Table of Contents

SBCS-Informatics

This is a README explaining how to use the Apocrita HPC resource, it contains:

- Instructions on how to submit jobs and use Apocrita generally
- An introduction to the command line and unix
- A slightly more advanced section with tips and tricks for the more seasoned user who has gotten to know Apocrita and Unix better

Click on the menu on the left to navigate through the website. This document is also available as a single pdf download.

Contributing

There are two main ways of contributing content or asking questions about this document.

1. Using GitHub, clicking the Edit link will take you to the repository of this document. There, you can either make a GitHub issue or a pull request.
2. You can find contact details here if you would rather send an email.

Using Unix and Apocrita

This section will help new users to get started using Apocrita, with information on how to log in and how to submit jobs as well as some handy information on the system and how it works.

The information here can also be seen as the first place to look when a question about Apocrita comes up. If there is any unanswered questions after that feel free to use the contact page.

1. What is Apocrita?

Cluster Structure

Apocrita is a High Performance Computing cluster that consists of many computers, called **nodes**, which are all interconnected. The user logs into one specific node, called a **head node**, from which the rest of the cluster can be accessed. Importantly, ***no jobs are run directly on the head node***. Instead, whenever the user wants to run a job, the details of the job is submitted to a software that handles scheduling of work on the cluster and it is dispatched to a work node when one is available.

Nodes

- 150 normal use nodes, two 6-core processors with 24GB RAM
- 11 so called **fat nodes** with four 12-core processors and 512GB RAM. These nodes are used for heavier calculations, in particular ones which require a lot of memory.

Data

Each user has a specific account on Apocrita which will have a data quota of 50GB in their home directory. Often this is nowhere near enough to run the analysis needed, therefore there is a so called **scratch** space for temporary files. This is where most people do their work but its important to note that that **scratch area is not backed up** and you should treat these as purely temporary. In the future a time-limit may be implemented, automatically deleting old files.

We are working on setting up an archival data space where the user can store data long term, for things like raw and processed data from finished projects that cannot be deleted but probably wont be used a lot in the future. In many cases there is project specific storage implemented.

System

Apocrita is running Scientific Linux 6.2.

2. Connecting to Apocrita

Get an account on the cluster

In order to log in you will need to get an account from ITS research. You request a user using the online ITS research form linked to below, when a user has been set up for you they will contact you with the details.

Request account form

This user account will have certain limitations, for example, a user has a set data quota on their home directory so that the cluster isnt overburdened. Additionally you will only have access to files and folders which have been specifically granted. Talk to your PI, perhaps there is a directory specific to your lab.

Logging in from Windows

Download PuTTY. In the host name window enter the server address e.g.

```
frontend1.apocrita.hpc.qmul.ac.uk
```

Enter the correct port number (22) and check on correct connection type (SSH). Now go to the data tab under connection and in auto-login username box enter your username (e.g.bt12345). Go back to the session tab and in the saved sessions box enter a name for this log in (e.g. frontend1). Save!

Now double click the name which should open up the Apocrita window. In the window, after your username, double click to bring up a highlighted bar. Now copy password and then right click to paste it into the box. The password won't appear but it should log on.

You can also use MobaXterm to log in.

Logging in from Mac

It's much easier on Macs as Apple's OS X is unix-based. If you just need the commandline, open Terminal (this is accessible from Spotlight or in Applications/Utilities).

```
ssh [youruserID]@frontend[1/2].apocrita.hpc.qmul.ac.uk
```

e.g.:

```
ssh btw000@frontend1.apocrita.hpc.qmul.ac.uk
```

If prompted, enter the password given by ITS research. You can also set up SSH keys that will be used when you log in from your computer instead of a password.

Display

Finally, if you're planning on using any graphical interface, specifically something which requires plots, add the X11 fowarding flag, `-X`. This instructs the terminal forward display items to your computer, that will allow you to see your plots etc. This is useful when working in R and viewing plots, otherwise you'll have to save them to .pdfs and download them

every time.

```
ssh -X btw000@frontend1.apocrita.hpc.qmul.ac.uk
```

Usefully, the `screen` feature allows you to keep a process running on a server without the need to be consistently connected to the network (e.g. when you're carrying your laptop between home and campus and can't be connected to the internet). This is worth remembering for if/when you have a job that requires a lot of time to run.

