



Quick Start Guide

Contents

Useful links	2
SAFE User Guide	3
Login	3
Sign Up	3
Terms and Conditions.....	5
Main Page (Example shows a fake user)	5
Your user account(s).....	6
Creating an SSH key Accessing Our Machines How to Upload a new SSH Key.....	7
Purpose of an SSH Key.....	7
Benefits of SSH Keys	7
Private Key	7
Public Key	8
How to create your SSH Key?.....	8
Backing Up	12
Uploading your SSH Key to SAFE/ Updating your SSH Key	12
Submitting a job to JADE2	13
The Slurm Scheduler	13
The module tool	19
Using Containerised Applications	21
How and where to save your data on JADE2	25

Useful links

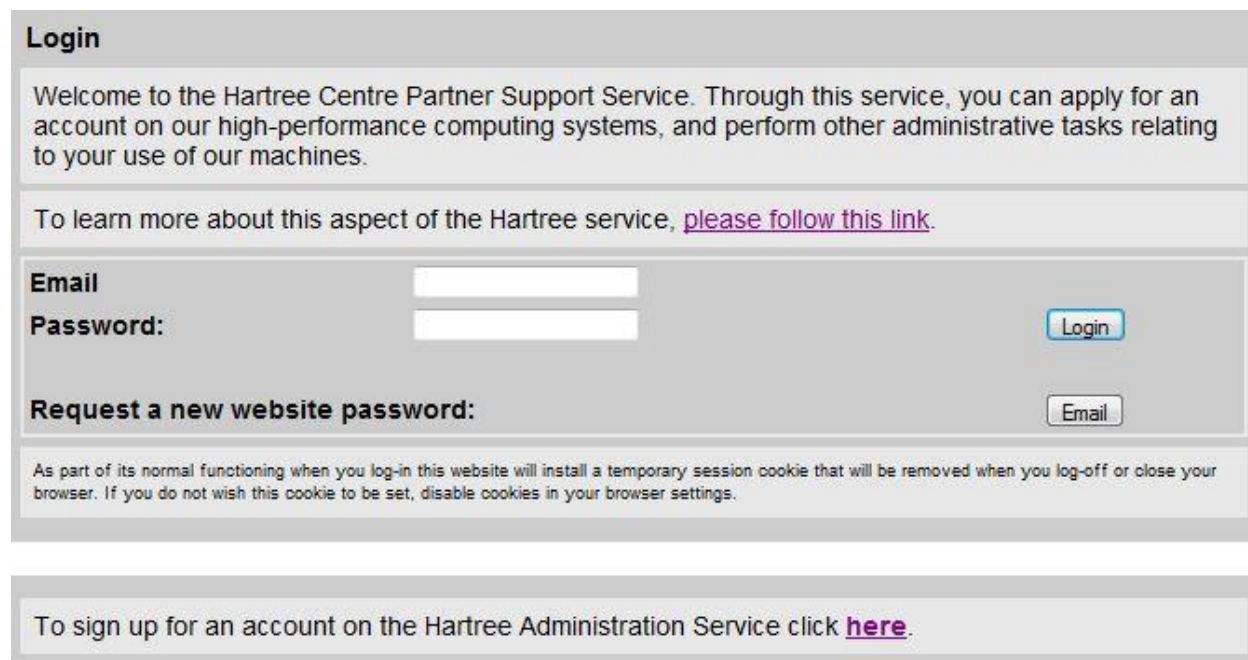
- To gain access to the machine please create a Hartree SAFE account (if you do not have one already):
<https://um.hartree.stfc.ac.uk/hartree/signup.jsp>
- Please sign up for a Hartree ServiceNow account, this will allow you to access support and information for JADE 2:
<https://stfc.service-now.com/hartreecentre>
- Further information on how to use the machine can be found via service-now:
https://stfc.service-now.com/kb?id=kb_search&kb_knowledge_base=ad6a44fc1b609050dbb77449cd4bcb96
- Slurm documentation: <https://slurm.schedmd.com>
- Main JADE2 page: <https://um.hartree.stfc.ac.uk/hartree>
- JADE2 Docs: <http://docs.jade.ac.uk/en/latest/jade/connecting.html>
- JADE2 Forum/Service: <https://stfc.service-now.com/hcssp/>
- Slurm interactive jobs: <https://hpc-uit.readthedocs.io/en/latest/jobs/interactive.html>
- Singularity container building: https://sylabs.io/guides/3.0/user-guide/build_a_container.html

SAFE User Guide

The online Service Administration system is at URL:
<https://um.hartree.stfc.ac.uk/hartree/login.jsp>

SAFE is the principal method used to manage project and user accounts. It also lets you view resource usage and submit queries. Here we provide a step by step guide.

