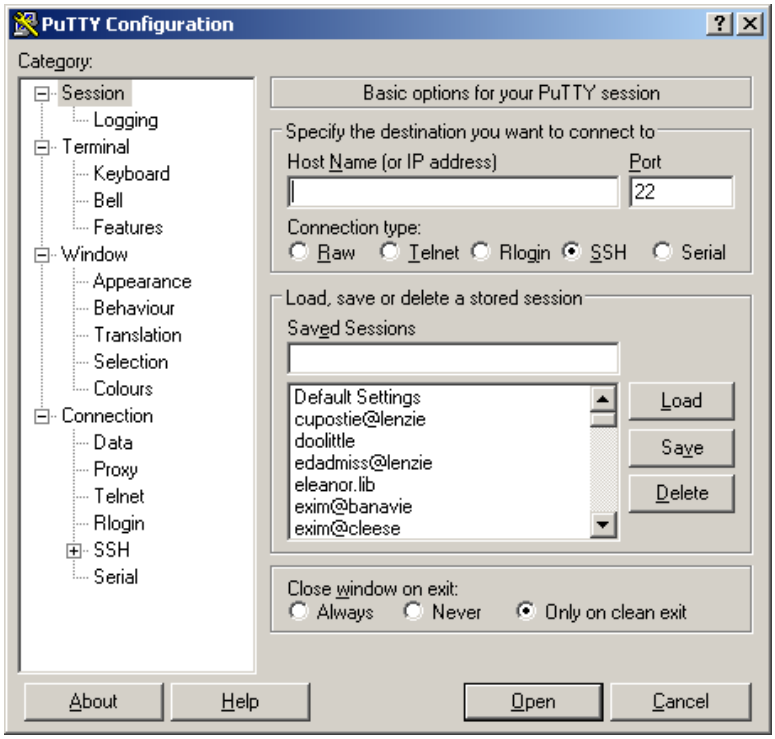
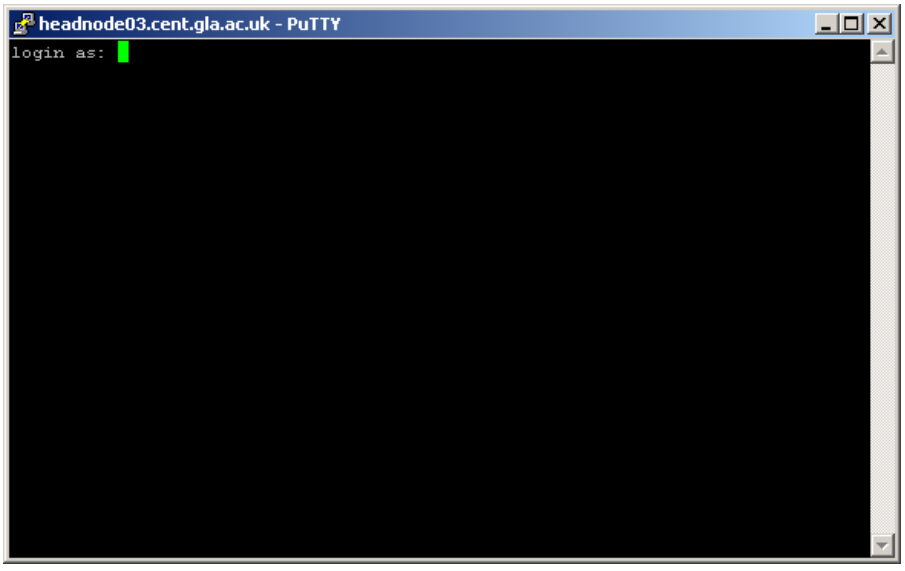
Using PuTTY

Download PuTTY

In the start menu you will find a programme called **PuTTY** – you can use this to connect to the headnode. When you start the programme you will be presented with a box like this.