3. Launching a Job on Apocrita

When you log in to Apocrita you will be accessing a **head node**. It is very important that you do not simply run your scripts on the head node. These nodes are there to accomodate logins to the cluster, not workload. Instead, there is a scheduling program running on the cluster which takes instructions to your job and distributes the total load over all the worker nodes.

Using/Accessing an Apocrita machine to do work.

There are three methods to get your job running on the cluster. Generally speaking, if you `ssh` into Apocrita's frontend, you can instruct it to carry out your jobs on your quota of space using any of these methods.

There are two types of instruction for this:

- `qlogin` or `qrsh`: gives you access to one machine
- `qsub`: submits a job you describe in a file to the cluster

`qsub`

This is the easiest way to run your job on Apocrita. Even so, it is not as simple as running your command. Remember, don't run jobs on the head nodes.

You need to write a script with the instructions for your job, below you find the simplest version of such a script.

```
#!/bin/sh
#$ -cwd              # Set the working directory for the job to the current directory
#$ -V
#$ -l h_rt=24:0:0    # Request 24 hour runtime
#$ -l h_vmem=1G      # Request 1GB RAM
./code               # Your code goes here
```

After you have written and saved the script you feed it to `qsub` as such

```
qsub job_script.sh
```

There are several more options you can add to the header of your script which for example allows for more cores to be used.

For more details have a look at the ITS research website, or if you feel like you can take it, the qsub man page.

`qlogin` and `qrsh`

These commands will connect you to one of the worker nodes with the requested resources available for you to use. This will allow you to run your scripts/jobs interactively on a command line. This can be useful if you want to have more freedom in what you are doing than a script allows, but should be avoided for longer jobs as the frontends are rebooted fairly frequently which may kill your job. These commands take the same options as `qsub`.

Monitoring your job

Use `qstat` to show the status of all your jobs on Apocrita, and `qdel` to delete a job from the queue.

4. Basic UNIX Navigation

Where am I?

Files are organised in your allocated server space into folders, or 'directories', just like on your regular computer. You can check your current working directory (i.e. where you are) with the 'print working directory' command, `pwd`.

Checking the contents of directory

It's easy to check the contents of the directory you're currently in, using the list command, `ls`. Typing `ls` alone will list the filenames. However, if you require extra information you can add 'flags' after the command to give the computer further instructions:

```
ls          #prints filenames in a list
ls -a       #prints all filenames, including hidden files
ls -l       #'long' list, displays info including permissions.
ls -h       #prints the sizes of files in units you can read
ls -tr      #t for time sorted, r for reverse
```

The above flags (amongst others) can be combined together in one string for convenience. You may use one `-` (to indicate a flag) followed by all required characters, or separate them individually:

```
ls -lhatr          #has the same effect as
ls -l -h -a -t -r
```

This bit will print a list of files with all of the above information.

Changing directories

Navigate through directories using the 'change directory' command, `cd`. It will assume you're looking for a directory name that is within your current working directory

Change directory

```
cd directory_name/
```

Go back "up" one directory:

```
cd ../
cd ../../                   #you can go several at a time
cd ../../anthr_dir/subdir2  #as many up and down as you want
```

Return to home directory

```
cd ~/
```

or just

```
cd
```

> *Note: On Mac, it's possible to click and drag the desired location of a directory or file from the finder (by the icon) to the terminal. Just type `cd` followed by a space, click & drag, hit enter. You can do this from any starting point.*

5. Playing with Files

Creating a new directory

The `mkdir` command will create a directory inside your current location.

```
mkdir newdirectoryname
```

Moving files in local machine

The `mv` command will remove a file or directory from its current location and place it elsewhere. The syntax is, "move, current location, new location". It can also be used to rename files.

If the new location is a file, the file is renamed:

```
mv oldfile newfile
```

or if the target is a directory, the file is moved:

```
mv oldfile newdirectory/
```

Delete files or directories

Delete files with the 'remove' command, `rm`:

`rm filename`

or for a directory `rmdir`, the directory has to be empty:

`rmdir directory/` You can use `rm` to remove directories with the `-r` option, it removes files in a directory recursively. Be careful when using `rm`, once you hit enter, the files are gone. After adding or removing files and directories, you can check and make sure it's worked using the `ls` command.

