IT Services – CSF Quick Start Guide

Authors: *Jonathan Boyle, George Leaver, Richard Mervin, Penny Richardson*
Version: *1.6*
Date: *October 2014*

1. Syntax Used in the Instructions

Text boxes in `monospace` represents commands to be typed in exactly. Be careful to copy all spaces and characters correctly, and to enter commands on a single line.
Text boxes in ***`monospace-bold-italics`*** should be replaced as described. For example: insert your standard University IT username if it says ***`username`***

2. Unix/Linux operating systems

Wiki page, including online intro course: wiki.rac.manchester.ac.uk/community/UNIX

The CSF uses the Linux operating system, which is a Unix-like operating system.

3. To connect to the Computational Shared Facility (CSF)

CSF page: ri.itservices.manchester.ac.uk/csf/getting-started-on-the-csf/connecting-to-the-csf/

1. On the training PC (or your own desktop / laptop) open a Terminal application using:
   Windows: MobaXterm *Portable Edition* (mobaxterm.mobatek.net/download-home-edition.html)
   Linux: Applications > Systems Tools > Terminal
   MacOS: Go > Utilities > Terminal
2. Enter the following line at the terminal window prompt, replacing ***`username`*** with your University id (username used to log in to your email, for example), and press enter
   `ssh -X username@csf.itservices.manchester.ac.uk`
   Note: `X` is uppercase.
3. If asked `Are you sure you want to continue connecting (yes/no)?` enter `yes`
4. Enter your ***`password`*** when asked. If you have not previously been given a CSF password this will be provided by your tutor. Note: non-CSF users will be given an account that remains active during the current round of training courses; these will be closed on Friday 28th November 2014.
5. You should now have a new prompt in the terminal window which will be one of the following:
   `[username@login1 ~]$` or `[username@login2 ~]$` (if not, please tell your tutor).
6. If this is the first time you have connected to the CSF you **must change your password** by entering `yppasswd` and following the instructions (enter old password then new password twice).
   IMPORTANT: Make sure you remember your new password; you will need it for today and for any other training courses that uses the CSF. If you forget your password:
   - If on a training event please tell your tutor
   - At other times please email the CSF helpdesk: its-ri-team@manchester.ac.uk

You should now be logged in to the CSF and, if it is the first time you have logged in, you should have changed the supplied password to something private (and secure). If attending a training event please ask for help if you have not successfully completed the above steps.

4. Modules

CSF page: ri.itservices.manchester.ac.uk/csf-apps/software/Modules/

Modules are used on the CSF to set your environment and to access resources and programs (enter `man module` for more info). Use the following `module` commands:

`module avail`	see all available modules (use names to load / unload)
`module list`	see your currently loaded modules
`module load` ***`modulename`***	load a module called ***`modulename`*** (name can include `/`)
`module unload` ***`modulename`***	unload a module called ***`modulename`***

5. To download files required for training

To install files required for training enter

```
module load training/training_material
```

Running this command will install files for all our training courses in a directory (called `training`) in your home folder. You can run this command again to replace any missing files or to get updates when your tutor makes them available without overwriting files *you* have edited.

6. Compiling

CSF page: ri.itservices.manchester.ac.uk/csf-apps/software/Applications/Compilersintel/

Note: our training environment uses two 12-core Intel Xeon E5-2640 (2.5GHz) *Sandy Bridge* compute nodes, each with 64GB ram. We use the Intel compilers to compile Fortran and C/C+ programs. MPI training uses the module for the standard (i.e. non-InfiniBand) network on the CSF. Alternative options can be found on the CSF wiki page.

The steps to compiling are

1. Load the necessary module
2. Run the compiler to generate an executable program from your source code

The following table lists the modules and compile commands used in our training.

Language	**Parallelisation**	***`modulename`***	***`CompileCommand`***
Fortran	None	`compilers/intel/fortran/12.0.5`	`ifort`
	OpenMP	`compilers/intel/fortran/12.0.5`	`ifort -openmp`
	MPI	`mpi/intel-12.0/openmpi/1.6`	`mpif90`
C/C++	None	`compilers/intel/c/12.0.5`	`icc` or `icpc`
	OpenMP	`compilers/intel/c/12.0.5`	`icc -openmp`
	MPI	`mpi/intel-12.0/openmpi/1.6`	`mpicc` or `mpi++`

Loading the module

Enter the following, where ***`modulename`*** is replaced by text from the above table:

```
module load modulename
```

Running the compiler

In general, to run the compiler, enter:

```
CompileCommand SourceFile.ext -o ExecutableName
```

where:

CompileCommand	is replaced by text from the above table
SourceFile.ext	is replaced by the name of your source code with the correct file extension, e.g.
	ext is `f90` for Fortran
	ext is `c` for C
	ext is `cpp` for C ++
ExecutableName	is replaced by the name of the executable you want to create, and **must** follow `-o`

For example, to compile C code: `icc mycode.c -o myapp`

Optimisation can be controlled using the **`-O`*n*** flag, where **`O`** is uppercase letter O (not zero) and ***n*** is a number representing optimisation level between 0 (no optimisation) and 3 (aggressive optimisation). For example to compile *without* optimisation (as required in MPI and OpenMP courses) add the `-O0` flag, i.e. enter:

```
CompileCommand SourceFile.ext -O0 -o ExecutableName
```

Some training courses use the `make` utility as an alternative method for compilation. If this is the case you will be instructed by your tutor.