The SAFE front page has a “Login” form and a link to “Sign Up”.



Login

Welcome to the Hartree Centre Partner Support Service. Through this service, you can apply for an account on our high-performance computing systems, and perform other administrative tasks relating to your use of our machines.

To learn more about this aspect of the Hartree service, [please follow this link](#).

Email

Password:

Request a new website password:

As part of its normal functioning when you log-in this website will install a temporary session cookie that will be removed when you log-off or close your browser. If you do not wish this cookie to be set, disable cookies in your browser settings.

To sign up for an account on the Hartree Administration Service click [here](#).

Image of SAFE login dialog

Login

(If you have already signed up) - Type your username (this is your email address) and password into the form to log into the SAFE system.

If you have forgotten your password, click - ' Email' next to 'Request a new website password'.

Sign Up

Complete the form with your details to apply for an individual user account. You will normally do this if you are part of an approved Hartree Centre Project and your Principle Investigator (PI) or Company Account Manager (CAM) has asked you to do so.

Your Details:

Email Address	<input type="text"/>
Nationality	United Kingdom <input type="button" value="v"/>
Title (Mr, Mrs, Dr etc.)	<input type="text"/>
First Name	<input type="text"/>
Initials	<input type="text"/>
Last Name	<input type="text"/>
Institution	<input type="text"/>
Department	<input type="text"/>
Phone number (include Int. dialing code e.g. +44 for UK)	<input type="text"/> + followed by numbers and spaces
Opt out of user Emails	<input type="checkbox"/>
Address Line 1	<input type="text"/>
Address Line 2	<input type="text"/>
Address Line 3	<input type="text"/>
Address Line 4	<input type="text"/>
Town/City	<input type="text"/>
Postcode	<input type="text"/>
Country	United Kingdom <input type="button" value="v"/>
SSH Public key	<input type="text"/> <input type="button" value="Browse..."/>
PersonalCertificate	<input type="text"/>

Your postal address will look like this.
You must re-enter your department and institution if you wish it to appear in the postal address

	Your name Address Line 1 Address Line 2 Address Line 3 Address Line 4 Town/City PostCode Country
--	---

Image of SAFE personal details dialog

If any details were typed in incorrectly or they change, you have the ability to edit these details after you log in.

Please note: we require an institutional or corporate email address. If you use, for example, a gmail address you will see a field error message in the field such as "Forbidden email address gmail"

In order to create your SSH key via Windows/Linux/Unix/Mac - [Click here](#)

Terms and Conditions

During signing up you will be asked to agree to the STFC Acceptable Use Policy. The policy is described here: <http://www.stfc.ac.uk/aup>.

Once you submit the form you will be asked to accept the Terms and Conditions of Access.

Please read these Terms and Conditions. By clicking the 'I accept the Terms and Conditions of Access' button you will be entering into a contract with us. If you do not agree to these Terms and Conditions, we will not be able to register you or allow you to use the service. Before accepting the Terms and Conditions, please note: you can change any of the details you have input by clicking your browser's BACK button and then editing them. You can also change them later by returning to this website. These Terms and Conditions are offered only in English. You may read the terms of the agreement here.

Click the button to accept the Terms and Conditions of Access.

You will then be sent an email with your SAFE Web ID and password. Your access to the SAFE interface will always be via this Web ID (your email address) and web password.

When you have obtained a password and logged in you will receive a welcome message and after continuing you may well be asked to change your Web password due to expiry. Enter the password supplied and then the new password you want to use.

Once your password has been updated, you will be sent to the SAFE 'Main' menu page (see below).

Please note: SAFE is for managing accounts on projects and viewing your budget allocation. Help with the Hartree Centre machines can be found on the Self Service Portal.

Main Page (Example shows a fake user)

This menu displays information such as:

- *Your details* - This is where you can keep your personal details up to date.
- *Information about your project and group* - Here you can view the projects you are part of and the groups.
- *Request Join Project* - By clicking on 'Request Join Project' you can request to join a project. Choose a Project from a drop down menu and enter the signup password for the Project. This signup password will have been supplied to you by the PI of the Project. Click on 'Request' to obtain a UserID. It will then be the responsibility of your PI to 'process' and accept your request.