Viewing different parts of an existing file

`less` shows a small portion of the file, `more` shows a larger portion (according to the manual, it will display your file 'one screen's worth of lines at a time'). `head` displays the first ten lines, and `tail` displays the last ten lines.

```
less filename.extension
more filename.extension
```

The head and tail commands can be modified to show a specific number of lines, with a flag `-n`, where n= number of lines:

```
head -15 file.txt   #displays the first 15 lines
tail file.txt       #displays the last 10 lines (default)
```

Searching for a pattern within a file

The `grep` function searches for a particular pattern of characters. The syntax for the grep function is: 'grep, "pattern", file-to-look-in'. e.g.

```
grep "scaffold" genome.fasta
```

6. Copying & Downloading files

Files can be copied in an around your computer, in and around your space on the servers, and between the two. The basic copy function, `cp`, is for within locations (i.e. your computer or the server). To go between the two, use 'secure copy', `scp`.

The basic syntax is 'cp, original file location, new file'.

Between a remote server and local machine

From local file to server:

```
scp /path/to/local/file.txt server:/path/to/server/directory_or_newfile.txt
```

For example:

```
scp /Users/bob/Desktop/data.csv btw666@frontend1.apocrita.hpc.qmul.ac.uk:/home/btw666/archive/2013/data.csv
```

From server to local directory on local machine:

```
scp btw666@frontend1.apocrita.hpc.qmul.ac.uk:/home/btw666/myoutput.pdf .
```

> *Note, "`.`" means "current directory".*

Within your home computer, or the servers

This is just the same, except replace `scp` with `cp`.

```
cp /path/to/file.extension /new/directory/
```

Using a graphical user interface (GUI)

Cyberduck (on Mac) or FileZilla (on Windows). These two are GUI programs which offer file-sharing via a straightforward drag-and-drop or by menu navigation.

Both programs have a facility for attaching your account and passwords to them so they'll log in automatically. Make sure you check in the preferences/options/settings to set the file transfer protocol to "ssh", as that's not usually the default.

> *Note that while these systems are convenient for moving and organising your files, that's all they do. Alas, it's back to the command line to do anything with your files.*

Downloading files from the web

`wget` is short for "web-get", and will download a target file into your current working directory, e.g.

```
wget http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
```

Here wget is used to download an old version of itself

7. Dealing with Compressed files

As disk space is always limited and factoring in the large number of users, it is vital that everyone takes care storing their data properly on Apocrita. Always compress and archive files that aren't used.

Files

In genomics it is very common to have big files which contain sequence information or mapping files that can tell software and user where each sequence read fits against a certain reference. These files are essentially text files like any other and are often human-readable (for your convenience). Unfortunately that means these types of files take up an enormous amount of space, but there are ways to mitigate this issue. Compression really just means that files are manipulated in such a way that information is denser, making them smaller, although this also means that they cannot be viewed/read unless you reverse the process. The simpler the file, the smaller it gets after compression because there wasn't much information to begin with. It is important to **compress anything that you are not currently using**, as this will save a lot of space on the cluster.

> *Remember that some tools and applications are able to work with compressed files.*

Directories

In some cases it isnt a single large file that is causing storage issues. Some programs create a large system of folders filled with lots and lots of tiny files. This can add up quickly and because of the way file systems work there is a "minimum size" that a file can occupy on disk (if you want to know more about block size). The solution to this problem is to make an archive of the directory. In a Linux/Unix environment the most common archiver is a program called `tar`. `tar` was created to handle problems with block size and writes a single new file, often called tarball, containing everything in the directory. This is not compressed so what you often see is compressed tar archives where the tarball has been run through `gzip`. You should do this as well.

`tar`

Use the `tar` command to create, and extract, archives of folders.

```
tar -c directory/ > directory.tar
```

`-c` is for create. This creates a new file called `directory.tar` but the original directory is still there. You can now remove the directory.

`gzip`

Gzip is the go-to program to use for compressing files on any Unix system. Here is how simple it is to use:

```
gzip file
```

This zips the file up and gives it the .gz extension, note that this replaces the file with the compressed version.

