Command Line (CLI) BioSciences

Make your science easier with high performance computing (HPC)

Dr Keiran Rowell & Nathan Glades - *Digital Health Hub,* UNSW - Run 2 - 14th January 2026

HPC

Why leave your laptop? When you have to: Data, Silicon, Time, $$

UNSW SYDNEY

You are escaping:

- Wait, which desktop hard drive did I leave my data on?
- My calculations don't finish before the power saver kicks in.
- It takes me days to install all the programs I need.
- My computer just isn't powerful enough for all the data I've gathered.
- I can't access my work off-campus.

So you can do your science:

- I can get help for difficult to install programs!
- I can share my work and scripts with scientists doing the same thing.
- I can save work hours by not manually doing the same thing.
- I can use the power of industrial-grade computers linked together.
- I can let the scheduler handle running my jobs and go do something else.

ssh

Text interfaces: efficiency, automation, adaptability

UNSW SYDNEY

The command line is a starting barrier because there's *no visual feedback*

```
keiran@discworld:~
```

It will be silent if things 'just work'

Otherwise, it will throw back error messages to the terminal

```
keiran@discworld:~ ls non-existent_data
"non-existent_data": No such file or directory (os error 2)
```

Error messages are highly specific, which means we can solve our problems

Can write down exactly what to do into a shell script (. sh)

```
keiran@discworld:~ bash hello_discworld.sh
Hello Discworld!
```

ssh

Text interfaces: a blank page means things are simple

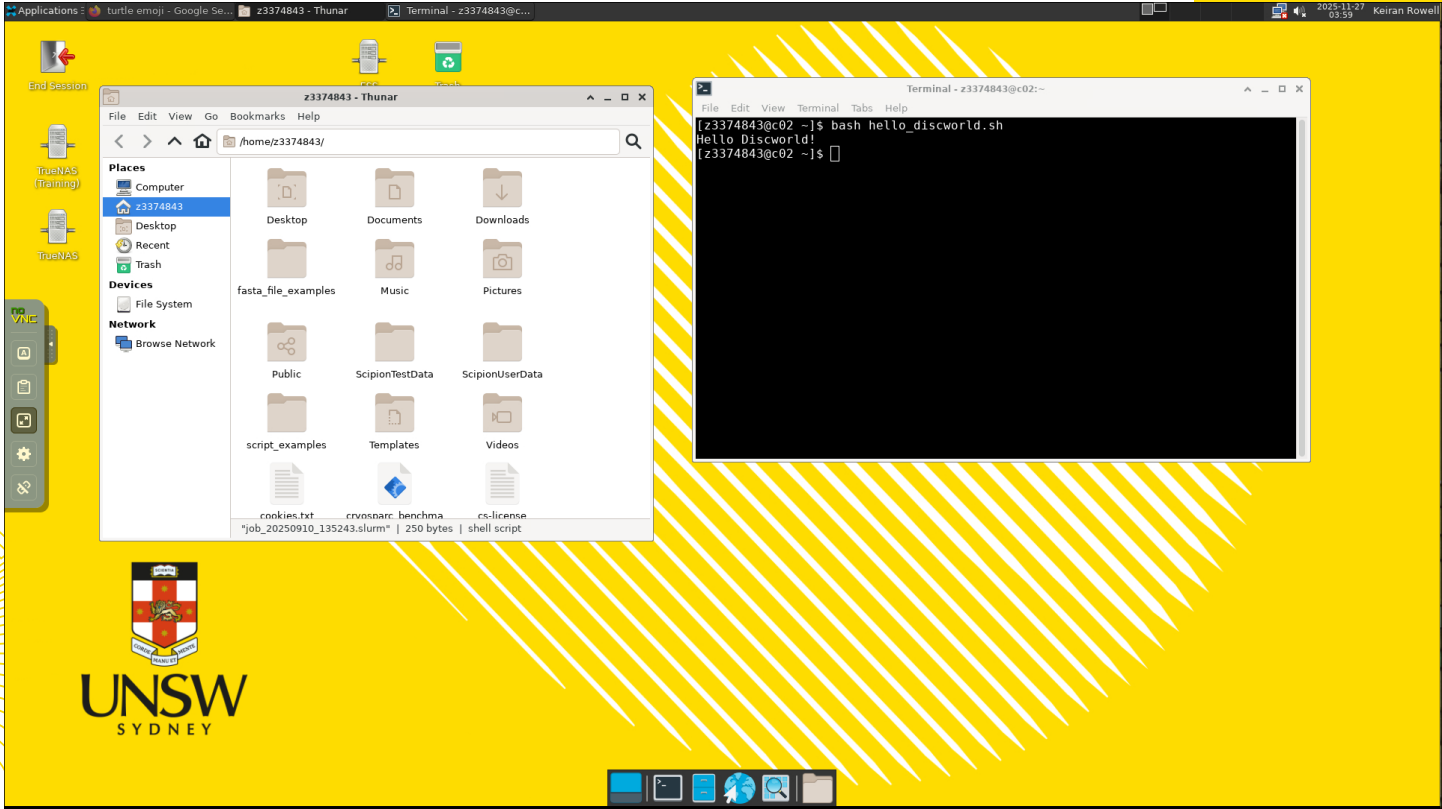
UNSW SYDNEY

- All text can be edited (we've made the `edit` command)
- Everything is a file
- The permissions of each file can be controlled
- Managing data is faster
- You can automate recipes (shell scripts `.sh`) of other scientists
- `tldr program` *or* `program --help` tells you usage info

sdesktop

Graphical interfaces: interaction, structures, inspection

- Science is *visual*
- 'desktop': successful computing metaphor
- 3D molecules
- Image analysis requires **user choice**



sbatch

Pooled resources: 'set-and-forget', 24x7, scalable, **powerful**

- Hardware pooled — request your bucket
- Dip your toe in before the highboard

- Testing (start small):
  - Memory
  - Time per data size
  - CPU cores
  - GPU processing

Trial in an interactive job

UNSW SYDNEY

`sinteractive`

Interactive requests: testing, debugging, trials, flexible

UNSW SYDNEY

Interactive jobs give you a 'live' terminal while still contained in its lane.