Once you have joined a Project, other boxes will be present.

Your user account(s)

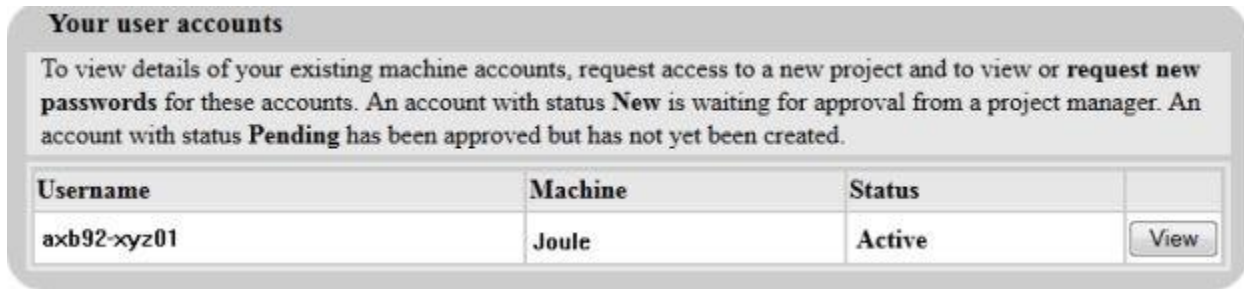


Image showing the SAFE list of your user accounts

A new user account is generated for you for each project you join. This means you may have multiple user accounts. This section shows you a list of all of your accounts.

You will be allocated a base user name identifier of the form *xxxnn* e.g. *axb92* (if you have no middle initial an 'X' is used instead). This will be combined later with the Group ID for each project to which you belong, to give an overall UserID of the form *xxxnn-aaann* e.g. *axb92-xyz01*.

You can view your Username(s), Machine and Status.

- An account with status *New* is waiting for approval from a project manager.
- An account with status *Pending* has been approved but has not yet been created.
- An account with status *Active* has been approved has been created - therefore please check your emails.

For 'Active' users, clicking 'View' takes you to a page where you can request a New Password, View your machine password (requires web password) and add an additional certificate.

Creating an SSH key | Accessing Our Machines | How to Upload a new SSH Key

JADE2 Machine details: [Jade2.hartree.stfc.ac.uk](https://jade2.hartree.stfc.ac.uk) - Access Requires SSH key

JADE2 accounts will not be processed until an SSH key is uploaded to your [SAFE](#) account, you can do this by following these steps.

Purpose of an SSH Key

The purpose of an SSH key is for authentication purposes only and serves to confirm that you really are who you say you are when logging onto the systems. SSH itself then encrypts the data flowing over the connection.

SSH keys come in pairs - a "private" key and a "public" key. We have provided instructions below on how to generate a pair of SSH keys on different computer platforms and what you need to do before accessing our systems.

Benefits of SSH Keys

One advantage of using SSH keys for logging on to the Hartree Centre systems is that you may have multiple user ids on the systems, but we will set them up with the same SSH public key so that you don't need to remember and maintain multiple passwords.

If you wish to log in to our systems from multiple machines, for example, a home and work PC, then you have two options to make this easy:

Copy your private key (probably called `id_rsa`) between your different systems.

Add more public keys to the file `~/.ssh/authorized_keys` in your accounts on the Hartree Centre systems.

Since we are using SSH to know that it's really you who is logging on to the Hartree Centre systems, how do you know that it's really the Hartree Centre that you're logging on to?

The first time during connection you will receive a prompt that displays the "RSA key fingerprint" and if you want to be certain you can check that this matches the fingerprint of the appropriate system. If it doesn't match, it's possible although unlikely that you are connecting to someone "pretending" to be us.

Please feel free to contact us via [REPORT IT](#) on the portal homepage if you are concerned. Once you agree that it's OK, the details are stored and you don't get prompted again.

Private Key

The private key is something you should keep to yourself and guard safely. If you create the key on a Unix/Linux/Mac system it must not be "world readable", so that no user on the same system can see it. (Use `chmod 600 ~/.ssh/id_rsa` to correct the file permissions if you get a warning about this.) You may further protect your SSH private key on your machine when you create it by creating a "passphrase". This means that every time you use SSH you will be prompted to enter this passphrase,

and so that anyone else who manages to copy your private key will be unable to use it unless they also know the passphrase.

