

Using the Babraham Compute Cluster

Version 1.4.0

Licence

This manual is © 2013-14, Simon Andrews.

This manual is distributed under the creative commons Attribution-Non-Commercial-Share Alike 2.0 licence. This means that you are free:

- to copy, distribute, display, and perform the work
- to make derivative works

Under the following conditions:

- Attribution. You must give the original author credit.
- Non-Commercial. You may not use this work for commercial purposes.
- Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a licence identical to this one.

Please note that:

- For any reuse or distribution, you must make clear to others the licence terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.
- Nothing in this license impairs or restricts the author's moral rights.

Full details of this licence can be found at

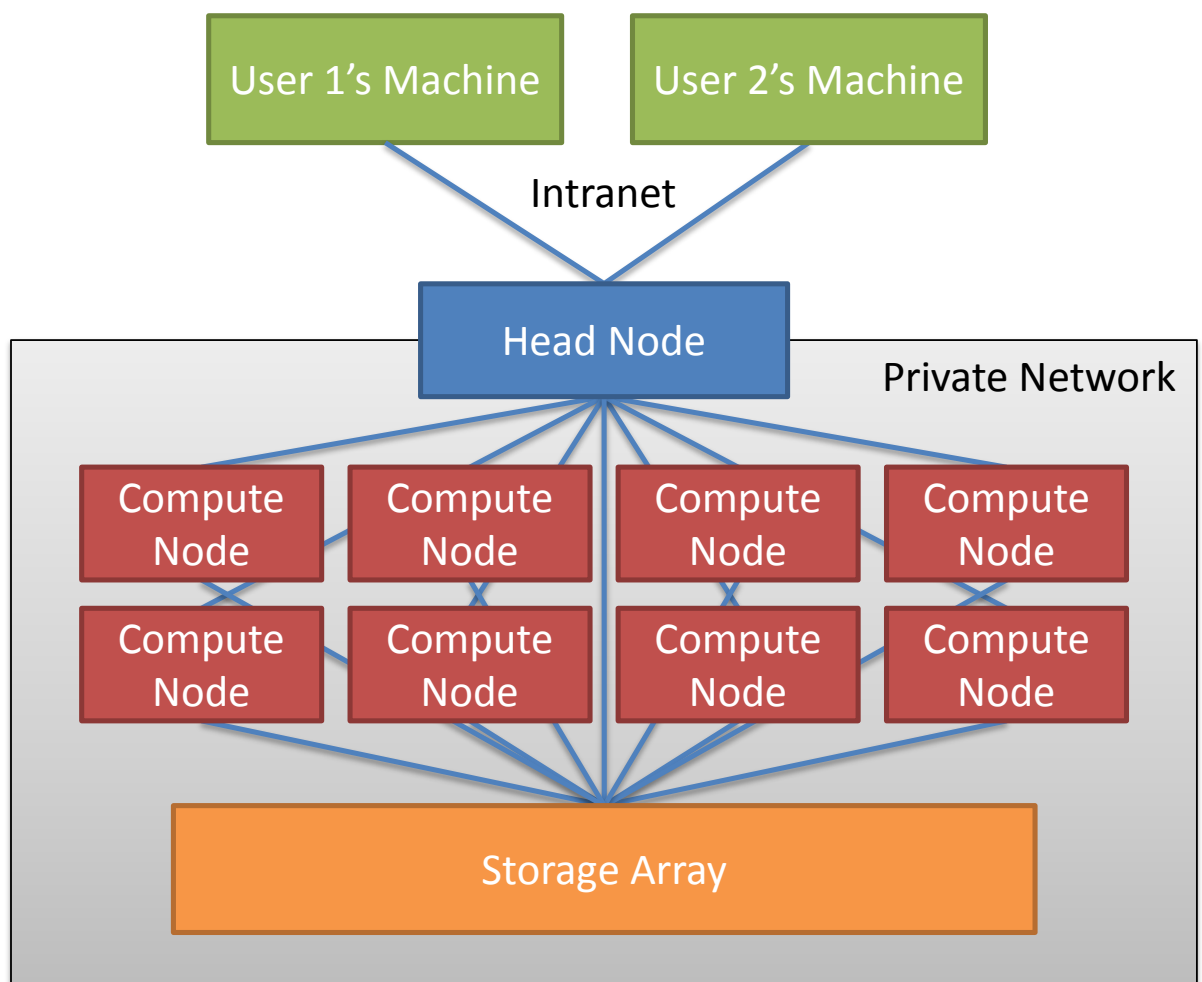
<http://creativecommons.org/licenses/by-nc-sa/2.0/uk/legalcode>

What is the cluster

The Babraham compute cluster is a shared computational resource which is now available to anyone at the institute doing research which requires a powerful computing platform.

The system is a 384 node linux cluster which can be accessed via a command line interface. It has a dedicated 160TB storage array attached directly to it and can also access a separate 320TB storage array containing various public datasets. It has a wide variety of common bioinformatics packages installed and should provide a powerful platform for all manner of large scale computational analyses.

In structure the cluster consists of a set of physical servers. The servers are split into compute nodes and head nodes. When using the cluster you only normally directly interact with the head node and it can communicate with the processing nodes to actually carry out the work you request.



Using the cluster

Registering to use the cluster

Before you can use the cluster you need to be registered as a user on the system. To do this please contact Simon Andrews in bioinformatics and arrange a time to briefly discuss what work you want to do on the cluster and set up your account.

Software required to use the cluster

The cluster is accessed via a linux command line interface, but it can also support the running of graphical programs on the cluster which are displayed on your local desktop. To access the cluster you will therefore require a terminal program to make the initial connection and provide a command line interface. If you want to run graphical programs you will need an X server program which will allow remote windows to be displayed on your machine.

Windows

Windows users will need to install two pieces of software to access the cluster. To allow a command line connection they should use PuTTY, which can be found in the Molbio Software Repository along with all of the other windows bioinformatics packages. For graphical displays you will need a package called Exceed which you'll need to get from computing. This is a commercial package for which we have a site license so there is no individual charge to have this installed on your machine.

OSX

OSX users can use the X11.app (now called XQuartz.app in the latest OSX release) as their X server which should be installed in their Applications folder in the Utilities subfolder. The X11 app also provides an xterm application which can be used to connect to the cluster.

Linux

Linux uses X11 to display all of its graphics so no additional X server is required. Linux users can use whichever terminal application comes with their chosen desktop environment (Terminal, Konsole, xterm etc) to connect to the server.

Connecting to the cluster

Although the cluster has many processing nodes within it all access to the cluster happens through a single node called the 'head node'. This is the machine you connect to, and from the head node you can submit jobs which are automatically passed out to appropriate compute nodes to be run.