Great for anything you haven't figured out yet.

```
keiran@discworld:~ sinteractive --time 2:00:00
keiran@discworld:~ echo "I do what I want"
I do what I want
```

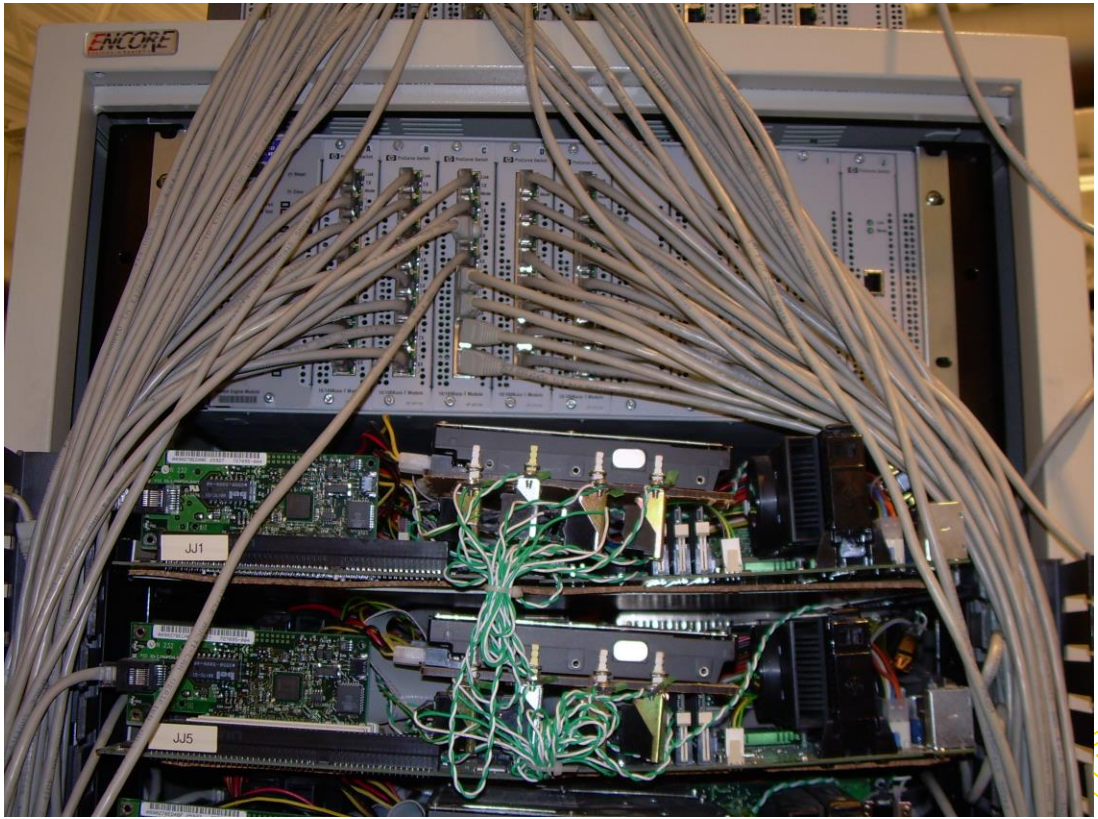
The graphical sdesktop is still contained inside an interactive job.

Once constructed, hand your job off to the scheduler.

A watched pot never boils

sdraft

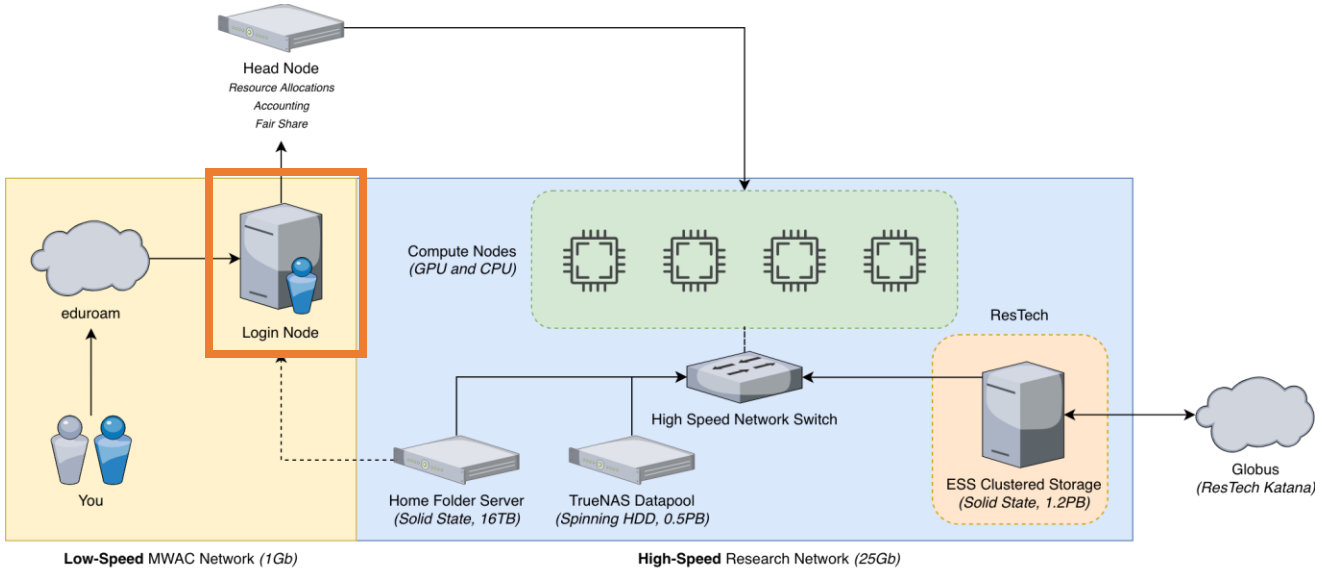
Resource requests: job time, memory, cpus, gpus



Source: Carlo Nardone, Computer Museum: First Google Cluster rack (detail of networking)

- Don't worry about model compatibility
  - That's for the computer engineers
- Think about how your science:
  - Scales – doubles, squared?
  - Runs – Parallel, sequential?
  - Size – memory, disk space?

