there are available resources, the scheduler will provide an interactive session running on a compute node. Notice the change of prompt and the node the job is executing on, which will no longer be a frontend node. Type `qstat` and find your running session in the list of your jobs, named QLOGIN.

3. Load an *R* module and run

   ```
   Rscript /data/teaching/workshop/common/4/countdown.R
   ```

   to execute a trivial R script from within the interactive session. Observe the output which should show decrementing numbers from 5 to 0.

4. Use `exit` to end the interactive job session.

5. Notice the change of prompt which will be a frontend node. Please do not start any computational work when connected to a frontend. All jobs must be submitted via `qsub` or `qlogin`.

Tutorial 5 - Using Resources Effectively

Objective:	To submit jobs which use HPC resources effectively.
Prerequisite:	A text editor (e.g. *vim*).

Instructions:

1. Open a text editor to create a job script.
2. Add the following text and save the file as *bad_job.sh*:

```
#!/bin/bash
#$ -cwd
#$ -j y
#$ -pe smp 2
#$ -l h_vmem=3G

sleep 30s

# Run blast
module load blast+
blastn -db /data/teaching/workshop/common/5/files/nt \
       -query /data/teaching/workshop/common/5/example_seqs.fa \
       -out example_seqs_blastout.txt
```

   Note: The trailing backslash character breaks a single line command over multiple lines, to aid readability.

3. Submit the job script to the HPC cluster for processing.
4. Check which cluster node the job is running on with `qstat`.
5. Run `ssh -t NODE top -Hu ${USER}` to log into the compute node running your job and observe the process table output to see that the process *blastn* is only using up to 100% CPU (1 core) and one thread, meaning the job is using fewer cores than requested, because we requested 2 cores. Press `q` to return to the prompt.

   Note: Replace NODE with the actual node your job is running on.

6. Add `-num_threads 4` to the end of the *bad_job.sh* script. The complete `blastn` command should look like:

```
blastn -db /data/teaching/workshop/common/5/files/nt \
       -query /data/teaching/workshop/common/5/example_seqs.fa \
       -out example_seqs_blastout.txt \
       -num_threads 4
```

   Note: Don't forget to add a backslash character to the `-out` parameter.

This will instruct blast to use 4 threads during execution, which will overload a compute node (because the job only requested 2 cores), resulting in a slower performance for your job and other user's also running on the same node. Please do not do this for production jobs.

7. Submit the job script to the HPC cluster and run the command to observe the process table output on the node running your job, to see that *blastn* is now running on 4 threads, even though we only requested 2 cores.

8. Replace `-num_threads 4` with `-num_threads ${NSLOTS}` in the *bad_job.sh* script.

   We are now using the `NSLOTS` variable which is assigned at job runtime to the number of cores requested - this is the correct method when running multi-threaded applications on the HPC cluster. Feel free to rename the script now it is no longer bad.

9. Submit the job script to the HPC cluster and run the command to observe the process table output on the node running your job, to see that *blastn* is now correctly running on 2 threads.

10. When the job has completed, examine the output file `example_seqs_blastout.txt`.

11. Confirm the exit status of your completed job by substituting JOBID with your job number in the following command:

    ```
    jobstats -j JOBID
    ```

    The `STATUS` column displays the job exit status. Jobs which exit with code 0 ran successfully (and are coloured in green), all other numerical codes indicate an error occurred.

References

- Logging into the HPC - https://docs.hpc.qmul.ac.uk/intro/login/
- Application Modules - https://docs.hpc.qmul.ac.uk/using/UsingModules/
- Submitting Jobs - https://docs.hpc.qmul.ac.uk/using/
- Job Script Builder - https://docs.hpc.qmul.ac.uk/using/jobscriptbuilder/
- Interactive Jobs - https://docs.hpc.qmul.ac.uk/using/#interactive-jobs
- Productivity tips for HPC users - https://blog.hpc.qmul.ac.uk/productivity-tips-for-apocrita-cluster-users.html#content

Solutions

Tutorial 3

Modify the resources in the example.sh script, increasing the core request to 4 and the total memory to 8GB. Add a job name with the #$ -N NAME parameter and repeat steps 3-5 and observe the change of output from qstat.

```
#!/bin/bash
#$ -cwd
#$ -j y
#$ -pe smp 4
#$ -l h_rt=1:0:0
#$ -l h_vmem=2G
#$ -m bea
#$ -N myjobname
```