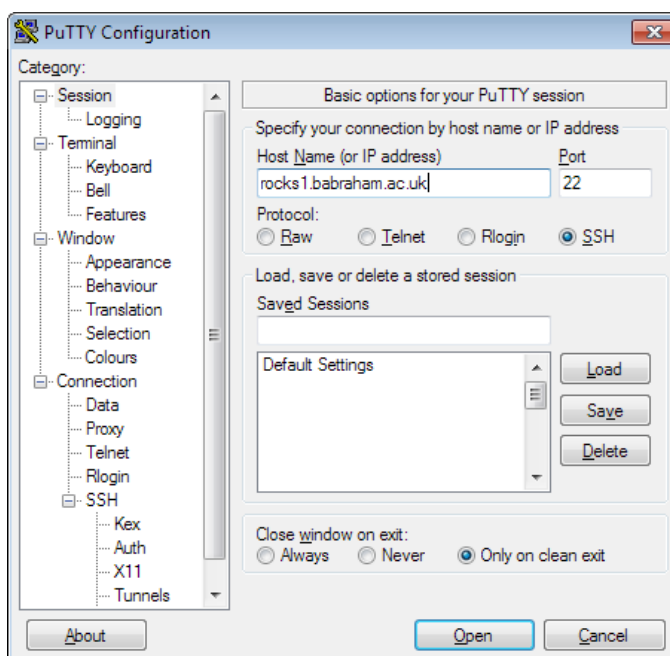
The address of the head node is **rocks1.babraham.ac.uk** and a connection to this node must be made via SSH. Details of how to do this are given below.

Windows

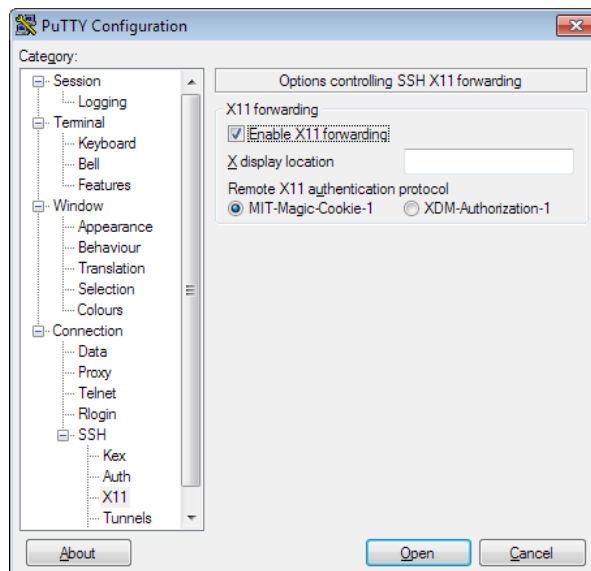


Before connecting to the head node ensure that the Exceed program is running (Start > All Programs > OpenText Exceed > Exceed). You will see a splash screen appear and then disappear and then you should just see the Exceed icon in your toolbar, there will be nothing else shown on the main part of your screen.

Once Exceed is running you can use PuTTY to connect to the cluster. When you first run PuTTY you will see the main configuration screen.

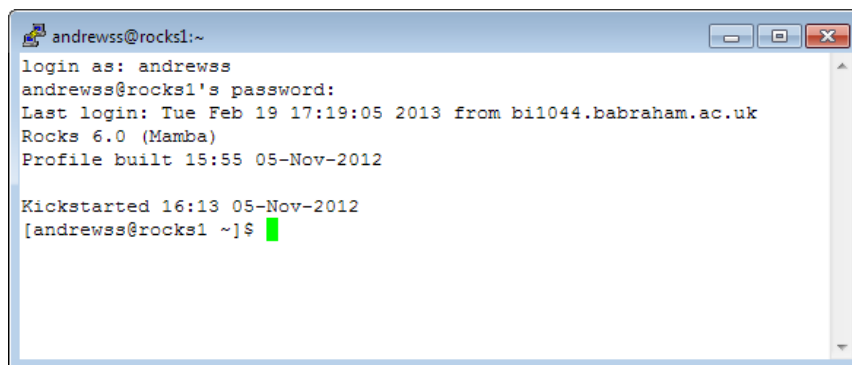


On the front screen you need to enter the address of the head node in the "Host" box, which is **rocks1.babraham.ac.uk**. You also need to move to the Connection > SSH > X11 option, and tick the box which says **"Enable X11 forwarding"**.



You can now save these settings by putting a name in the “**Saved Sessions**” box on the main screen and then pressing Save. In future you can then connect to the cluster by simply double clicking on the saved session in the saved session list.

When you start a session you will see a PuTTY screen appear from where you can enter your login details. The details you use will be your normal windows username and password.



OSX

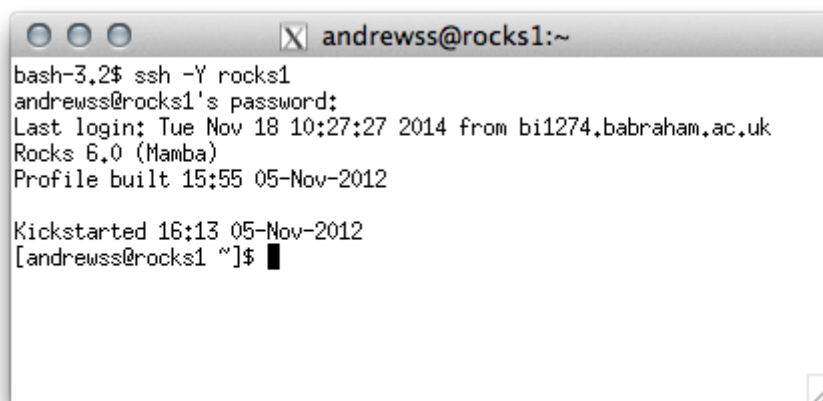
To connect to the cluster first start the X11.app from your applications folder. When you start this you should see an xterm terminal window appear. If it doesn't appear then you can start one manually by going to X11 > Applications > Terminal.

Once the terminal appears you can connect to the cluster by running:

```
ssh -Y rocks1.babraham.ac.uk
```

[On older versions of X11.app you may need to use -X instead of -Y]

...and using your password to connect.

A terminal window titled 'andrewss@rocks1:~' showing the execution of an SSH command. The user 'andrewss' connects to 'rocks1'. The terminal displays the password prompt, login success message, system version (Rocks 6.0), and the prompt '[andrewss@rocks1 ~]\$'.

Linux

Linux users can connect to the cluster from whichever terminal application they normally use. The command line would be the same as for OSX.

```
ssh -Y rocks1.babraham.ac.uk
```

If the username you're using on your local linux session isn't the same as your windows username then you can specify a specific user to connect as using

```
ssh -Y username@rocks1.babraham.ac.uk
```

File Management

Where to save data

The cluster has an attached storage system onto which you can save your data. There are three main areas into which you can put data and which are appropriate for different kinds of data. All of the areas are situated under the **/bi** directory and you shouldn't try to save data outside of these locations.

Every user has a home directory under **/bi/home/[username]** and this is where you'll start each session on the cluster. The home directory is suitable for relatively small files which are only relevant to you. Examples might be scripts that you're working on or configuration or results files.