Cluster Architecture

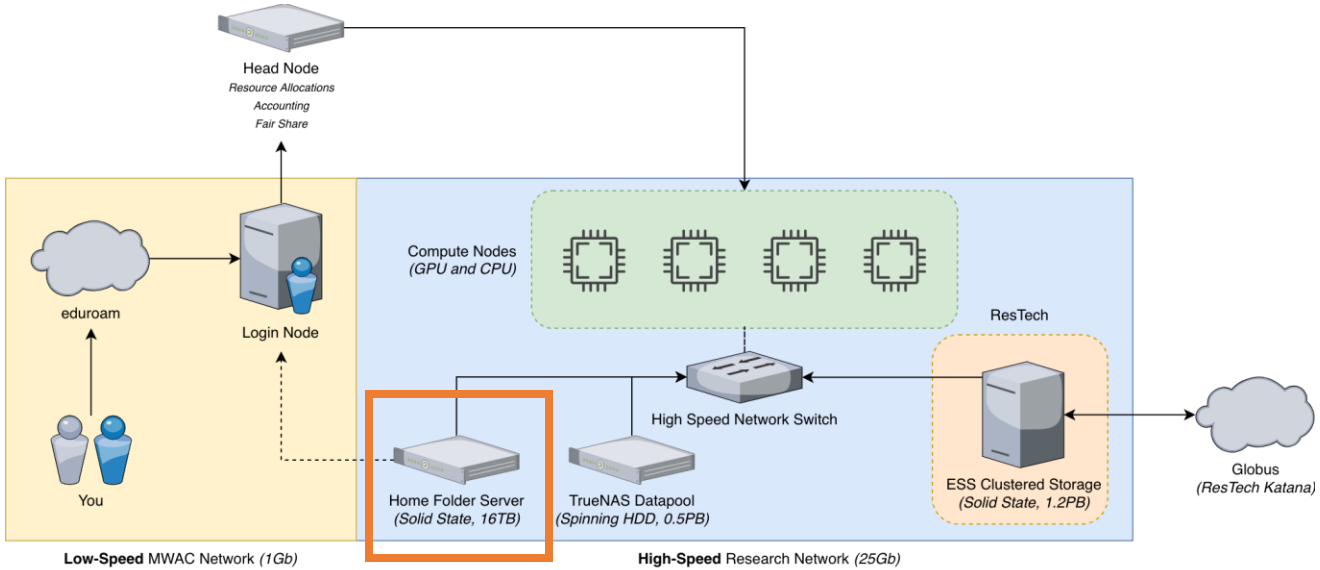


Login node
The login node is how you access the compute cluster. It is a low resource node that can provide access to some cluster resources

On the login node, you can
- Start, stop, and manage jobs
- Access your home folder
- Move small datasets to-and-from your home folder
- Run persistent shell sessions

You cannot
- Access your research data
- Run compute workloads
- Access a GPU
- Copy large datasets

Cluster Architecture



Home Folder Server
This server contains your home directory. This data is yours and cannot be shared with anyone else. It is accessible at /home/{Your zID}

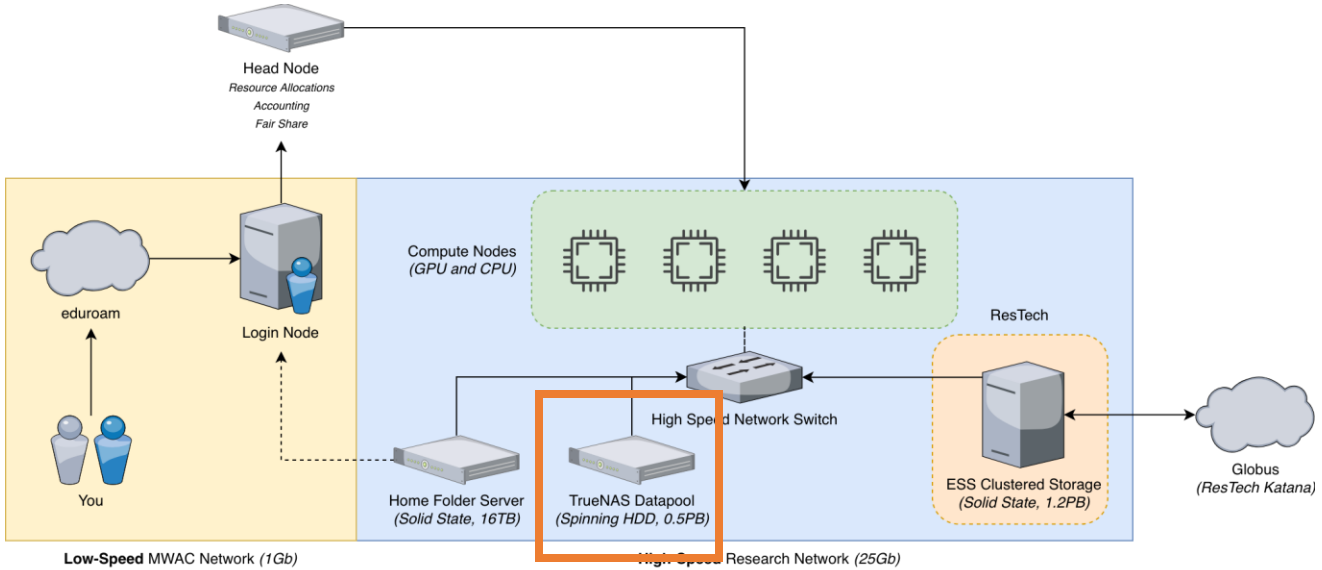
Do keep in your home
- Scripts
- Logs
- Documents
- Configuration Files

Don't keep in your home
- Research data
- Large shared datasets
- Projects

Backups and snapshots
Nightly snapshots of your home folder are taken and backed up to an external data source. Snapshots are kept for 14 days. Files accidentally deleted, modified, or corrupted can be recovered within 14 days.

Usage and quota
Your home folder is on fast SSD storage. However, due it it's (relatively) small size, your home folder is limited to **100GB** in size by default. This limit can be increased temporarily, if your workload demands

