Artemis HPC
Training
Introduction to the Artemis HPC

Hayim Dar, Nathaniel Butterworth
sih.training@sydney.edu.au

Sydney Informatics Hub
A Core Research Facility

HPC Access
Example: ssh -Y <username>@hpc.sydney.edu.au

Course Docs at https://goo.gl/7D2YFn
Introduction

- Sydney Informatics Hub (a Core Research Facility) alongside ICT

- Under the Research Portfolio, Core Research Facilities provide access to high-end research infrastructure, personnel and services: supporting quality research outcomes for USyd researchers
  

- Provides Artemis HPC and associated training and support
Informatics and Research Computing Expertise

Provides research computing expertise, training, and support

- Data science, analyses and support (bioinformatics, modelling, simulation, visualisation, machine learning, statistics)

- Training and workshops
  - High Performance Computing (HPC)
  - Programming (R, Python, Matlab, Scripting, GPU)
  - Code management (Git)
  - Bioinformatics (RNA-Seq, Genomics)

- Research Computing Support
  - Artemis HPC
  - Argus Virtual Research Desktop
  - Bioinformatics software support (CLC Genomics Workbench, Ingenuity Pathways Analysis)

- Events and Competitions
  - HPC Publication Incentive – High quality papers that acknowledge SIH and/or HPC/VRD
  - Artemis HPC Symposium
Overview of today’s course

- Connecting to Artemis
- Artemis directory structure and ‘projects’
- PBS Pro job scheduler
- PBS scripts
- Submitting and monitoring jobs
- Checking job output and resource usage
- Getting accounts on HPC and RDS via RDMP
Artemis HPC

- HPC stands for ‘High performance computing’
- Artemis is the name chosen for USyd’s HPC cluster
- [https://sydneyuni.atlassian.net/wiki/spaces/RC/pages](https://sydneyuni.atlassian.net/wiki/spaces/RC/pages)

- 3 Phases of Artemis, totalling:
  - 7,636 cores
  - 45 TB RAM
  - 108 NVIDIA V100 GPUs
  - 378 TB storage
  - CentOS 6.9
  - PBS Pro job scheduling software
  - 56 Gbps FDR InfiniBand
Artemis HPC

- Artemis is ideal for any computational analysis requiring:
  - Long walltime
  - High RAM usage
  - Analysis requiring multiple cores or multiple nodes
  - Large number of single-core jobs
  - Large input or output data

- Artemis is for everyone!
  - Goal to centralise USyd computational resources
  - This reduces the overheads of Faculty or School-based servers, so researchers do not have to worry about maintaining/updating the hardware and software
  - Provides a great incentive to funding bodies to fund projects where HPC is provided to the project at no cost by the host institution
Your use of the Artemis HPC

- No cost to USyd researchers!
- Access requires a unikey and valid RDMP (later)
- **Publications involving use of Artemis or other Sydney Informatics Hub services MUST acknowledge these free resources (this forms part of the user agreement)**

- **Suggested acknowledgement:**
  “The authors acknowledge the Sydney Informatics Hub and the University of Sydney’s high performance computing cluster Artemis for providing the high performance computing resources that have contributed to the research results reported within this paper.”

- **Benefits of acknowledgement:**
  - Publications with appropriate HPC acknowledgment are eligible for the **SIH HPC Publication Incentive**
  - Acknowledgements demonstrate research outputs arising from the USyd investment into Artemis and support staff: this drives further funding meaning ongoing upgrades and maintenance for our HPC and services
SIH HPC Publication Incentive

- **A $1000 prize pool!**
- To be eligible, papers must:
  1. Be published in a peer-reviewed journal
  2. Be authored or co-authored by a current University of Sydney staff member or student
  3. Include research supported by SIH or data analysed on Artemis HPC or NCI Raijin
  4. **Acknowledge the Sydney Informatics Hub** at the University of Sydney
  5. Ensure that the publication is listed under your Research Outputs on IRMA
  6. Not have been entered in a previous round of the incentive
- Contact [sih.info@sydney.edu.au](mailto:sih.info@sydney.edu.au) for more information
Connecting to Artemis

You connect to Artemis remotely: from any computer, any location, any operating system

You just need a terminal client, and if you are on an external network, you will first need to connect to VPN or use the Artemis Jump.
Accessing the Artemis HPC from your computer

- **Mac/Linux**: use the existing Terminal application

- **Windows**: install a terminal client – many available with various features, eg:

<table>
<thead>
<tr>
<th>Client</th>
<th>License</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xwin</td>
<td>Provided by USyd</td>
<td>Edit text in a graphical window</td>
</tr>
<tr>
<td>Cygwin</td>
<td>Free</td>
<td>Approximates a native Linux environment on Windows PC</td>
</tr>
<tr>
<td>Putty</td>
<td>Free</td>
<td>Easily change size and colour of text</td>
</tr>
</tbody>
</table>
Connecting to Artemis from: Mac

- Optional: Install XQuartz: [https://www.xquartz.org/](https://www.xquartz.org/)
- Go to Finder ➔ Applications ➔ Utilities
- Double-click Terminal
- Type the following into Terminal then hit enter:

  `ssh -Y ict_hpctrain<N>@hpc.sydney.edu.au`

**NOTE:**
If you see text enclosed in arrows, this means replace the text with content relevant to you

**NOTE:**
The `-Y` option only works if you installed XQuartz
Connecting to Artemis from: Mac

- Optional: Install XQuartz: [https://www.xquartz.org/](https://www.xquartz.org/)
- Go to Finder → Applications → Utilities
- Double-click Terminal
- Type the following into Terminal then hit enter:

```
ssh -Y ict_hpctrain<N>@hpc.sydney.edu.au
```

If connecting for the first time you will be asked if you trust the authenticity of host – type ‘yes’ then hit enter key.

NOTE:
If you see text enclosed in arrows, this means replace the text with content relevant to you.

NOTE:
The -Y option only works if you installed XQuartz.

Type the password then press ‘enter’.

Logon success
Where to download software for your own Windows machine?

  - Requires unikey credentials
  - Follow the instructions to install license server details

- **Putty**: [http://www.putty.org/](http://www.putty.org/)
  - Download and run ‘putty.exe’

- **Cygwin**: [https://www.cygwin.com/](https://www.cygwin.com/)
  - Download and run ‘setup-x86_64.exe’ (64 bit OS) or ‘setup-x86.exe’ (32 bit OS)
  - During setup, select additional packages to install. Recommended: openssh, x-org server, xinit
Connecting to Artemis from: **Windows with Xwin**

- Click the windows icon in the lower-right hand corner of your screen
- Type X-Win into the search bar
- Click the X-Win32 or X-Config options
- A configuration window will appear
Connecting to Artemis from: Windows with Xwin

- Along the right hand side under ‘New Connection’ select ‘Manual’
- Select ‘ssh’ for connection method then click ‘Next’
- In the ‘Connection Name’ field, enter any descriptive name eg `<name_HPC>`
- In the ‘Host’ field, enter: `hpc.sydney.edu.au`
- In the ‘Login’ field, enter your unikey
  - Training unikey: ict_hpctrain
- In the ‘Command’ field, enter:
  `/usr/bin/xterm -ls -fa `consolas` -fs 11`
- Click ‘Save’ then ‘Launch’

Letter ‘l’ not numeral 1
Connecting to Artemis from: Windows with Putty

- Launch Putty: A configuration window will appear
- In the ‘Host Name’ field, enter `hpc.sydney.edu.au`
- Leave port as 22 (default)
- Type a name for your connection into ‘Saved Sessions’
- Under ‘Connections’, expand ‘SSH’ and click on ‘X11’
- Select the check box next to ‘Enable X11 forwarding’
- Click ‘Session’ (top left)
- Click ‘Save’ (VIP step!) then ‘Open’
- At ‘login as’ enter your unikey (or ict_hpctrain<N>)
- Enter your unikey (or training) password
Connecting to Artemis from: Windows with Putty

- At ‘login as’ enter your unikey (or ict_hpctrain<N>)
- Enter your unikey (or training) password and hit enter
  - Once you have a saved session, you can launch it by selecting the saved session name, clicking ‘Load’ then ‘Open’
  - The first time you use an X11 command, you will be asked if you allow the connection. Select ‘Always’
1. Open your Putty configuration window again
2. Click on your saved session name
3. Click ‘Load’
4. Under ‘Window’, select ‘Appearance’
5. Under ‘Font Settings’, click ‘Change’
6. Change the font as desired
7. Click ‘OK’
8. Click ‘Session’
9. Click ‘Save’
10. Click ‘Open’ to launch your new-look terminal
Artemis directory structure and ‘projects’