Most of the important data should go into your group share which is situated under **/bi/group**. If you're not sure which group you're in you can just run 'groups' on the command line which will tell you. How files inside an individual group share are organised is left up to the group so you'll need to discuss with the other members of your group how you are going to manage this.

All data saved in either your home directory or the group share will be backed up to a second storage cluster at regular intervals. The backups are filesystem snapshots so although they provide some resilience they do not provide long term backups. If you accidentally delete something from one of these areas you need to get it restored as quickly as possible as the snapshots will eventually roll over and the data will be permanently lost.

For data which doesn't need to be backed up there is a separate storage area which you can use and you should try to use this area where possible so that we don't end up wasting space in the backups for data which can easily be recreated. This scratch area can be found under **/bi/scratch** and its permissions are open so that anyone can write data into this area.

For all of the storage areas please try to manage the data you create. We have a relatively large storage area, but with the size of datasets we are now creating it will start to fill up surprisingly quickly. At the moment we do not put limits on the amount of space which can be taken up by individual users or groups, but this relies on people managing their storage usage and cleaning up data which is no longer required.

Transferring data to and from the cluster

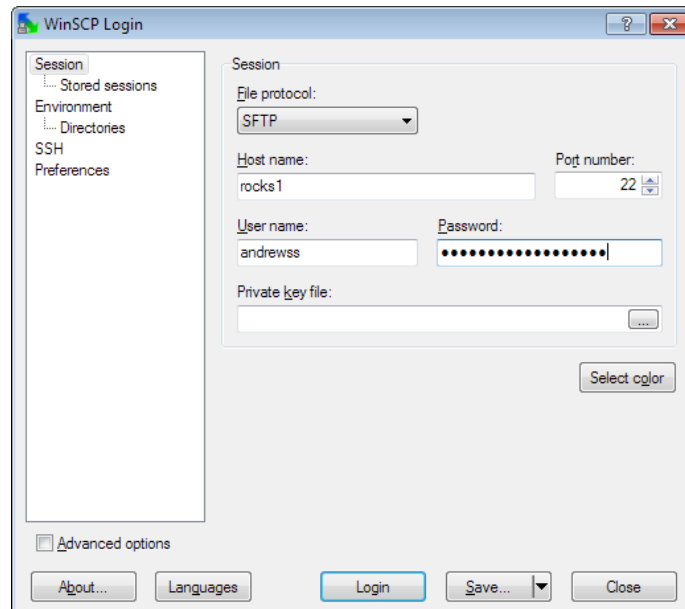
The main data stores on the cluster are only visible from nodes within the cluster, so if you want to move data on or off the cluster you will need to do this manually.

Data transfers to and from the cluster can be done using scp or sftp using the same credentials you use to log into the cluster.

Windows

On windows you will need to use an scp client to do your data transfers. The one we would recommend is WinSCP which can be found in the Molbio Software Repository. This is a simple transfer program which integrates with Windows Explorer to allow you to drag and drop files between the cluster and your desktop.

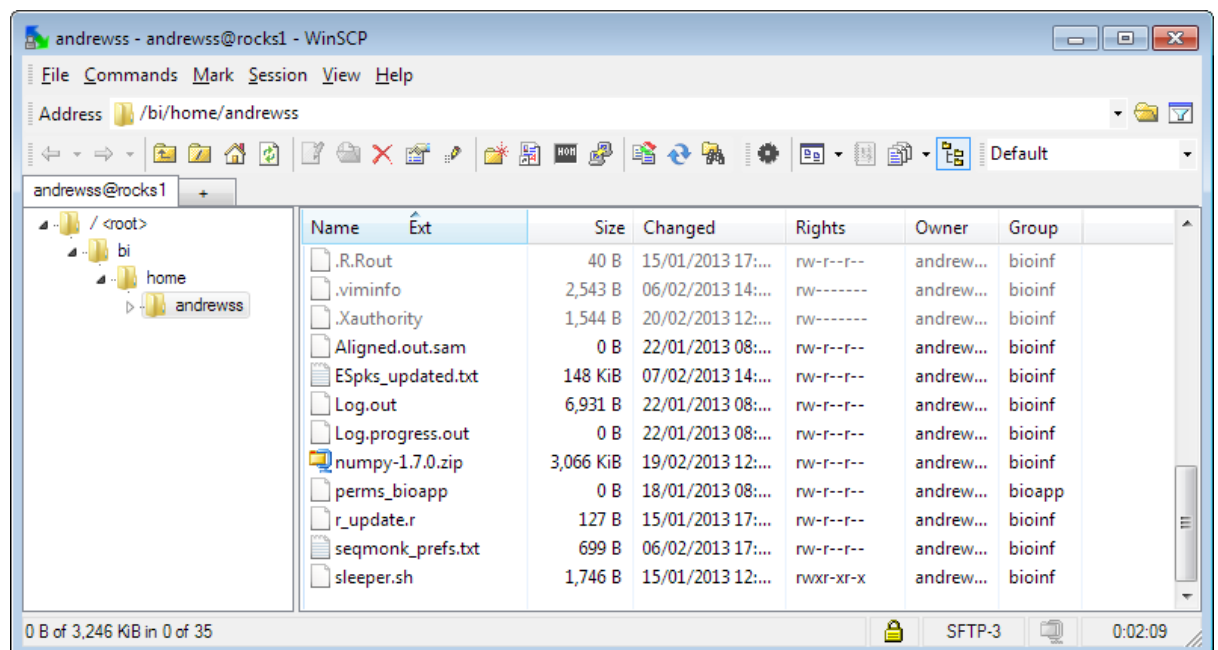
When you launch the program you see a setup window into which you can put the details of the server and your account:



You can also go to Preferences and change the default interface to be the Explorer interface so it looks like another windows explorer window.

You can save the settings you've entered by pressing the save button, and the saved session will be shown to you the next time you start the program.

Once you've logged in you will see a file window containing your data on the cluster. You can use this to navigate to the directory you want to use for your transfer and then dragging data into or out of this window into a normal windows explorer window to actually perform the transfer.



OSX

On OSX you can either transfer data using a command line interface or a graphical scp client. On the command line you can use the `scp` program to copy data from a remote host. The syntax for `scp` is virtually the same as for a normal `cp` except that you can prefix either side of the transfer with a host name to specify the machine you want to pull data from.