In addition, if you have created a passphrase, you can use the `ssh-add` command on your computer in conjunction with an implementation of `ssh-agent` so that your passphrase only needs to be entered once and is saved in your computer's memory. This may be convenient if you need to log on and off repeatedly.

Public Key

You can safely distribute your public key to anyone with whom you want to communicate. Anyone who has your public key can use it to decrypt messages you encrypt with your private key. Your public key is required to be uploaded into our [SAFE](#) system in order to gain access to the systems.

How to create your SSH Key?

On Windows

You can use [PuTTYgen](#) to create your key and [PuTTY](#) to access the systems. Alternatively, you could use [Cygwin](#) and follow the instructions below in the section "On Linux".

These instructions relate to PuTTYgen. Please note, we do not support or provide PuTTY or PuTTYgen.

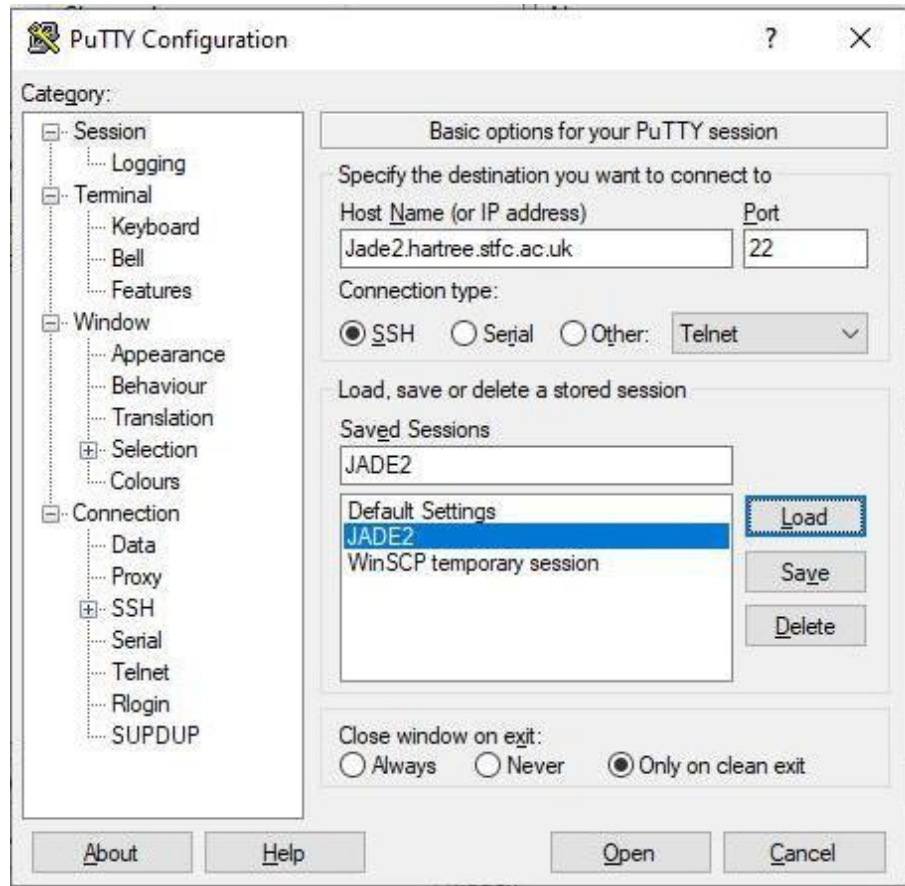
First, download [PuTTY](#) and [PuTTYgen](#) from the [external link](#) and install.

01. Start [PuTTYgen](#).
02. Check that SSH-2 RSA is checked.
03. The minimum 'Number of bits' is 1024 but you can change this number to 2048 for increased security by typing this number in the 'Number of bits in a generated key' box at the bottom of the window.
04. Press 'Generate'.
05. Move your mouse randomly in the small screen in order to generate the key pairs.
06. A key comment will be generated, which will identify the key (useful when you use several SSH keys).
07. You can choose to enter a passphrase which has to be confirmed. The passphrase is used for extra protection for your key. You will be asked for it when you connect via SSH.
08. Click "Save public key" to save your public key. Give it a meaningful name and use the format [.pub](#)
09. Click "Save private key" to save your private key. You could give it the same name as the public key, but suffixed with "priv" to show that they are part of a pair. For example, if you called the public key "mykey", call the private key "mykey_priv". Make a note of the public key filename, as you will need it when logging in to the Hartree Centre systems.

