Submitting jobs to the partitions on Viking.

To run a job on one of Vikings compute node you need to tell the Slurm scheduler what you want to run, where the files are and how much resource you will need.

There are two different ways to do this.

Batch Jobs

These are non-interactive sessions where a number of tasks are batched together into a job script, which is then scheduled and executed by Slurm when resources are available.

- you write a job script containing the commands you want to execute on the cluster
- you request an allocation of resources (nodes, cpus, memory)
- the system grants you one, or more, compute nodes to execute your commands
- your job script is automatically run
- your script terminates and the system releases the resources

Interactive Sessions

These are similar to a normal remote login session, and are ideal for debugging and development, or for running interactive programs. The length of these sessions is limited compared to batch jobs however, so once your development is done, you should pack it up into a batch job and run it detached.

- you request an allocation of resources (cpus, memory)
- the system grants you a whole, or part, node to execute your commands
- you are logged into the node
- you run your commands interactively
- you exit and the system automatically releases the resources

Interactive Jobs

Exercise 1: Run a simple interactive job on Viking

In this exercise we will begin to run simple jobs on Viking.

The most basic type of job you can run is an interactive job on the command line. Type the following into the terminal

```
[abc123@login1(viking) ~]$ srun --ntasks=1 --time=00:30:00 --pty /bin/bash
```

What do you see?

```
[abc123@login1(viking) ~]$ srun --ntasks=1 --time=00:30:00 --pty /bin/bash
srun: job 6485884 queued and waiting for resources
srun: job 6485884 has been allocated resources
[abc123@node069 [viking] ~]$ pwd
/users/abc123
[abc123@node069 [viking] ~]$ echo "hello" exit
hello
[abc123@node069 [viking] ~]$ exit
exit
[abc123@login1(viking) ~]$
```

```
[abc123@login1(viking) ~]$ srun --ntasks=4 --time=00:30:00 --pty /bin/bash
```

```
[abc123@login1(viking) ~]$ srun --ntasks=4 --time=00:30:00 --pty /bin/bash
srun: job 6485885 queued and waiting for resources
srun: job 6485885 has been allocated resources
[abc123@node071 [viking] ~]$
```

You may find you have to wait before your job is running.

The above example creates a single task (one core) interactive session in the default partition "nodes", with a 30 minute time limit. The second example creates a four task (four cores) session.

To terminate the session, exit the shell by typing "exit" and pressing enter.

When there is resource available you will be placed directly onto a compute node, in the above cases node069 and node071. Here you can run your workload as normal.

These sorts of jobs can be useful but you will have to run each one separately with manual intervention. If you have many jobs to run the best method would be to use a slurm job script

Batch Jobs

Job scripts

If you have a number of jobs to run the best method is to write a **slurm job script**. It is also easier to share job scripts with other member of your research group.

You can download an example job script here (viking_job1.slurm) and copy them to Viking using the instructions on copying files to Viking. Or you can copy them once logged into Viking using the following commands.

```
[abc123@login1(viking) ~]$ cd scratch
[abc123@login1(viking) scratch]$ cp -r /mnt/lustre/groups/viking-examples/hpc_course/jobscripts .
[abc123@login1(viking) scratch]$ ls
jobscripts
```

This is what the job script looks like. At the start there are a number of instructions for Slurm which will help place you workload in the queue and decide where and when resources are available for you job to run.

The following lines need to be updated if you wish to use this job script

Your username here

#SBATCH --mail-user=abc123@york.ac.uk # Where to send mail

Your account code here

#SBATCH --account=PROJECTCODE # Project account

```
#!/bin/bash
#SBATCH --job-name=simple                # Job name
#SBATCH --mail-type=BEGIN,END,FAIL       # Mail events (NONE, BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=abc123@york.ac.uk    # Where to send mail
#SBATCH --ntasks=1                       # Run on a single CPU
#SBATCH --mem=1gb                        # Job memory request
#SBATCH --time=00:01:00                  # Time limit hrs:min:sec
#SBATCH --output=basic_job_%j.log        # Standard output and error log
#SBATCH --partition=nodes                                 # Job queue
#SBATCH --account=PROJECTCODE            # Project account


module purge                                                        # purge any loaded modules
module load lang/Python/3.7.0-foss-2018b # Load a module within a job script

echo My working directory is `pwd`
echo Running job on host:
echo -e '\t'`hostname` at `date`
echo
echo
echo Job completed at `date`
```

Tips of your job scripts.

You will need to make sure you have following directives defined in your job scripts. Its is especially important to try to measure how resource (time, memory, CPUs) your job will need. This can take time and a few iterations of job to fully understand it, but if you ask for less resource going forward your job will spend less time queuing.

Managing time

The *time* directive is the time that the job take to run. It can be omitted in which case the default time is 8 hours. The job will be killed if it exceeds this time. Specifying a shorter and more accurate *time* will allow your job to start execution sooner. The *time* can be specified in two ways:

1. In the job script using the "#SBATCH --time=02:00:00" directive (RECOMMENDED)

The form of the time is "HH:MM:SS".

Controlling memory usage

You need to understand the memory requirements of your program. A job has a default memory allocation and sometimes you will need to request more memory so your program can run.