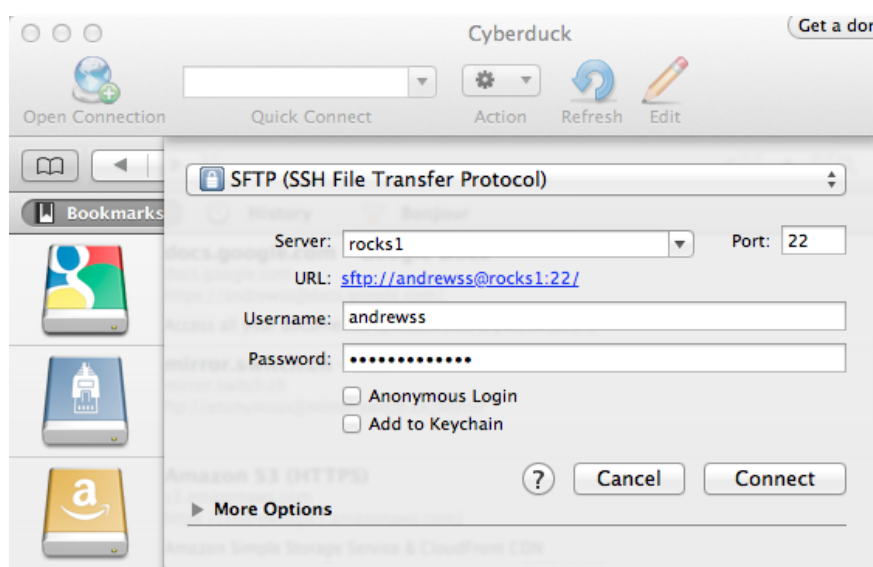
For example, if you wanted to copy a file called data.txt from your group share to your local data volume you could use a command such as:

```
scp rocks1:/bi/group/mygroup/data.txt /Volumes/Data/
```

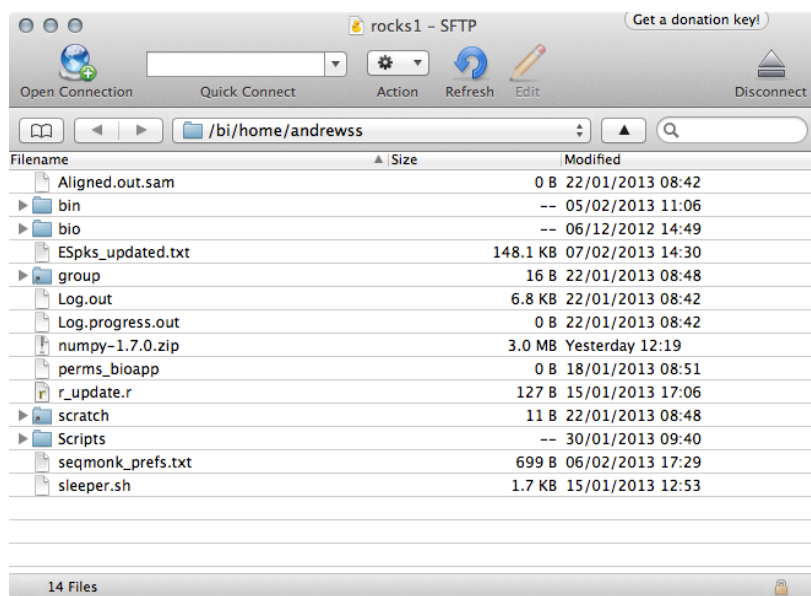
To send a file called sendme.txt from the current directory to a folder called Upload in your cluster home directory you could use:

```
scp sendme.txt rocks1:/bi/home/myusername/Upload/
```

If you want to use a graphical client then the cyberduck client works well for moving data by scp. You can connect to the server by selecting “Open Connection” and using SFTP as the transfer method and your normal username and password as authentication



Once you have logged in you will see a graphical view of your files on the server and you can drag and drop data in and out of this window.



Linux

Linux users can use a command line copy as described in the OSX section. There is also normally an SFTP client built into the file manager for whichever desktop environment you're using. Additionally there are generic file transfer programs such as filezilla which can run under linux and which provide a graphical interface to SFTP data transfers.

Software packages

Once you are logged into the cluster you will need to set up your environment to add in whichever software packages you want to be able to use. There are very few packages available within the default shell you get having just logged in but it's easy to add in what you need.

Most of the software on the cluster is controlled by a system called environment modules. This allows you to install a lot of different packages on the cluster, including multiple versions of the same package, or even conflicting packages and then provides a mechanism for each user to select which of these packages they want to use in their session.

All of the options relating to environment modules are accessed through a program called 'module'.

To see which packages are available to you on the cluster you can run the command

```
module avail
```

This will produce a list of the packages and versions you can choose from.

```
$ module avail
```

```
----- /usr/share/Modules/modulefiles -----
dot          module-info  null          use.own
module-cvs    modules      rocks-openmpi

----- /etc/modulefiles -----
openmpi-x86_64

----- /bi/apps/modulefiles -----
R/2.15.2                cufflinks/2.0.2        python/2.7.3
SBMLsimulator/1.0-rc2   cutadapt/1.1           python3/3.3.0
bedtools/2.17.0         cutadapt/1.2.1         queue/2.3.9
bioperl/1.2.3           ensemblapi/70          samtools/0.1.18
bismark/0.7.7           fastq_screen/0.4       seqmonk/0.23.1
bowtie/0.12.9           fastqc/0.10.1         seqmonk/0.24.0.devel
bowtie2/2.0.5           gatk/2.3.9             sratoolkit/2.1.9
bowtie2/2.0.6           hicup/0.3.0            star/2.2.0
bwa/0.6.2               java/1.6.0_37          tabix/0.2.6
cellDesigner/4.3        java/1.7.0_09          tophat/2.0.7
cmake/2.8.10.2          macs/1.4.2             trim_galore/0.2.5
copasi/4.8-build-35-static meme/3.5.7             vcftools/0.1.10
copasi/4.9.45           ngs/0.1                weeder/1.4.2
```

Most of the interesting packages will be in the last section of this list. You can see that each package has both a name and an associated version number. Some packages have multiple versions installed whereas others will have only one.

To use a package you use the 'module load' command. You can choose to specify a specific module to load (eg bowtie2/2.0.5) or you can just use the module name (bowtie2) in which case you'll get the latest version of that module.

Loading a module will immediately make that software package available in your current session as demonstrated below.

```
$ bismark --version
-bash: bismark: command not found
```

```
$ module load bismark
$ bismark --version
```

```

      Bismark - Bisulfite Mapper and Methylation Caller.
Bismark v0.7.7 Copyright 2010-12 Felix Krueger, Babraham Bioinformatics
      www.bioinformatics.babraham.ac.uk/projects/
```

If you're not sure what modules you've already loaded then you can see this using 'module list'. Modules know about dependencies on other modules, so loading one module may also load several others to allow it to work.

```
$ module list
No Modulefiles Currently Loaded.
```

```
$ module load trim_galore
$ module list
Currently Loaded Modulefiles:
  1) java/1.7.0_09          3) cutadapt/1.2.1
  2) fastqc/0.10.1         4) trim_galore/0.2.5
```

If you need a package which isn't available either by default or through the environment modules then please come to speak to someone in the bioinformatics group who can help you to work out the best way to add in the software you need.

You can also configure your account to automatically load some modules whenever you log in. To do this you initially need to run the command shown below. This only ever needs to be run once on your account.

```
echo "module add null" >> ~/.bashrc
```

Once you've done that you can use the commands below:

`module initlist` – shows you which modules you're currently loading by default

`module initadd [module name]` – adds a new module to the initially loaded list

`module initrm [module name]` – removes a module from your initially loaded list

Running jobs

Although you connect to the cluster through the head node, this machine isn't designed to do any real processing work. You can run small jobs such as copying and uncompressing data, on the head node and you can also use this to run editors for writing code and setting up jobs, but any significant computational work should not be run there.