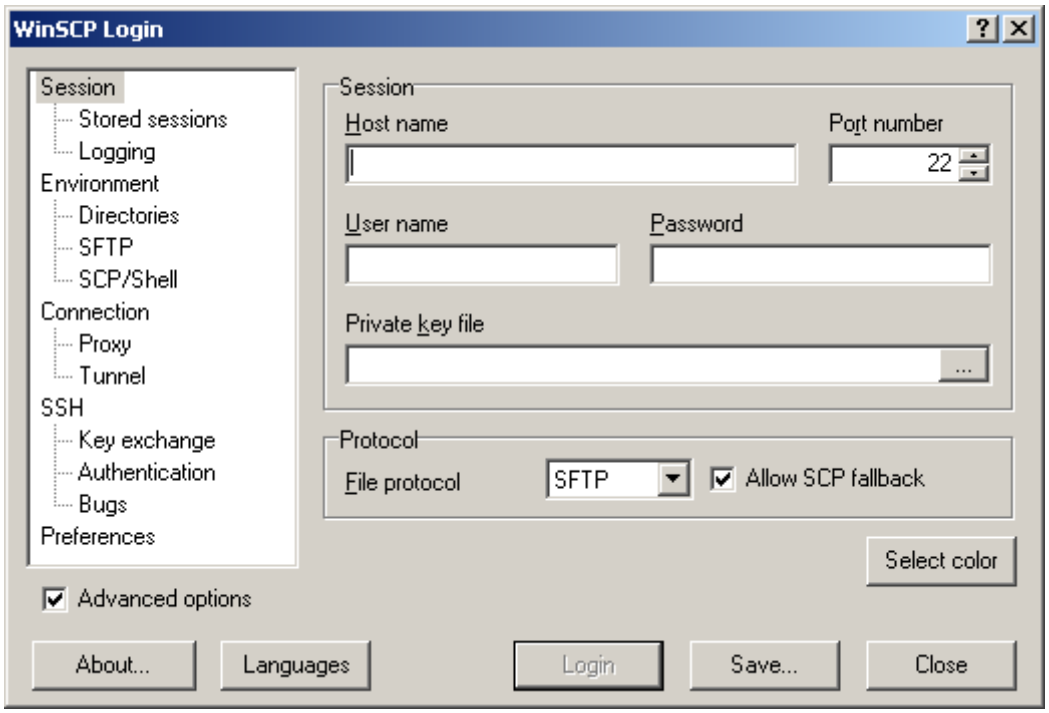
Put the name **headnode04.cent.gla.ac.uk** into the hostname box and ensure the connection type is SSH then click open. This will open a command line interface on the headnode.



From here you can log in with your username & password for the cluster (for SSH keys setup you can contact us). To end the terminal session type exit & press the enter key.

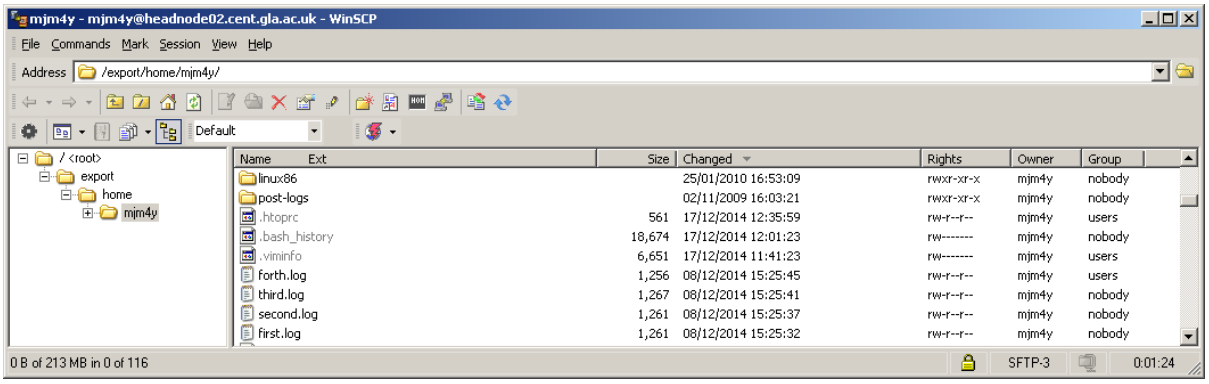
Using WinSCP

[Download WinSCP]



In the start menu you will find a programme called WinSCP – when you start it you will be presented with a box like this.

Put **headnode04.cent.gla.ac.uk** into the hostname box and enter your username and password. Then click login and you will get a window like this



From here you can use windows programmes to view and edit files. You can also drag & drop files between here and your local machine. One gotcha – the display does not automatically refresh so you will need to refresh the screen by clicking the refresh screen button or pressing the F5 key.

To end the session just close the window as you would do with any other windows programme.

1) Log into the cluster using your username & password

2) If you have not already done so – change your password using the command passwd

```
passwd

Changing password for user mjmtest1.
Changing password for mjmtest1
(current) UNIX password:
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

3) If you have not already done so setup passwordless ssh

When prompted to enter a password just press return, to work with PBS/Torque it needs to have no password.