Extracting compressed files, e.g. .zip .tar.gz .tgz

The command for unzipping a file depends on the type of archive it is (i.e. its extension)

```
unzip file.zip          #for .zip
gunzip file.gz          #for .gz
tar -zxvf file.tar.gz   #for .tar.gz
tar -zxvf file.tgz      #for .tgz
```

```
tar -jxvf file.tar.bz   #for .tar.bz
```

> *Notable is that a file may have any extension, it is actually just a part of the file name. However, using proper extensions is a way of letting the user know what kind of file it is. When you move, archive and unzip files etc, make sure that you keep correct extensions on your files, or maybe you wont remember how to open it next time.*

8. Compiling a piece of software

Software for Unix systems is generally distributed as archives. To install a software package:

Download it (perhaps with `wget`). Then decompress it (perhaps with `unzip` or `gunzip`). Then check the installation instructions, usually in an `INSTALL` or `README` file. Often this will involve typing `make` in the directory.

Some readme files are more useful than others. If there is no help in the readme, but there is a file called 'makefile' in the directory, just type `make` and that should work.

9. R

You can run R in the Unix command line. To do this, simply type `R`.

This essentially turns the command window into the R console you're familiar with. From here, you can do all of the same things as you can in regular R. There are a few things worth noting:

Reading files

Alas, the command line is completely mouseless, so the `file.choose()` command for reading data using `read.table()` or `read.csv()` is no longer available. Instead, type in the name of the file you want in inverted commas. Fortunately, R will look for the file in the working directory you were in when you started the console, so it may actually be even more straightforward. However, if you're not in the same directory, you'll need to type the path the same way as you would when copying, moving, or navigating files & folders.

```
mydata <- read.table("data.txt", header=T)
otherdata <- read.csv("work-monthly/RData/data.csv")
```

Creating pdf file of a plot

Unless you're using a screen, you won' t be able to see any plots you've made in R and so this will be necessary. Even if you are, once you've made the plot you want for your paper, you'll still need to save it. R can do this by opening its own workspace inside a pdf file, called a 'device'. Other devices include the Quartz & X11 windows you're already familiar with looking at your plots on.

First, call the `pdf()` function to tell it what the file should be called, then make your plot as you like it. To stop working in that file call the `dev.off()` function. You can then download it from the server to view it.

```
pdf("plot.pdf")
plot(object)
dev.off()
```

More information on R's ability to use devices (it can do more than just PDFs, and also have several open at once) can be found in the help files.

10. Galaxy

There is a Galaxy server set up on Apocrita.

Login

https://galaxy.hpc.qmul.ac.uk/ - This is the link to the web interface.

To log in, use your entire QMUL email address ie j.doe@qmul.ac.uk as username and your Apocrita password.

SFTP

You can use SFTP to transfer large files to the Galaxy server. Use a GUI program to connect to the server using SFTP and transfer your files. The hostname is the same as the website, galaxy.hpc.qmul.ac.uk.

Remember to set SFTP and not FTP.

Specifics

The actual server runs on frontend2 but it uses DRMAA to submit jobs to the cluster queue. It will run jobs as the "galaxy" user, that means that the user starting the job is not the one that executes it on the cluster, but the username is listed in the job name in the queue.

For an unknown reason the command line version of sftp may or may not work for transfering files. There seems to be some issue with permissions of files sometimes. Use Cyberduck if you can. Contact Adrian Lärkeryd or ITS-Research if you're having issues with the file transfer.

This server is still very much experimental and things change. If something is not working or you need a new tool installed do not hesitate to contact Adrian.

Repeatexplorer

Is set to use almost an entire fat node on Apocrita. Unfortunately it doesnt allow for dynamically changing the resource requirement, thus even a small job will request a lot of resources from the cluster and be stuck in the SGE queue for longer than necessary. The main requirement of this tool seems to be memory and the biggest nodes on the cluster have 512GB of RAM. It is possible that some very large jobs could exceed that limit and be killed.

11. Handy Shortcuts

- Get your quota information `qmquota -s`
- Make a soft link to a file `ln -s /path/to/source/file.txt ./destination_file.txt`
- Find username or information on a user with `finger <search-term>`

Apocrita job stats

ITS-R has a webpage for statistics where you can see the load on the cluster but more importantly you can see your completed jobs.

stats.hpc.qmul.ac.uk

On the left hand menu there is a View your job detail in which you can see all the jobs ran on the cluster recently.