The cluster runs a system called gridengine which is a queuing system which can take in large numbers of jobs and distribute them from the head node to one of the 128 compute nodes. They can then run in parallel and the results are returned back to the head node, from where you can collect them. The queue can manage submissions from multiple users and it can take in far more requests than can simultaneously be processed by the cluster, and will assign them to the available compute nodes in the most efficient manner possible. We would therefore strongly urge you to use the queue for your large computational jobs.

Requesting resources

One of the key aspects to the way that gridengine works is that it has to match the resources required for each submitted job to those available on the cluster. In this way it can ensure that it makes optimal use of the resources it has and processes as many jobs as possible at the same time. When you submit a job to the cluster therefore it is important that you tell the cluster, as best you can, how much computational resource your job is going to require. There are two types of resource which are important:

1. The amount of memory (RAM) your job will need
2. The number of CPU cores your job will use

If you don't specify these resources then a small default will be used instead (1 core and 1GB RAM), so if your job will use a significant amount of memory or more than 1 CPU core then you must specify this when you launch it.

The options for specifying resources are the same for all of the different submission types listed below and should be appended to any of the batch submission commands (interactive sessions are not limited by resources).

To specify that your process needs to use a large amount of memory (more than 1GB) you should use the option `-l h_vmem=2G` and replace the `2G` with whatever amount of memory you require. You should note that if your job exceeds the amount of memory you have requested then it will be killed, so make sure you ask for enough. If you don't know how much to request then ask someone in bioinformatics for advice.

If you are running a multi-threaded application which requires more than one CPU core to complete then you should tell the scheduler how many cores it is going to use. The options to do this are `-pe cores 8` which would specify 8 cores. Simply change the last number to the number of cores you actually require. The options below would be suitable for a job which requires 8 cores and 2GB RAM.

```
-l h_vmem=2G -pe cores 8
```

Submitting to the queue

There are three different ways to interact with the gridengine queue and they're appropriate for different types of job.

1) Interactive sessions

An interactive session simply provides you with a login shell on a free compute node. You don't need to specify what it is that you're going to run and you can run as many commands within the compute session as you need to. The session will continue until you manually log out of the compute node and come back to the head node.

Interactive sessions are only normally used where you want to run a single intensive application which requires user interaction. An example would be running a graphical application such as SeqMonk, where you don't want the processing to be done on the head node so you'd start an interactive session on a compute node and launch the program from there.

You shouldn't use interactive sessions to run multiple large compute jobs, it's much better to use the other submission options for these. Firstly this is because jobs started within an interactive session can't be monitored by the main gridengine queue so may end up clashing with other running jobs. Also you can only start jobs on the physical server you are assigned in an interactive session, whereas the main queue can distribute your jobs over all the physical servers in the cluster where they will complete much more quickly.

To start an interactive session simply type 'qlogin' on the head node.

\$ qlogin

```
Your job 1062 ("QLOGIN") has been submitted
waiting for interactive job to be scheduled ...
Your interactive job 1062 has been successfully scheduled.
Establishing /bi/apps/ssh_wrapper/qlogin_wrapper session to host
compute-0-3.local ...
Last login: Fri Feb  1 09:05:53 2013 from rocks1.local
Rocks Compute Node
Rocks 6.0 (Mamba)
Profile built 14:16 21-Jan-2013

Kickstarted 14:23 21-Jan-2013
[andrewss@compute-0-3 ~]$
```

2) Interactive jobs

The second type of submission is an interactive job. Instead of starting a session on a remote node this option sends a single command to a remote node where it is processed. It looks from the user side as if you are running the job on the head node, but behind the scenes the processing is actually happening on a processing node.

The command to launch this type of job is `qrsh`, which you can follow by the command you want to run. As a simple example you can run "`qrsh hostname`" to show that the command is really being passed to a remote node for processing. In the example below 10 `qrsh` calls to 'hostname' are made and you can see the job being passed to different servers in the cluster.

```
$ for i in {1..10}; do qrsh hostname; done
compute-0-0.local
```

```
compute-0-1.local
compute-0-2.local
compute-0-0.local
compute-0-1.local
compute-0-2.local
compute-0-0.local
compute-0-1.local
compute-0-2.local
compute-0-0.local
```

There are a couple of extra options you probably want to use when running `qrsh`. By default it will start up a new login session on a compute node to run your command which will change the modules you have loaded and will put you in a different directory. This means that commands which work on the head node won't work on the compute node if they rely on any changes in the local environment, such as loading modules.

```
$ module load bowtie
$ bowtie --version
bowtie version 0.12.9
64-bit
Built on igml
Sun Dec 16 14:36:32 EST 2012
Compiler: gcc version 4.1.2 20080704 (Red Hat 4.1.2-50)
Options: -O3 -m64 -Wl,--hash-style=both
Sizeof {int, long, long long, void*, size_t, off_t}: {4, 8, 8, 8, 8, 8}

$ qrsh bowtie --version
bash: bowtie: command not found
```

In order to preserve the local environment when running the command you need to add the `'-v'` option to the `qrsh` command.

```
$ qrsh -v bowtie --version
bash: module: line 1: syntax error: unexpected end of file
bash: error importing function definition for `module'
bowtie version 0.12.9
64-bit
Built on igml
Sun Dec 16 14:36:32 EST 2012
Compiler: gcc version 4.1.2 20080704 (Red Hat 4.1.2-50)
Options: -O3 -m64 -Wl,--hash-style=both
Sizeof {int, long, long long, void*, size_t, off_t}: {4, 8, 8, 8, 8, 8}
```

You will see that the command now works, but produces a couple of error messages at the top. These are the result of a bug in gridengine, which is annoying but doesn't affect the function of the commands so you can safely ignore these.

The other option you will commonly pass to `qrsh` is the `'-cwd'` option which moves you to the same directory on the compute node as you were in when you ran the `qrsh` command.

You will need to add this if you are reading or writing any files which are accessed via a relative path.

```
$ pwd
/bi/group/bioinf

$ qssh pwd
/bi/home/andrewss

$ qssh -cwd pwd
/bi/group/bioinf
```

You can run multiple `qssh` jobs by putting them into the background but we have found that if you do this on too large a scale then the jobs won't automatically complete, but will stall at the end until you put them back into the foreground at which point they terminate cleanly. You will also need to keep the login session active whilst any `qssh` jobs are running since any output is returned into the shell from which the jobs were launched.

3) Non-interactive jobs

The preferred way to submit any large batch jobs is through the use of the '`qsub`' command. This command is largely similar to the `qssh` command but with a couple of important differences. The main difference between the two in terms of their functionality is that `qssh` is supposed to be used for jobs where your session will stay active for the duration of the job, whereas `qsub` can be used for jobs which can be submitted and then left to run with no further user interaction.

