Katana User Documentation

Release 1.0

Lachlan Simpson

Sep 22, 2020

CONTENTS:

1 Help and Support **3**
- 1.1 Scheduled Maintenance . . . 3
- 1.2 Contact the Research Technology Services team . . . 3
- 1.3 Katana System Status and Known Issues . . . 4
- 1.4 Katana Terms of Use . . . 4

2 Using Katana **5**
- 2.1 About Katana . . . 5
- 2.2 Accessing Katana . . . 6
- 2.3 Running Jobs on Katana . . . 9
- 2.4 GitHub . . . 18
- 2.5 Using OnDemand . . . 19

3 Storage **21**
- 3.1 Storage Locations . . . 21
- 3.2 Katana Data Mover . . . 24
- 3.3 How to use the UNSW Data Archive . . . 26

4 Software **29**
- 4.1 Environment Modules . . . 29
- 4.2 Biosciences . . . 32
- 4.3 Java . . . 32
- 4.4 Intel Compilers and Software Libraries . . . 32
- 4.5 Operating Systems . . . 32
- 4.6 Perl . . . 33
- 4.7 Python . . . 33
- 4.8 Python Virtual Environments . . . 34
- 4.9 Virtual Environments from the inside . . . 38
- 4.10 Jupyter Notebooks . . . 39
- 4.11 R and RStudio . . . 40
- 4.12 SAS . . . 40
- 4.13 Stata . . . 40
- 4.14 TMUX . . . 41
- 4.15 Compressing Large Directories . . . 41

5 Reference Data **43**

6 Frequently Asked Questions **45**
- 6.1 General FAQ . . . 45
- 6.2 Scheduler FAQ . . . 45
- 6.3 Storage FAQ . . . 49

6.4 Expanding Katana 51
6.5 Acknowledging Katana 51

7 Glossary **53**

8 News **55**

Index **57**

This document is available in `pdf format` for offline reading.

CHAPTER

ONE

HELP AND SUPPORT

1.1 Scheduled Maintenance

Attention: There is no maintenance scheduled for the moment.

1.2 Contact the Research Technology Services team

Katana issues including: functional issues, software installation, reference data sets, general questions: Email the IT Service Centre, including the word **Katana** in the subject line.

Note: This is the best and primary way to get help from UNSW Research Technology Services beyond this document. You *must* use your UNSW email address *or* your zid. Without this information, we have no idea who you might be.

When writing your email, please include a clear and detailed description of the issue experienced, including error messages and node name. Something like "It doesn't work" doesn't help us help you! If at all possible, include the steps someone else needs to do to reproduce the problem, the job identifier, the date and time of your problem and on which Katana node it occurred, the script filename and the directory you were running from.

Example of a bad request i'm trying to do some work on katana, but it seems that the server is slow or not responsive at times. i'm logged in from inside unsw today, so working from home shouldn't be the issue.

Example of a great request When I tried to run Sentaurus TCAD today (2020-05-01) on Katana I got this error message regardless of structures I wanted to simulate:

"Job failed Error: Child process with pid '116643' got the signal 'SIGSEGV' (segmentation violation) gjob exits with status 1"

My job ran on k052 with jobid 300000, my zid is z2134567

For face to face support: Hacky Hour Thursdays, 3pm.

For questions about research data at UNSW on storage, movement or Data Management Plans, please email the Research Data Team

1.3 Katana System Status and Known Issues

No known issues at the moment.

1.4 Katana Terms of Use

Any use of Katana is covered by the Conditions of Use - UNSW ICT Resources.

Important: Katana is not suitable for highly sensitive data. You should use the UNSW Data Classification scheme to classify your data and learn about managing your research data by visiting the Research Data Management Hub.

CHAPTER

TWO

USING KATANA

2.1 About Katana

Katana is a shared computational cluster located on campus at UNSW that has been designed to provide easy access to computational resources. With over 4000 CPU cores spread over a large number of compute nodes each with up to 1Tb of memory, Katana provides a flexible compute environment where users can run jobs that wouldn't be possible or practical on a desktop or laptop.

Katana is powerful on it's own, but can be seen as a training or development base before migrating up to systems like as Australia's peak HPC system Gadi, located at NCI. Research Technology Services also provide training, advice and support for new users or those uncertain if High Performance Computing is the right fit for their research needs.

2.1.1 System Configuration

- RPM based Linux OSes. RedHat on the management plane, CentOS on the nodes
- PBSPro version 19.1.3
- Large global scratch at `/srv/scratch`, local scratch at `$TMPDIR`
- 12, 48, 100, 200 hour *Walltime* queues with prioritisation

2.1.2 Compute

- Hetergenous hardware: Dell, Lenovo, Huawei.
- roughly 170 nodes

2.1.3 GPU Compute

The most popular use of these nodes is for Tensorflow.

- four GPU capable nodes, Tesla V100-SXM2, 32GB
- three are dedicated for the department that owns them
- one is general use for all researchers

2.2 Accessing Katana

Anyone at UNSW can apply for a general account on Katana. This level is designed for those that think Katana would suit their research needs or will typically use less than 10,000 CPU hours a quarter. This level still gets access to the same level of support including software installation, help getting started or running their jobs. The only difference is the number of compute jobs that can be run at any time and how long they can run for - general users can only use a 12 hour *Walltime*.

If your needs require more CPU hours or consulation, some Faculties, Schools and Research Groups have invested in Katana and have a higher level of access. Users in this situation should speak to their supervisor.

2.2.1 Requesting an Account

To apply for an account you can send an email to the UNSW IT Service Centre giving your zID, your role within UNSW and the name of your supervisor or head of your research group.

2.2.2 Connecting to Katana

Note: When you are connecting to Katana via `katana.restech.unsw.edu.au` you are connecting to one of two login nodes `katana1.restech.unsw.edu.au` or `katana2.restech.unsw.edu.au`. If you have a long running *TMUX* open, you will need to login to the node on which it was started.

Linux and Mac

From a Linux or Mac OS machine you can connect via ssh in a terminal:

```
laptop:~$ ssh z1234567@katana.restech.unsw.edu.au
```

Windows

From a Windows machine a SSH client such as PuTTY or MobaXTerm is required.

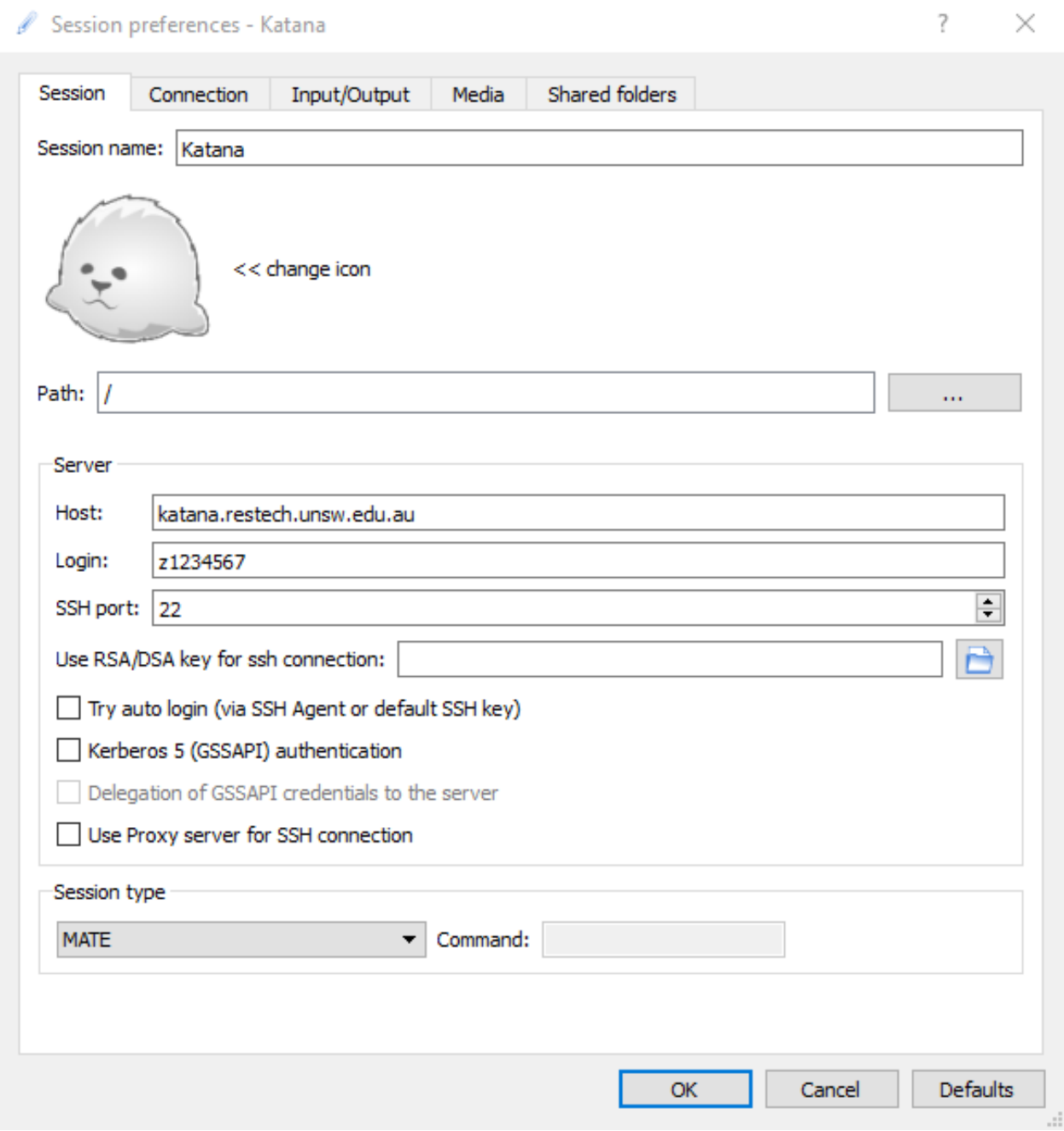
Graphical sessions

Warning: Please use the provided *Using OnDemand* option rather than the following. It's significantly easier and faster.

Some software - **Ansys**, *Jupyter Notebooks*, **Matlab**, and *R and RStudio* being among the most popular - are easier with a graphical session. If you require an interactive graphical session to Katana then you can use the X2Go client.