7. To submit jobs to the CSF compute nodes

CSF page: ri.itservices.manchester.ac.uk/csf/getting-started-on-the-csf/starting-with-sge/

When you log in to the CSF you are connected to the *login node*. This is a lightweight node used for tasks such as compiling code and managing files. You should **not** run your executables on this node though - it doesn't have many cores or much memory. Instead, you are going to make use of the much more powerful *compute nodes* in the CSF. You do this by making a request to the *Sun Grid Engine* (SGE) batch system to run your executables. This will run your code on suitable compute nodes, according to the number of cores and other resources you request. If many other jobs are running you may have to wait until suitable compute nodes become available, but the queue system manages this fairly. The following sections describe how to submit your jobs to the SGE batch system (from the login node).

7.1. Batch jobs

From the login node, jobs are submitted using the `qsub` command to send jobs to the SGE batch system. This will select appropriate compute nodes to run on. `qsub` uses various flags to set job parameters (enter `man qsub` for more info). The flags can be given in a *job script* file, which is read by `qsub` or can be given as command line flags. It is also possible to specify some options as command line flags and some in a job script.

The most common *flags* used by the `qsub` ***flags*** command you will use are:

`-cwd`	to run the job in the current directory
`-l` ***resource***	(lowercase letter l) to use specific resources, e.g. `-l course` to use compute nodes reserved for training (but not outside of training events).
`-N` ***name***	to set job name
`-o` ***filename***	(lowercase letter o) to set file name for the standard output stream
`-V`	(uppercase letter V) to export the current environment to the compute node
`-pe` ***name num***	to set the **p**arallel **e**nvironment and **num**ber of cores (not required for 1 core jobs) e.g.
	`-pe smp.pe 8` for shared memory jobs using 8 cores (suitable for OpenMP code)
	`-pe orte.pe 24` for distributed memory jobs using 24 cores (suitable for MPI code)

Jobs submitted using the `-l course` flag will receive higher priority during a training event. The flag cannot be used outside of training events and jobs will be scheduled using the default priority. Remove the `-l course` flag outside of training events.

By default standard output and error streams appear in files named ***jobname*****.o*****JOBID*** and ***jobname*****.e*****JOBID*** respectively, where ***JOBID*** is a unique number given the job by SGE when you submit your job to the batch system. By default ***jobname*** is the name of the script file or command. Alternative job names can be specified using the `-N` flag, and alternative output (and error) file names can be specified using `-o` (and `-e`) flags.

After submitting a job, run `qstat` to monitor progress and `qdel` ***JOBID*** to delete jobs.

Method 1 – using a job script file

This method uses a text file in which lines starting with **#$** (followed by a space) set the flags. After listing SGE flags all commands to be run are listed. For example, the following script submits a serial (i.e. single core) job to run an executable called ***MyExecutable*** in the current directory. If this script is saved to a file called ***QsubScript*** the job can be submitted by entering `qsub` ***QsubScript*** at the prompt.

```
#$ -V
#$ -cwd
#$ -o OutputFileName
#$ -l course

./MyExecutable
```

To run shared memory parallel (OpenMP) jobs add

```
#$ -pe smp.pe n
```

where ***n*** is replaced by the number of cores up to a maximum of 12.

For distributed memory parallel (MPI) jobs use

```
#$ -pe orte.pe n
```

where ***n*** is replaced by the number of cores up to a maximum of 24 for training events.

Note: that when running jobs outside of the training environment (if you have full CFS access) you are encouraged to use `orte-12-ib.pe` ***n*** where ***n*** is a multiple of 12. This will give better performance than `orte.pe` due to use of faster InfiniBand networking. In this case you would also have use: `module load mpi/intel-12.0/openmpi/1.6-ib` to set up the correct environment.

Method 2 – enter flags at the command line

Simply type `qsub` followed by the list of flags and values, ending with the command to be executed. If the command to be executed is a binary executable (i.e., not a job-script as described above) then you should add `-b y` to indicate a binary executable.

For example, the script shown above is equivalent to

```
qsub -b y -V -cwd -l course -o OutputFileName ./MyExecutable
```

7.2. Interactive jobs

CSF page:
ri.itservices.manchester.ac.uk/csf/getting-started-on-the-csf/using-an-app-with-a-gui-x11-and-qrsh/

Interactive jobs are submitted using `qrsh`, for example to request a single core interactive job:

```
qrsh -V -cwd -l course CommandToRun
```

or to request a distributed memory parallel job that uses `mpirun` enter:

```
qrsh -V -cwd -l course -pe orte.pe n mpirun -np n CommandToRun
```

replacing ***CommandToRun*** with whatever it is you want to run, and for parallel jobs replacing both ***n*** with the number of processes/cores. Outside training events `course` should be replaced by `inter` when using qrsh.

8. Post training help

For general help using the CSF please visit:
ri.itservices.manchester.ac.uk/csf/overview/support-overview/.

For further information please contact the relevant help desk available at
itservices.manchester.ac.uk/research/contacts/.