Cluster Architecture

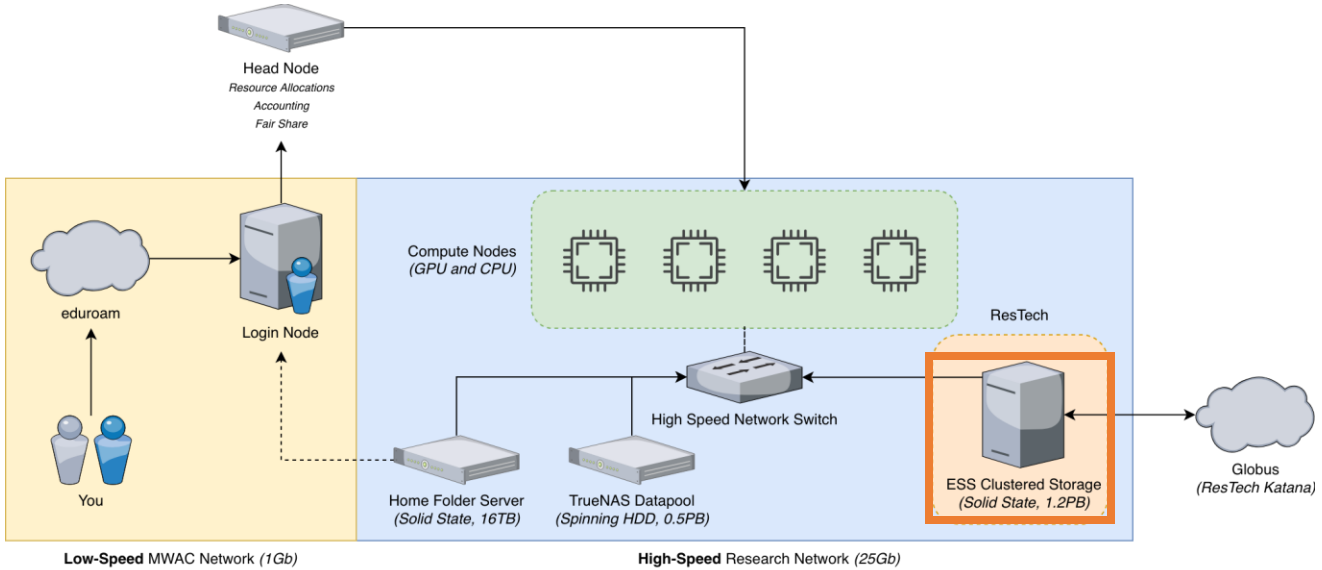


TrueNAS Datapool
This server is in the process of having it's data moved onto the ESS. However, it is still used for training. It is accessible on **compute nodes only** at /data/truenas

Do keep on the TrueNAS
- Tutorial datasets
- Tutorial projects
- Intermediary files
- Archive/backups of files in your home folder

Usage and quota
Due to it's large size, the TrueNAS has no quota limitations. However, it is running spinning hard-disk drives, so access and data streaming may be considerably slower

Cluster Architecture



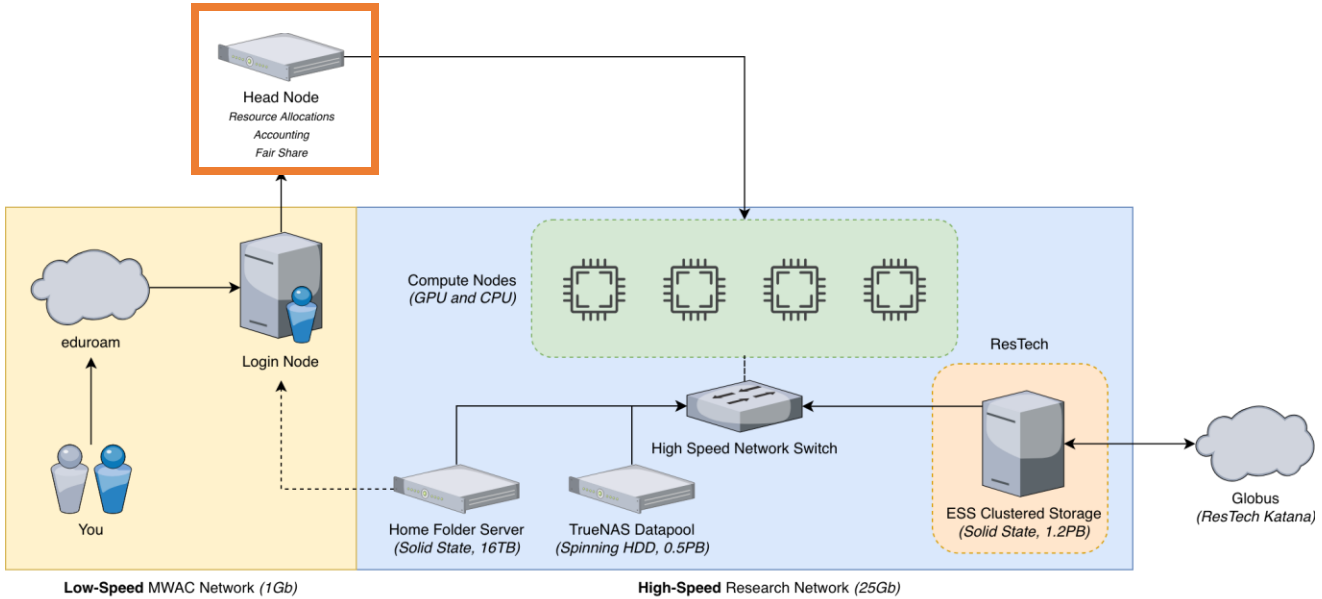
ResTech ESS
This data storage is managed by Research Technology. It is made of fast solid state flash storage It is accessible on **compute nodes only** at
/data/ess/

Do keep on the ESS
- Collected datasets
- Projects

Usage and quota
Usage and quota is handled by Research Technology. If you have collected data from a microscope at UNSW, it will be available on the ESS

Globus
Globus can be used to transfer research data into and out of UNSW over the internet. If you have collected data outside of UNSW, you can move it to UNSW via Globus.