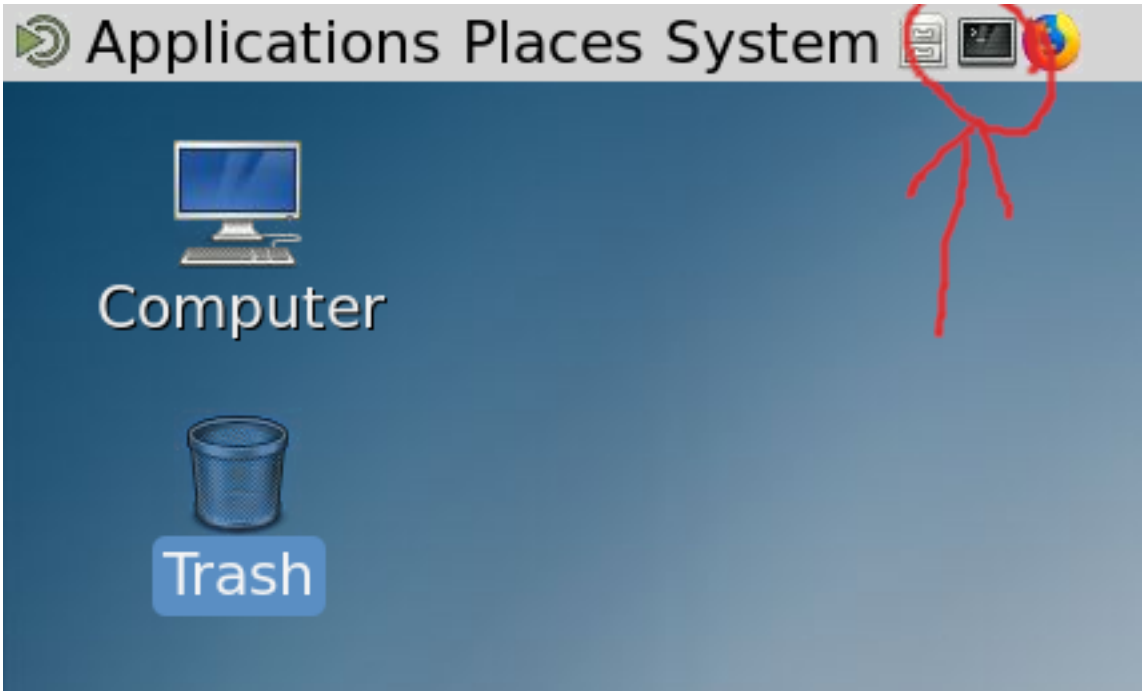
Start X2Go and create a session for Katana. The details that you need to enter for the session are:

```
Session name: Katana
Host: katana.restech.unsw.edu.au
Login: zID
Session type: Mate
```



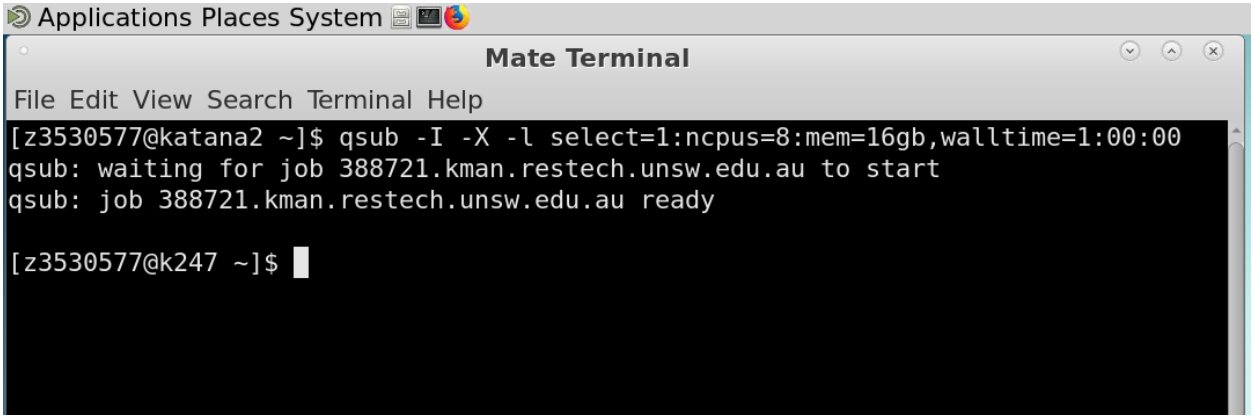
Warning: The usability of a graphical connection to Katana is highly dependent on network latency and performance.

Once you have logged into a Katana desktop, you should start a terminal

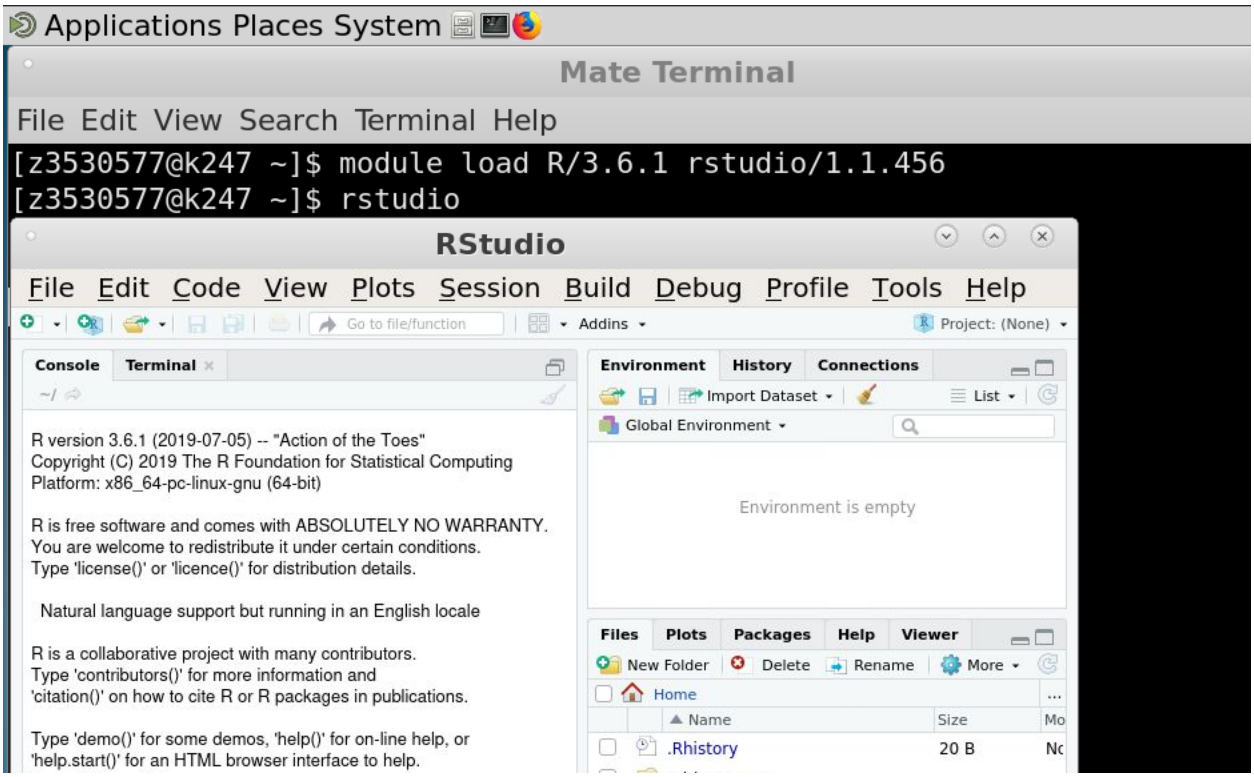


Then run an interactive session. Here you can see a command similar to what you would run for an interactive session with 8 CPUs and 16 GB for one hour. You will probably need more time. You can tell your interactive session has started when you see the name of the machine change - in this image I am on k247.

```
qsub -I -X -l select=1:ncpus=8:mem=16gb,walltime=1:00:00
```



Once that's started, you can load the modules and run the command line name of the software you want. That is how you run Graphical Interfaces or GUIs using Katana's grunt.



2.3 Running Jobs on Katana

2.3.1 Brief Overview

The *Login Node* of a cluster is a shared resource for all users and is used for preparing, submitting and managing jobs. Never run any computationally intensive processes on the login nodes. Jobs are submitted from the login node, which delivers them to the *Head Node* for job and resource management. Once the resources have been allocated and are available, the job will run on one or more of the compute nodes as requested.

Different clusters use different tools to manage resources and schedule jobs - OpenPBS and SLURM are two popular systems. Katana, like NCI's Gadi, uses OpenPBS for this purpose.

Jobs are submitted using the `qsub` command. There are two types of job that `qsub` will accept: an *Interactive Job* and a *Batch Job*. Regardless of type, the resource manager will put your job in a *Queue*.

An **interactive job** provides a shell session on a *Compute Nodes*. You interact directly with the compute node running the software you need explicitly. Interactive jobs are useful for experimentation, debugging, and planning for **batch jobs**.

In contrast, a *Batch Job* is a scripted job that - after submission via `qsub` - runs from start to finish without any user intervention. The vast majority of jobs on the cluster are batch jobs. This type of job is appropriate for production runs that will consume several hours or days.

To submit a *Batch Job* you will need to create a job script which specifies the resources that your job requires and calls your program. The general structure of *A Job Script* is shown below.

Important: All jobs go into a *Queue* while waiting for resources to become available. The length of time your jobs wait in a queue for resources depends on a number of factors.

The main resources available for use are Memory (RAM), *CPU Core* (number of CPUs) and *Walltime* (how long you want the CPUs for). These need to be considered carefully when writing your job script, since the decisions you make will impact which queue your jobs ends up on.

As you request more memory, the number of available queues goes down. The memory limits, in GB, at which the number of queues decreases are 124, 180, 248, 370, 750 and 1000.

Similarly, when considering the number of CPU cores, the available resources reduce at 16, 20, 24, 28, 32, 44 and 64 CPU cores.

Walltime provides the biggest constraint on your submitted jobs. The points at which resource availability is reduced are 12, 48, 100 and 200 hours

Jobs have the following restrictions:

- Jobs of up to 12 hours can run anywhere on the cluster
- Jobs of up to 100 hours can run on nodes belonging to your group and the general nodes
- Jobs of up to 200 hours can only run on nodes belonging to your group

2.3.2 Interactive Jobs

An interactive job or interactive session is a session on a compute node with the required physical resources for the period of time requested. To request an interactive job, add the -I flag (capital i) to `qsub`. Default sessions will have 1 CPU core, 1GB and 1 hour

For example, the following two commands. The first provides a default session, the second provides a session with two CPU core and 8GB memory for three hours. You can tell when an interactive job has started when you see the name of the server change from `katana1` or `katana2` to the name of the server your job is running on. In these cases it's `k181` and `k201` respectively.

```
[z1234567@katana1 ~]$ qsub -I
qsub: waiting for job 313704.kman.restech.unsw.edu.au to start
qsub: job 313704.kman.restech.unsw.edu.au ready
[z1234567@k181 ~]$
```

```
[z1234567@katana2 ~]$ qsub -I -l select=1:ncpus=2:mem=8gb,walltime=3:00:00
qsub: waiting for job 1234.kman.restech.unsw.edu.au to start
qsub: job 1234.kman.restech.unsw.edu.au ready
[z1234567@k201 ~]$
```

Jobs are constrained by the resources that are requested. In the previous example the first job - running on `k181` - would be terminated after 1 hour or if a command within the session consumed more than 8GB memory. The job (and therefore the session) can also be terminated by the user with `CTRL-D` or the `logout` command.

Interactive jobs can be particularly useful while developing and testing code for a future batch job, or performing an interactive analysis that requires significant compute resources. Never attempt such tasks on the login node – submit an interactive job instead.

2.3.3 Batch Jobs

A batch job is a script that runs autonomously on a compute node. The script must contain the necessary sequence of commands to complete a task independently of any input from the user. This section contains information about how to create and submit a batch job on Katana.

Getting Started

The following script simply executes a pre-compiled program ("myprogram") in the user's home directory:

```
#!/bin/bash

cd $HOME

./myprogram
```

This script can be submitted to the cluster with `qsub` and it will become a job and be assigned to a queue. If the script is in a file called `myjob.pbs` then the following command will submit the job with the default resource requirements (1 CPU core with 1GB of memory for 1 hour):

```
[z1234567@katana1 ~]$ qsub myjob.pbs
1237.kman.restech.unsw.edu.au
```

As with interactive jobs, the `-l` (lowercase L) flag can be used to specify resource requirements for the job:

```
[z1234567@katana ~]$ qsub -l select=1:ncpus=1:mem=4gb,walltime=12:00:00 myjob.pbs
1238.kman.restech.unsw.edu.au
```

If we wanted to use the GPU resources, we would write something like this - note that because of configuration of machines, you should request: *ncpus=(#ngpus*8):mem=(#ngpus*46)*

```
[z1234567@katana ~]$ qsub -l select=1:ncpus=8:ngpus=1:mem=46gb,walltime=12:00:00␣
↪myjob.pbs
1238.kman.restech.unsw.edu.au
```

A Job Script

Job scripts offer a much more convenient method for invoking any of the options that can be passed to `qsub` on the command-line. In a shell script, a line starting with # is a comment and will be ignored by the shell interpreter. However, in a job script, a line starting with #PBS can be used to pass options to the `qsub` command.