```
ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/export/home/mjmtest1/.ssh/id_rsa):
Created directory '/export/home/mjmtest1/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /export/home/mjmtest1/.ssh/id_rsa.
Your public key has been saved in /export/home/mjmtest1/.ssh/id_rsa.pub.
The key fingerprint is:
53:31:07:6e:f4:14:bb:87:34:83:bd:78:83:be:86:f0
mjmtest1@headnode04.cent.gla.ac.uk
```

This will create a few files in ~/.ssh/. Then you need to create an authorised keys file and restrict access to the new file.

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

4) Then copy the training directory over to your home area

```
cp  -R    /export/home/training    ~/
```

There is now a folder called training in your home area containing some test scripts we are going to use

You will need to edit some of the files so that the scripts will run for you

arrayjob.sh
arrayjob2.sh
arrayjob4.sh
cleanup.sh
stageinout.sh

In all of these files search for GUID and change it to your username. In arrayjob2.sh and arrayjob4.sh you should also change YOUREMAILADDRESS to your email address.

File editors are available on linux e.g. vi emacs or you can edit with windows programmes using WinSCP.

1) First open a PuTTY terminal on the headnode.

Type hostname and press return. Note the answer you got back

Open an interactive session on one of the execution nodes – type qsub –I

```
qsub -I
qsub: waiting for job
247367.headnode03.cent.gla.ac.uk to start
qsub: job 247367.headnode03.cent.gla.ac.uk ready

Prologue Args:
Job ID: 247367.headnode03.cent.gla.ac.uk
User ID: mjm4y
Group ID: users
MAchine: node041.hpc.gla.ac.uk
-bash-3.2$
```

Now type hostname and press return again. You will see that your PuTTY terminal is now running commands on one of the execution nodes.

To end the session on the execution node just type exit & press return, this will bring your session back to the headnode

2) change directory into the training folder:

```
cd training
```

Now you want to submit the script one.sh to the cluster by typing:- qsub one.sh

```
-bash-3.2$ qsub one.sh
247377.headnode03.cent.gla.ac.uk
-bash-3.2$
```

Note here you get presented with a job number and returned to the command prompt. Your job is now scheduled to run on one of the execution nodes. While it is running try typing qstat, qstat –a, qstat –f JOBNUMBER and see what output you get. When it completes you will see the output files in the directory you submitted the job from.

```
-bash-3.2$ ls -l one*
-rwxr--r-- 1 mjmtest1 mjmtest2   43 Dec 17 16:52 one.sh
-rw------- 1 mjmtest1 mjmtest2    0 Dec 18 15:23 one.sh.e247377
-rw------- 1 mjmtest1 mjmtest2 1843 Dec 18 15:25 one.sh.o247377
```

The job number is included in the name of the output files so that you can correctly match the output files to the job you submitted. These are basic text files so you can open them up with an editor and read the contents.

3) So far you have just been given minimum requirements and other defaults – let us start asking for specific resources – cpu time of 15 hours. We can do this by typing:

```
qsub –l cput=15:00:00 one.sh
```

While this is running try using qstat to query the job.
Submit another job asking for 100 hours cpu time and 2 processors on a node

```
qsub -l cput=100:00:00,nodes=1:ppn=2 one.sh
```

Now use qstat again to see your job in the queue.

4) Let us try putting the options into the script. First make a copy of one.sh

```
-bash-3.2$ cp one.sh oneB.sh
```

Now edit oneB.sh to add the following lines after the first line (#! /bin/bash) but before the main part of the script (sleep 100 in this case)

```
#PBS -l cput=15:00:00
#PBS –N myname
```

Changing myname to whatever name you want

Then submit oneB.sh

```
qsub oneB.sh
```

When you use qstat to query it you will see that it now has a script name of whatever you put in as myname and a requested time of 15 hours. When the job completes the filename of the output & error files will be myname.o12345 and myname.e12345.

Edit oneB.sh again adding another line

```
#PBS -l nodes=1:ppn=4
```

Now submit the job again and have a look at it with qstat.

5) Now try adding other options to the script e.g. walltime rather than cput in the –l option, and using the –m & -M options. In all cases use qstat to see what happens with your job.

6) Have a look at the output of qstat –q, qstat –Q and qstat –Qf.

7) Finally remove the output and error files, either with winscp or on the linux command line with rm.