All user accounts on Artemis are linked to at least one ‘project’

Projects are registered using a Research Data Management Plan (rdmp.sydney.edu.au)
Artemis directory tree

Full pathname:
/project/Project1/Experiment1/Scripts/abc

*File suffixes are irrelevant to Linux, but useful to you
Artemis directory tree

- When you first log in, you are situated in your home directory:
  /home/<unikey>
  /home/ict_hpcTrain

  - Home directories have a tiny 10 Gb quota
  - Do not do any project work in here: use instead your **project directory**

**REMEMBER:**
If you see text enclosed in arrows, this means replace the text with content relevant to you
Artemis directory tree

- All project directories are located in:
  /project

  - Each project has its own directory with 1 Tb space
  - This is a **shared directory amongst all members of the project**
  - To ensure quota is not exceeded, **check usage** and **utilise scratch space**
  - Project is not permanent storage and is not backed up, so **utilise Research Data Store** (RDS, later)

- You (or your Lead CI) selected the short project name in the RDMP
  - You can find out which project(s) you’re a member of by typing ‘groups’ on the command line.
  - Projects you are a member of begin with RDS- and end with -RW

```
[ict_hpctrain1@login3 ~]$ groups
linuxusers HPC-ACL RDS-ICT-HPCTRAINING-RW RDN- USERS RDN-CORE-Training
RDS-CORE-Training-RW
[ict_hpctrain1@login3 ~]$ 
```

Remember these names!
Artemis directory tree

- All project directories are located in:
  
```
/project
```

- The `ict_hpctrain<N>` accounts are members of two projects, which means this account has access to two project directories:
  - `/project/HPCTRAINING`
  - `/project/Training`

```
[ict_hpctrain1@login3 ~]$ groups
linuxusers HPC-ACL RDS-ICT-HPCTRAINING-RW RDN-_USERS RDN-CORE-Training
RDS-CORE-Training-RW
[ict_hpctrain1@login3 ~]$ 
```

"short names"

Remember these names!
Artemis directory tree

- All project directories are located in: `/project`

- Check usage with the `pquota` command:

```
[ict_hpctrain1@login3 Training]$ pquota
Disk quotas for user ict_hpctrain1 (uid 508194):
  Filesystem  used  quota  limit  grace  files  quota  limit  grace
   /home     2.191M   10G   11G    -     34     0     0     -
Disk quotas for group RDS-ICT-HPCTRAINING-RW (gid 12524):
  Filesystem  used  quota  limit  grace  files  quota  limit  grace
   /project  125.7G   1T   1.074T    -    5342     0     0     -
Disk quotas for group RDS-CORE-Training-Rw (gid 14206):
  Filesystem  used  quota  limit  grace  files  quota  limit  grace
   /project  50.26G   1T   1.074T    -   1592     0     0     -
[ict_hpctrain1@login3 Training]$ 
```
Artemis directory tree

- Each project has a corresponding 'scratch space' situated in:
  
    /scratch

    - Same project name is used, e.g. your project is called 'Training', you will have access to:
      
      /scratch/Training
      /project/Training

- Using scratch space for input and output files is recommended because:
  
  - Disk quota is vast and not capped to 1 Tb like project directories
  - Ensures there is space to write your output files:
    
    - If your quota is exceeded during an analysis, your job will likely fail and/or the output will be non-existent or truncated
  - Remember project is shared with other group members, so there may be other group members filling up the quota at the same time as you

- **Scratch is not permanent storage**: after your job completes, back up output to RDS (see the data transfer course!) and delete what is no longer needed. Data in /scratch is removed after being inactive for 3 months, data in /project is removed after being inactive for 6 months! Inactive means the project has submitted less than 1 CPU hour.
Artemis directory tree

```
```

Full pathname:

```
/project/Project1/Experiment1/Scripts/abc
```

*File suffixes are irrelevant to Linux, but useful to you*
Artemis directory tree

- All software is installed in:
  `/usr/local`

- To see a list of installed software, run:
  `module avail`

- If the software you want is not installed:
  You can request any open-source (free) software, as long as it has a Linux command line version
  - Go to the ICT Self-Service Portal at https://sydney.service-now.com/selfservice/ict_services.do
  - Select ICT Services → Research → High Performance Computing Request
  - Under ‘Category’, select ‘Application (new software or different version)’
  - Use the text box to name the software and version you want installed, including the download URL

  You will be sent TrackIT notifications regarding the status of your request
Change into your project directory

If using your own HPC account, your project directory is:
/project/<YourProject>

The long way: /project/RDS-<FAC>-<YourProject>-RW

If using a training account, your project directory for today is:
/project/Training

The long way: /project/RDS-CORE-Training-RW

- Short and long project names are both equally valid
- Some programs only work with the long name, so it is worth knowing both versions
Change into your project directory

When you see this bullet point, it means ‘do something’

- Change into your project directory now

If using your own HPC account:
\texttt{cd /project/YYYYYourProject}

If using a training account:
\texttt{cd /project/Training}

![Diagram showing directory structure with user accounts and project folders]
Make a working directory for today

- Make a directory, call it your name
- Change into it
- Work in here for the rest of today

```
mkdir <YourName>
cd <YourName>
```

(remember to replace the content inside the arrows with your details)

- We will use this location to store our raw data and to write output files

Very important:

You are sharing the same project space with other users today – you all have read and write permissions to each other’s directories!!

So please be sure to only work within your own named directory.
Prepare the input data

- Today, the sample data can be found here:
  /project/Training/sample_data.tar.gz
- Everyone: please double check you are in your working directory before running the extract command, because you are all members of the same ‘project’ today
- eg: I would be in /project/Training/stephen
  - Change to your directory
    cd <yourname>
  - Extract the archive:
    tar -zxvf ../sample_data.tar.gz
  - Move all files out of ‘Sample_data’
    mv sample_data/* .
  - Delete the empty ‘Sample_data’ directory
    rmdir sample_data

If your system has decompressed the archive (depending on your settings this may happen automatically), use instead:

```
tar -xvf Sample_data.tar
```

The “..” here means ”the directory above this one. Only run this command from your own directory:

```
/project/Training/<YourName>
```
PBS Pro
Job scripts
Artemis ‘queues’
Text editing on Artemis HPC
### PBS Pro

- We now have our directory set up with input data – we can submit jobs to Artemis!
- Why “submit jobs” and not just “run jobs”?
- Unlike other single-node Linux servers you may have used, Artemis is a *cluster* of computers
- Jobs are allocated to nodes on the cluster by *scheduling software* – in this case, PBS Pro
- PBS Pro manages the workload on Artemis by scheduling jobs to run at different times and on different nodes, to ensure the cluster is being used to optimal efficiency

The practical difference to you:

- Instead of running your job directly on the command line to be immediately executed, you must submit it to PBS Pro
- You can submit your job using a PBS script (covered today), directly on the command line for single-command jobs, or by entering commands in ‘interactive mode’
- The scheduler may run your job right away, or it may be placed in a queue and start later
PBS Pro

- When your job starts is determined by:
  - What resources you have requested (RAM, cores, walltime)
  - What resources are currently available on the HPC
  - What other jobs are in queue
  - ‘Fair share’ user model – factors recent usage to ensure no monopoly on resources

- Resource requests and relevant job information is provided to PBS Pro using ‘directives’
  - Either saved in a PBS script or entered manually on the command line
Now we will create and examine your first PBS script

- Copy the file ‘index.pbs’ and call the new file ‘basic.pbs’
  ```bash
  cp index.pbs basic.pbs
  ```

- Open basic.pbs with your preferred text editor…

Copying scripts is a great way to avoid retyping directives for every new job! Just be sure to modify the directives to suit each analysis.
Text editors on the HPC

<table>
<thead>
<tr>
<th>Editor</th>
<th>Ease of use</th>
<th>Power/flexibility</th>
<th>GUI option*</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEdit</td>
<td>High</td>
<td>Low</td>
<td>Yes</td>
</tr>
<tr>
<td>Nano</td>
<td>Moderate to High</td>
<td>Low</td>
<td>No</td>
</tr>
<tr>
<td>Emacs</td>
<td>High with gui Moderate without GUI</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>Vi/vim</td>
<td>Low</td>
<td>High</td>
<td>No</td>
</tr>
</tbody>
</table>

If you have an X Server installed (e.g., Xwin, Xquartz):
- GEdit and Emacs will run as a GUI (point-and-click)
- Make sure to include –X or –Y flag in ssh connection command
- Run GEdit or Emacs GUI with the ampersand (‘&’) character. This runs the program in the background, i.e., you can use your terminal and edit text at the same time