Here is an overview of the different parts of a job script which we will examine further below. In the following sections we will add some code, explain what it does, then show some new code, and iterate up to something quite powerful.

For the previous example, the job script could be rewritten as:

```
#!/bin/bash

#PBS -l select=1:ncpus=1:mem=4gb
#PBS -l walltime=12:00:00

cd $HOME

./myprogram
```

This structure is the most common that you will use. The top line must be `#!/bin/bash` - we are running bash scripts, and this is required. The following section - the lines starting with `#PBS` - are where we will be configuring how the job will be run - here we are asking for resources. The final section shows the commands that will be executed in the configured session.

The script can now be submitted with much less typing:

```
[z1234567@katana ~]$ qsub myjob.pbs
1239.kman.restech.unsw.edu.au
```

Unlike submission of an interactive job, which results in a session on a compute node ready to accept commands, the submission of a batch job returns the ID of the new job. This is confirmation that the job was submitted successfully. The job is now processed by the job scheduler and resource manager. Commands for checking the status of the job can be found in the section *Managing Jobs on Katana*.

If you wish to be notified by email when the job finishes then use the `-M` flag to specify the email address and the `-m` flag to declare which events cause a notification. Here we will get an email if the job aborts (`-m a`) due to an error or ends (`-m e`) naturally.

```
#PBS -M your.name.here@unsw.edu.au
#PBS -m ae
```

The output that would normally go to screen and error messages of a batch job will be saved to file when your job ends. By default these files will be called `JOB_NAME.oJOB_ID` and `JOB_NAME.eJOB_ID`, and they will appear in the directory that was the current working directory when the job was submitted. In the above example, they would be `myjob.o1239` and `myjob.e1239`. You can merge these into a single file with the `-j oe` flag. The `-o` flag allows you to rename the file.

```
#PBS -j oe
#PBS -o /home/z1234567/results/Output_Report
```

When a job starts, it needs to know where to save it's output and do it's work. This is called the *current working directory*. By default the job scheduler will make your *current working directory* your home directory (`/home/z1234567`). This isn't likely or ideal and is important that each job sets its current working directory appropriately. There are a couple of ways to do this, the easiest is to set the *current working directory* to the directory you are in when you execute `qsub` by using

```
cd $PBS_O_WORKDIR
```

There is one last special variable you should know about, especially if you are working with large datasets. The storage on the compute node your job is running on will always be faster than the network drive.

If you use the storage close to the CPUs - in the server rather than on the shared drives, called *Local Scratch* - you can often save hours of time reading and writing across the network.

In order to do this, you can copy data to and from the local scratch, called `$TMPDIR`:

```
cp /home/z1234567/project/massivedata.tar.gz $TMPDIR
tar xvf massivedata.tar.gz
my_analysis.py massive_data
cp -r $TMPDIR/my_output /home/z1234567
```

There are a lot of things that can be done with PBSPro, but you don't and won't need to know it all. These few basics will get you started.

Here's the full script as we've described. You can copy this into a text editor and once you've changed our dummy values for yours, you only need to change the last line.

```
#!/bin/bash

#PBS -l select=1:ncpus=1:mem=4gb
#PBS -l walltime=12:00:00
#PBS -M your.name.here@unsw.edu.au
#PBS -m ae
#PBS -j oe
#PBS -o /home/z1234567/results/Output_Report

cd $PBS_O_WORKDIR

./myprogram
```

2.3.4 Array Jobs

One common use of computational clusters is to do the same thing multiple times - sometimes with slightly different input, sometimes to get averages from randomness within the process. This is made easier with array jobs.

An array job is a single job script that spawns many almost identical sub-jobs. The only difference between the sub-jobs is an environment variable `$PBS_ARRAY_INDEX` whose value uniquely identifies an individual sub-job. A regular job becomes an array job when it uses the `#PBS -J` flag.

For example, the following script will spawn 100 sub-jobs. Each sub-job will require one CPU core, 1GB memory and 1 hour run-time, and it will execute the same application. However, a different input file will be passed to the application within each sub-job. The first sub-job will read input data from a file called `1.dat`, the second sub-job will read input data from a file called `2.dat` and so on.

Note: In this example we are using brace expansion - the {} characters around the bash variables - because they are needed for variables that change, like array indices. They aren't strictly necessary for `$PBS_O_WORKDIR` but we include them to show consistency.

```
#!/bin/bash

#PBS -l select=1:ncpus=1:mem=1gb
#PBS -l walltime=1:00:00
#PBS -j oe
#PBS -J 1-100

cd ${PBS_O_WORKDIR}

./myprogram ${PBS_ARRAY_INDEX}.dat
```

There are some more examples of array jobs including how to group your computations in an array job on the UNSW Github HPC examples page.

2.3.5 Splitting large Batch Jobs

If your batch job can be split into multiple steps you may want to split one big job up into a number of smaller jobs. There are a number of reasons to spend the time to implement this.

1. If your large job runs for over 200 hours, it won't finish on Katana.
2. If your job has multiple steps which use different amounts of resources at each step. If you have a pipeline that takes 50 hours to run and needs 200GB of memory for an hour, but only 50GB the rest of the time, then the memory is sitting idle.
3. Katana has prioritisations based on how many resources any one user uses. If you ask for 200GB of memory, this will be accounted for when working out your next job's priority.
4. There's no other way to say this, but because there are more resources for 12 hour jobs, seven or eight 12 hour jobs will often finish well before a single 100 hour job even starts.

2.3.6 Get information about the state of the scheduler

When deciding which jobs to run, the scheduler takes the following details into account:

- are there available resources
- how recently has this user run jobs successfully
- how many resources has this user used recently
- how long is the job's Walltime
- how long has the job been in the queue

You can get an overview of the compute nodes and a list of all the jobs running on each node using `pstat`

```
[z1234567@katana2 src]$ pstat
k001  normal-mrcbio           free          12/44   200/1007gb  314911*12
k002  normal-mrcbio           free          40/44    56/ 377gb  314954*40
k003  normal-mrcbio           free          40/44   375/ 377gb  314081*40
k004  normal-mrcbio           free          40/44    62/ 377gb  314471*40
k005  normal-ccrc             free           0/32     0/ 187gb
k006  normal-physics          job-busy      32/32   180/ 187gb  282533*32
k007  normal-physics          job-busy      32/32   180/ 187gb  284666*32
k008  normal-physics          free           0/32     0/ 187gb
k009  normal-physics          job-busy      32/32   124/ 187gb  314652*32
k010  normal-physics          free           0/32     0/ 187gb
```

To get information about a particular node, you can use `pbsnodes` but on it's own it is a firehose. Using it with a particular node name is more effective:

```
[z1234567@katana2 src]$ pbsnodes k254
k254
     Mom = k254
     ntype = PBS
     state = job-busy
     pcpus = 32
     jobs = 313284.kman.restech.unsw.edu.au/0, 313284.kman.restech.unsw.edu.au/1,␣
↪313284.kman.restech.unsw.edu.au/2
     resources_available.arch = linux
     resources_available.cpuflags = avx,avx2,avx512bw,avx512cd,avx512dq,avx512f,
↪avx512vl
```

(continues on next page)

(continued from previous page)

```
    resources_available.cputype = skylake-avx512
    resources_available.host = k254
    resources_available.mem = 196396032kb
    resources_available.ncpus = 32
    resources_available.node_weight = 1
    resources_available.normal-all = Yes
    resources_available.normal-qmchda = Yes
    resources_available.normal-qmchda-maths_business-maths = Yes
    resources_available.normal-qmchda-maths_business-maths-general = Yes
    resources_available.vmem = 198426624kb
    resources_available.vnode = k254
    resources_available.vntype = compute
    resources_assigned.accelerator_memory = 0kb
    resources_assigned.hbmem = 0kb
    resources_assigned.mem = 50331648kb
    resources_assigned.naccelerators = 0
    resources_assigned.ncpus = 32
    resources_assigned.ngpus = 0
    resources_assigned.vmem = 0kb
    resv_enable = True
    sharing = default_shared
    last_state_change_time = Thu Apr 30 08:06:23 2020
    last_used_time = Thu Apr 30 07:08:25 2020
```

2.3.7 Managing Jobs on Katana

Once you have jobs running, you will want visibility of the system so that you can manage them - delete jobs, change jobs, check that jobs are still running.

There are a couple of easy to use commands that help with this process.

qstat

Show all jobs on the system

`qstat` gives very long output. Consider piping to `less`

```
[z1234567@katana2 ~]$ qstat | less
Job id            Name             User             Time Use S Queue
----------------  ---------------- ----------------  -------- - -----
245821.kman       s-m20-i20-200h   z1234567                0 Q medicine200
280163.kman       Magcomp25A2      z1234567          3876:18: R mech700
282533.kman       Proj_MF_Nu1      z1234567          3280:08: R cosmo200
284666.kman       Proj_BR_Nu1      z1234567          3279:27: R cosmo200
308559.kman       JASASec55        z1234567          191:21:3 R maths200
309615.kman       2020-04-06.BUSC  z1234567          185:00:5 R babs200
310623.kman       Miaocyclegan     z1234567          188:06:3 R simigpu200
...
```

List just my jobs

You can use either your **ZID** or the *Environment Variable* `$USER`

```
[z2134567@katana2 src]$ qstat -u $USER
kman.restech.unsw.edu.au:
                                                            Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
315230.kman.res z2134567 general1 job.pbs       --    1   1    1gb 01:00 Q   --
```

If you add the `-s` flag, you will get slightly more status information.

```
[z1234567@katana2 src]$ qstat -su z1234567

kman.restech.unsw.edu.au:
                                                            Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
315230.kman.res z1234567 general1 job.pbs     61915   1   1    1gb 01:00 R 00:03
   Job run at Fri May 01 at 14:28 on (k019:mem=1048576kb:ncpus=1:ngpus=0)
315233.kman.res z1234567 general1 job.pbs       --    1   1    1gb 01:00 Q   --
    --
```

List information about a particular job

```
[z1234567@katana2 src]$ qstat -f 315236
Job Id: 315236.kman.restech.unsw.edu.au
    Job_Name = job.pbs
    Job_Owner = z1234567@katana2
    job_state = Q
    queue = general12
    server = kman.gen
    Checkpoint = u
    ctime = Fri May  1 14:41:00 2020
    Error_Path = katana2:/home/z1234567/src/job.pbs.e315236
    group_list = GENERAL
    Hold_Types = n
    Join_Path = n
    Keep_Files = n
    Mail_Points = a
    mtime = Fri May  1 14:41:00 2020
    Output_Path = katana2:/home/z1234567/src/job.pbs.o315236
    Priority = 0
    qtime = Fri May  1 14:41:00 2020
    Rerunable = True
    Resource_List.ib = no
    Resource_List.mem = 1gb
    Resource_List.ncpus = 1
    Resource_List.ngpus = 0
    Resource_List.nodect = 1
    Resource_List.place = pack
    Resource_List.select = 1:mem=1gb:ncpus=1
    Resource_List.walltime = 01:00:00
    substate = 10
```

(continues on next page)

(continued from previous page)

```
    Variable_List = PBS_O_HOME=/home/z1234567,PBS_O_LANG=en_AU.UTF-8,
        PBS_O_LOGNAME=z1234567,
        PBS_O_PATH=/home/z1234567/bin:/usr/lib64/qt-3.3/bin:/usr/lib64/ccache:
        /usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/opt/pbs/bin,PBS_O_M
        AIL=/var/spool/mail/z1234567,PBS_O_SHELL=/bin/bash,PBS_O_WORKDIR=/home
        /z1234567/src,PBS_O_SYSTEM=Linux,PBS_O_QUEUE=submission,PBS_O_HOST=kat
        ana2
    etime = Fri May  1 14:41:00 2020
    eligible_time = 00:00:00
    Submit_arguments = -W group_list=GENERAL -N job.pbs job.pbs.JAZDNgL
    project = _pbs_project_default
```

qdel

Remove a job from the queue or kill it if it's started. To remove an array job, you must include the square braces and they will need to be escaped. In that situation you use `qdel 12345\[\]`. Uses the `$JOBID`

```
[z1234567@katana2 src]$ qdel 315252
```

qalter

Once a job has been submitted, it can be altered. However, once a job begins execution, the only values that can be modified are `cputime`, `walltime`, and `run_count`. These can only be reduced.