The other major difference between `qssh` and `qsub` is that whereas `qssh` allows you to simply specify a command to run, `qsub` expects you to produce a script which contains the commands you want to run in the job. The output of `qsub` is also presented as a set of files, rather than being sent back to the terminal from which the job was launched.

A simple script submitted via `qsub` might look like this:

```
#!/bin/bash
bowtie --version
```

You would submit this file (called `bowtie_version.sh`) to the queue as follows:

```
$ qsub -V -cwd bowtie_version.sh
Your job 1094 ("bowtie_version.sh") has been submitted

$ ls -ltr
-rw-r--r-- 1 andrewss bioinf 28 Feb 21 11:08 bowtie_version.sh
-rw-r--r-- 1 andrewss bioinf 0 Feb 21 11:09 bowtie_version.sh.e1094
-rw-r--r-- 1 andrewss bioinf 84 Feb 21 11:09 bowtie_version.sh.o1094
```

Since we added the '`-cwd`' option the output comes back to the same directory as we started in, and you get two files produced, one for the standard output from the job and the other for the standard error. The names will be the same as the script you submitted and will use `e[job_id]` and `o[job_id]` as suffixes by default. You can change the names of the output files

by using the `-o` and `-e` options to `qsub`. If you want to merge your output together into a single file you can also add `-j y`.

```
$ qsub -cwd -V -o out.txt -e err.txt bowtie_version.sh
Your job 1095 ("bowtie_version.sh") has been submitted
```

```
$ ls -ltr
-rw-r--r-- 1 andrewss bioinf 28 Feb 21 11:08 bowtie_version.sh
-rw-r--r-- 1 andrewss bioinf  0 Feb 21 11:13 err.txt
-rw-r--r-- 1 andrewss bioinf 84 Feb 21 11:13 out.txt
```

```
$ qsub -cwd -V -o all.txt -j y bowtie_version.sh
Your job 1096 ("bowtie_version.sh") has been submitted
```

```
$ls -ltr
-rw-r--r-- 1 andrewss bioinf 28 Feb 21 11:08 bowtie_version.sh
-rw-r--r-- 1 andrewss bioinf 84 Feb 21 11:14 all.txt
```

If you don't want to have to write your commands into a file then you can issue a command directly by telling `qsub` that your job is a 'binary' job as opposed to a script, using the option `-b y`.

```
$ qsub -cwd -b y hostname
Your job 233604 ("hostname") has been submitted
```

```
$ls -ltr
total 27
-rw-r--r-- 1 andrewss bioinf  0 Nov 18 10:18 hostname.e233604
-rw-r--r-- 1 andrewss bioinf 18 Nov 18 10:18 hostname.o233604
```

```
$ cat hostname.o233604
compute-0-1.local
```

With an explicit output filename then this can be much cleaner

```
$ for i in {1..10}; do qsub -cwd -o hostname.out -j y -b y hostname; done
Your job 233606 ("hostname") has been submitted
Your job 233607 ("hostname") has been submitted
Your job 233608 ("hostname") has been submitted
Your job 233609 ("hostname") has been submitted
Your job 233610 ("hostname") has been submitted
Your job 233611 ("hostname") has been submitted
Your job 233612 ("hostname") has been submitted
Your job 233613 ("hostname") has been submitted
Your job 233614 ("hostname") has been submitted
Your job 233615 ("hostname") has been submitted
```

```
$ ls
hostname.out
```

```
$ cat hostname.out
compute-0-1.local
compute-0-1.local
compute-0-2.local
compute-0-1.local
compute-0-1.local
compute-0-1.local
```

```
compute-0-1.local
compute-0-4.local
compute-0-1.local
compute-0-1.local
```

Rather than have all of the jobs called 'hostname' you can pass the `-N` option to give them a more individual name.

```
$ for i in {1..10}; do qsub -cwd -o hostname.out -N hostname_${i} -j
y -b y hostname; done
Your job 233627 ("hostname_1") has been submitted
Your job 233628 ("hostname_2") has been submitted
Your job 233629 ("hostname_3") has been submitted
Your job 233630 ("hostname_4") has been submitted
Your job 233631 ("hostname_5") has been submitted
Your job 233632 ("hostname_6") has been submitted
Your job 233633 ("hostname_7") has been submitted
Your job 233634 ("hostname_8") has been submitted
Your job 233635 ("hostname_9") has been submitted
Your job 233636 ("hostname_10") has been submitted
```

Changing Default Options

If you have a set of options which you always add to your `qsub` or `qrun` commands then rather than having to type them out each time you can have them added to every future command by default.

To do this you need to create a file called `.sge_request` (note the leading dot) and put it in the top level of your home directory. Within this file you can put one option per line which will all then be added to your commands by default after that.

For example if your `.sge_request` file looks like this:

```
$ cat .sge_request
-cwd
-V
-b y
-j y
```

You can simply run:

```
$ qsub hostname
Your job 233665 ("hostname") has been submitted

$ ls
hostname.o233665
```

As the `-cwd -V -b y -j y` will all have been added from the file. If you need to remove these options then you can add `-clear` to the start of your command to remove the default options.

Monitoring jobs in the queue

Once you have submitted some jobs to the queue there will be no indication of progress until the job is returned to you. However you can monitor the current state of the queue to see what jobs you have running and where they are.

For simple monitoring there is a command line tool called `qstat` which will show you what is currently in the queue. If you run this on its own it will show you a summary of all of the jobs you currently have running.

```
$ qstat
job-ID prior name user state submit/start at queue slots ja-task-ID
-----
1135 0.00000 test_job_1 andrewss qw 02/21/2013 11:27:59 1
1136 0.00000 test_job_2 andrewss qw 02/21/2013 11:27:59 1
1137 0.00000 test_job_3 andrewss qw 02/21/2013 11:27:59 1
1138 0.00000 test_job_4 andrewss qw 02/21/2013 11:27:59 1
1139 0.00000 test_job_5 andrewss qw 02/21/2013 11:27:59 1
1140 0.00000 test_job_6 andrewss qw 02/21/2013 11:27:59 1
1141 0.00000 test_job_7 andrewss qw 02/21/2013 11:27:59 1
1142 0.00000 test_job_8 andrewss qw 02/21/2013 11:27:59 1
1143 0.00000 test_job_9 andrewss qw 02/21/2013 11:27:59 1
1144 0.00000 test_job_1 andrewss qw 02/21/2013 11:27:59 1
```