If not:
- Nano is the easiest to use if you are an inexperienced shell user
Nano in Xwin or Putty/Mac terminal

- Nano opens inside the terminal window
- Commands to save, close etc are typed
- Commands are shown in the editor at all times making it easy to remember them
Emacs (or GEdit) in Xwin

- Xwin can open GEdit in a GUI window – this means you can use your mouse to point-and-click commands from the menu if you prefer.
- Including the ‘&’ (run command in the background) means your terminal remains active while GEdit is open, eg

```bash
$ gedit &
```

** (gedit:15765): WARNING **: Could not load theme icon gtk-home: Icon 'gtk-home' not present in theme
## GEdit and nano: basic commands

<table>
<thead>
<tr>
<th>Operation</th>
<th>Nano command</th>
<th>GEdit command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open new empty file</td>
<td><code>nano</code></td>
<td><code>gedit &amp;</code></td>
</tr>
<tr>
<td>Open new file + name it</td>
<td><code>nano NewFile</code></td>
<td><code>gedit NewFile &amp;</code></td>
</tr>
<tr>
<td>Open existing file</td>
<td><code>nano ExistingFile</code></td>
<td><code>gedit ExistingFile &amp;</code></td>
</tr>
<tr>
<td>Save</td>
<td><code>control + o</code></td>
<td><code>control + s</code></td>
</tr>
<tr>
<td>Close/exit</td>
<td><code>control + x</code></td>
<td><code>control + q</code></td>
</tr>
</tbody>
</table>

**Note:**
- Only use ‘&’ if you have an X Server (eg Xwin, Xquartz) installed, or the editor will not open.
Important note for Windows users

- Windows and Linux use different character encoding
- If you create a file on Windows and copy to Linux, this will very often end in a failure to read the file correctly (it depends on the software you use and the file you create)
- You might see an error like this when trying to run a script on HPC which is affected by this problem:

  ```bash
  -bash: /var/spool/PBS/mom_priv/jobs/1397159.pbsserver.SC: /bin/bash^M: bad interpreter: No such file or directory
  - In this example, with an exit status of 126
  ```
Important note for Windows users

- In some cases (e.g., plain text files), this can be identified and corrected using 2 Linux commands:

  - `file <filename>` - describes file type
  - `dos2unix <filename>` - converts files to Unix/Linux format

```
[cwil2281@login2 ~]$ file from_excel.txt
from_excel.txt: ASCII text, with CRLF line terminators
[cwil2281@login2 ~]$ file from_HPC.txt
from_HPC.txt: ASCII text
[cwil2281@login2 ~]$ dos2unix from_excel.txt
dos2unix: converting file from_excel.txt to UNIX format ...
[cwil2281@login2 ~]$ file from_excel.txt
from_excel.txt: ASCII text
[cwil2281@login2 ~]$  
```
Windows users

- Some text editors can save files with Artemis compatible line endings. eg Notepad++

Select Unix line endings before saving your file
Important note for Windows users

In summary:

- Be aware of Windows vs Linux line endings
- If you encounter strange, seemingly inexplicable errors, investigate this as a source of the problem
- On HPC, use ‘file’ and ‘dos2unix’ to check and convert files – but remember that this does not solve the issue in all cases
- Try to perform creation and editing of scripts intended for HPC on the HPC directly
Open basic.pbs with your preferred text editor…

Nano:

```
nano basic.pbs
```

or

GEdit from Xwin terminal:

```
gedit basic.pbs &
```

Three essential components of a PBS script:

1. **Directives**: instructions to PBS Pro, inc. resource requests and details about your job
2. **Module loads**: if your job needs some software, ‘loading’ the software adds it to your path for the current session
3. **Commands that make up the job**: can be anything you would enter into the terminal – Linux commands, calls to external software or scripts etc
#!/bin/bash

# Create indexes for reference sequence
#PBS -P Project
#PBS -N Index
#PBS -l select=1:ncpus=1:mem=4GB
#PBS -l walltime=00:10:00
#PBS -M your.name@sydney.edu.au
#PBS -m abe
#PBS -q defaultQ

# Load modules
module load bwa
module load samtools

path=/project/<YourProject>/<YourName>

# BWA index:
bwa index -a bwtsw ${path}/canfam3_chr5.fasta

# SAMtools index:
samtools faidx ${path}/canfam3_chr5.fasta
1. Directives

`#!/bin/bash

# Create indexes for reference sequence
#PBS -P Project
#PBS -N Index
#PBS -l select=1:ncpus=1:mem=4GB
#PBS -l walltime=00:10:00
#PBS -M your.name@sydney.edu.au
#PBS -m abe
#PBS -q defaultQ

'#: a comment, ignored by the system. Useful for descriptive notes to yourself

`#` then `PBS` (no space): a directive, important instruction to the PBS scheduler
1. Directives

```bash
#!/bin/bash

# Create indexes for reference sequence
#PBS -P Project
#PBS -N Index
#PBS -l select=1:ncpus=1:mem=4GB
#PBS -l walltime=00:10:00
#PBS -M your.name@sydney.edu.au
#PBS -m abe
#PBS -q defaultQ
```
# Directives

```bash
#!/bin/bash

# Create indexes for reference sequence
#PBS -P Project
#PBS -N Index
#PBS -l select=1:ncpus=1:mem=4GB
#PBS -l walltime=00:10:00
#PBS -M your.name@sydney.edu.au
#PBS -m abe
#PBS -q defaultQ
```
1. Directives

```bash
#!/bin/bash

# Create indexes for reference sequence
#PBS -P Project
#PBS -N Index
#PBS -l select=1:ncpus=1:mem=4GB
#PBS -l walltime=00:10:00
#PBS -M your.name@sydney.edu.au
#PBS -m abe
#PBS -q defaultQ
```
1. Directives

```bash
#!/bin/bash

# Create indexes for reference sequence
#PBS -P Project  # Short project name
#PBS -N Index    # Job name (short)
#PBS -l select=1:ncpus=1:mem=4GB  # Nodes, cores and RAM
#PBS -l walltime=00:10:00  # Max run time hrs:mins:secs
#PBS -M your.name@sydney.edu.au
#PBS -m abe
#PBS -q defaultQ
```

If you're unsure of run time: overestimate, because the system will kill your job if it goes over the requested time!
1. Directives

```
#!/bin/bash
# Create indexes for reference sequence
#PBS -P Project Short project name
#PBS -N Index Job name (short)
#PBS -l select=1:ncpus=1:mem=4GB Nodes, cores and RAM
#PBS -l walltime=00:10:00 Max run time hrs:mins:secs
#PBS -M your.name@sydney.edu.au
#PBS -m abe Send yourself an email when the job aborts (a), begins (b) and ends (e). Optional.
#PBS -q defaultQ
```

If you’re unsure of run time: overestimate, because the system will kill your job if it goes over the requested time!
1. Directives

```bash
#!/bin/bash
# Create indexes for reference sequence
#PBS -P Project  # Short project name
#PBS -N Index    # Job name (short)
#PBS -l select=1:ncpus=1:mem=4GB  # Nodes, cores and RAM
#PBS -l walltime=00:10:00  # Max run time hrs:mins:secs
#PBS -M your.name@sydney.edu.au
#PBS -m abe  # Send yourself an email when the job aborts (a), begins (b) and ends (e).
#PBS -q defaultQ  # Queue to submit job to
```

If you're unsure of run time: overestimate, because the system will kill your job if it goes over the requested time!
select = number of ‘chunks’ of compute resources your job requires
ncpus = number of CPUs/cores your job requires
mem = maximum amount of RAM your job requires

- The above example is a single-core job using a maximum of 4 GB RAM
- RAM is total RAM per chunk, so if you have a 4-core job that requires 4 GB per CPU, request 16 GB RAM
  - Max. RAM is enforced: your job will be killed by PBS Pro if it exceeds the requested amount
  - Min. RAM is 1 GB
- If your job can run on multiple CPUs, you can increase the value at ‘ncpus’
- If your job can run across multiple nodes, you can increase the value at ‘select’
- Only request ‘select’ and ‘ncpus’ >1 if you are certain that your software or code can make use of the additional resources; over-requesting wastes resources and will likely cause your job to wait in queue longer than necessary. Be sure to include the right flags/syntax in your code to utilise extra cores/nodes
- For an MPI job, specifying a greater value at ‘select’ rather than at ‘ncpus’ is likely to result in shorter queue times, as the scheduler can more easily find many smaller chunks of available compute resources rather than fewer larger chunks, eg

  select=1:ncpus=16:mem=64GB  versus  select=16:ncpus=1:mem=4GB
Queues on the Artemis HPC