Users can only lower resource requests on queued jobs. If you need to increase resources, contact a systems administrator. In this example you will see the resources change - but not the `Submit_arguments`

```
[z1234567@katana2 src]$ qsub -l select=1:ncpus=2:mem=128mb job.pbs
315259.kman.restech.unsw.edu.au
[z1234567@katana2 src]$ qstat -f 315259
Job Id: 315259.kman.restech.unsw.edu.au
    ...
    Resource_List.mem = 128mb
    Resource_List.ncpus = 2
    ...
    Submit_arguments = -W group_list=GENERAL -N job.pbs -l select=1:ncpus=2:mem=128mb␣
↪job.pbs.YOOu3lB
    project = _pbs_project_default

[z1234567@katana2 src]$ qalter -l select=1:ncpus=4:mem=512mb 315259; qstat -f 315259
Job Id: 315259.kman.restech.unsw.edu.au
    ...
    Resource_List.mem = 512mb
    Resource_List.ncpus = 4
    ...
    Submit_arguments = -W group_list=GENERAL -N job.pbs -l select=1:ncpus=2:mem=128mb␣
↪job.pbs.YOOu3lB
    project = _pbs_project_default
```

2.3.8 Tips for using PBS and Katana effectively

Keep your jobs under 12 hours if possible

If you request more than 12 hours of `WALLTIME` then you can only use the nodes bought by your school or research group. Keeping your job's run time request under 12 hours means that it can run on any node in the cluster.

Important: Two 10 hour jobs will probably finish sooner that one 20 hour job.

In fact, if there is spare capacity on Katana, which there is most of the time, six 10 hours jobs will finish before a single 20 hour job will. Requesting more resources for your job decreases the places that the job can run

The most obvious example is going over the 12 hour limit which limits the number of compute nodes that your job can run on but it is worth . For example specifying the CPU in your job script restricts you to the nodes with that CPU. A job that requests 20Gb will run on a 128Gb node with a 100Gb job already running but a 30Gb job will not be able to.

Running your jobs interactively makes it hard to manage multiple concurrent jobs

If you are currently only running jobs interactively then you should move to batch jobs which allow you to submit more jobs which then start, run and finish automatically. If you have multiple batch jobs that are almost identical then you should consider using array jobs

If your batch jobs are the same except for a change in file name or another variable then you should have a look at using array jobs.

2.4 GitHub

Version control systems like GitHub record changes to files over time, allowing you to go back to older versions, and create new branches for experimentation.

Version control is most useful for keeping track of programming code and documentation - large text based corpora that aren't suitable for storing in databases.

UNSW has an organisation in GitHub for researchers and staff, joining this organisation gives you access to the UNSW-only repositories. There is even a UNSW specific LaTeX thesis template!

- Restech-HPC - Example job scripts for Katana and NCI
- UNSW-Data-Archive - Scripts for uploading to and downloading from the UNSW Data Archive
- UNSW-eNotebook-LabArchives - UNSW eNotebook (LabArchives) widgets

We encourage researchers to use the Citation File Format when writing code.

2.4.1 How to sign up to the UNSW GitHub organisation

1. Open https://myapps.microsoft.com
2. Look to see if GitHub.com is in the list. If not then:

a. Click on "+ Add app" at the top of the page

b. Click on "GitHub.com" and press "Add"

c. Wait until it appears in your list. This could take up to 40 minutes and you may need to refresh your browser.

3. Click on "GitHub.com" icon

a. You will see "Single sign-on to UNSW Sydney", press the green "continue" button to validate your UNSW identity with GitHub.

b. At this point GitHub requires you to maintain a local account and password as well as verifying your UNSW identity. You can create a new GitHub identity, or sign in with an existing identity. In either case, because you've navigated from myapps and validated your UNSW identity, you will be connected to the UNSW GitHub organisation.

2.5 Using OnDemand

There are a number of GUI based software packages that are offered through a web interface. The requirements are a Katana account and for the user to be either on campus or connected to the UNSW VPN.

These can be accessed from https://ood.restech.unsw.edu.au

After logging in, you can:

- access your `/home/`, `/srv/scratch/` and any project files from the Files menu. While you can upload files through this interface, please don't upload anything larger than about 10MB.
- See your active jobs in a web based view of the traditional `qstat` command
- start a new interactive app

CHAPTER

THREE

STORAGE

3.1 Storage Locations

The storage on Katana is split into several different types, each of which serves a different purpose.

Important: We have just said *each of which serves a different purpose*. Despite that, there will be overlap. And personal preference. In most cases it will be obvious where to put your information. If it isn't and you need help with your decision making, you can email the Research Data team for advice. They are friendly people.

3.1.1 Cluster Home Drive

Location

```
/home/z1234567
```

Also known as

```
$HOME
```

Attributes

- Backed up
- 10Gb limit
- By default only user has access
- Able to be used everywhere on Katana

Best used for

Source code and programs

3.1.2 Global Scratch

Location

```
/srv/scratch/z1234567
```

Also known as

```
/srv/scratch/$USER
```

Attributes

- Not available to users in STUDENT groups such as ESTUDENT.
- Not backed up
- Generally 16Tb shared between multiple users
- By default only user has access
- Able to be used everywhere on Katana

Best used for

Data files

3.1.3 Shared Scratch

Only available to groups of users that have requested it - primarily for teams that share data sets or results.

Location

```
/srv/scratch/name
```

Attributes

- Not backed up
- Spaced based on group requirements
- All users in the group have access
- Able to be used everywhere on Katana

Best used for

Shared data files and copies of programs

3.1.4 Local Scratch

Found on the node on which your job is running.

Location

The location is created by the job scheduler as part of initialising the running of the job.

Also known as

```
$TMPDIR
```

Attributes

- Not backed up
- Only exists whilst job is running
- 200Gb shared between node users
- Storage located on compute node so good for compute

Best used for

Much fast completion of jobs that require large datasets to be near the CPU, calculations and temp files.

3.1.5 UNSW Research Storage

Location

```
/home/z1234567/sharename
```

Also known as

```
$HOME/sharename
```

Attributes

- Backed up
- Only available on Katana head node

Best used for

Shared and user data files.

3.2 Katana Data Mover

Also known as `kdm` or `kdm.restech.unsw.edu.au`

If you have data that you would like to copy to or within the Katana cluster, archive or even compress and decompress you should use the Katana Data Mover - also known as the KDM server - rather than using the head node. This section contains instructions on how to use KDM server.

If you are familiar with using Linux commands to copy or move files then you can do that directly by logging on to `kdm.restech.unsw.edu.au` via `ssh` in the same way that you would log in to Katana and then use the `cp`, `mv` and `rsync` commands that you would normally use under Linux.

If you are not familiar with using the Linux command line for moving or copying files then the easiest way to move files around is to use client software such as FileZilla. Once you have connected to `kdm.restech.unsw.edu.au` using your zID and zPass you should see a remote view which corresponds to the files sitting on Katana. You can then use the FileZilla interface to move files and folders around.

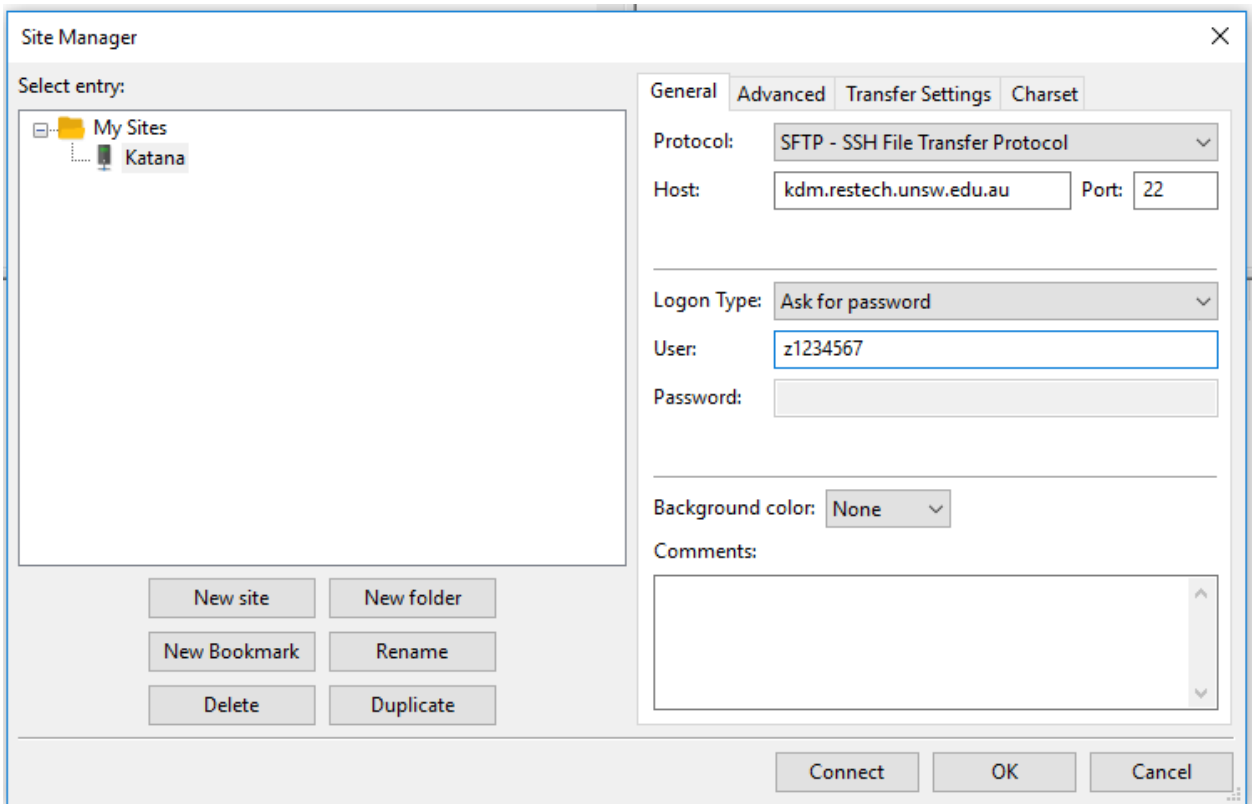
Note: We require people to "move data" through the data mover. We have hundreds of users, most of whom have data ranging from very large to impossibly large. This is why we have the `kdm`. If you are transferring a couple of small text files - job scripts for instance - you can copy directly to the Katana. But we would ask you to keep it to a minimum, and nothing bigger than 2-3 MB.

3.2.1 Copying Files To and From a Cluster

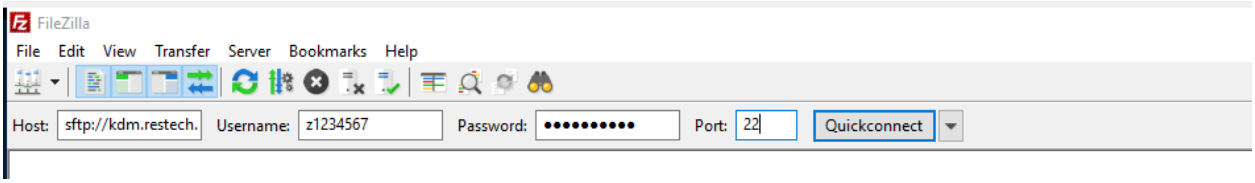
The method of transferring files to and from clusters depends on your local machine. If you are a Linux user then you should use rsync and if you are a Windows user then you should download and install WinSCP or FileZilla

Filezilla

Once you have installed Filezilla you can go into the site manager and create a new site in the site manager using the settings below.