Cluster Architecture

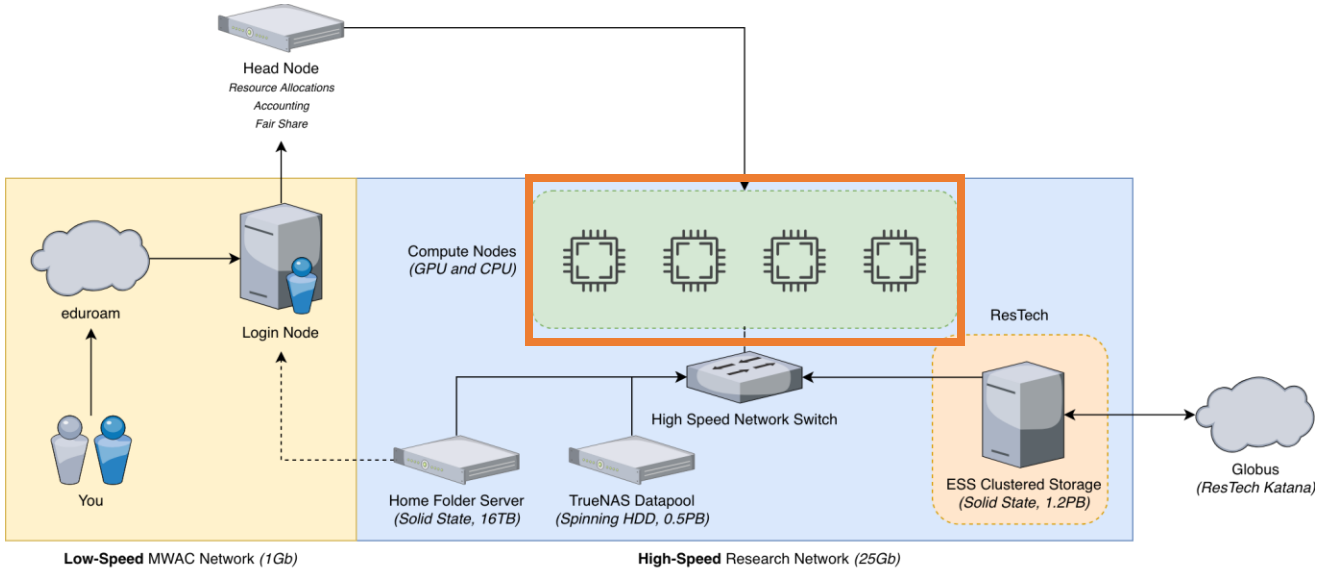


Head Node
The head node is what manages the entire compute cluster. It looks after resource allocations, and managed the queue when no resources are available

Billing & Accounting
This server keeps track of cluster usage per user and per billing account. If you are a member of the facility, this server will assist in automating the billing process

Fair Share & QoS
The cluster will provide "best effort" quality of service to ensure that there are always resources available. However, when the cluster is busy, your job may sit in queue. Fair Share ensures your job is priorities based on your past usage.

Cluster Architecture

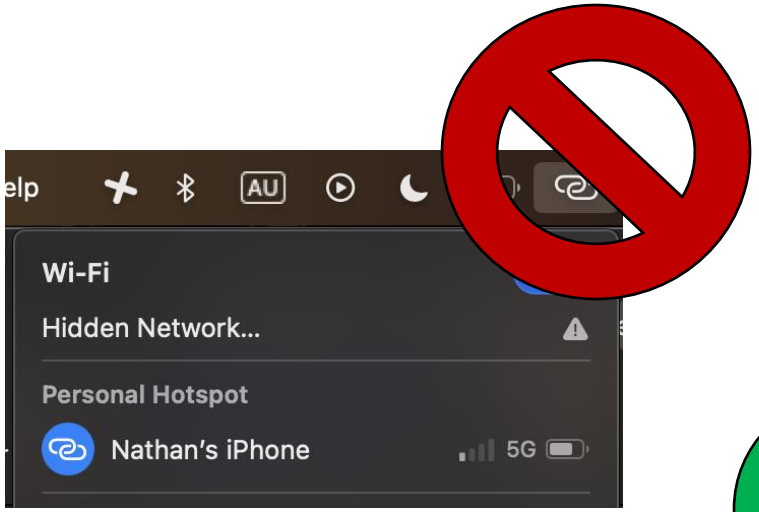


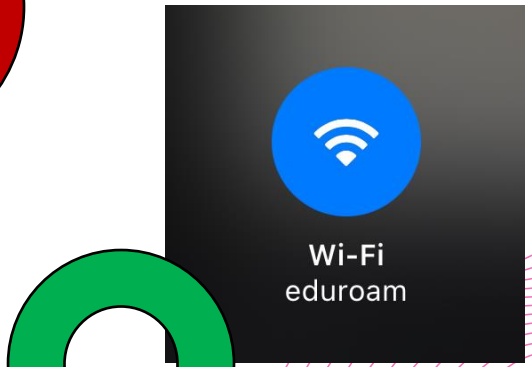
Compute Nodes
These are the servers that perform work. They are equipped with powerful CPUs, GPUs, and a lot of RAM.

Resources
Currently, the cluster's compute resources feature:
- 9 GPUs
- 176 CPU cores
- 386 GB of RAM

Accessing the cluster

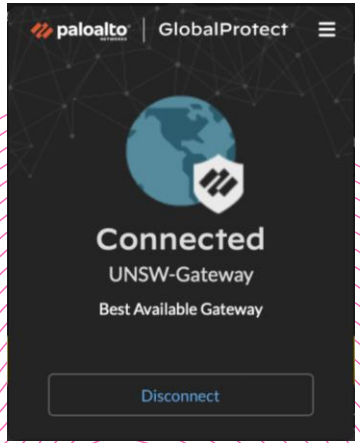
Step 0: Ensure correct network connection





(With UNSW Credentials)

OR



Accessing the cluster

Step 1: Open a terminal application

MacOS

Terminal

Windows

Command Prompt

Linux

Your preferred terminal emulator

Accessing the cluster

Step 2: Connect to the login node

```
% ssh z3545907@discworld.analytical.unsw.edu.au
```

Accessing the cluster

Step 3: Success

Accessing compute

Interactive Session

Note the change of the hostname. This shows that we are now on a compute node

Accessing compute

Requesting longer sessions

```
Good afternoon, Nathan!
[z3545907@discworld0 ~]$ sinteractive --time 12:00:00
[z3545907@c01 ~]$
```

This job will run for 12 hours

Accessing compute

Requesting more resources

```
[z3545907@discworld0 ~]$ sinteractive --mem 16G --cpus 4 --time 04:00:00
[z3545907@c01 ~]$
```

This job will run for **4 hours** with **16GB of RAM** and **4 CPU cores**

Accessing compute

Run with a GPU

```
[z3545907@discworld0 ~]$ sinteractive --gpu
[z3545907@c01 ~]$
```

This job will run with a GPU

Accessing compute

Queue status

```
[z3545907@discworld0 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1784 prod-dw-s cryospar z3535074  R 1-07:51:40      1 c01
              1861 prod-dw-s cryospar z5025120  R    7:05:28      1 c00
              1865 prod-dw-s cryospar z5025120  R    6:51:36      1 c00
              1862 prod-dw-s cryospar z5025120  R    7:00:51      1 c00
              1886 prod-dw-s sdesktop z3545907  R       6:18      1 c01
```