Advanced

This section contains more advanced information as well as tips and tricks for users who know their way around. Make sure you know what a command found here does before running it. If you are curious about something and want to know if it can be applied to your situation or research, contact us.

1. Apocrita Specifications

Apocrita cluster nodes

Thin nodes

- 150 Nodes
- Dual 6-core Intel Westmere (E5645) - 2.4GHz
- Memory 24GB

Hyperthreading is currently enabled on the Intel CPUs but each (serial) job is allocated a single real core

Fat Nodes

- 11 Nodes
- Four 12-core AMD Bulldozer (6234) - 2.4GHz
- Memory 512GB

Interconnect

Gigabit Ethernet

Queueing system

Sun Grid Engine 8.0.0e

Compilers

Intel, Solaris Studio, Open64, Portland

Parallel libraries

OpenMPI

SSH-able nodes

SM11

A "set free" fat node that SBCS users can ssh directly into. The GPFS is mounted and it should work like any other part of Apocrita, just that the Sun Grid Engine doesnt schedule jobs here so that it is free for users to handle themselves. Check whether or not the machine is free (top, htop) before you start something, and maybe use a nice value. To log in, just type `ssh sm11` when logged into a frontend. You can also make an ssh shortcut in your config file, explained below.

- Four 12-core AMD Opteron(TM) Processor 6234 - 2.4GHz
- Mem 512GB

- 16TB local scratch disk

Prometheus

Purchased with NERC money by Nichols & Wurm.

This machine is not connected to Apocrita. It runs Ubuntu 14.04 and is administrated by SBCS users. A user has to be created for anyone who wants to use this hardware, see below for contact details.

- 2 10-core Intel Xeon CPU E5-2680 v2s clocked at Min:1199.953Mhz Max:2538.265Mhz with hyperthreading ON
- Kernel 3.18.9-031809-generic x86_64
- Mem 512GB
- 13TB local scratch disk
- Runs Ubuntu
- Docker installed

Contact: a.larkeryd@qmul.ac.uk, y.wurm@qmul.ac.uk, r.a.nichols@qmul.ac.uk

VM21 & VM22

The VMs are two nodes of the Apocrita cluster on which a KVM is running. These were set up in order to have Docker running on the cluster, however this is not yet up and running properly. Users can log in to these nodes and run their programs, however there are some caveats. Only a few core modules are available at the moment (module avail). There is also a possibility that the virtual machine is slowing the nodes down. Benchmarks are to be held to determine exact implications of this. Possible that one machine will be reinstalled without the KVM.

- VM21 and VM22 are running on frontend5 and frontend6 respectively
- Four 10-core Intel Xeon E5-4640 v2 - 2.20GHz
- Mem 516GB

GPU

There are no GPUs on Apocrita, but there is one node attached to Taurus with two NVidia C2070 GPU cards in it.

2. SSH Keys

Keys

You can set up a pair of SSH keys for a more secure as well as password-less login to Apocrita. This is done by having a private key on your machine, and a matching public key on the remote server, when you try to log in these two match up and let you in without having to type the user password. The private key should never be shared with anyone as it will allow that person access to your login. This is why you should always **protect your private key** with a passphrase.

Key Generation

1. Open a terminal window (on Windows, use MobaXterm)
2. Enter `ssh-keygen` and hit enter
3. You will see `Enter file in which to save the key (/home/username/.ssh/id_rsa):` on the screen. Just hit enter here which will save the keys in their default location.
4. The program will now ask you for a passphrase. Please enter one (it is possible to create a key without it but don't, it's to protect from someone getting hold of your private key.)
5. There is a message telling you that the key pair has been created, the public key is now located in /home/my_username/.ssh/id_rsa.pub and the private key is /home/my_username/.ssh/id_rsa.

6. You are now ready to copy the **public key** to Apocrita

Public Key Copy

The process is different depending on which operating system you are using.

Windows and Linux

Here its very simple, open a terminal (or MobaXterm window) and type in `ssh-copy-id btw000@login.hpc.qmul.ac.uk` using your own username. Thats it. Now try your connection `ssh -X btw000@login.hpc.qmul.ac.uk` !