You can also use the Quick Connect bar as shown here:



From my computer to Katana Home

To copy the directory `/home/1234567/my-directory` from your local computer to Katana scratch. The trailing `:` is important!

```
me@localhost:~$ rsync -avh /path/to/my-directory z1234567@kdm.restech.unsw.edu.au:
```

From my computer to Katana Scratch

```
me@localhost:~$ rsync -avh /path/to/my-directory z1234567@kdm.restech.unsw.edu.au:/
↪srv/scratch/z1234567
```

From Katana to my computer

First, you need to make sure the data is in either your Home directory or your scratch

If the data is in `/home/z1234567/my-remote-results` and you want it in your home directory:

```
me@localhost:~$ rsync -avh z1234567@kdm.restech.unsw.edu.au:my-remote-results /home/
↪me/
```

If the data is in `/srv/scratch/my-remote-results` and you want it in your home directory:

```
me@localhost:~$ rsync -avh z1234567@kdm.restech.unsw.edu.au:/srv/scratch/my-remote-
↪results /home/me
```

Note: *TMUX* is available if your data is large and the rsync might take a long time.

3.3 How to use the UNSW Data Archive

The UNSW Data Archive is the primary research storage facility provided by UNSW. The Data Archive gives UNSW researchers a free, safe and secure storage service to store and access research data well beyond the life of the project that collected that data.

To help researchers make use of this system the Katana Data Mover has a script that you can use to copy files from Katana into a project on the Data Archive system.

Note: To use this script you must have access to the UNSW Data Archive which requires setting up a Research Data Management Plan.

The best documentation on how to use the Data Archive is found on their website:

- using the web application
- using SFTP
- using the Command Line

To see what versions of the Data Archive script are available log on to `kdm.science.unsw.edu.au` and type

```
module avail unswdataarchive
```

Use the help command for usage

```
module help unswdataarchive/2020-03-19
```

To generate a token send an email to the IT Service Centre asking for a Data Archive token to be generated. A service desk request for an authentication token to be generated needs to indicate a Data Archive namespace (/UNSW_RDS/Dxxx or /UNSW_RDS/Hxxx) as a scope for the token.

Warning: This advice has poor results. The help file is too long for most screen sizes and there's no pagination in modules version < 4. Last line should include a location that the researcher can read directly (using less)

3.3.1 Initial Setup

To use the Data Archive you need to set up a configuration file. Here's how to create the generic config in the directory you are in:

```
[z1234567@kdm ~]$ module add unswdataarchive/2020-03-19
[z1234567@kdm ~]$ get-config-file
```

To generate a token send an email to the IT Service Centre asking for a Data Archive token to be generated. A service desk request for an authentication token to be generated needs to indicate a Data Archive namespace (/UNSW_RDS/Dxxx or /UNSW_RDS/Hxxx) as a scope for the token

Then edit the configuration file `config.cfg` and to change the line that looks like `token=`

If you haven't generated a token you can also upload content using your zID and zPass by adding the following line to the file `config.cfg` and you will be asked for your zPass when you start the upload.

```
user=z1234567
```

3.3.2 Starting a data transfer

To get data **into** the archive, we use `upload.sh`

```
upload.sh /path/to/your/local/directory /UNSW_RDS/D0000000/your/collection/name
```

To get data **from** the archive, we use `download.sh`

```
download.sh /UNSW_RDS/D0000000/your/collection/name /path/to/your/local/directory
```

CHAPTER

FOUR

SOFTWARE

This section starts with Environment Modules to help you get started using the software we have installed on Katana.

Each section after that explains tips and tricks for using that software on Katana and a link to an example batch script in our Github examples.

4.1 Environment Modules

Environment Modules offer a simple means of customising your environment to access the required versions of installed software and this section provides information on how they are used on Katana.

When we use modules, we are changing our "environment", hence the name.

4.1.1 How do I discover what software is available?

```
[z1234567@katana ~]$ module avail

--------------------------- /share/apps/modules/intel ----------------------------
intel/11.1.080(default)  intel/12.1.7.367  intel/13.0.1.117  intel/13.1.0.146

---------------------------- /share/apps/modules/pgi -----------------------------
pgi/13.7

--------------------------- /share/apps/modules/matlab ---------------------------
matlab/2007b          matlab/2010b          matlab/2012a(default)
matlab/2008b          matlab/2011a          matlab/2012b
matlab/2009b          matlab/2011b          matlab/2013a
```

4.1.2 What if the software that I want is not on the list?

If you require software installed on the cluster, email the IT Service Centre detailing the software that you would like installed and that you would like to have it installed on Katana. Please include links and desired version numbers.

4.1.3 How do I add a particular version of software to my environment?

```
[z1234567@katana1 ~]$ module add matlab/2018b
```

or

```
[z1234567@katana1 ~]$ module load matlab/2018b
```

4.1.4 How do I remove a particular version of software from my environment?

```
[z1234567@katana1 ~]$ module rm matlab/2018b
```

or

```
[z1234567@katana1 ~]$ module unload matlab/2018b
```

4.1.5 How do I remove all modules from my environment?

```
[z1234567@katana1 ~]$ module purge
```

4.1.6 Which versions of software am I currently using?

```
[z1234567@katana1 ~]$ module list
Currently Loaded Modulefiles:
 1) intel/18.0.1.163   2) matlab/2018b
```

4.1.7 How do I find out more about a particular piece of software?

You can find out more about a piece of software by using the module help command. For example:

```
[z1234567@katana1 ~]$ module help mrbayes

----------- Module Specific Help for 'mrbayes/3.2.2' --------------

MrBayes 3.2.2 is installed in /apps/mrbayes/3.2.2

This module was complied against beagle/2.1.2 and openmpi/1.6.4 with MPI support.

More information about the commands made available by this module is available
at http://mrbayes.sourceforge.net
```

4.1.8 How do I switch between particular versions of software?

```
[z1234567@katana1 ~]$ module switch matlab/2018b matlab/2017b
```

4.1.9 How can I find out what paths and other environment variables a module uses?

```
[z1234567@katana1 ~]$ module show mothur/1.42.3
-------------------------------------------------------------------
/apps/modules/bio/mothur/1.42.3:

module-whatis     Mothur 1.42.3
conflict     mothur
setenv         MOTHUR_ROOT /apps/mothur/1.42.3
prepend-path     PATH /apps/mothur/1.42.3/bin
setenv         LAST_MODULE_TYPE bio
setenv         LAST_MODULE_NAME mothur/1.42.3
setenv         LAST_MODULE_VERSION 1.42.3
-------------------------------------------------------------------
```

4.1.10 Why does the cluster forget my choice of modules?

Environment modules only affect the particular session in which they are loaded. Loading a module in one SSH session will not affect any other SSH session or even any jobs submitted from that session. Modules must be loaded in every session where they will be used.

4.1.11 How can I invoke my module commands automatically?

The best way of doing this is to add your Module commands to your job scripts. This approach is useful for preserving the required environment for each job. For example:

```
#!/bin/bash

#PBS -l nodes=1:ppn=1
#PBS -l vmem=4gb
#PBS -j oe

module purge
module add intel/18.0.1.163

cd ${PBS_O_WORKDIR}

./myprog
```

Perl, Python and R all have their own library/module systems - CPAN, PyPI and CRAN. If a library or module you want from one of these sources isn't installed in the module, please email us at IT Service Desk

4.2 Biosciences

Bioconductor, BioPerl, BioPython, Blast+, Mothur are all installed.

4.3 Java

Java is installed as part of the Operating System but we would strongly recommend against using that version - we cannot guarantee scientific reproducibility with that version. Please use the java modules.

Each Java module sets

```
_JAVA_TOOL_OPTIONS -Xmx1g
```

This sets the heap memory to 1GB. If you need more, set the environment variable `_JAVA_OPTIONS` which overrides `_JAVA_TOOL_OPTIONS`

```
export _JAVA_OPTIONS="-Xmx5g"
```

4.4 Intel Compilers and Software Libraries

Research Technology Services has a licence for Intel Compiler Collection which can be accessed by loading a module and contains 3 groups of software, namely compilers, libraries and a debugger. This software has been optimised by Intel to take advantage of the specific capabilities of the different intel CPUs installed in the Intel based clusters.

- **Compilers**
  - Intel C Compiler (icc)
  - Intel C++ Compiler (icpc)
  - Intel Fortran Compiler (ifort)
- **Libraries**
  - Intel Math Kernel Library (MKL)
  - Intel Threading Building Blocks (TBB)
  - Intel Integrated Performance Primitives (IPP)
- **Debugger**
  - Intel Debugger (idbc)

4.5 Operating Systems

Katana nodes are running either RedHat 7.8 (management plane) or CentOS 7.8 (compute nodes).

Research software is installed in modules so that they can be loaded or unloaded as necessary. This way we can offer and run multiple versions of each package at the same time.

4.6 Perl

The default version of Perl on Katana is 5.16.3 which is provided by CentOS 7 and can be found at `/usr/bin/perl`.

This is an older version of Perl. We have Perl 5.28.0 installed as a module.

It is common for perl scripts to begin with

```
#!/usr/bin/perl
```

If you are using the Perl module, you will need to change the first line to

```
#!/usr/bin/env perl
```

4.7 Python

It is common for python scripts to begin with

```
#!/usr/bin/python
```

If you are using a Python module, you will need to change the first line to

```
#!/usr/bin/env python
```

or more the likely

```
#!/usr/bin/env python3
```

4.7.1 Conda and Anaconda

We get a lot of questions about installing Conda and Anaconda. Unfortunately neither are designed to be installed in multi-user environments.

You are able to install them into your home directory and we encourage you to do so.

Alternatively, many packages will give you an option for a `pip install` - if this is an option, we recommend you use `python virtual environments`.

4.7.2 Virtual Environments

We encourage researchers to utilise the power of *Python Virtual Environments* if they are developing their own software or want to use packages that aren't installed.

4.8 Python Virtual Environments

4.8.1 or how to run a python library until the admins install it (or what to do if they wont install it)

Background

Sometimes you will need to use a particular version of Python, or you will need to use a set of Python libraries that aren't available in the provided installation.

In these cases, we can use what's known as a *virtual environment* or *venv*. A venv gives us a static version of Python in our home directory in which we can install any packages we like.

In this tutorial we will see how to set up a venv and explain what's happening under the hood. Then we will show how you can use one in your development process. As a note, in this walk through we will refer to packages and libraries as "packages". When we use these terms, we are referring to a collection of files, written in Python, usually with some versions and potentially some requirements of their own.

Setting up the default environment

According to the Python documentation, we will run something like this `python3 -m venv /path/to/new/virtual/environment`. This is not a directory that we need to see or need to spend time in actively, so it's ok to make it hidden. This is an important distinction - the *venv* should not be where you are doing your development. It's meant to be flexible - as soon as you fill it with development code, it's no longer flexible. Also, you want your development code backed up or in a repository - it is unnecessary bloat to add the Python software to that backup or repository.

We will make a directory in which we can keep many venvs. What I found was that once I started using venvs, it didn't make any sense to do Python development without them.

In Linux we can make a directory or file invisible by naming it with a leading dot:

```
[z1234567@katana2 ~]$ mkdir /home/z1234567/.venvs/
```

Setting up the virtual environment - creation and activation

I'll be using the latest version of Python available to me. Since this is one of the modules that we offer, I can use it with the understanding that it will be there indefinitely.

```
[z1234567@katana2 ~]$ module load python/3.7.4
[z1234567@katana2 ~]$ which python3
/apps/python/3.7.4/bin/python3
z1234567@katana2 ~]$ python3 -m venv /home/z1234567/.venvs/venv-tutorial-1
```

That's it, we are done. If you want to take a look under the hood, see what's in my virtualenv?