Uploading Your SSH Key to PuTTY (For Windows Users)

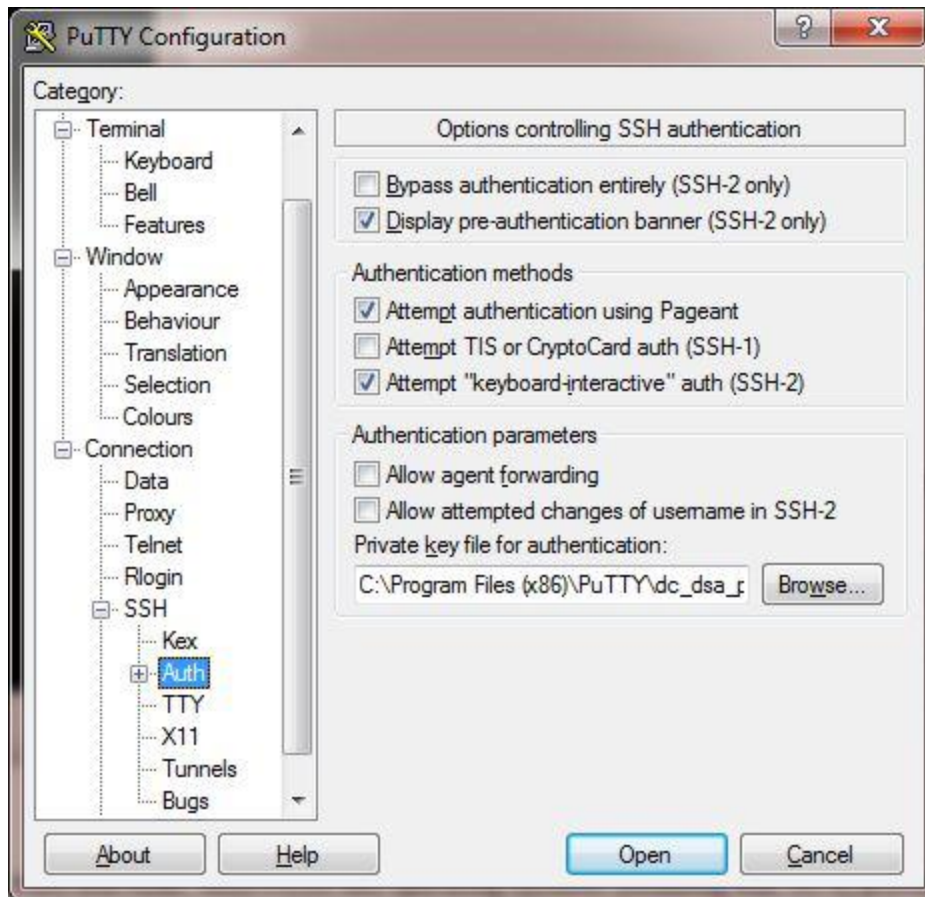
If PuTTY is your chosen software to use, please find below our instructions: (Again, please note PuTTY is not supported by the Hartree Centre)

- Start PuTTY and type [Jade2.hartree.stfc.ac.uk](https://jade2.hartree.stfc.ac.uk) into the "Host" box.
- In order to save your session for future use, give it a name in the "Saved Sessions" box. Now click the Save button.



- You can also enter your login ID in the 'Data' window (under Connections) in the box called Auto-login username.
- To add your SSH key, in the navigation tree on the left, expand the + next to SSH, and then click on Auth.
- Click the Browse button and select the filename of the private key you created previously.

For example:



You can also add your UserID so that you don't have to enter it each time you logon (if you are the sole user of the desktop machine)

- Go to the Connection > Data screen and type in your userID in the Auto-login username box.
- Be sure to click on Session in the navigation tree (you may need to scroll up) to return to the first page, and click Save again.

Now you are ready to login, and you have a ready-made PuTTY session for next time.

- Click Open to proceed or double-click the session name.
- During first time logging in - Select Yes when prompted to accept the server's host key.