'squeue' will show all jobs (running and queued) on the cluster

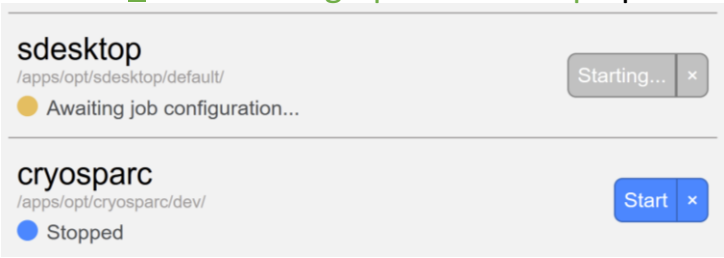
command line

`tldr program` • shows quick use

```
[z3374843@discworld0 ~]$ tldr cp

  cp

  Copy files and directories.
  More information: https://www.gnu.org/software/coreutils/manual/html_node/

  - Copy a file to another location:
    cp path/to/source_file path/to/target_file

  - Copy a file into another directory, keeping the filename:
    cp path/to/source_file path/to/target_parent_directory

  - Recursively copy a directory's contents to another location (if the dest
    cp [-r|--recursive] path/to/source_directory path/to/target_directory
[z3374843@discworld0 ~]$ tldr diff

  diff

  Compare files and directories.
  More information: https://manned.org/diff.

  - Compare files (lists changes to turn `old_file` into `new_file`):
    diff old_file new_file

  - Compare files, ignoring white spaces:
    diff [-w|--ignore-all-space] old_file new_file
```

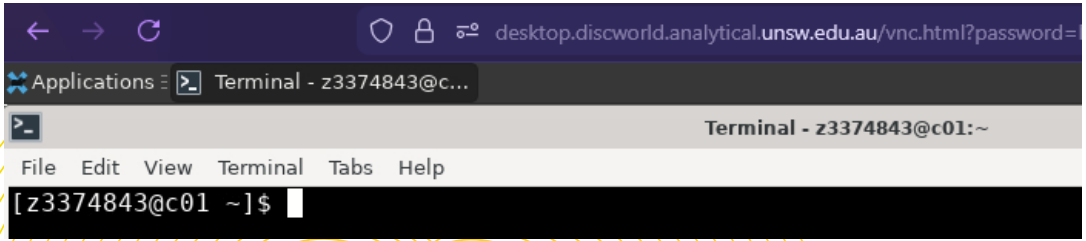
portal.discworld.analytical.unsw.edu.au

You can have 1 interactive graphical desktop open at once



(default desktop = no powerful GPU, just interface)

You can have *as many* command line terminals
(*via* `sdesktop` or SSH) open at once as you like



✨ AI copilots

Are great at writing & explaining typical tasks

We have tried to make the command line more *readable* than standard (zsh, default on Mac)
Can still run the 'typical' shell (`bash program.sh`) but look up zsh/our docs. **We can change your shell**

Text editing

Keeping things in text makes it simple to `edit`

```
z3374843@discworld:~ edit hello_discworld.sh
1 echo "Hello Discworld!"
```

Cntrl + S

```
hello_discworld.sh (1,1) | ft:shell | unix |
Saved hello_discworld.sh
```

Cntrl + Q

```
z3374843@discworld:~
```

The `edit` command opens an editor (`micro`) with common muscle-memory (not most efficient)

To save time with jobs, `sdraft` creates a template – you `edit` in the details

```
z3374843@discworld:~ sdraft hello_discworld_x10
1  #!/bin/bash
2
3  #SBATCH --job-name=hello_discworld_x10 # Job name
4  #SBATCH --time=00:10:00                # Time limit (HH:MM:SS)
5  #SBATCH --cpus-per-task=1              # CPUs per task
6  #SBATCH --mem=4GB                      # Memory per node
7
8  for repetition in {0..10}
9  do
10     bash hello_discworld.sh
11 done
```

```
z3374843@discworld:~
Hello Discworld!
Hello Discworld!
Hello Discworld!
Hello Discworld!
Hello Discworld!
Hello Discworld!
Hello Discworld!
Hello Discworld!
Hello Discworld!
Hello Discworld!
Hello Discworld!
```

UNSW SYDNEY

References online

Discworld User Docs

unsw.atlassian.net/wiki/spaces/DUD/overview

Command-line **tools**

Way faster for day-to-day tasks with big datasets
You're in control – of managing *personal* data. Includes permissions & `rm`
Remember – still have file browser in `sdesktop`

- Folders are called `directories` in Linux land
- `paths` are as `/full/path/to/file` or, in shorthand: `.` 'here' `./` 'here folder', `../` 'folder above'
- `/home` = `~` takes you home quickly
- Up arrow (`↑`) goes back through your shell commands
- `Cntrl + r` starts a 'reverse search' that lets you search for old commands by a particular word
- `Cntrl + c` cancels whatever action you were doing
- `ls -al` will give you *all* files, including the ones hidden by the magic `.` in front
  It *also* shows `permissions` of who can `read`, `write` (hence overwrite/delete), run/`execute` files
- `du` shows you the 'disk usage' of a folder – that is, where all the file storage is going
- `find` will help you locate where a particular file is. (we also have the more efficient `fd`)
- `exit` (or sometimes `quit`) will get your out of most programs, including logout from Discworld
- `cd -` will take you back to the last folder you were in