```
-bash-3.2$ ls
arrayjob2.sh      arrayjob.sh  input           oneB.sh.o247588  one.sh.e247587  stageinout.sh
arrayjob4.sh      cleanup.sh   oneB.sh         one.sh           one.sh.o247390  three.sh
array-jobs-files  four.sh      oneB.sh.e247588 one.sh.e247390   one.sh.o247587  two.sh


-bash-3.2$ rm oneB.sh.e247588 oneB.sh.o247588 one.sh.e247390 one.sh.e247587 one.sh.o247390
one.sh.o247587

-bash-3.2$
```

1) Type pbsnodes –a and look at the output

2) Copy the file oneB.sh to oneC.sh and then edit oneC.sh and change the sleep 100 line to sleep 300 (job will now run for 300 seconds). Submit the job with qsub oneC.sh and then look at the output of tracejob jobnumber

```
tracejob 12345
```

3) Submit some more jobs and try out the commands showq, showstart & checkjob

4) Look at the output of pbstop

```
pbstop  -n -c 64 -m 1
```

5) Submit a few jobs and then try to cancel them with qdel

6) Submit some more jobs and cancel them with canceljob

7) Clean up the error and output files as before

From within your training directory.

1) Submit an arrayjob

```
qsub –t 1-4 arrayjob.sh
```

Now use the different options of qstat to query what is happening with your job, remember you can get more information by adding the –t option. So try qstat, qstat –a, qstat –n and then qstat –t, qstat –at, qstat –nt.

2) Now I want you to try to limit the number of running tasks to two.

```
qsub –t 1-4%2 arrayjob.sh
```

Again look at the running job with qstat – you should now see 2 of your jobs in the held state until the first ones finish.

3) The file arrayjob2.sh has the qsub options in the file itself and arrayjob4.sh starts an array of 12 tasks with a limit of 4 running.

Now start an arrayjob and use tracejob with the array options to query what has happened with the job and the individual tasks.

Wait till all the tasks have finished and use tracejob again.

4) You should now try using qdel with arrays – use arrayjob4.sh as this gives more tasks

```
qsub arrayjob4.sh
```

Now try to delete certain tasks with qdel jobid[arrayid]

Submit the job again and delete some tasks with qdel –t.

5) Clean up the output and error files as before. You will also need to delete the logfiles one.log etc.

1) A look at dependencies – start a job running from your training folder

```
qsub one.sh
```

Take a note of the job number and use it when you submit another job

```
qsub –W depend=afterok:12345 two.sh
```

Now use qstat to observe what is happening - you can see that two.sh waits in the hold state until one.sh completes with no errors.

2) Try the example from the notes

```
qsub one.sh
qsub two.sh
qsub three.sh
qsub four.sh
```

Note the jobid's of the four jobs and now submit the cleanup job (the job id's may not be sequential)

```
qsub –W depend=afterok:12345:12346:12347:12348 cleanup.sh
```

Now use qstat to observe how your jobs progress

3) We can turn this round and submit cleanup.sh first with a dependency of 4 jobs.

```
qsub –W depend=on:4 cleanup.sh
```

Note the jobid and then submit the four jobs as follows

```
qsub –W depend=beforeok:12345 one.sh
qsub –W depend=beforeok:12345 two.sh
qsub –W depend=beforeok:12345 three.sh
qsub –W depend=beforeok:12345 four.sh
```

Again use qstat to observe the progress.

4) Job synchronisation, for this task use qstat between each job submission, you should observe that the jobs remain in the queue until all four are submitted AND there is sufficient free resource for all 4 jobs to run.

```
qsub –W depend=synccount:3 one.sh
```

Note the jobid.

```
qsub –W depend=syncwith:12345 two.sh
qsub –W depend=syncwith:12345 three.sh
qsub –W depend=syncwith:12345 four.sh
```

5) arrayjobs – submit an arrayjob and another job which depends on it completing first

```
qsub arrayjob2.sh
```

Note the job id and then submit the cleanup job

```
qsub –W depend=afterokarray:12345[] cleanup.sh
```

Watch the progress of the jobs with qstat –t options

Now we can let cleanup run before all tasks in the array have completed like this

```
qsub arrayjob4.sh
```

Note the jobid, this is a 12-task array to give more flexibility. And submit cleanup.sh to run after 6 tasks have finished.

```
qsub –W depend=afterokarray:12345[][6] cleanup.sh
```

Again, monitor progress with qstat -t

6) Look at the contents of the file stageinout.sh, you will see that it has the stagein and stageout options . Also have a look at the file input. Note the naming convention for the files on the execution node – this is because you are using a shared file area (/tmp) and you have to consider that other people might use the same file name.

Now submit this script.

```
qsub stageinout.sh
```

Monitor with qstat as usual. After the job has completed there is now another file in your home area called output.log.

7) cleanup the output, error & log files as before.