# The CSC Cluster
# `tinis`

The CSC cluster `tinis` is the university's newest computing cluster, coming online in November 2015. It currently consists of 190 active nodes, each containing two Intel Xeon E5-2630 v3 processors with hyperthreading disabled, and are interconnected by QLogic TrueScale InfiniBand. This worksheet will guide you through using the cluster for the first time.

## 1 Logging in

To log in to the cluster, use `ssh` like so:

```
ssh username@tinis.csc.warwick.ac.uk
```

`username` was provided to you when you created your account. If you have not yet registered an account then refer to the registration guide provided in seminar 2.

## 2 Compiling code

To compile MPI code on `tinis`, certain environment modules must be loaded. Run the following command after logging in:

```
module load intel impi
```

This loads MPI libraries and compiler wrappers that enable and simplify the compilation of MPI code. These modules must be loaded every time you log in - this can be automated by adding the above command to your `.bash_profile`.

## 3 Running a job

`tinis` is composed of a single login node and many compute nodes. After logging in to `tinis` you will be at the login node, which should only be used for compiling code and running lightweight tasks. All compute-intensive tasks should be run on the compute nodes, which is achieved by requesting access to compute nodes as an interactive job, or by submitting a job to the batch system which will in turn allocate it to compute nodes.

## 3.1 Interactive jobs

Running your MPI program in an interactive job is recommended if you are testing it or debugging, as you can monitor its execution in real time. To request an interactive job run the following command in the terminal:

```
salloc -p cs402 --nodes=2 --ntasks-per-node=16 --mem-per-
    cpu=128 --time=00:10:00
```

- `-p cs402` – request resource from the CS402 partition.
- `--nodes=2` – request 2 nodes.
- `--ntasks-per-node=16` – request 16 processors per node, for a total of 32 processes.
- `--mem-per-cpu=128` – request 128MB of memory for each process.
- `--time=00:10:00` – request the resource for 10 minutes. If you are still in this interactive job after 10 minutes then it will be terminated. It is beneficial to not request more time than necessary as quick jobs are prioritised over long jobs.

After running this `salloc` command you should see output similar to the following:

```
salloc: Job is in held state, pending scheduler release
salloc: Pending job allocation 270882
salloc: job 270882 queued and waiting for resources
salloc: job 270882 has been allocated resources
salloc: Granted job allocation 270882
[csrcmp@login1 ~]$
```

The resource request has now been granted. Note that I have been returned to the login node. `srun` is used to run an MPI program on the allocated resource, creating one MPI process for each requested processor:

```
srun ./helloWorld
```

This will run the chosen program interactively across the allocated compute processors. To terminate the interactive job, just execute `exit`.

```
[csrcmp@login1 ~]$ exit
exit
salloc: Relinquishing job allocation 270882
salloc: Job allocation 270882 has been revoked.
```

## 3.2 Batch jobs

Running in interactive mode has a few limitations, mainly in terms of the amount of time you have, and the number of processors you can request. This is why we will need to use batch jobs.

Submitting a batch job differs to an independent job in two ways. Firstly, a batch script must be prepared that contains the resource request followed by a call to `srun`. The second difference is you do not use `salloc`, instead you use `msub`. An example batch script is shown below:

```
#!/bin/bash
#MSUB -q cs402
#MSUB -l nodes=2:ppn=16
#MSUB -l pmem=128mb
#MSUB -l walltime=00:5:00
srun ./helloMPI
```

To submit the batch job use `msub`:

```
msub ./job.msub
```

`msub` will return the job ID, allowing you to monitor and cancel the job:

```
[csrcmp@login1 hpc]$ msub ./job.msub

271276
[csrcmp@login1 hpc]$
```

All batch jobs will by default write `stdout` and `stderr` to a log file named `slurm-job_id.out`, in the directory in which the job was submitted.
For a full list of `msub` options see:
https://computing.llnl.gov/tutorials/moab/#Submit
See the `tinis` Wiki for more information:
https://wiki.csc.warwick.ac.uk/twiki/bin/view/Main/TinisUserGuide#MPI_jobs

## 4 Managing Jobs

To view your jobs use `showq -u your_username`. To kill one of your jobs use the command `scancel job_id`, for example:

```
scancel 271276
```