Next, we need to **activate** our venv. This makes our virtualenv our current environment. To activate, we execute `source /path/to/venv/bin/activate`. Note that after activation, the prompt changes to make it clear you are now in a venv. You can see the change in which versions of `python3` and `pip3` are available before and after activation:

Before we activate our environment

```
[z1234567@katana2 ~]$ which python3; which pip3
/apps/python/3.7.4/bin/python3
/apps/python/3.7.4/bin/pip3
```

Activation

```
[z1234567@katana2 ~]$ source ~/.venvs/venv-tutorial-1/bin/activate
```

After activation, our python binaries are not the defaults, but the versions in our *venv*

```
(venv-tutorial-1) [z1234567@katana2 ~]$ which python3; which pip3
~/.venvs/venv-tutorial-1/bin/python3
~/.venvs/venv-tutorial-1/bin/pip3
```

pip3 - the Python package manager ("the *Package Installer for Python*")

Using pip3 we can see whats installed and install new packages. You will often see packages give installation advice for pip (Conda is another popular system).

Now that we are using the venv, we can list what's in the venv, and then install a new package:

```
(venv-tutorial-1) [z1234567@katana2 ~]$ pip3 list
Package    Version
---------- -------
pip        19.0.3
setuptools 40.8.0
You are using pip version 19.0.3, however version 20.0.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

At this point - before any work is done, and while using your venv - it's a great time to perform that update.

```
(venv-tutorial-1) [z1234567@katana2 ~]$ pip install --upgrade pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/54/0c/
↪d01aa759fdc501a58f431eb594a17495f15b88da142ce14b5845662c13f3/pip-20.0.2-py2.py3-
↪none-any.whl (1.4MB)
    100% |████████████████████████████████| 1.4MB 1.5MB/s
Installing collected packages: pip
  Found existing installation: pip 19.0.3
    Uninstalling pip\-19.0.3:
    Successfully uninstalled pip\-19.0.3
Successfully installed pip-20.0.2
(venv-tutorial-1) [z1234567@katana2 ~]$ pip install --upgrade setuptools
Collecting setuptools
  Downloading setuptools-46.1.1-py3-none-any.whl (582 kB)
     |████████████████████████████████| 582 kB 13.5 MB/s
Installing collected packages: setuptools
  Attempting uninstall: setuptools
    Found existing installation: setuptools 40.8.0
          Uninstalling setuptools\-40.8.0:
        Successfully uninstalled setuptools\-40.8.0
Successfully installed setuptools-46.1.1
(venv-tutorial-1) [z1234567@katana2 w~]$ pip3 list
Package    Version
---------- -------
pip        20.0.2
setuptools 46.1.1
```

Installing software

And then package installation is as easy as using `pip install ...`:

```
(venv-tutorial-1) [z1234567@katana2 ~]$ pip install numpy
Collecting numpy
  Downloading numpy-1.18.2-cp37-cp37m-manylinux1*x86*64.whl (20.2 MB)
     |████████████████████████████████| 20.2 MB 38 kB/s
Installing collected packages: numpy
Successfully installed numpy-1.18.2
(venv-tutorial-1) [z1234567@katana2 ~]$ pip list
Package    Version
---------- -------
numpy      1.18.2
pip        20.0.2
setuptools 46.1.1
```

Exiting the venv, and coming around again

To leave a venv, you use the `deactivate` command like this:

```
(venv-tutorial-1) [z1234567@katana2 ~]$ deactivate
[z1234567@katana2 ~]$
```

Notice how the prompt returned to the way it was? Let's create a new venv:

```
[z1234567@katana2 ~]$ python3 -m venv /home/z1234567/.venvs/scipy-example
[z1234567@katana2 ~]$ ls -l ~/.venvs/
total 0
drwx------. 5 z1234567 unsw 69 Mar 23 15:07 scipy-example
drwx------. 5 z1234567 unsw 69 Mar 23 11:45 venv-tutorial-1
[z1234567@katana2 ~]$ source ~/.venvs/scipy-example/bin/activate
(scipy-example) [z1234567@katana2 src]$
(scipy-example) [z1234567@katana2 src]$ pip list
Package    Version
---------- -------
pip        19.0.3
setuptools 40.8.0
You are using pip version 19.0.3, however version 20.0.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

When we install SciPy, it automatically knows to install NumPy, a dependency:

```
(scipy-example) [z1234567@katana2 ~]$ pip install scipy
Collecting scipy
  Downloading scipy-1.4.1-cp37-cp37m-manylinux1*x86*64.whl (26.1 MB)
     |████████████████████████████████| 26.1 MB 95 kB/s
Collecting numpy>=1.13.3
  Using cached numpy-1.18.2-cp37-cp37m-manylinux1*x86*64.whl (20.2 MB)
Installing collected packages: numpy, scipy
Successfully installed numpy-1.18.2 scipy-1.4.1
(scipy-example) [z1234567@katana2 ~]$ pip list
Package    Version
---------- -------
numpy      1.18.2
pip        20.0.2
```

(continues on next page)

(continued from previous page)

```
scipy      1.4.1
setuptools 46.1.1
```

If you want to install an older version, it's relatively easy

```
(old-scipy-example) [z1234567@katana2 ~]$ pip install scipy==1.2.3
Collecting scipy==1.2.3
  Downloading https://files.pythonhosted.org/packages/96/e7/
↪e06976ab209ef44f0b3dc638b686338f68b8a2158a1b2c9036ac8677158a/scipy-1.2.3-cp37-cp37m-
↪manylinux1_x86_64.whl (24.8MB)
    100% |████████████████████████████████| 24.8MB 239kB/s
Collecting numpy>=1.8.2 (from scipy==1.2.3)
  Using cached https://files.pythonhosted.org/packages/b7/ce/
↪d0b92f0283faa4da76ea82587ff9da70104e81f59ba14f76c87e4196254e/numpy-1.18.2-cp37-
↪cp37m-manylinux1_x86_64.whl
Installing collected packages: numpy, scipy
Successfully installed numpy-1.18.2 scipy-1.2.3
(old-scipy-example) [z1234567@katana2 src]$ pip list
Package    Version
---------- -------
numpy      1.18.2
pip        20.0.2
scipy      1.2.3
setuptools 46.1.1
```

That's a quick introduction to how you can install Python packages locally.

Special Cases

Say for instance you want to use software X in a Jupyter Notebook. X is already

installed on Katana.

In that case, your workflow would be:

- load the module in question
- create the Virtual Environment with the flag `--system-site-packages`
- install software in question with an understanding that you might not be able to get the latest release

For example, using the Katana TensorFlow installation and a desire for Jupyter:

```
[z1234567@katana1 ~]$ module load tensorflow/1.14gpu
[z1234567@katana1 ~]$ python3 -m venv /home/z1234567/.venvs/tf --system-site-packages
[z1234567@katana1 ~]$ source ~/.venvs/tf/bin/activate
(tf) [z1234567@katana2 ~]$ pip install jupyter
```

This will throw errors because there are a collection of packages missing in relation to the latest Jupyter. They shouldn't affect your ability to run *Jupyter Notebooks* with tensorflow.

4.9 Virtual Environments from the inside

We've built a venv in our `~/.venvs` directory. Let's take a look inside. This presumes you have used the command

```
[z1234567@katana2 src]$ python3 -m venv /home/z1234567/.venvs/venv-tutorial-1
```

to set up your virtualenv.

Here is a quick overview of the basics.

We can see there is a directory in `~/.venvs` that has the same name as the virtualenv we created.

```
[z1234567@katana2 ~]$ ls -l ~/.venvs/
total 0
drwx------.  5 z1234567 unsw   69 Mar 23 11:45 venv-tutorial-1
```

Inside that directory we can see some more directories. The two important directories here are `bin` and `lib`.

```
[z1234567@katana2 ~]$ ls -l ~/.venvs/venv-tutorial-1/
total 8
drwx------. 2 z1234567 unsw 4096 Mar 23 11:45 bin
drwx------. 2 z1234567 unsw    6 Mar 23 11:45 include
drwx------. 3 z1234567 unsw   22 Mar 23 11:45 lib
lrwxrwxrwx. 1 z1234567 unsw    3 Mar 23 11:45 lib64 -> lib
-rw-------. 1 z1234567 unsw   83 Mar 23 11:45 pyvenv.cfg
```

In `bin` you will see executables. The main one of note is `activate`.

```
[z1234567@katana2 ~]$ ls -l ~/.venvs/venv-tutorial-1/bin/
total 36
drwx------. 2 z1234567 unsw 4096 Mar 23 11:45 .
drwx------. 5 z1234567 unsw   69 Mar 23 11:45 ..
-rw-r--r--. 1 z1234567 unsw 2235 Mar 23 11:45 activate
-rw-r--r--. 1 z1234567 unsw 1291 Mar 23 11:45 activate.csh
-rw-r--r--. 1 z1234567 unsw 2443 Mar 23 11:45 activate.fish
-rwxr-xr-x. 1 z1234567 unsw  266 Mar 23 11:45 easy_install
-rwxr-xr-x. 1 z1234567 unsw  266 Mar 23 11:45 easy_install-3.7
-rwxr-xr-x. 1 z1234567 unsw  248 Mar 23 11:45 pip
-rwxr-xr-x. 1 z1234567 unsw  248 Mar 23 11:45 pip3
-rwxr-xr-x. 1 z1234567 unsw  248 Mar 23 11:45 pip3.7
lrwxrwxrwx. 1 z1234567 unsw    7 Mar 23 11:45 python -> python3
lrwxrwxrwx. 1 z1234567 unsw   30 Mar 23 11:45 python3 -> /apps/python/3.7.4/bin/
↪python3
```

In `lib` we need to traverse a few more directories, but eventually we will see where the packages are installed. As you can see, `pip` and `setuptools` are already installed. These are the default:

```
[z1234567@katana2 ~]$ ls -l ~/.venvs/venv-tutorial-1/lib/python3.7/site-packages/
total 16
-rw-------. 1 z1234567 unsw  126 Mar 23 11:45 easy_install.py
drwx------. 5 z1234567 unsw   90 Mar 23 11:45 pip
drwx------. 2 z1234567 unsw 4096 Mar 23 11:45 pip-19.0.3.dist-info
drwx------. 5 z1234567 unsw   89 Mar 23 11:45 pkg_resources
drwx------. 2 z1234567 unsw   40 Mar 23 11:45 **pycache**
drwx------. 6 z1234567 unsw 4096 Mar 23 11:45 setuptools
drwx------. 2 z1234567 unsw 4096 Mar 23 11:45 setuptools-40.8.0.dist-info
```

4.10 Jupyter Notebooks

4.10.1 or, the "'it's slightly frustrating, but can be done using the tools" way of running Juypter on Katana'

Background

Due to the nature of the Katana system, it's not possible to serve content to the internet from Katana. Juypter notebooks are, essentially, python engines served via http. As a result, getting a notebook working requires a little more work than usual, but should get adequate results. This documentation is minimal, because it points to two other sets of documenation that have more information. WARNING: while it's possible to run Jupyter Notebooks on Katana, it's not an ideal system to run Jupyter on (at the moment). As a result, you may find that you experience is degraded - it's highly dependant on network latency. We would recommend against it and will not be able to provide usability or performance support. We can continue to provide you with Python support, and Jupyter support.

We need to do a two main steps to prepare, and then each connection will require two steps to connect to your notebook.

Preparation

These two steps are independant of each other - there is no need to do them in order.