- The nodes on Artemis are partitioned into different ‘queues’
- Each queue is a collection of nodes dedicated to run different kinds of jobs and each has different maximum resources allowed on that particular queue
- Some queues have a higher or lower ‘fair share’ penalty than others
- If you leave off the ‘-q’ directive, your job will go to the default queue, which is called ‘defaultQ’
- Depending on the resources you have requested, jobs submitted to ‘defaultQ’ might be placed on:
  - Small, Normal or Large
  - GPU
  - High memory

```
#PBS -q defaultQ
```

- High memory jobs are those that request > 20 GB RAM per core or > 123 GB RAM per chunk, up to a maximum of 6 TB RAM
- GPU jobs are requested in a special way:
  ```
  #PBS -l select=1:ncpus=1:ngpus=1
  ```
- There are 4 other queues you can request:
  - small-express, with ```#PBS -q small-express```
  - scavenger, with ```#PBS -q scavenger```
  - dtq, with ```#PBS -q dtq```
  - interactive, this is via the command line for an interactive job (not submitted with a job script):
  ```
  qsub -I -P <project> -l select=1:ncpus=1:mem=4GB,walltime=1:00:00
  ```
# Artemis HPC queue resource limits

<table>
<thead>
<tr>
<th>Queue</th>
<th>Max walltime</th>
<th>Max cores per job</th>
<th>Max cores per user</th>
<th>Memory per core</th>
<th>Memory per node</th>
<th>Fair share weighting</th>
<th>Request by</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>1 day</td>
<td>24</td>
<td>96</td>
<td>&lt; 20 GB</td>
<td>&lt;123 GB</td>
<td>1x</td>
<td>#PBS -q defaultQ</td>
</tr>
<tr>
<td>normal</td>
<td>7 days</td>
<td>96</td>
<td>96</td>
<td>&lt; 20 GB</td>
<td>&lt;123 GB</td>
<td>1x</td>
<td>#PBS -q defaultQ</td>
</tr>
<tr>
<td>large</td>
<td>21 days</td>
<td>288</td>
<td>288</td>
<td>&lt; 20 GB</td>
<td>&lt; 123 GB</td>
<td>1x</td>
<td>#PBS -q defaultQ</td>
</tr>
<tr>
<td>high mem</td>
<td>21 days</td>
<td>192</td>
<td>192</td>
<td>&gt; 20 GB</td>
<td>123 GB – 6 TB</td>
<td>5x</td>
<td>#PBS -q defaultQ</td>
</tr>
</tbody>
</table>
| gpu             | 7 days       | 120               | 120                | N/A             | <123 GB         | 5x                   | #PBS -q defaultQ
|                 |              |                   |                    |                 |                 | #PBS -l select=N:ncpus=N:ngpus=N               |
| small-express   | 1 day        | 24                | 96                 | N/A             | <123 GB         | 5x                   | #PBS -q small-express                           |
| scavenger       | 2 days       | 288               | 288                | N/A             | <123 GB         | 0x                   | #PBS -q scavenger                               |
| dtq             | 10 days      | 2                 | 16                 | N/A             | < 16 GB         | 0x                   | #PBS -q dtq                                     |
| interactive     | 4 hours      | 4                 | 4                  | N/A             | <123 GB         | 10x                  | Command-line interactive session, include ‘-I’ flag |

Artemis HPC queue resource limits
Artemis HPC Overall Resource Limits

- Maximum jobs queued per user: 200
- Maximum cores per user: 600
- If you need to submit more jobs at once, you can use array jobs (won’t be covered today)
- Array jobs can have up to 1,000 sub-jobs.
- Therefore, if you take full advantage of array jobs, you could submit 200 x 1,000 = 200,000 jobs simultaneously!
What is the defaultQ?

- If you don’t specify a queue, your job will be placed in this queue. Once placed in this queue, your job will be routed to one of the following internal queues based on the resources requested in your PBS script:
  - small
  - normal
  - large
  - highmem (High Memory)
  - GPU
What is the scavenger queue?

- Non-urgent jobs that you don’t mind how quickly they complete
- These jobs have ZERO impact on your fair share!
- The down side is they are subject to termination if a non-scavenger job needs the resources your scavenger job is using
  - This means the job may be paused and resume later
  - If it is paused for too long it will be killed by PBS Pro and you will need to restart it
What is the data transfer queue (dtq)?

- A queue dedicated to running data transfer and file input/output jobs
- These jobs have ZERO impact on your fair share!
- The only queue on Artemis with direct access to RCOS
  - RCOS = Research Computing Optimised Storage, the NFS option of Research Data Store (RDS)
  - The workshop ‘Data Transfer and RDS for HPC’ will explain in more detail
- No compute jobs allowed
  - Data transfer or file management tasks only
  - Pure compute jobs will be terminated
  - Jobs can use CPU power to do file compression/encryption/data transfer
Fair share

- Calculated at the level of ‘project’ (Faculty is no longer involved)
  - All users in the same project contribute to the same fair share weighting
- Your fair share weighting is increased as you use more computation time
  - Higher fair share weight means your priority in the Artemis queue decreases
- Half-life of your fair share weight is 2 weeks
Queues summary

- The complex queue structure is there for you to take advantage of IF YOU CHOOSE TO
- For a novice user or those who don’t wish to understand the queues in depth, just submit to defaultQ 😊

- For urgent jobs (that fit within the max parameters) you can submit to small-express and cop the increased fair share penalty
- For non-urgent jobs and users who care about fair share weight, you can submit to scavenger
- High mem jobs are automatically recognised by PBS Pro if they ask for > 123 GB RAM
- GPU jobs are automatically recognised by PBS Pro if they request ‘ngpus’
#!/bin/bash

# Create indexes for reference sequence
#PBS -P Project
#PBS -N Index
#PBS -l select=1:ncpus=1:mem=4GB
#PBS -l walltime=00:10:00
#PBS -M your.name@sydney.edu.au
#PBS -m abe
#PBS -q defaultQ

# Load modules
module load bwa
module load samtools

path=/project/<YourProject>/<YourName>

# BWA index:
bwa index -a bwtsw ${path}/canfam3_chr5.fasta

# SAMtools index:
samtools faidx ${path}/canfam3_chr5.fasta

1. Directives

2. Module loads

3. Commands to run the job
Within each session (connection to Artemis) you must **load** the modules you want to use. In other words – tell the system which programs you need to run your job. They will be added to your path for the current session.

```
# Load modules
module load bwa
module load samtools
```

- To see what modules are available on the HPC, run the command:
  
  `module avail`

- To see what versions of a program are available (for example, Matlab):
  
  `module avail |& grep matlab`
# Create indexes for reference sequence
#PBS -P Project
#PBS -N Index
#PBS -l select=1:ncpus=1:mem=4GB
#PBS -l walltime=00:10:00
#PBS -M your.name@sydney.edu.au
#PBS -m abe
#PBS -q defaultQ

# Load modules
module load bwa
module load samtools

path=/project/<YourProject>/<YourName>

# BWA index:
bwa index -a bwtsw $path/canfam3_chr5.fasta

# SAMtools index:
samtools faidx $path/canfam3_chr5.fasta
Any command can be entered in a PBS script. Which commands will depend on what kind of job you are running.

**How do you know what commands to include?**

1. **Choose** the program/s you want to run
2. **Read** the manual/user guide
3. **Work out** the commands and what parameters/arguments are required
4. **Enter** them into the PBS script
Any command can be entered in a PBS script. Which commands will depend on what kind of job you are running.