UNSW SYDNEY

References online

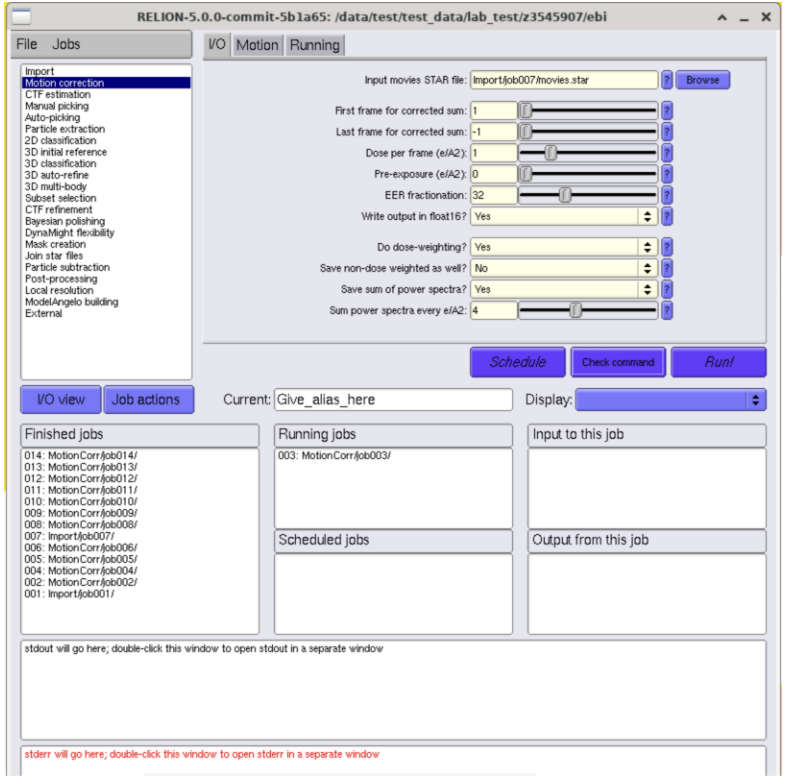
Discworld User Docs

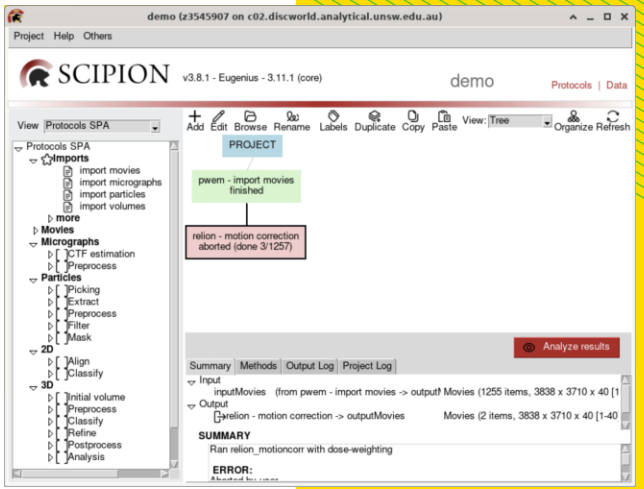
unsw.atlassian.net/wiki/spaces/DUD/overview

Live work & support time

(we have the room all day)

Running Jobs

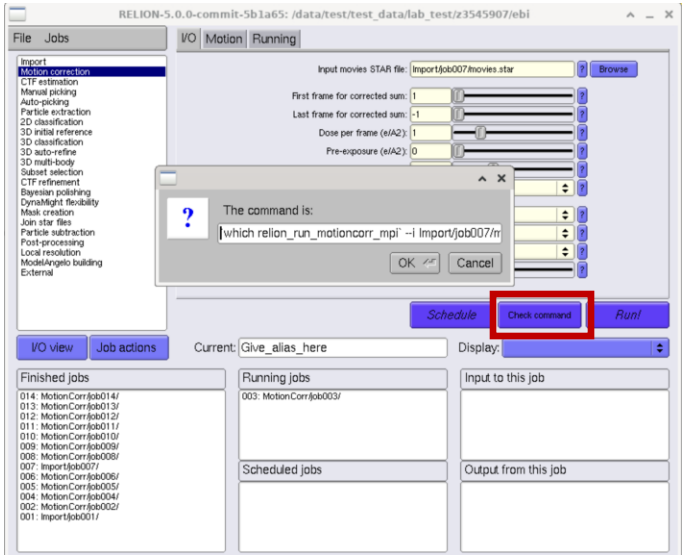




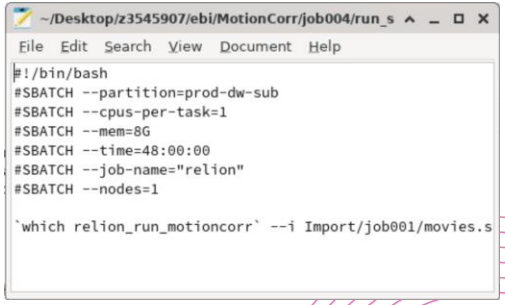
UNSW SYDNEY

What is **actually** going on

Under the hood, these applications are just a front-end to constructing jobs. You can see this best in RELION.



run_submit.script



The check command button and run_submit.script files show us what is really going on. Under the hood, its just **running commands on the compute cluster.**

What if we did this ourselves?

This probably raises some questions:

How would I do this?

Why would I want to do this?

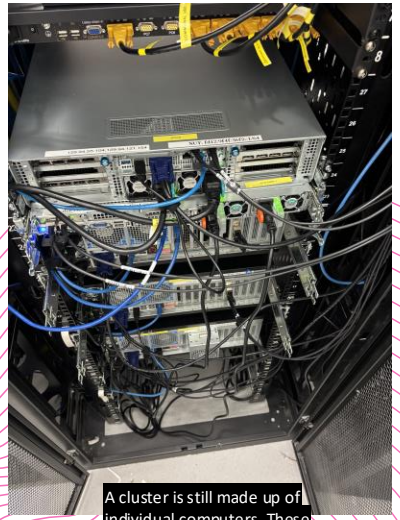
When should I do this?

Batching and array jobs

UNSW SYDNEY

Understanding the benefits

BUT…



Understanding the benefits

”Run this job for me, please”

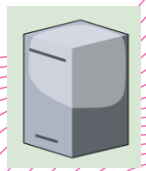
1 GPU
4 CPUs
64 GB RAM

“Okay! Your job ID is 1234”

2 GPUs
64 CPUs
64 GB RAM

2 GPUs
64 CPUs
64 GB RAM

2 GPUs
64 CPUs
64 GB RAM



2 GPUs
64 CPUs
64 GB RAM

Understanding the benefits

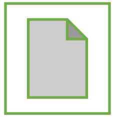
"Please run this job for me"

1 GPU
4 CPUs
16 GB RAM

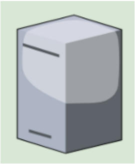
"Okay, your job is waiting in queue"

"Actually, this is a small job. I can start it now, and it won't affect other's wait times"

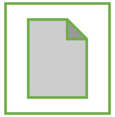
2 GPUs
64 CPUs
64 GB RAM



1 GPU
16 CPUs
16 GB RAM

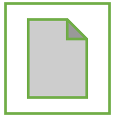


2 GPUs
64 CPUs
64 GB RAM

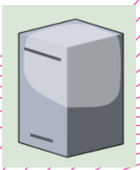


1 GPU
16 CPUs
16 GB RAM

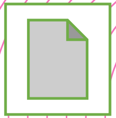
2 GPUs
64 CPUs
64 GB RAM



1 GPU
16 CPUs
16 GB RAM



2 GPUs
64 CPUs
64 GB RAM



1 GPU
16 CPUs
16 GB RAM

Understanding the benefits

”Run this job for me, please”

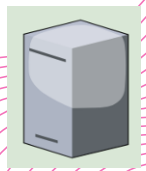
8 GPU
256 CPUs
256 GB RAM

“Sorry, I can’t fit this job anywhere!”