Mac

One of the few times having a mac will make you suffer extra work. You will have to manually copy your **public key** to a file located in your home directory on Apocrita.

1. Open a terminal and go to your home directory with `cd`
2. Use `scp` to copy the public key to Apocrita with this command `scp .ssh/id_rsa.pub btw000@login.hpc.qmul.ac.uk:~/`
3. Login to Apocrita `ssh btw977@login.hpc.qmul.ac.uk`
4. `cat ~/id_rsa.pub >> ~/.ssh/authorized_keys`
5. You can now log out and try the connection again to see if your keys work!

> *If you have Homebrew installed on your Mac you can use it to install `ssh-copy-id` and go from there.*

Shortcuts

Another convenience tip is to add shorthand names for your ssh logins. The address to the apocrita head node is quite long and arguably tedious to write. So, this being computer science, of course there is a setup that will allow you to simply type something like `ssh apocrita` on the commandline to connect.

1. `cd ~/.ssh`
2. Open or create the config file `nano config`
3. Add the following, you may call the Host whatever you like, here I'm using "apocrita":

```
Host apocrita
 Hostname frontend1.apocrita.hpc.qmul.ac.uk
 IdentityFile ~/.ssh/id_rsa
 User btw000
 ServerAliveInterval 300
```

4. Save and quit the editor
5. Change permissions of the file `chmod 600 ~/.ssh/config`
6. Try `ssh apocrita`

Another example, using apocrita (our newly created shortcut to frontend1) as a proxy to connect to SM11. This way all you need to connect to SM11 is `ssh sm11` .

```
Host sm11
    Hostname sm11
    User btw000
    ProxyCommand ssh apocrita nc %h %p
```

You may or may not want X11 forwarding and there are other options.

3. `zsh` defaults

These are instructions to set up your shell with some useful `zsh` settings like better autocompletion and history etc. It also contains a small set of useful aliases.

All you need to do

Save your own `.zshrc` file

This is the file that says what zsh should do when you log in. You have one in your home directory which you will have to remove in order for this to work. This way you can keep your previous `.zshrc` if you feel like going back.

```
cd
mv .zshrc .zshrc.BAK
```

Run the small installation

```
source /data/SBCS-Informatics/zsh_extensions
```

You should find yourself in a helpful shell. Have fun.

Details

`zsh` 5.0.7

The default version of zsh that every SBCS user has (unless explicitly changed) is version 4.3.10 which doesn't contain all the functionality used here. Because of that a check is made when logging in and if the version of `zsh` isnt 5.0.7 it is loaded.

Oh My Zsh

Oh My Zsh is a large package of settings, plugins and themes for zsh which is used as foundation for these defaults. It has a ton of options but it is kept fairly simple here, including the theme. It does allow for much more customisation which you can do for yourself.

Per directory history

This plugin is enabled to give you a comman line history that is specific to the directory you are in. That way when you go back to the folder where you carried out your analysis all those months ago you can just press the up-arrow on your keyboard and it will go through what last happened in this particular directory (instead of the failed installation of that python package you were wrestling with yesterday.)

Customising further

Perhaps you want some more fancy features than what is provided here? There are a multitude of themes that change appearance of your shell, or maybe its syntax highlighting you crave. The setup followed above inserted a sourcing of our general `.zshrc` file in your local version. That file is located in your home directory. In order to change the settings you need to replace your `.zshrc.` with the contents of the SBCS-Informatics-zshrc. That way you can change anything you like! Add plugins, try the random theme, add your own aliases or even build your own functions. You can of course write your own .zshrc and use other packages.

There are other shells available as well. `fish` for example, which claims to be "Finally, a command line shell for the 90s". However, `fish` does have slightly different syntax than `bash` and `zsh`, the two of which are very similar. Feel free to keep

looking, there are other more obscure things out there.

4. Oneliners

- Long list of bioinformatic and non-bio oneliners
- Get all sbcs users: `ldapsearch -x cn=sbcs | grep memberUid | sort`. Can email to @qmul.ac.uk directly

Contact

Please send an email if you need anything. If you have a question, or more information, you would like to add to the documentation, feel free to use the EDIT link up top and do a github pull request.

Adrian Lärkeryd

- a.larkeryd@qmul.ac.uk