If you add `-f` to the command it will break the jobs down based on the physical servers on which they are being run and will show you some summary information on how heavily loaded each of those servers is.

```
$ qstat -f
queueName qtype resv/used/tot. load_avg arch states
-----
all.q@compute-0-0.local BIP 0/2/64 1.53 lx26-amd64
1137 0.55500 test_job_3 andrewss r 02/21/2013 11:28:09 1
1141 0.55500 test_job_7 andrewss r 02/21/2013 11:28:09 1
-----
all.q@compute-0-1.local BIP 0/2/64 1.57 lx26-amd64
1138 0.55500 test_job_4 andrewss r 02/21/2013 11:28:09 1
1142 0.55500 test_job_8 andrewss r 02/21/2013 11:28:09 1
-----
all.q@compute-0-2.local BIP 0/4/64 1.50 lx26-amd64
1135 0.55500 test_job_1 andrewss r 02/21/2013 11:28:09 1
1139 0.55500 test_job_5 andrewss r 02/21/2013 11:28:09 1
1143 0.55500 test_job_9 andrewss r 02/21/2013 11:28:09 1
-----
all.q@compute-0-3.local BIP 0/3/64 1.51 lx26-amd64
1136 0.55500 test_job_2 andrewss r 02/21/2013 11:28:09 1
1140 0.55500 test_job_6 andrewss r 02/21/2013 11:28:09 1
1144 0.55500 test_job_1 andrewss r 02/21/2013 11:28:09 1
```

If you want to see jobs for all users you can use `qstat -u "*"`

```
$ qstat -u "*"
job-ID prior    name         user          state submit/start at   queue                          slots ja-task-ID
-----
 1161 0.55500 test_job_1 andrewss      r   02/21/2013 11:31:39 all.q@compute-0-3.local         1
 1162 0.55500 test_job_2 andrewss      r   02/21/2013 11:31:39 all.q@compute-0-1.local         1
 1163 0.55500 test_job_3 andrewss      r   02/21/2013 11:31:39 all.q@compute-0-0.local         1
 1164 0.55500 test_job_4 andrewss      r   02/21/2013 11:31:39 all.q@compute-0-2.local         1
 1165 0.55500 test_job_5 andrewss      r   02/21/2013 11:31:39 all.q@compute-0-3.local         1
 1166 0.55500 test_job_6 andrewss      r   02/21/2013 11:31:39 all.q@compute-0-1.local         1
 1167 0.55500 test_job_7 andrewss      r   02/21/2013 11:31:39 all.q@compute-0-0.local         1
 1168 0.55500 test_job_8 andrewss      r   02/21/2013 11:31:39 all.q@compute-0-2.local         1
 1169 0.55500 test_job_9 andrewss      r   02/21/2013 11:31:39 all.q@compute-0-3.local         1
 1170 0.55500 test_job_1 andrewss      r   02/21/2013 11:31:39 all.q@compute-0-1.local         1
 1171 0.55500 another_te bigginsl      r   02/21/2013 11:34:09 all.q@compute-0-1.local         1
 1172 0.55500 another_te bigginsl      r   02/21/2013 11:34:09 all.q@compute-0-2.local         1
 1173 0.55500 another_te bigginsl      r   02/21/2013 11:34:09 all.q@compute-0-0.local         1
 1174 0.55500 another_te bigginsl      r   02/21/2013 11:34:09 all.q@compute-0-1.local         1
 1175 0.55500 another_te bigginsl      r   02/21/2013 11:34:09 all.q@compute-0-2.local         1
 1176 0.55500 another_te bigginsl      r   02/21/2013 11:34:09 all.q@compute-0-0.local         1
 1177 0.55500 another_te bigginsl      r   02/21/2013 11:34:09 all.q@compute-0-3.local         1
 1178 0.55500 another_te bigginsl      r   02/21/2013 11:34:09 all.q@compute-0-1.local         1
 1179 0.55500 another_te bigginsl      r   02/21/2013 11:34:09 all.q@compute-0-2.local         1
 1180 0.55500 another_te bigginsl      r   02/21/2013 11:34:09 all.q@compute-0-0.local         1
```

For a more graphical view of the queue you can use the `qmon` program. This will initially open up a small toolbar.



The top left two buttons allow you to view either the list of active jobs or the load state of the machines in the cluster.

QMON +++ Job Control

GRID ENGINE

Job Control

Pending Jobs		Running Jobs		Finished Jobs	
JobId	Priority	JobName	Owner	Status	Queue
1161	0.55500	test_job_1	andrewss	r	all.q@compute-0-
1162	0.55500	test_job_2	andrewss	r	all.q@compute-0-
1163	0.55500	test_job_3	andrewss	r	all.q@compute-0-
1164	0.55500	test_job_4	andrewss	r	all.q@compute-0-
1165	0.55500	test_job_5	andrewss	r	all.q@compute-0-
1166	0.55500	test_job_6	andrewss	r	all.q@compute-0-
1167	0.55500	test_job_7	andrewss	r	all.q@compute-0-
1168	0.55500	test_job_8	andrewss	r	all.q@compute-0-
1169	0.55500	test_job_9	andrewss	r	all.q@compute-0-
1170	0.55500	test_job_1	andrewss	r	all.q@compute-0-
1171	0.55500	another_te	bigginsl	r	all.q@compute-0-
1172	0.55500	another_te	bigginsl	r	all.q@compute-0-
1173	0.55500	another_te	bigginsl	r	all.q@compute-0-
1174	0.55500	another_te	bigginsl	r	all.q@compute-0-
1175	0.55500	another_te	bigginsl	r	all.q@compute-0-
1176	0.55500	another_te	bigginsl	r	all.q@compute-0-
1177	0.55500	another_te	bigginsl	r	all.q@compute-0-

Refresh Submit Tickets Force Suspend Resume Delete Reschedule Select All Why? Hold Priority Qalter Clear Error Customize Done Help

QMON +++ Cluster Queues

Cluster Queue Control

Refresh

Tickets

Customize

Done

Help

Add

Clone

Modify

Delete

Show Detached Settings

☐ Force

Suspend

Resume

Disable

Enable

Reschedule

Clear Error

Load

☐ c
 ☐ a
 ☐ A
 ☐ E

Explain

Cluster Queues		Queue Instances					Hosts			
Host	Arch	#CPU	#Sock	#Core	LoadAvg	%CPU	MemUsed	MemTotal	SwapUse	SwapTot
compute-0-0	lx26-amd64	64	4	32	0.62	1.0%	5.6G	504.9G	0.0	1000.0M
compute-0-1	lx26-amd64	64	4	32	0.66	1.0%	5.8G	504.9G	0.0	1000.0M
compute-0-2	lx26-amd64	64	4	32	0.56	0.9%	2.0G	126.1G	2.6M	1000.0M
compute-0-3	lx26-amd64	64	4	32	0.61	1.0%	2.0G	126.1G	864.0K	1000.0M

Removing jobs from the queue

If you decide that having started a job you no longer want it to run then you can delete it from the queue at any time. You can only delete your own jobs though.