2 GPUs
64 CPUs
64 GB RAM

2 GPUs
64 CPUs
64 GB RAM

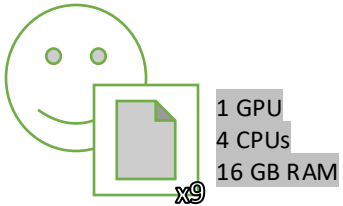
2 GPUs
64 CPUs
64 GB RAM



2 GPUs
64 CPUs
64 GB RAM

Array job?

”Run this 9x array job for me, please”



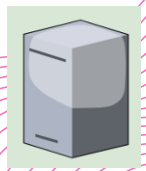
1 GPU
4 CPUs
16 GB RAM

”This job is now waiting in queue”

2 GPUs
64 CPUs
64 GB RAM

2 GPUs
64 CPUs
64 GB RAM

2 GPUs
64 CPUs
64 GB RAM



2 GPUs
64 CPUs
64 GB RAM

Array job?

From this array job, we got to use:

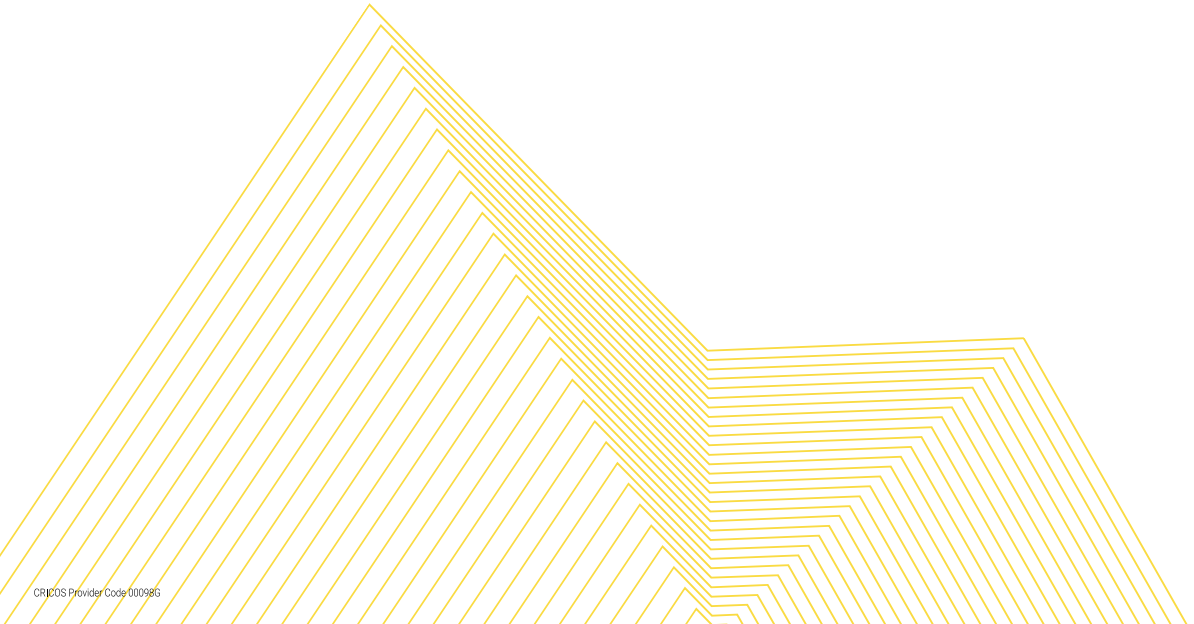
1 GPU
4 CPUs
16 GB RAM

9 GPUs
36 CPU cores
144 GB of RAM

As a single job, this **could not fit on a single compute node.**
But the job still ran, since it was structured as an array job.

2 GPUs
64 CPUs
64 GB RAM

Structuring an array job



UNSW SYDNEY

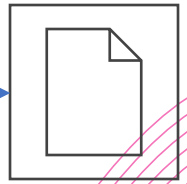
Traditional ”Monolith” job

samplesheet.csv

Sample ID	Path
0	/data/ess/z1234567/samples/0.fasta
1	/data/ess/z1234567/samples/1.fasta
2	/data/ess/z1234567/samples/2.fasta
3	/data/ess/z1234567/samples/3.fasta
4	/data/ess/z1234567/samples/4.fasta

Alignment Job (**Job ID 1234**)

Output



Will generate an alignment for each sample one-by-one.

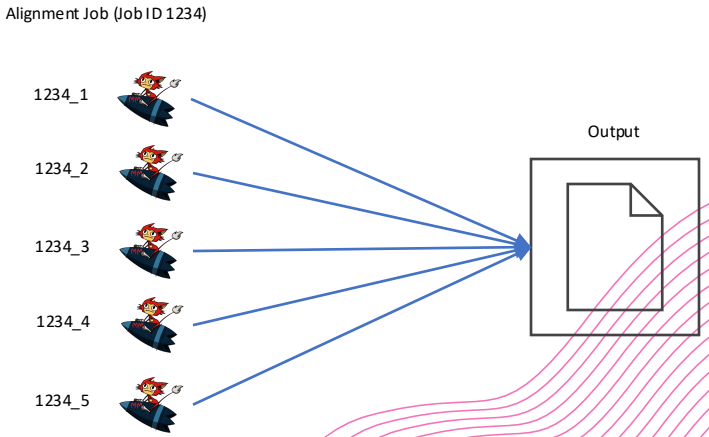
```
for i in samples:
    mmseqs_align(i, output_path)
```

On a small scale, this works okay. But what if we had **100 samples**, or **1000 samples**?

Array job

samplesheet.csv

Sample ID	Path
0	/data/ess/z1234567/samples/0.fasta
1	/data/ess/z1234567/samples/1.fasta
2	/data/ess/z1234567/samples/2.fasta
3	/data/ess/z1234567/samples/3.fasta
4	/data/ess/z1234567/samples/4.fasta



Jobs can be run **simultaneously**
(Given available resources)