**How do you know what commands to include?**

1. **Choose** the program/s you want to run
2. **Read** the manual/user guide
3. **Work out** the commands and what parameters/arguments are required
4. **Enter** them into the PBS script

**General minimum structure of a software call:**

Program name, +/- functions, +/- arguments, input data

An argument (-a) and its value (bwtsw)
Preparing a simple job

- Now that you understand what the aspects of a PBS script are and the Artemis queues that run your jobs, we will modify basic.pbs into a very simple job
  - Copying PBS directives from one job script to another is common practice, and saves you having to re-type everything
  - Always be sure to edit the directives to suit the job at hand

- In today’s course you will submit 3 practice jobs
- For each job, you will need to:
  1. Open the .pbs script
  2. Make some changes to the script in a text editor
  3. Save the changes
  4. Submit the job
  5. Check the status of the job
  6. Check the output of the job
Preparing a simple job

➤ Now to edit your basic.pbs script:

1. Adjust your project name at ‘-P’ to reflect your Project
   - Training login users – remember your project is ‘Training’
2. Change the job name at ‘-N’ to your first name
3. The script currently requests one core – this is fine, our basic job won’t multithread so leave ncpus=1
4. The script currently requests 10 minutes wall time – this is too much, reduce it to 2 minutes
5. What about RAM? We won’t be doing heavy compute so reduce it to the minimum of 1GB
6. Insert your email address at –M
7. Change the queue at ‘-q’ to the defaultQ queue (if it isn’t already defaultQ)
8. Now delete everything below the directives, including module loads, comments and commands. We will replace this content with our own commands.
Preparing a simple job

9. Add this line to your script, replacing the project and <YourName> as appropriate:

```bash
cd /project/Training/<YourName>
```

PBS considers your ‘working directory’ to be your home directory, even if you are in project when you submit the job. Thus it is VIP to specify location, either by using ‘cd’ or full pathnames for input and output files.
Preparing a simple job

9. Add this line to your script, replacing the project and <YourName> as appropriate:
   ```bash
cd /project/Training/<YourName>
   ```

   PBS considers your ‘working directory’ to be your home directory, even if you are in project when you submit the job.
   Thus it is VIP to specify location, either by using ‘cd’ or full pathnames for input and output files.

10. Now add some commands to the script: remember that any command you would normally put on the command line can be embedded within a script:

   ```bash
   mkdir New_job
   cp basic.pbs New_job/copy.pbs
   perl hello.pl <YourName>
   ```
Preparing a simple job

9. Add this line to your script, replacing the project and `<YourName>` as appropriate:
   `cd /project/Training/<YourName>`

PBS considers your ‘working directory’ to be your home directory, even if you are in project
when you submit the job
Thus it is VIP to specify location, either by using ‘cd’ or full pathnames for input and output files

10. Now add some commands to the script: remember that any command you would normally
put on the command line can be embedded within a script

   `mkdir New_job`
   `cp basic.pbs New_job/copy.pbs`
   `perl hello.pl <YourName>`

Regular Linux commands can be included, just as they would on the
command line
Calls to external scripts can be included
Preparing a simple job

11. Save your script, eg

Nano:

$ctrl + o$

Emacs (keystrokes):

$ctrl + x, ctrl + s$

Emacs (GUI menu):

File $\rightarrow$ Save
Submitting & monitoring jobs
Checking job output & resource usage
Submitting a job

- Essential PBS commands:

  qsub job.pbs  
  Submit the job

  qstat -x jobID  
  Check the status of a submitted job (‘x’ flag allows historical query)

  qstat -xf jobID  
  As above, showing detailed job information

  qstat -u <unikey>  
  Query all queued and running jobs for user

  qdel jobID  
  Delete a submitted job

  qstat -T | grep <unikey>  
  Show estimated start time of queued jobs
Submitting a job

- Essential PBS commands:

  qsub job.pbs  Submit the job

  Submit your job with the ‘qsub’ command:
  qsub basic.pbs

  Check it’s status with the ‘qstat’ command:
  qstat -x <jobID>

  Note the ID that was printed to screen: this is your job ID and can be used to query your job’s status

  - You can specify jobID with or without the ‘.pbsserver’ suffix

  Submitting a job
  [cwi12281@login2 Cali]$ qsub basic.pbs
  1107528.pbsserver
  [cwi12281@login2 Cali]$ qstat -x 1107528
  Job id     Name          User          Time Use  S   Queue
  1107528.pbsserver Cali     cwi12281     00:00:00 F small-express
Checking on a job’s output

- Great, it’s finished!

1. Did it do what you expected?
   
   Need to know what you expected

2. Did it successfully complete?
   
   Can check your email report

3. Did it run without errors?
   
   Check the job logs and output files
Checking on a job’s output

1. Did it do what you expected?

- Consider what each command in the job was supposed to do:
  
  ```
  mkdir New_job
  cp basic.pbs New_job/copy.pbs
  ```

  We should see a new directory called ‘New_job’, containing a file called copy.pbs
  
  ➢ Check this with ‘ls’

  ```bash
  [cwil2281@login2 Cali]$ ls  
  align.pbs Cali.e1107528 Cali.o1107528_usage index.pbs  
  basic.pbs Cali.o1107528 hello.pl  
  [cwil2281@login2 Cali]$ ls New_job/  
  copy.pbs  
  [cwil2281@login2 Cali]$
  ```
Checking on a job’s output

1. Did it do what you expected?

   perl hello.pl <YourName>

   This command ran a perl script which prints your name to ‘standard output’:
   - If we ran it on the command line, we would see the message in the terminal window
   - Since it was run from within a PBS script, the message will be in the **standard output log file created by PBS**
   - By default, PBS will create **three log files**:
     - one for standard output, named JobName.oJobID
     - one resource usage summary, named JobName.oJobID_usage
     - one for standard error, named JobName.eJobID

   ➢ Check if the perl script ran correctly with the ‘**cat**’ command

```
-rw-r--r-- 1 cwi12281 RDS-FVS-DingDom-RW  478 Nov 30 15:04 Cali.o1107528_usage
-rw------- 1 cwi12281 RDS-FVS-DingDom-RW  31 Nov 30 15:04 Cali.o1107528
-rw------- 1 cwi12281 RDS-FVS-DingDom-RW  54 Nov 30 15:04 Cali.e1107528
[cwi12281@login2 Cali]$ cat Cali.o1107528
Hello, world! My name is Cali.
```

Checking on a job’s output

2. Did it successfully complete?

- Since we included the –M and –m directives, we can check our email to view a report on the completed job
- Exit status of 0 is good
  - Anything else is bad
- You can also see exit status in the usage log or by running `qstat -xf JobID`
Checking on a job’s output

3. Did it run without errors?

- Check your ‘.e’ log file. For this job, it should be empty

- For some software, an error-free job will yield an empty error log
- For others, there may be content in the error log that is not necessarily indicative of an error
- So if you see an empty error log: great! If not, check what it contains

Tips:
- Run `wc -l` first to get a line count of the error log
- If it’s short, ‘cat’ is an easy way to view it’s contents
- If it’s long, ‘more’ is preferable
- Running ‘tail’ on the log to view the last 10 lines can often quickly reveal a fatal error
- Use ‘grep –i error’, or ‘grep –i warning’ etc to search the log for potential error messages
Checking on a job’s output

3. Did it run without errors?

- **Example:** A job is killed by the system because it ran out of walltime

```
root <adm@mgmt1.hpc.syi
PBS JOB 425901.mgmt1

To: Coll Willet

PBS Job ID: 425901.mgmt1
Job Name: Index_Cali
Execution terminated
Exit_status=271
resources_used.cput=00:00:10
resources_used.mem=77000k
resources_used.ncpus=1
resources_used.rmem=311764k
resources_used.walltime=50:00:42
```

```
[cwil2281@login1 Cali]$ tail Index_Cali.o425901
[bwa_index] Pack FASTA... 0.72 sec
[bwa_index] Construct BWT for the packed sequence...
[BWTIncCreate] textLength=80000000, availableWord=17629100
[BWTIncConstructFromPacked] 10 iterations done, 29079360 characters processed,
=>> PBS: job killed: walltime 10 exceeded limit 2
```

Submitting jobs without a PBS script

- Very simple jobs can be run without writing a PBS script
- For example, `hello.pl` could be submitted from the command line:

```bash
qsub -P Training -- /usr/local/perl/5.24.0/bin/perl \
/project/Training/<Your Name>/hello.pl <Your Name>
```

1. Did it do what you expected?
   The output will be written to `STDIN.o<jobid>`

2. Did it successfully complete?
   `qstat -xf <jobid> | grep 'Exit_status'`
   0 if successful, > 0 if there was an error

3. Did it run without errors?
   `cat STDIN.e<jobid>`

Note: Only full path names work when using this method
Preparing a job using modules

- We will now look at a slightly more complex job that calls on other software to process input data and write output data.

  ➢ Open index.pbs
    - This creates indexes of the canfam3_chr5.fasta DNA reference sequence.
    - The indexes are required to perform the next job, which maps DNA sequence fragments back to the reference.
Creating an analysis pipeline

- When building pipelines (series of tasks required for a complete analysis) for your own data analysis, you can choose to split out key stages into separate jobs, or include them all within the one job.

