# CEU - CSD3 Manual

## Table of Contents

## Access to CSD3

Apply for a CSD3 account: https://www.hpc.cam.ac.uk/rcs-application

Access to CSD3 is currently using RAVEN credentials. If a user does not have RAVEN, get in touch with Ank for access.

ssh CRSid@login.hpc.cam.ac.uk
password: RAVEN password

Each user will have a scratch space with a 1TB quota at the path below:

```
cd  ~/rds/hpc-work
```

Access to CEU resources will depend on specific projects that the user is involved in. At the moment this is named after the sheets within the excel file that describes CEU data holdings on CSD3:

CEU data holdings
These can be found on CSD3 at the path:

```
cd ~/rds/rds-*
```

## Access to RDS project space

If you don't see the CEU RDS projects (e.g. post_qc_data) in your ~/rds/ folder, you may need to first accept the terms and conditions on your HPC account page: https://selfservice.uis.cam.ac.uk/account/

You should see the following listed under the "Research Data Store" heading: "genetic_resources", "legacy_projects", "post_qc_data", "pre_qc_data", "projects", "public_databases", and "results". If you see in red "T&Cs not yet accepted", click the project name and accept the terms and conditions. The project will then be mounted to your ~/rds/ folder within a few hours by someone in the HPC team.

If you do not see these on your HPC account page, please contact Ank Michielsen <am2710@medschl.cam.ac.uk>.

# Transfer of data from cardio/cordis to CSD3

Most of the data has been transferred by Wojciech and those not in the spreadsheet that the user would like to transfer from cardio/cordis to CSD3 can be done using 'rsync' until **31 August 2019**, **when all the data on cardio/cordis will be deleted.**

Users have been given read access to most directories within the CEU data holdings area. Write access is restricted analysts and PIs working on specific projects. If you have write access and would like to transfer files to a directory on CSD3, follow the steps/commands below:

*commands are shown in **blue,** options are in **grey** and path/arguments are provided in **green**

## Option 1

Step 1: login to CSD3 as above
Step 2: navigate to the folder on CSD3 where you would like to copy the files from cardio/cordis using the command below:

```
cd [path_on_csd3]
```

Step 3: copy files from cardio to CSD3 using **rsync**

```
rsync -rltEDvz crsid@cardio-login.hpc.cam.ac.uk:[path_on_cardio]/ .
```

Note: There's a "." in the end of the command above to denote current working directory

Use the command below for cordis:

```
rsync -rltEDvz crsid@cordis-login.hpc.cam.ac.uk:[path_on_cordis] .
```

## Option 2

Step 1: login to cardio as usual
Step 2: Navigate to the folder containing the files or directories you would like to copy to CSD3

```
cd [path_on_cardio]
```

Step 3: copy files from cardio to CSD3 using **rsync**

```
rsync -rltEDvz  * crsid@login.hpc.cam.ac.uk:[path_on_csd3]/
```

Note: Use * as above if you would like to copy all the files including sub directories to the path on CSD3. If you just need a single directory or file, replace * with the directory or file name.

## Running jobs on CSD3

**Converting your cardio jobs to CSD3 jobs:**

Running jobs on CSD3 is similar to running jobs on cardio, with a few important differences. When you ssh to CSD3 you will be placed on one of the 16 login nodes (named login-e-1 through login-e-16). From these, you submit jobs to the SLURM queuing system as you would on cardio. You will need to make a few minor modifications to your existing job submission scripts:

- You will need to specify an account. In your job submission scripts you will need to add a line with **#SBATCH -A [ACCT_NAME]** where **[ACCT_NAME]** is the chargeable account and service level (SL) to assign the job. This can also be provided as a command line argument at submission, e.g. **sbatch -A [ACCT_NAME]**. Run the **mybalance** command to view the accounts available to you. More details on service levels and accounts are provided below.

- You will need to change the requested partition/queue to one or more of the following: **skylake**, or **skylake-himem**. E.g. **#SBATCH -p skylake**.

    - The **skylake** queue contains nodes with 32 CPUs and 6GB of RAM per CPU.
    - The **skylake-himem** queue contains nodes with 32 CPUs with 12GB of RAM per CPU.

    Unlike cardio and cordis, CPU and memory resources are directly tied to each other. If you request a set number of CPUs without specifying the memory, you will receive the number of CPUs x the amount of memory per CPU. For example, 3 cores on **skylake** will give you 3x6GB=18GB of RAM. Conversely, if you request a specific amount of memory you will be assigned the required number of cores. For example, if your job requires 60GB of ram then you would receive 20 cores on **skylake** or 10 cores on **skylake-himem**.

- Unlike cardio you may run jobs that are not intensive on the login node, for example R sessions. This is advisable in situations where you might normally request an interactive session on cardio. There is a watchdog script that will kill any processes taking too many resources on a login node. See the HPC policies on Interactive sessions for details.