To remove a job first use `qstat` to find the job id for the job you want to remove. You can then use `qdel` to remove the job. You can specify multiple ids separated by spaces.

```
$ qstat
```

job-ID	prior	name	user	state	submit/start	at queue	slots	ja-task-ID
1183	0.00000	test_job_1	andrewss	qw	02/21/2013	11:48:34	1	
1184	0.00000	test_job_2	andrewss	qw	02/21/2013	11:48:34	1	
1185	0.00000	test_job_3	andrewss	qw	02/21/2013	11:48:34	1	
1186	0.00000	test_job_4	andrewss	qw	02/21/2013	11:48:34	1	
1187	0.00000	test_job_5	andrewss	qw	02/21/2013	11:48:34	1	
1188	0.00000	test_job_6	andrewss	qw	02/21/2013	11:48:34	1	
1189	0.00000	test_job_7	andrewss	qw	02/21/2013	11:48:34	1	
1190	0.00000	test_job_8	andrewss	qw	02/21/2013	11:48:34	1	
1191	0.00000	test_job_9	andrewss	qw	02/21/2013	11:48:34	1	
1192	0.00000	test_job_1	andrewss	qw	02/21/2013	11:48:34	1	

\$ qdel 1183 1184 1185

andrewss has registered the job 1183 for deletion
andrewss has registered the job 1184 for deletion
andrewss has registered the job 1185 for deletion

\$ qstat

job-ID	prior	name	user	state	submit/start	at queue	slots	ja-task-ID
1186	0.555	test_job_4	andrewss	r	02/21/2013	11:48:39	all.q@compute-0-1.local	1
1187	0.555	test_job_5	andrewss	r	02/21/2013	11:48:39	all.q@compute-0-2.local	1
1188	0.555	test_job_6	andrewss	r	02/21/2013	11:48:39	all.q@compute-0-3.local	1
1189	0.555	test_job_7	andrewss	r	02/21/2013	11:48:39	all.q@compute-0-1.local	1
1190	0.555	test_job_8	andrewss	r	02/21/2013	11:48:39	all.q@compute-0-0.local	1
1191	0.555	test_job_9	andrewss	r	02/21/2013	11:48:39	all.q@compute-0-2.local	1
1192	0.555	test_job_1	andrewss	r	02/21/2013	11:48:39	all.q@compute-0-3.local	1

Clusterflow Pipelines

To make life easier for people running batches of jobs using common bioinformatics tools we have written a pipelining system called clusterflow (or cf for short) which makes the process of defining and submitting `qsub` jobs much easier. Clusterflow provides a series of pipelines which allow you to easily run sets of analyses, potentially involving several different programs, with a single simple command.

Accessing Clusterflow

Clusterflow itself is just another software package installed as an environment module. To import clusterflow into your current session you simply need to do:

```
module load cf
```

This provides access to the `cf` program which is what you use for all interaction with the clusterflow system.

Configuring clusterflow

Clusterflow has the ability to send you emails to inform you about the progress of jobs you are running. Before using clusterflow for the first time you should set up your notification configuration. To do this you simply run the command below and follow the prompts.

```
cf --make_config
```

Modules and Pipelines

The two main core concepts in clusterflow are modules and pipelines. A module is simply a wrapper for a single program (FastQC, Bismark, Tophat etc). A pipeline is a structured set of modules arranged in a particular order (eg run FastQC, then trim galore, then tophat). To see the modules available in clusterflow you can run:

```
$ cf --list_modules
=====
Cluster Flow - available modules
=====
Available modules:
  Directory ./
  Directory /bi/home/andrewss/clusterflow/modules/ (not found)
  Directory /bi/apps/clusterflow/0.1/modules/
    - bismark_align
    - bismark_deduplicate
    - bismark_messy
    - bismark_methXtract
      - bismark_tidy
      - bowtie
      - bowtie1
      - bowtie2
      - cf_download
      - cf_run_finished
      - cf_runs_all_finished
      - fastq_screen
      - fastqc
      - hicup
```


- sra_abidump
- sra_fqdump
- tophat
- trim_galore

To see the pipelines you run:

```
$ cf --list_pipelines
=====
Cluster Flow - available pipelines
=====
Installed pipelines:
  Directory ./
  -
  Directory /bi/home/andrewss/clusterflow/pipelines/
  - pbat
  - trim_bowtie
  - trim_bowtie2
  - trim_bowtie_miRNA
  Directory /bi/apps/clusterflow/0.1/pipelines/
  - bismark
  - bismark_pbat
  - bismark_singlecell
  - fastq_bismark
  - fastq_bowtie
  - fastq_hicup
  - fastq_pbat
  - fastq_tophat
  - sra_bismark
  - sra_bismark_RRBS
  - sra_bowtie
  - sra_bowtie1
  - sra_bowtie2
  - sra_bowtie_miRNA
  - sra_hicup
  - sra_pbat
  - sra_tophat
  - sra_trim
  - trim_bowtie_miRNA
  - trim_tophat
```

Running clusterflow commands

To run a clusterflow pipeline you simply issue a command with the following structure

```
cf [name of module or pipeline] [files you want to process]
```

Clusterflow will then create and submit a series of qsub jobs so that all of the programs in the pipeline are run for all of the files you want to process.

There are two other options you may need to add to your cf command. If your pipeline needs to map data to a reference genome then it will need to know which genome you want to use. There are a set of pre-configured genomes already available and to access these you run:

```
$ cf --list_genomes
```

Once you have found the name of the genome you want to use you can just append that to the clusterflow command:

```
cf --genome GRCm38 [name of module or pipeline] [files you want to process]
```

The other option you can set is whether a set of sequence files you're submitting are single or paired end. The program will try to automatically identify paired end files based on their names, but if you see that the automatic detection has incorrectly identified your paired end data as single end, or if you have files with paired end names that you want to process individually then you can force single or paired end mode by adding `--single` or `--paired` to the `cf` command. If you force the use of paired end mode then the program will assume that pairs of files will be placed next to each other in the list of files you submit.

```
cf --paired [name of module or pipeline] [paired list of files]
```

Monitoring clusterflow jobs

Clusterflow jobs will appear in the normal gridengine queue alongside other `qsub` jobs so you can just use `qstat` to see them. You will find that large pipelines will submit a large number of jobs such that the output of `qstat` can become hard to read. To make things easier to understand clusterflow therefore has its own monitoring program which shows you the pipelines you've submitted and their progress. To access this you simply run:

```
cf --qstat
```

..or if you want to see pipelines from all users:

```
cf --qstatall
```

Further help with clusterflow

You can see more clusterflow options by running:

```
cf --help
```

If you want to try writing your own pipelines (easy) or modules (hard) then you can also look at the full clusterflow manual which is available from the projects web site at

<http://www.bioinformatics.babraham.ac.uk/projects/clusterflow/>

Problems

If you encounter any problems with the cluster please get in touch with Simon Andrews in the bioinformatics group, Mark Thompson in computing or Nicolas Rodriguez in computational neurobiology who should be able to help, or to point you to the right person to speak to.

Further help

This document is intended to give you the information you need to start working in the cluster environment. If you need more general help with working in a linux command line environment then please ask a member of the bioinformatics group for the separate guide which covers the basic unix commands you will need to work on the cluster. You can also approach anyone in the bioinformatics group to discuss any work you want to do on the cluster and they will be able to help you with this.