- You should consider:
  - Which parts of a workflow need to be run once only, and which must be re-run for every new set of inputs?
  - Where would you like to perform ‘checkpointing’ before continuing the analysis?
    - This could mean checking results to decide if further steps are needed, deciding on parameters for the next stage of the pipeline, splitting your job into smaller stages to ensure they can run within the maximum walltime for the queue you want to run on, etc etc
    - Long running jobs that hold information in RAM for extended periods of time are recommended to periodically write the data to disk. This ensures that if the job fails (due to error, hardware malfunction etc) that all of the compute is not lost, and the job can be resumed by reading back in from disk instead of starting all over again.
  - Are there stages of a workflow that use a lot more or less resources than other stages?
    - It is best to group portions of a pipeline into sections based on resource use. Eg if you had a 5-step analysis where only one step multi-threaded on 24 cores and the rest was single-core, for the duration of the 4 single-core steps you have 23 cores sitting idle on the machine (23 cores that you have queued longer for…)

Preparing a job using modules

Now to edit your script:

1. Adjust your project name at ‘-P’ to reflect your Project
   - Training login users – remember your project is ‘Training’
2. Change the job name at ‘-N’ to ‘Index_<yourName>’
3. The script currently requests one core – this is fine, our basic job won’t multithread so leave ncpus=1
4. The script currently requests 10 minutes wall time – this is too much, reduce it to 5 minutes
5. The file to be indexed is small so reduce RAM to the minimum (1GB)
6. Insert your email address at –M
7. Request your job be placed in the small-express queue (we want it to run right now)
Preparing a job using modules

7. Edit the value of the variable ‘io’ – you need to update ‘Project’ and ‘YourName’

- A variable is a key concept in bash (Linux) and other programming languages. A variable simply stores some value
- We won’t explore programming today, but variables are very handy for making your PBS scripts ‘portable’ (easy to reuse) from one experiment/run to the next

\[ \text{io} = /\text{project/}<\text{Project}>/<\text{YourName}> \] → a variable containing the directory pathname for your input files

- By using \${\text{io}} \ for input and output path names, we only have to type our directory path once!

- Edit the value of ‘io’ here

  \[ \text{io} = /\text{scratch/DingDom/Cali} \]

It’s value will be read by the script here and here

- # BWA index:
bwa index -a bwtsw \${\text{io}}/canfam3_chr5.fasta

- # SAMtools index:
samtools faidx \${\text{io}}/canfam3_chr5.fasta
Submitting the job and checking status

- Save the script
- Submit the job: `qsub index.pbs`
- Periodically check it’s status using the ‘qstat’ command: `qstat -x jobID`

```
[cwi12281@login1 Cali]$ qstat -x 1107583
Job id   Name          User   Time Use S Queue
-----------------        ------- ------ -----
1107583.pbsserver Cali_Index  cwi12281 00:00:00 R small-express
[cwi12281@login1 Cali]$ qstat -x 1107583
Job id   Name          User   Time Use S Queue
-----------------        ------- ------ -----
1107583.pbsserver Cali_Index  cwi12281 00:00:00 R small-express
[cwi12281@login1 Cali]$ qstat -x 1107583
Job id   Name          User   Time Use S Queue
-----------------        ------- ------ -----
1107583.pbsserver Cali_Index  cwi12281 00:00:22 F small-express
```

Remember: `jobID` is a unique number assigned to your job by the system.

If you forget the `jobID`? Run this command:

`qstat -u <unikey>`
Check the completed job

- Before continuing with the next stage of the data analysis, you should always check the job ran successfully

1. Did it do what you expected?
   Need to know what you expected → we should see some new output files

2. Did it successfully complete?
   Check your email report or usage log → make sure your exit status is zero

3. Did it run without errors?
   Check the job log files → check the error log for errors and that the output is correct
Check the completed job

1. Did it do what you expected?

- Run the `ls -lh` command on your scratch directory

  - Output should be 6 index files – these have the same prefix as the input sequence file, but with an additional suffix
  - Don’t view these, most are not human readable. We know they are OK if they exist, are not empty and the exit status and error logs reveal no errors

```
-rw-r--r-- 1 cwi12281 RDS-FVS-BioServ-RW 5.9K Sep 25 10:43 canfam3_chr5.fasta.amb
-rw-r--r-- 1 cwi12281 RDS-FVS-BioServ-RW 40 Sep 25 10:43 canfam3_chr5.fasta.ann
-rw-r--r-- 1 cwi12281 RDS-FVS-BioServ-RW 39M Sep 25 10:43 canfam3_chr5.fasta.bwt
-rw-r--r-- 1 cwi12281 RDS-FVS-BioServ-RW 19 Sep 25 10:43 canfam3_chr5.fasta.fai
-rw-r--r-- 1 cwi12281 RDS-FVS-BioServ-RW 9.6M Sep 25 10:43 canfam3_chr5.fasta.pac
-rw-r--r-- 1 cwi12281 RDS-FVS-BioServ-RW 20M Sep 25 10:43 canfam3_chr5.fasta.sa
```

6 index files
2. Did it successfully complete?

- Check your email summary or usage log for an exit status of zero

The information in the job summary about what ‘mem’ and ‘walltime’ resources were used can help you decide on appropriate resource requests for future similar jobs.

You can also get this information from the usage log.

The completion email is handy to give you a heads up when your job has completed, especially for queued or long running jobs.
Check the completed job

3. Did it run without errors?

➢ Check your standard error log file

- Is it empty? If not, investigate… (it shouldn’t be empty for this job as BWA program writes things to standard error)
- If it’s not too long, view contents with ‘cat’
- You could search for error messages using ‘grep’, a pattern matching search command

➢ Enter this command (replace your name and job ID)

```bash
grep -i error Index_<YourName>.eJobID
```

This performs a case-insensitive (the `-i` flag) for the word ‘error’ in the log file

```
[cwil2281@login1 Cali]$ grep -i error Index_Cali.e425902
```

No errors found 😊
Check the completed job: usage logs

4. Check actual usage versus requested resources
   - Artemis produces a usage log file for all completed jobs
   - Similar to the email report, but with additional information making it easier to determine whether your resource requests were appropriate

   Why is this important?
   - For you: over-requesting can result in longer queue time – use information in your usage log to adjust directives for your next similar job
   - For others: over-requesting leads to inaccurate estimations of queued job start times

   Usage log naming convention:
   
   JobName.oJobID_usage
Check the completed job: usage logs

4. Check actual usage versus requested resources

```
[cw12281@login1 Cali]# cat Cali_Index.o1107583_usage
Job Id: 1107583, pbsserver for user cw12281 in queue small-express
Job Name: Cali_Index
Project: RDS-FVS-DingDom-RW
Exit Status: 0

Walltime requested: 00:05:00 ; Walltime used: 00:00:26
Lpus requested: 1 :
   Cpu Time: 00:00:16 ; Cpu percent: 101
Mem requested: 1gb :
   Mem used: 156696kb
VMem requested: None :
   VMem used: 156696kb
PMem requested: None :
   PMem used: None
```

Way too much wall time! If you were to rerun this job, you would use this information to make more appropriate requests next time.
Check the completed job: usage logs

4. Check actual usage versus requested resources

```
[cwil2281@login1 Cali]$
$ cat Cali_Index.o1107583_usage
Job Id: 1107583, pbsserver for user cwil2281 in queue small-express
Job Name: Cali_Index
Project: RDS-FVS-DingDom-RW
Exit Status: 0
Walltime requested: 00:05:00 ; Walltime used: 00:00:26
Cpus requested: 1 ;
Cpu Time: 00:00:16 ; Cpu percent: 101
Mem requested: 1gb ; Mem used: 156696kb
VMem requested: None ; VMem used: 156696kb
PMem requested: None ; PMem used: None
```

153 MB used – we can’t reduce this job to less than 1 GB, but for real jobs you should check ‘mem used’ to make sure your request is appropriate.
Queued jobs

- Hopefully our jobs won’t queue today since they are tiny and running on small-express
- Why do jobs with fewer cores/RAM generally leave the queue sooner than larger jobs?
  - This is not a sign of the system favouring small jobs over large – we have dedicated nodes for running large jobs and there is no difference in priority of a large versus small jobs (except for small-express which is weighted high)
  - It is a product of the fact that on a large cluster running many different types of jobs, it is easier for the scheduler to find small free spaces than it is to find large free spaces
  - An over-simplified example: a 24-core node is running 1 x 12-core job and 1 x 8-core job. This leaves 4 free cores. You have submitted a 6-core job and are waiting in queue. Later, Jane submits a 2-core job which runs before yours, because it can fit on the 4 free cores while yours cannot
  - This does not mean it is always better to submit single-core jobs compared to multi-core jobs

<table>
<thead>
<tr>
<th>Busy cores</th>
<th>Job 1, ncpus=12</th>
<th>Busy cores</th>
<th>Job 2, ncpus=8</th>
<th>Free cores</th>
</tr>
</thead>
</table>

Queue:
1. Your job, ncpus=6
2. Jane’s job, ncpus=2
3. …
Queued jobs

Why are jobs queued?
- The system does not have the amount of resources you requested spare right now
- Order in the queue is affected by your fair share weighting

When will your job come off the queue and start running?
- As soon as the resources you need are available, considering your place in the queue

How to avoid/minimise queue time?
- Don’t over-request resources
  - Processors: only request what you need
  - For multi-threading or MPI jobs: use trial and error to find a good balance between speed (number of CPUs requested) and queue time
  - Yes, this is indeed complicated by cluster load at the time you submit
  - Walltime and RAM: try and make reasonably educated requests (plus a little bit spare just in case, esp. time) – your usage logs are useful here
How to tell if a job has been queued?