--mem=<*size[units]*> Specify the real memory required per node. Default units are megabytes.

For example:

    #SBATCH --mem=1gb

Redirecting output

The %j in the -o (--output) line tells SLURM to substitute the job ID in the name of the output file. You can also add a -e or --error with an error file name to separate output and error logs.

Filename patterns

There are several useful placeholders that can be used in filenames which will be automatically filled in by SLURM. The full list of these can be found in the sbatch man page, under the *`filename pattern`* heading.

Specifying a project code

Please use the following line in your batch scripts, so that we can associate your work with your project:

    #SBATCH --account=YOUR-PROJECT-CODE

Specifying a project account code is mandatory.

Exercise 2: Submitting your job to Viking

Now we will submit your job to the Viking cluster

To Submit a job run

```
[abc123@login1(viking) scratch]$ sbatch viking_job1.slurm
```

To see your job in the queue run the squeue commnd. The job running below has a jobid of 147875

```
[abc123@login1(viking) scratch]$ squeue -u abc123
JOBID  PARTITION NAME     USER ST TIME NODES NODELIST(REASON)
147875 nodes     viking_j.s abc123 R  0:04 1     node170
```

When you submit your job it will take the environment you have setup with it. For example if you have loaded a particular program using the module system a job run on node124 will know where the program is stored. It will also output files from the place you submitted your job from (in the above example your scratch directory) unless you specify otherwise.

We do set partition limits on the various queues on Viking. With this job example we requested resource on the nodes partition, but if you wanted a lot of memory or access to GPUs you will have to request a different partition. To find out about the different partitions available please read VK12) Available Resource Partitions and their limits. The majority of jobs will default to the nodes partition which has the most compute nodes. Please try to not request more memory or more time than you need, it will just mean you will wait in the queue for longer. When your jobs complete you will be issued with an email on how efficient your job is. Try to reduce what you request to a sensible level. Here are some examples

- You requested 8GB memory but your job only used 2 GB. Reduce to 3GB of memory in your job scripts.
- You requested a time of 4 hours but your job only ran in 1 hour. Request 2 hours the next time you submit your job.

To delete a job from the queue use the *scancel [options] <jobid>* command, where jobid is a number referring to the specified job (available from *squeue*).

```
[abc123@login1(viking) scratch]$ scancel 147876
```

A user can delete all their jobs from the batch queues with the *-u* option:

$ scancel -u=<userid>

If your job has failed or you are interested in finding out more information about a job that has already completed you can use the sacct command.

To look at your job history run sacct -j jobid

```
[abc123@login1(viking) scratch]$ sacct -j 147876
JobID        JobName    Partition  Account    AllocCPUS  State      ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
147876       simple.job nodes      dept-proj+ 1          CANCELLED+ 0:0
147876.batch batch                 dept-proj+ 1          CANCELLED  0:15
```

See VK3) Submitting Jobs to Viking and VK4) Job script configuration for more information.

Further information on your job status

Job State Codes

When submitting a job, the job will be given a "state code" (ST) based on a number of factors, such as priority and resource availability. This information is shown in the squeue commands. Common states are:

State	Explanation
R (Running)	The job is currently running.
PD (Pending)	The job is awaiting resource allocation.
CG (Completing)	Job is in the process of completing. Some proccesses on some nodes may still be active.
F (Failed)	Job terminated on non-zero exit code or other failure condition.

Job Reason Codes

The REASON column from the *squeue* command provides useful information on why a job is in the current state. Some of these reasons may be one or more of the following:

REASON	Explanation
AssociationJobLimit	The job's association limit has reached it's maximum job count.
AssociationResourceLimit	The job's association has reached some resource limit.
AssociationTimeLimit	The job's association has reached it's time limit.
BeginTime	The job earliest start time has not yet been reached.
Cleaning	The job is being requeued is still cleaning up from it's previous execution.
Dependency	The job is waiting for a dependent job to complete.
JobHeld Admin	The job has been held by the admin.
JobHeld User	The job has been held by the user.
JobLaunchFailure	The job could not be launched. This may be due to a file system problem, invalid program name, etc.

NodeDown	A node required by the job is not available at the moment.
PartitionDown	The partition required by the job is DOWN.
PartitionInactive	The partition required by the job is in an Inactive state and unable to start jobs.
PartitionNodeLimit	The number of nodes required by this job is outside of the partition's node limit. Can also indicate that required nodes are DOWN or DRAINED.
PartitionTimeLimit	The job exceeds the partition's time limit.
Priority	One or more higher priority jobs exist for this partition or advanced reservation.
ReqNodeNotAvail	Some node specifically required for this job is not available. The node may currently be in use, reserved for another job, in an advanced reservation, DOWN, DRAINED, or not responding. Nodes which are DOWN, DRAINED, or not responding will be identified as part of the job's "reason" field as "UnavailableNodes". Such nodes will typically require the intervention of a system administrator to make available.
Reservation	The job is awaiting its advanced reservation to become available**.**

Why is my job not running?

Many people ask why their job is not running. Here is a great video explaining how Slurm works and how it schedules your jobs.