1. You will need to be able to connect to Katana via Remote Desktop. There are Instructions on how to do this in *Graphical sessions*.
2. a. Load one of the python modules *or*
2. b. You will need to create a Python Virtual Environment (*Python Virtual Environments*), activate it, and install Jupyter via the command `pip install jupyter`

Reconnection

Everytime you would like to use your Jupyter notebook, you will need to:

- Remote Desktop into Katana
- open a terminal
- start an interactive session - `qsub -I` - and wait for it to start
- in the interactive session, load any relevant modules (if applicable)
- activate your virtual environment
- grab the IP address of the node you are on: `hostname -I | awk '{print $1}'`
- launch Jupyter Notebook like this: `jupyter-notebook --ip=$(hostname -I | awk '{print $1}') --no-browser`
- copy one of the first two options (the `file:///` or `http://10.197.34.xxx:8888`) into firefox in the remote desktop session

Caveat: if you get an error about a port being busy, launch Jupyter with this command, susbstituting XXXX with a large number:

```
jupyter-notebook --port=XXXX
```

IMPORTANT FINAL POINT

We will forcibly kill Jupyter notebooks we find running on katana1 or katana2 (collectively known as "the head nodes") without warning or explanation beyond this paragraph. All jupyter notebooks must be started in an *Interactive Jobs* on Katana.

4.11 R and RStudio

R is installed as a module. Each version has a number of libraries installed within it.

If you would like a new library installed, please email the IT Service Centre with Katana R Module installation in the subject line.

It is possible to use RStudio on Katana.

4.12 SAS

The 64-bit version of SAS is available as a module.

By default SAS will store temporary files in `/tmp` which can easily fill up leaving the node offline. In order to avoid this we have set the default to `$TMPDIR` to save temporary files in `/var/tmp` on the Katana head node and local scratch on compute nodes. If you wish to save temporary files to a different location you can do that by using the `-work` flag with your SAS command or adding this line to your `sasv9.cfg` file:

```
-work /my/directory
```

4.13 Stata

Stata is availt as a module.

When using Stata in a pbs batch script, the syntax is

```
stata -b do StataClusterWorkshop.do
```

If you wish to load or install additional Stata modules or commands you should use findit command on your local computer to find the command that you are looking for. Then create a directory called `myadofiles` in your home directory and copy the .ado (and possibly the .hlp) file into that directory. Now that the command is there it just remains to tell Stata to look in that directory which can be done by using the following Stata command.

```
sysdir set PERSONAL $HOME/myadofiles
```

4.14 TMUX

tmux is available on Katana. It can be used to manage multiple sessions, including keeping them alive despite a terminal losing connectivity or being shutdown.

When you login to Katana using the terminal, it is a "live" session - if you close the terminal, the session will also close. If you shut your laptop or turn off the network, you will also kill the session.

This is fine, except when you have a long running program - say you are downloading a large data set - and you need to leave campus.

In these situations, you can use `tmux` to create an **interruptible session**. `tmux` has other powerful features - multiple sessions and split screens being two of many features.

To start tmux, type `tmux` at the terminal. A new session will start and there will be a green information band at the bottom of the screen.

Anything you start in this session will keep running even if you are disconnected from that session regardless of the reason. Except when the server itself is rebooted. That - and I hope this is obvious - will kill all sessions.

If you do get detached, you can re-attach by logging into the same server and using the command `tmux a`

tmux is similar to another program called `screen` which is also available.

4.15 Compressing Large Directories

If you want to compress large directories or directories with a large number of files, we recommend a simple tool called tgzme developed by one of our researchers.

It's essentially a smart shell script wrapped around the one line command:

```
tar -c $DIRECTORY | pigz > $DIRECTORY.tar.gz
```

We thank Dr. Edwards for his contribution.

CHAPTER

FIVE

REFERENCE DATA

We keep a number of reference data sets available on Katana at `/data/` so that we don't accidentally - for instance - end up with 150 copies of the Human Genome in user's home directories.

As these are reference data, they don't change often and we can update them as necessary.

Table 1: Katana Data Sets

Directory	Description	Update Schedule	URL
annovar	Reference datasets that come with software installation.	Installed when software is installed.	annovar.openbioinformatics.o
antismash	Reference files and commands for antismash version 4.2.0	Version specific database installed when software is installed	antismash.secondarymetaboli
blastn	NCBI nr, nt, refseq_genomic and refseq datasets	Updated on release 6 times a year	www.ncbi.nlm.nih.gov/refseq
blastv5	Version 5 of NCBI nr, nt, refseq_genomic and refseq datasets.	Updated on release 6 times a year.	www.ncbi.nlm.nih.gov/refseq
diamond	Diamond reference databases for versions 0.8.38, 0.9.10, 0.9.22 and 0.9.24. Database format periodically changes.	Updated when NCBI nr databases are updated.	ab.inf.uni-tuebingen.de/software/diamo
gtdbtk		Version specific database installed when software is installed.	
hapcol	Reference datasets that come with software installation.	Installed when software is installed.	hapcol.algolab.eu
hg19	Human reference genome hg19 (GRCh37).	Fixed reference. Never updated.	www.ncbi.nlm.nih.gov/grc
interproscan	Reference datasets for InterProScan versions 5.20-59.0 and 5.35-74.0	Version specific database installed when software is installed.	www.ebi.ac.uk/interpro
itasser	Reference datasets for I-TASSER plus link to current nr database.	Version specific databases installed when software is installed plus link to nr database (see blast above).	zhanglab.ccmb.med.umich.ed TASSER
kaiju	Reference databases for all versions of Kaiju. Same databases for all versions.	Databases installed when software is installed.	kaiju.binf.ku.dk
matam	Reference databases for all MATAM versions.	Version specific database installed when software is installed.	github.com/bonsai-team/matam
megan	Reference databases for all MEGAN versions.	Version specific database installed when software is installed.	ab.inf.uni-tuebingen.de/software/megan
repeatmasker	Reference datasets for RepeatMasker version 4.0.7	Version specific database installed when software is installed.	www.repeatmasker.org
trinotate	Reference databases for all versions of Kaiju. Same databases for all versions.	Databases installed when software is installed.	trinotate.github.io

CHAPTER

SIX

FREQUENTLY ASKED QUESTIONS

6.1 General FAQ

6.1.1 Where is the best place to store my code?

The best place to store source code is to use version control and store it in a repository. This means that you will be able to keep every version of your code and revert to an earlier version if you require. UNSW has a central github account, but we encourage you to create your own.

6.1.2 I just got some money from a grant. What can I spend it on?

There are a number of different options for using research funding to improve your ability to run computationally intensive programs. The best starting point is to *Contact the Research Technology Services team* to figure out the different options.

6.1.3 Can I access Katana from outside UNSW?

Yes, if you have an account then you can connect to Katana from both inside and outside UNSW. Some services - like remote desktops - will not be as responsive as inside the UNSW network.

6.2 Scheduler FAQ

6.2.1 Does Katana run a 32 bit or a 64 bit operating system?

Katana *Compute Nodes* run a 64 bit version of the CentOS distribution of Linux. Currently version 7.8. The *Head Node* runs RedHat 7.8.

6.2.2 How much memory is available per core and/or per node?

The amount of memory available varies across the cluster. To determine how much memory each node has available use the 'pbsnodes' command. Roughly, you can safely use 4GB per core requested. You can request more memory but it may delay time spent in the queue.

6.2.3 How much memory can I use on the login node for compiling software?

The login nodes have a total of 24GB of memory each. Each individual user is limited to 4GB and should only be used to compile software. If you need more, do it in an *Interactive Job*.

6.2.4 Why isn't my job making it onto a node even though it says that some nodes are free?

There are three main reasons you will see this behavior. The first of them is specific to Katana and the other two apply to any cluster.

Firstly, the compute nodes in Katana belong to various schools and research groups across UNSW. Any job with an expected run-time longer than 12 hours can only run on a compute node that is somehow associated with the owner of the job. For example, if you are in the CCRC you are entitled to run 12+ hour jobs on the General nodes and the nodes jointly purchased by CCRC. However, you cannot run 12+ hour jobs on the nodes purchased by Astrobiology, Statistics, TARS, CEPAR or Physics. So you may see idle nodes, but you may not be entitled to run a 12+ hour job on them.

Secondly, the idle nodes may not have sufficient resources for your job. For example, if you have asked for 100GB memory but there are only 50GB free on the "idle node".

Thirdly, there may be distributed memory jobs ahead of your job in the queue which have reservations on the idle nodes, and they are just waiting for all of their requested resources to become available. In this case, your job can only use the reserved nodes if your job can finish before the nodes are required by the distributed memory job. For example, if a job has been waiting a week (yes, it happens) for `walltime=200,cpu=88,mem=600GB` (very long, two whole nodes), then those resources will need to be made available at some point. This is an excellent example of why breaking your jobs up into smaller parts is good HPC practice.

6.2.5 How many jobs can I submit at the one time?

Technically you can submit as many jobs as you wish. The queuing system run by the scheduler is designed to prevent a single user flooding the system - each job will reduce the priority of your next jobs. In this way the infrequent users get a responsive system without impacting the regular users too much.

Whilst there is not a technical limit to the number of jobs you can submit, submitting more that 2,000 jobs at the one time can place an unacceptable load on the job scheduler and your jobs may be deleted without warning. This is an editorial decision by management.

6.2.6 What is the maximum number of CPUs I can use in parallel?

As many as your account and queue will allow you. But there are trade-offs - if you ask for 150 CPUs (~5 full servers) you might be waiting more than a couple of months for your job to run.

If you are regularly wanting to run large parallel jobs (16+ cores per job) on Katana you should consider speaking to *Help and Support* so that they are aware of your jobs. They may be able to provide you additional assistance on resource usage for parallel jobs.

6.2.7 Why does my SSH connection periodically dsconnect?

With all networks there is a limit to how long a connection between two computers will stay open if no data is travelling between them. Look to set your ServerAliveInterval or Keep Alive interval to 60 in your secure shell software (putty, ssh).

6.2.8 Can I change the job script after it has been submitted?

Yes you increase the resource values for jobs that are still queued, but even then you are constrained by the limits of the particular queue that you are submitting to. Once it has been assigned to a node the intricacies of the scheduling policy means that it becomes impossible for anyone including the administrator to make any further changes

6.2.9 Where does Standard Output (STDOUT) go when a job is run?

By default Standard Output is redirected to storage on the node and then transferred when the job is completed. If you are generating data you should redirect `STDOUT` to a different location. The best location depends on the characteristics of your job but in general all `STDOUT` should be redirected to local scratch.

6.2.10 How do I figure out what the resource requirements of my job are?

The best way to determine the resource requirements of your job is to be generous with the resource requirements on the first run and then refine the requirements based on what the job actually used. If you put the following information in your job script you will receive an email when the job finishes which will include a summary of the resources used.

```
#PBS -M z1234567@unsw.edu.au
#PBS -m ae
```

6.2.11 Can I cause problems to other users if I request too many resources or make a mistake with my job script?

Yes, but it's extremely unlikely. We used to say no, but that's not strictly true. The reality is that if something breaks it's usually your job hitting the odd corner case we didn't account for. It doesn't happen often.

6.2.12 Will a job script from another cluster work on cluster X?

It depends on a number of factors including the sceduling software. Some aspects are fairly common across different clusters (e.g. walltime) others are not. You should look at the cluster specific information to see what queuing system is being used on that cluster and what commands you will need to change. You wont find a cluster that doesn't have knowledgable support that can help you migrate.

6.2.13 How can I see exactly what resources (I/O, CPU, memory and scratch) my job is currently using?

From *outside* the job, you can run `qstat -f <jobid>`.

If, for instance, you wanted to measure different steps of your process, then inside your jobscript you can put `qstat -f $PBS_JOBID`

For fine grain detail, you may need to get access to the worker node that the job is running on:

```
qstat -nru $USER
```

then you can see a list of your running jobs and where they are running. You can then use ssh to log on to the individual nodes and run `top` or `htop` to see the load on the node including memory usage for each of the processes on the node.

6.2.14 How do I request the installation or upgrade of a piece of software ?

If you wish to have a new piece of software installed or software that is already installed upgraded please send an email to the IT Service Centre from your UNSW email account with details of what software change you require and the cluster that you would like it changed on.