`qstat -x JobID`

```
[cwil2281@login2  Training_content]$ qstat -x 240757_mgmt1

<table>
<thead>
<tr>
<th>Job id</th>
<th>Name</th>
<th>User</th>
<th>Time</th>
<th>Use</th>
<th>S</th>
<th>Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>240757_mgmt1</td>
<td>Index</td>
<td>cwil2281</td>
<td></td>
<td></td>
<td>Q</td>
<td>compute</td>
</tr>
</tbody>
</table>
```

Q = queued
Can I get an estimate of start time?

- The system can **estimate** a start time **assuming** all currently running and queued jobs run to completion, using **all** the resources that they have requested.
- This estimate is a **worst-case scenario**: your job will usually always start well ahead of that time because:
  - Some jobs will be terminated or removed from the queue by the user.
  - Some jobs will fail early due to an error.
  - Most jobs will run for less walltime than requested.

```
qstat -T | grep yourUnikey
```

```
[cwil2281@login2 Training_content]$ qstat -T | grep cwil2281
240776.mgmt1 cwil2281 compute Index -- 1 12 -- 02:00 Q Sa 18
[cwil2281@login2 Training_content]$`

Estimated start time
Sat at 6pm
Delete a job

- You may want to delete a job if:
  - It has been queued and you would like to reconfigure your requests to try and leave the queue earlier
  - You realise you have made a mistake in the script
  - You simply no longer want to run it

- You can stop a job that is running or in the queue with this command:

  qdel jobID
Preparing another job using modules

- We will now edit and submit align.pbs
  - This job uses the index files created by index.pbs to align the ‘fastq’ DNA sequence files against the ‘fasta’ reference sequence file
  - It is essential that the index job ran correctly for this job to run (so check now if you haven’t already)

- Open align.pbs
Preparing another job using modules

- Edit the script:
  1. Adjust your project name at ‘-P’ to reflect your Project
     - Training login users – remember your project is ‘Training’
  2. Change the job name at ‘-N’ to ‘Align_yourName’
  3. The script currently requests 2 processors (ncpus=2) – the software ‘BWA’ can multithread, so we need to **make sure that we tell BWA to use the 2 processors we’ve requested**
  4. The script currently requests 20 minutes wall time – probably too much, reduce to 10 mins
  5. For a ~3 GB mammalian genome, BWA uses just over 3 GB RAM per core, but this is only 1 chromosome so reduce it to 1 GB per core
  6. Insert your email address at -M
  7. Edit the value of the variable ‘io’ – you need to update Project and YourName
  8. Note that the final output files are copied to your project directory into a new directory called ‘Output’ inside the directory you’re working in
  9. Save the script
Handy use of a variable

io=/project/Training/test
mkdir ${io}/Output

#Load modules:
module load bwa
module load samtools

#Align reads to reference with BWA:
bwa mem -M -t 2 -R '\@RG\tID:134\tPL:illumina\tPU:CONNEACXX\tSM:MS_134' \\
   ${io}/canfam3_chr5.fasta ${io}/134_R1.fastq.gz \\
   ${io}/134_R2.fastq.gz | samtools view -bSho ${io}/Output/134.bam -

#Sort by genomic coordinate with SAMtools:
samtools sort ${io}/Output/134.bam ${io}/Output/134_sorted

#Index alignment with SAMtools:
samtools index ${io}/Output/134_sorted.bam
io=/project/Training/test
mkdir ${io}/Output

#Load modules:
module load bwa
module load samtools

#Align reads to reference with BWA:
bwa mem -M -t 2 -R '@RG\tID:134\tPL:illumina\tPU:CONNEACXX\tSM:MS_134' \n ${io}/canfam3_chr5.fasta ${io}/134_R1.fastq.gz \n ${io}/134_R2.fastq.gz | samtools view -bShO ${io}/Output/134.bam -

#Sort by genomic coordinate with SAMtools:
samtools sort ${io}/Output/134.bam ${io}/Output/134_sorted

#Index alignment with SAMtools:
samtools index ${io}/Output/134_sorted.bam

Handy use of a variable

The backslash means that the command is continued on a new line - tells the shell to ignore the newline character
io=/project/Training/test
mkdir ${io}/Output

#Load modules:
module load bwa
module load samtools

#Align reads to reference with BWA:
bwa mem -M -t 2 -R 'KG\tID:134\tPL:illumina\tPU:CONNEACXX\tSM:MS_134'
  ${io}/canfam3_chr5.fasta ${io}/134_R1.fastq.gz \n  ${io}/134_R2.fastq.gz | samtools view -b $io/Output/134.bam -

#Sort by genomic coordinate with SAMtools:
samtools sort $io/Output/134.bam $io/Output/134_sorted

#Index alignment with SAMtools:
samtools index $io/Output/134_sorted.bam

Handy use of a variable

The backslash means that the command is continued on a new line - tells the shell to ignore the newline character.

Be aware of how much disk space you’re using. /project/Training has a 1 TB limit. Check how much disk space your files are using with the `ls -lh` command.
Preparing another job using modules

- Have a look at the three commands that comprise this job
  - Looking at ‘Align reads to reference’ command, can you see where we have directed BWA to use 2 processors?
    - The ‘t’ flag specifies ‘threads’ (processors/cores) to BWA. How multithreading is specified is specific to each software, so make sure you check the software documentation
    - **Always make sure the number of cores your directives request are utilised by the software** – BWA does not know to run on 2 cores just because the PBS directive has requested them

```bash
# Align reads to reference with BWA:
bwa mem -M -t 2 -R '@RG\tID:134\tPL:illumina\tPU:CONNEACXX\tSM:MS_134' \
${io}/canfam3_chr5.fasta ${io}/134_R1.fastq.gz \
${io}/134_R2.fastq.gz | samtools view -bSho ${io}/134.bam -
```
Preparing another job using modules

- Have a look at the three commands that comprise this job
  - Also note the directory paths pointing to the file inputs and outputs
    - By using full pathnames (instead of using `cd` as in the basic.pbs script) we can explicitly use different locations on the system
    - **Triple check that you have no typos in your directory paths** – if you try and write to a location that doesn’t exist, your job will terminate!

- Example error when file/pathnames are incorrect:

```
[wcll2281@login1 Cali]$ tail Align.e435y40
[M::mem_pestat] low and high boundaries for proc [M::mem_pestat] skip orientation FF
[M::mem_pestat] skip orientation RR
[M::mem_process_seqs] Processed 198020 reads in 19,303 CPU sec, 9,231 real sec
[samopen] SAM header is present, 1 sequences. [main_samview] fail to open "}/Output/134.bam" for writing. open: No such file or directory
[bam_sort_core] fail to open file /project/KDS-PVS-bioServ-RW/Cali/Output/134.bam
open: No such file or directory
[bam_index_build2] fail to open the BAM file.
[wcll2281@login1 Cali]$ 
```
Some quick revision!

- Take 15 minutes to work through the revision activity
  - Write down your answers as you work through the tasks
  - We will then go through the answers together
Revision activity

1. Submit align.pbs (also, submit the job now)

2. Check the status of the job

3. How might you know when the job is finished?

4. How much space are you allocated in /project, and how do you check how much you have used?

5. What log files are produced after a job completes, and what do they contain?

6. What do the following directives indicate to the scheduler:
   a) –P ______________________
   b) –N ______________________
   c) –q ______________________

7. Write the directive requesting 10.5 hours wall time

8. Write the directive to specify a 6-core job using 2 GB RAM per core

9. Your job uses the software ‘beast’. What needs to come before your command to run beast?

10. Where should all important input and output data be stored long-term?
Revision activity

1. Submit align.pbs

2. Check the status of the job

3. How might you know when the job is finished?

4. How much space are you allocated in project, and how do you check how much you have used?

5. What log files are produced after a job completes, and what do they contain?
Revision activity

1. Submit align.pbs
   qsub align.pbs
2. Check the status of the job

3. How might you know when the job is finished?

4. How much space are you allocated in project, and how do you check how much you have used?

5. What log files are produced after a job completes, and what do they contain?
Revision activity

1. Submit align.pbs
   `qsub align.pbs`
2. Check the status of the job
   `qstat -x JobID`
3. How might you know when the job is finished?
   ____________________________________________________________
   ____________________________________________________________
   ____________________________________________________________
4. How much space are you allocated in project, and how do you check how much you have used?
   ____________________________________________________________
5. What log files are produced after a job completes, and what do they contain?
   ____________________________________________________________
Revision activity

1. Submit align.pbs
   ```bash
   qsub align.pbs
   ```

2. Check the status of the job
   ```bash
   qstat -x JobID
   ```

3. How might you know when the job is finished?
   - The output of `qstat` shows an ‘F’ in the ‘S’ (status) column
   - We receive an email from the system indicating termination
   - We can see the log files for this job (they are only created at completion)

4. How much space are you allocated in project, and how do you check how much you have used?
   __________________________

5. What log files are produced after a job completes, and what do they contain?
   __________________________
Revision activity

1. Submit align.pbs
   `qsub align.pbs`
2. Check the status of the job
   `qstat -x JobID`
3. How might you know when the job is finished?
   - The output of `qstat` shows an ‘F’ in the ‘S’ (status) column
   - We receive an email from the system indicating termination
   - We can see the log files for this job (they are only created at completion)
4. How much space are you allocated in project, and how do you check how much you have used?
   - Space per project: 1 TB, command ‘pquota’
5. What log files are produced after a job completes, and what do they contain?
Revision activity

1. Submit align.pbs
   `qsub align.pbs`
2. Check the status of the job
   `qstat -x JobID`
3. How might you know when the job is finished?
   - The output of ‘qstat’ shows an ‘F’ in the ‘S’ (status) column
   - We receive an email from the system indicating termination
   - We can see the log files for this job (they are only created at completion)
4. How much space are you allocated in project, and how do you check how much you have used?
   - Space per project: 1 TB, command ‘pquota’
5. What log files are produced after a job completes, and what do they contain?
   - `JobName.oJobID` – standard output (things that are usually printed to terminal)
   - `JobName.eJobID` – standard error (OS or application error messages)
   - `JobName.oJobID_usage` – job resource usage
Revision activity

6. What do the following directives indicate to the scheduler:
   a) \(-P\) ______________________
   b) \(-N\) ______________________
   c) \(-q\) ______________________

7. Write the directive requesting 10.5 hours wall time ______________________

8. Write the directive to specify a 6-core job using 2 GB RAM per core ______________________

9. Your job uses the software ‘beast’. What needs to come before your command to run beast?
   ______________________

10. Where should all important input and output data be stored long-term?
    ______________________
Revision activity

6. What do the following directives indicate to the scheduler:
   a) \(-P\) project (abbreviated project name)
   b) \(-N\) job name (short descriptor)
   c) \(-q\) queue (defaultQ, scavenger, small-express, dtq)

7. Write the directive requesting 10.5 hours wall time

8. Write the directive to specify a 6-core job using 2 GB RAM per core

9. Your job uses the software ‘beast’. What needs to come before your command to run beast?

10. Where should all important input and output data be stored long-term?
6. What do the following directives indicate to the scheduler:
   a) \( -P \) project (abbreviated project name)
   b) \( -N \) job name (short descriptor)
   c) \( -q \) queue (defaultQ, scavenger, small-express, dtq)

7. Write the directive requesting 10.5 hours wall time
   \#PBS -l walltime=10:30:00

8. Write the directive to specify a 6-core job using 2 GB RAM per core

9. Your job uses the software ‘beast’. What needs to come before your command to run beast?

10. Where should all important input and output data be stored long-term?
6. What do the following directives indicate to the scheduler:
   a) \textit{\textbf{-P}} \textbf{project (abbreviated project name)}
   b) \textit{\textbf{-N}} \textbf{job name (short descriptor)}
   c) \textit{\textbf{-q}} \textbf{queue (defaultQ, scavenger, small-express, dtq)}

7. Write the directive requesting 10.5 hours wall time
   \#PBS \texttt{-l walltime=10:30:00}

8. Write the directive to specify a 6-core job using 2 GB RAM per core
   \#PBS \texttt{-l select=1:ncpus=6:mem=12GB} \textbf{or}
   \#PBS \texttt{-l select=6:ncpus=1:mem=2GB}

9. Your job uses the software ‘beast’. What needs to come before your command to run beast?
   
10. Where should all important input and output data be stored long-term?
Revision activity

6. What do the following directives indicate to the scheduler:
   a) \(-P\)  project (abbreviated project name)
   b) \(-N\)  job name (short descriptor)
   c) \(-q\)  queue (defaultQ, scavenger, small-express, dtq)

7. Write the directive requesting 10.5 hours wall time
   #PBS \(-l\) walltime=10:30:00

8. Write the directive to specify a 6-core job using 2 GB RAM per core
   #PBS \(-l\) select=1:ncpus=6:mem=12GB  \textit{or}
   #PBS \(-l\) select=6:ncpus=1:mem=2GB

9. Your job uses the software ‘beast’. What needs to come before your command to run beast?
   \texttt{module load beast}

10. Where should all important input and output data be stored long-term?
6. What do the following directives indicate to the scheduler:
   a) –P  project (abbreviated project name)
   b) –N  job name (short descriptor)
   c) –q  queue (defaultQ, scavenger, small-express, dtq)

7. Write the directive requesting 10.5 hours wall time
   #PBS -l walltime=10:30:00

8. Write the directive to specify a 6-core job using 2 GB RAM per core
   #PBS -l select=1:ncpus=6:mem=12GB  or
   #PBS -l select=6:ncpus=1:mem=2GB

9. Your job uses the software ‘beast’. What needs to come before your command to run beast?
   module load beast

10. Where should all important input and output data be stored long-term and why?
    The Research Data Store (RDS). RDS is backed up and Artemis is not!
Align.pbs - check for errors

- First, use `qstat` to check that your job has finished
  - `qstat -x <jobID>` (the “F” in the “S” column means it finished)

1. Did it do what you expected?
   Within your Output directory, you should see an alignment file (bam), a sorted bam and it’s index

2. Did it successfully complete?
   Check for exit status of zero

3. Did it run without errors?
   Check the error log

   Note that the error log is not empty – BWA sends it’s output to the error log
Align.pbs - check usage

➢ Check your usage log

Were your resource requests appropriate?

```
[wcil2281@login1 Cali]$ cat Cali_Align_o1107633_usage
Job Id: 1107633, psserver for user cwil2281 in queue small-express
Job Name: Cali_Align
Project: RDS-FVS-DingDom-RW
Exit Status: 0
Walltime requested: 00:05:00 : Walltime used: 00:03:16
Cpus requested: 2 : Cpu percent: 352
Mem requested: 2gb : Mem used: 474364kb
VMem requested: None : VMem used: 474364kb
PMem requested: None : PMem used: None
```

Much better than last time! Walltime can fluctuate depending on system usage – we have over-requested but only by 2 minutes

Willet et al 2015, PLoS ONE 10(2): e0117055a

The sequence alignment you just created reveals the causal mutation for canine spondylocostal dysostosis (genomic position cfa5:32,945,846)
Some final tips about log files

- You can have both standard output and standard error send to the same log file (.o), instead of 2 separate logs, by adding this directive:

  ```
  #PBS -j oe
  ```

- You can change the default name of your log files like this:

  ```
  #PBS -o output_MyJob.log
  #PBS -e error_MyJob.log
  ```

- Error and output logs are only readable by you (see permissions). To enable others in your project to read your logs (handy) add this directive:

  ```
  #PBS -W umask=022
  ```
Some final tips about variables

- PBS has a range of really useful ‘environment variables’
  - These are variables that are filled automatically when you submit a job
  - They are not covered today in the interest of time, however I encourage you to check them out to see how they could help simplify your scripting
- For example:

  **PBS_O_WORKDIR** - filled with the full pathname of the directory you are in when you submit your job

  **NCPUS** - the number of cpus requested
Backing up research data

- **Artemis is not backed up**
  - Store raw data, final output, job scripts etc on the research data store!

To learn how to back up your data and move data to and from Artemis HPC, come to the Data Transfer session this afternoon!
Thank you, please complete the survey

https://goo.gl/62oXpz