**Accounts and service levels:**

Access to the resources on CSD3 is in the order of our preferred usage is listed below. More details on what these levels mean can be found at:

CSD3 Service Levels and Policies

**Usage recommendations:**

1. **Service Level 3 (SL3)**

   Users will submit their job to the general queue (-p ) and use the core hours available to each PI. Currently each PI may receive **200,000 CPU core hours**, **8000 GPU hours** and **1000 KNL node hours per quarter**.

2. **Service Level 2 (SL2)**

   Users will submit their jobs to a queue (-q ) and the resources each job uses will be debited from the core hours purchased by the CEU. It is expected that only high priority jobs that won't fail are sent through SL2.

   There is a cost implication for use of SL2, and usage will be monitored.

3. **CEU dedicated partition**

   100 cores available in a few weeks once migration is complete and until February 2021. Details will be updated once this has been discussed and agreed.

## Submitting Jobs on CSD3

Further details on CSD3 please see the introduction to HPC slides and the CSD3 manual. An example slurm script is provided below, and can also be found when you log in to CSD3 under your home directory in the file ~/slurm_submit.peta4-skylake :

**Example SLURM script**
(credits: CSD3 documentation)

This is similar to cardio/cordis and is a simple bash script in the following format:

```
#!/bin/bash
command goes here...
```

This will submit a job and placed in a general queue with default parameters. You can add various parameters within the SLURM script to tell the scheduler which account (-A) should be used, which partition (-p) should the job be submitted and whether this should be placed in a queue with certain set priorities.

Below is an example of the general format in which options should be provided within a SLURM script

```
#!/bin/bash

#! Name of the job:
#SBATCH -J my_script

#! Account name for group, use SL2 for paying queue
#! Use mybalance to list project allocations:
#SBATCH -A MYPROJECT-CPU

#! Output filename:
#SBATCH --output=my_script.out

#! Errors filename:
#SBATCH --error=my_script.err

#! Number of nodes to be allocated for the job
#! (for single core jobs always leave this at 1):
#SBATCH --nodes=1

#! Number of tasks. By default SLURM assumes 1 task per node & 1 CPU per task.
#! (for single core jobs always leave this at 1):
#SBATCH --ntasks=1

#! How many cores will be allocated per task?
#! (for single core jobs always leave this at 1)
#SBATCH --cpus-per-task=1

#! Estimated runtime: hh:mm:ss (job is force-stopped after, if exceeded):
#SBATCH --time=36:00:00

#! Estimated maximum memory needed (job is force-stopped if exceeded):
#! RAM is allocated in ~5990mb blocks, you are charged per block used,
#! and unused fractions of blocks will not be usable by others.
#SBATCH --mem=5990mb

#! This is the partition name:
#SBATCH -p skylake

##SBATCH --mail-type=ALL

#! Don't put any #SBATCH directives below this line

#! Modify the environment seen by the application. For this example we need
the default modules.
```

```
#! This line enables the module command:
. /etc/profile.d/modules.sh

#! This line clears any modules currently loaded modules:
module purge

#! This line loads the basic environment:
module load rhel7/default-peta4

#! Add any other modules that need to be loaded for the application to run:

###############################################################################
#COMMANDS GO HERE
###############################################################################
```

**Estimating your resource usage:**

As mentioned above, the resources you are allocated depend on both the CPUs and amount of memory you require for your job. For the **skylake** queue each node has 32 CPUs and each CPU is allocated 6GB of RAM. For the **skylake-himem** queue each CPU has 12GB of RAM. You will be allocated the maximum number of cores and memory required given the CPU and memory resources you request. For example on the **skylake** queue if you request 10 CPUs you will be allocated 60 GB of RAM, or if you ask for 60 GB of RAM you will be allocated 10 CPUs. If you ask for 1 CPU and 60 GB of RAM, you will still be allocated 10 CPUs even though you've request only 1.

**If in doubt, simply request the amount of memory you need with --mem. E.g.:**
**sbatch --mem 8192 -A [ACCT_NAME] --time 2:0:0 -p skylake**
**This will ask the cluster for 8GB of RAM for 2 hours (giving you 2 CPUs on 1 node).**

There is no sure way to estimate the exact memory usage you will need for a job, but you can get an idea by listing the file sizes of the data you want to work, then adding an extra 20%. If you load in more memory, your job will be killed, in which case simply add more memory (e.g. multiply by 1.5) and try again until your job runs.

To find out the resources your job used you can use the **sacct** command with the job id slurm gave your job. For example:
**sacct -j [jobid] -o JobID,AllocCPUS,State,Elapsed,MaxRSS**
Will list for your job the number of CPUs allocated to the job, the state of the job (e.g. completed, or failed, see **man sacct** for details on other state codes), the amount of time the job ran for, and the maximum amount of memory it used.