6.2.15 Why is my job stuck in the queue whilst other jobs run?

The queues are not set up to be first-in-first-out. In fact all of the queued jobs sit in one big pool of jobs that are ready to run. The scheduler assigns priorities to jobs in the pool and the job with the highest priority is the next one to run. The length of time spent waiting in the pool is just one of several factors that are used to determine priority.

For example, people who have used the cluster heavily over the last two weeks receive a negative contribution to their jobs' priority, whereas a light user will receive a positive contribution. You can see this in action with the diagnose -p and diagnose -f commands.

6.2.16 You mentioned waiting time as a factor, what else affects the job priority?

The following three factors combine to generate the job priority.

- How many resources (cpu and memory) have you and your group consumed in the last 14 days? Your personal consumption is weighted more highly than your group's consumption. Heavy recent usage contributes a negative priority. Light recent usage contributes a positive priority.
- How many resources does the job require? Always a positive contribution to priority, but increases linearly with the amount of cpu and memory requested, i.e. we like big jobs.
- How long has the job been waiting in the queue? Always a positive contribution to priority, but increases linearly with the amount of time your job has been waiting in the queue. Note that throttling policies will prevent some jobs from being considered for scheduling, in which case their clock does not start ticking until that throttling constraint is lifted.

6.2.17 What happens if my job uses more memory than I requested?

The job will be killed by the scheduler. You will get a message to that effect if you have any types of notification enabled (logs, emails).

6.2.18 What happens if my job is still running when it reaches the end of the time that I have requested?

When your job hits it's *Walltime* it is automatically terminated by the scheduler.

6.2.19 200 hours is not long enough! What can I do?

If you find that your jobs take longer than the maximum WALL time then there are several different options to change your code so that it fits inside the parameters.

- Can your job be split into several independent jobs?
- Can you export the results to a file which can then be used as input for the next time the job is run?

You may want to also look to see if there is anything that you can do to make your code run better like making better use of local scratch if your code is I/O intensive.

6.2.20 Do sub-jobs within an array job run in parallel, or do they queue up serially?

Submitting an array job with 100 sub-jobs is equivalent to submitting 100 individual jobs. So if sufficient resources are available then all 100 sub-jobs could run in parallel. Otherwise some sub-jobs will run and other sub-jobs must wait in the queue for resources to become available.

The '%' option in the array request offers the ability to self impose a limit on the number of concurrently running sub-jobs. Also, if you need to impose an order on when the jobs are run then the 'depend' attribute can help.

6.2.21 In a pbs file does the MEM requested refer to each node or the total memory on all nodes being used (if I am using more than 1 node?

MEM refers to the amount of memory per node.

6.3 Storage FAQ

6.3.1 What storage is available to me?

Katana provides three different storage areas, cluster home drives, local scratch and global scratch. The storage page has additional information on the differences and advantages of each of the different types of storage. You may also want to consider storing your code using a version control service like GitHub. This means that you will be able to keep every version of your code and revert to an earlier version if you require.

6.3.2 Which storage is fastest?

In order of performance the best storage to use is local scratch, global scratch and cluster home drive.

6.3.3 Is any of the cluster based storage backed up?

The only cluster based storage that gets backed up is the cluster home drives. All other storage including local and global scratch is not backed up.

6.3.4 How do I actually use local scratch?

The easiest way of making use of local scratch is to use scripts to copy files to the node at the start of your job and from the node when your job finishes. You should also use local scratch for your working directory and temporary files.

6.3.5 Why am I having trouble creating a symbolic link?

Not all filesystems support symbolic links. The most common examples are some Windows network shares. On Katana this includes Windows network shares such as hdrive. The target of the symbolic link can be within such a filesystem, but the link itself must be on a filesystem that supports symbolic links, e.g. the rest of your home directory or your scratch directory.

6.3.6 What storage is available on compute nodes?

As well as local scratch, global scratch and your cluster home drive are accessible on the compute nodes.

6.3.7 What is the best way to transfer a large amount of data onto a cluster?

Use `rsync` to copy data to the KDM server. More information is above.

6.3.8 Is there any way of connecting my own file storage to one of the clusters?

Whilst it is not possible to connect individual drives to any of the clusters, some units and research groups have purchased large capacity storage units which are co-located with the clusters. This storage is then available on the cluster nodes. For more information please contact the Research Technology Service Team by placing a request with the IT Service Centre.

6.3.9 Can I specify how much file storage I want on local scratch?

If you want to specify the minimum amount of space on the drive before your job will be assigned to a node then you can use the file option in your job script. Unfortunately setting up more complicated file requirements is currently problematic.

6.3.10 Can I run a program directly from scratch or my home drive after logging in to the cluster rather submitting a job?

As the file server does not have any computational resources you would be running the job from the head node on the cluster. If you need to enter information when running your job then you should start an interactive job.

6.4 Expanding Katana

Katana has significant potential for further expansion. It offers a simple and cost-effective way for research groups to invest in a powerful computing facility and take advantage of the economies that come with joining a system with existing infrastructure. A sophisticated job scheduler ensures that users always receive a fair share of the compute resources that is at least commensurate with their research group's investment in the cluster. For more information please contact us.

6.5 Acknowledging Katana

If you use Katana for calculations that result in a publication then you should add the following text to your work.

```
This research includes computations using the computational cluster Katana supported␣
↪by Research Technology Services at UNSW Sydney.
```

If you are using nodes that have been purchased using an external funding source you should also acknowledge the source of those funds.

For information about acknowledging ARC funding

Your School or Research Group may also have policies for compute nodes that they have purchased.

6.5.1 Facilities external to UNSW

If you are using facilities at Intersect or NCI in addition to Katana they may also require some form of acknowledgement.

CHAPTER

SEVEN

GLOSSARY

Active Job Active jobs are jobs that have been assigned to a compute node and are currently running. These can be seen by running `qstat` and looking for an A in the second last column. See *Show all jobs on the system*

Array Job If you want to run the same job multiple times with slight differences (filenames, data source, etc), then you can create an array job which will submit multiple jobs for you from the one job script.

Batch Job A batch job is a job on a cluster that runs without any further input once it has been submitted. Almost all jobs on the cluster are batch jobs. All jobs are either batch jobs or *Interactive Job*.

Blade Some of the compute nodes Katana are called blade servers which allow a higher density of servers in the same space. Each blade consists of multiple CPUs with 6 or more cores.

Cluster A computer cluster is a set of connected computers that work together so that, in many respects, they can be viewed as a single system. Using a cluster is referred to as High Performance Computing or HPC. Most will have a *Management Plane* and several *Compute Nodes*.

Compute Nodes The compute nodes are where the compute jobs run. Users submit jobs from the *Login Node* and the *Job Scheduler* on the *Head Node* will assign the job to one or more compute nodes.

CPU Core Each node in the cluster has one or more CPUs each of which has 6 or more cores. Each core is able to run one job at a time so a node with 12 cores could have 12 jobs running in parallel.

Data Transfer Node The Data Transfer Node, also known as the *Katana Data Mover* (KDM), is a server that is used for transferring files to, from, and within the cluster. Due to the nature of moving data around, it uses a significant amount of memory and network bandwidth. This server is used to take that load off the *Login Node*.

Environment Variable Environment variables are variables that are set in Linux to tell applications where to find programs and set program options. They will start with a $ symbol. For example, all users can reference `$TMPDIR` in their *Job Script* in order to use *Local Scratch*

Global Scratch Global scratch is a large data store for data that isn't backed up. It differs from local scratch in that it is available from every node including the *Head Node*. If you have data files or working directories this is where you should put them.

Head Node The head node of the *Cluster* is the computer that manages job and resource management. This is where the *Job Scheduler* and *Resource Manager* run. It is kept separate from the *Login Node* so that production doesn't stop if someone accidentally breaks the *Login Node*.

Held Jobs Held jobs are jobs that cannot currently run. They are put into that state by either the server or the system administrator. Jobs stay held until released by a systems administrator, at which point they become *Queued Jobs*. These can be seen by running `qstat` and looking for an H in the second last column. See *Show all jobs on the system*

Interactive Job An interactive job is a way of testing your program and data on a cluster without negatively impacting the *Login Node*. Once a request has been submitted and accepted for an interactive job, the user will no longer be on the relatively small login nodes, and will have access to the resources requested on the *Compute Nodes*. In other words, your terminal session will move from a small (virtual) computer you share with many people to a large

computer you share with very few people. All jobs are either a *Batch Job* or an interactive job. Instructions on using *Interactive Jobs*

Job Scheduler The job scheduler monitors the jobs currenty running on the cluster and assigns *Queued Jobs* to *Compute Nodes* based on recent cluster useage, job resource requirements and nodes available to the research group of the submitter. In summary the job scheduler determines when and where a job should run. The job scheduler that we use is called PBSPro.

Job Script A job script is a file containing all of the information needed to run a *Batch Job* including the resource requirements and the actual commands to run the job.

Local Scratch Local scratch refers to the storage available internally on each compute node. Of all the different scratch directories this storage has the best performance however you will need to move your data into local scratch as part of your job script. You can use local scratch with the *Environment Variable* `$TMPDIR`

Login Node The login nodes of the cluster is the computer that you log in to when you connect to the cluster. This node is used to compile software and submit jobs.

Module The module command is a means of providing access to different versions of software without risking version conflicts across multiple users.

Management Plane The Management Plane is the set of servers that sit above or adjacent to the *Compute Nodes*. These servers are used to manage the system, manage the storage, or manage the network. User's have access to the *Login Node* and *Data Transfer Node*. Other servers include the *Head Node*.

MPI Message Passing Infrastructure (MPI) is a technology for running a *Batch Job* on more than one *Compute Nodes*. Designed for situations where parts of the job can run on independent nodes with the results being transferred to other nodes for the next part of the job to be run.

Network Drive A network drive is a drive that is independant from the cluster.

Queue All submitted jobs are put into a queue. Each queue has a collection of resources available to it. As those resources become available, new jobs will be assigned to those resources. Job prioritisation is done by the scheduler and depends on a number of factors including length of wait time and total resource use by user over the previous month.

Queued Jobs Queued jobs are eligible to run but are waiting for a *Compute Nodes* that matches their requirements to become available. Which idle job will be assigned to a compute node next depends on the *Job Scheduler*. These can be seen by running `qstat` and looking for a Q in the second last column. See *Show all jobs on the system*

Resource Manager A resource manager works with the *Job Scheduler* to manage running jobs on a cluster. Amongst other tasks it receives and parses job submissions, starts jobs on *Compute Nodes*, monitors jobs, kills jobs, and manages how many *CPU Core* are available on each *Compute Nodes*

Scratch Space Scratch space is a non backed up storage area where users can store transient data. It should not be used for job code as it is not backed up.

Walltime In HPC, walltime is the amount of time that you will be allocated when your job runs. If your jobs runs longer than the walltime, it will be killed by the *Job Scheduler*. It is used by the scheduler for helping allocate resources onto servers. On **Katana** it is also used to determine which *Queue* your job will end up in. The shorter the walltime, the more opportunity your job has to run which in turn means that it will start sooner. In short - it's harder to find 100 hours of space than it is to find 12 hours of space.

CHAPTER

EIGHT

NEWS

21/04/2020 - Jupyter has been installed on Katana as part of versions 3.7.3, 3.7.4 and the new 3.8.2 python modules.

INDEX

A

Active Job, 53
Array Job, 53

B

Batch Job, 53
Blade, 53

C

Cluster, 53
Compute Nodes, 53
CPU Core, 53

D

Data Transfer Node, 53

E

Environment Variable, 53

G

Global Scratch, 53

H

Head Node, 53
Held Jobs, 53

I

Interactive Job, 53

J

Job Scheduler, 54
Job Script, 54

L

Local Scratch, 54
Login Node, 54

M

Management Plane, 54
Module, 54
MPI, 54

N

Network Drive, 54

Q

Queue, 54
Queued Jobs, 54

R

Resource Manager, 54

S

Scratch Space, 54

W

Walltime, 54