The RSA key fingerprint should be `52:af:69:42:15:64:9d:91:e2:e9:69:58:41:bf:38:0e`.

- If you haven't previously saved your user-id - You will now be prompted via 'Login as:' here type your user id.

On Linux and Mac OSX

You can create an ssh key pair in a terminal window as follows: Use the `ssh-keygen` utility to create your key. For a 2048 bit RSA key do:

```
$ ssh-keygen -t rsa
```

For increased security, you can make an even larger key with the `-b` option. For example, for 4096 bits do:

```
$ ssh-keygen -t rsa -b 4096
```

01. When prompted, you can press Enter to use the default location (`/home/your_username/.ssh/id_rsa` on Linux, or `/Users/your_username/.ssh/id_rsa` on Mac) if you don't already have a key installed, or specify a custom location if you are creating a second key (or just want to for whatever reason).
02. You can choose to enter a passphrase at the prompt or just hit enter (twice) to leave the passphrase blank. Using a passphrase is more secure, but you will need to type it in each time you make a connection using this key pair. This is just a password used to unlock your key. If someone else gets a copy of your private key they will be able to log in as you on any account that uses that key, unless you specify a passphrase. If you specify a passphrase they would need to know both your private key and your passphrase to log in as you.
03. Your private key should now be in the location you specified, and your public key will be at that same location but with `.pub` tacked onto the filename.
04. Change to the directory containing your public key and display the key by typing `cat id_rsa.pub`
05. Highlight and copy the entire string beginning `"ssh-rsa"`
06. Go to the field "SSH Public key" in the SAFE sign up page and paste in the Key using CTRL+V, (or right click, and Paste).

Please note: at the time of creating a UserID, we use the most recent SSH Key. Additional keys that are uploaded to [SAFE](#) will not be updated automatically so if you want to use a different Key after the UserID is created you must tell us via selecting [REQUEST IT](#) from the portal.

Unix/Linux/Mac First Time Log in

ssh to the login node, providing the user-id you have been given, which will be in the form: axb92-gxh02.

For example:

```
yqk32924vig ~ $ ssh djc87-build@Jade2.hartree.stfc.ac.uk
```

```
OUTPUT: Last login: Thu Oct 18 09:55:39 2012 from rw46vig.dl.ac.uk
```

Or you could do:

```
ssh Jade2.hartree.stfc.ac.uk -l djc87-build
```

During the first time you connect, you will be asked to accept the server's host key.

Type Yes to accept it.

The RSA key fingerprint should be 52:af:69:42:15:64:9d:91:e2:e9:69:58:41:bf:38:0e.

Backing Up

It is also worth backing up your SSH key in an external format, eg. a secure USB drive. This way if you change your workstation it is possible to copy your SSH details to

Uploading Your SSH Key to SAFE/ Updating Your SSH Key

In order to update your SSH key:

- Simply log into [SAFE](#) and select 'Update personal details'.
- Add your new SSH key by copying and pasting from notepad to the text box 'SSH Public Key' or attach a file.

Here it is possible to add multiple keys, please ensure space is between each key. Once you have uploaded the key, it will then be necessary for your account to be manually updated. (Please note, this is not an instant update.)

Submitting a job to JADE2

Running software on the JADE2 system is accomplished with batch jobs, i.e. in an unattended, non-interactive manner. Typically a user would log into the JADE2 login node prepare a job script and submit it to the job queue.

For the list of available software on JADE2, please refer to the [module tool](#) section of this document.

The Slurm Scheduler

Running software on the JADE2 system is accomplished with batch jobs, i.e. in an unattended, non-interactive manner. Typically a user logs in to the JADE2 login nodes, prepares a job script and submits it to the job queue.

Jobs on JADE2 are managed by the [Slurm](#) batch system, which is in charges of:

- allocating the computer resources requested for the job,
- running the job and
- reporting the outcome of the execution back to the user.

Running a job involves, at the minimum, the following steps

- preparing a submission script and
- submitting the job to execution.

Commands

The table below gives a short description of the Slurm commands that are likely to be useful to most users:

Command	Description
sacct	report job accounting information about active or completed jobs
sbatch	submit a job script for later execution (the script typically contains one or more srun commands to launch parallel tasks)
scancel	cancel a pending or running job
sinfo	reports the state of partitions and nodes managed by Slurm (it has a variety of filtering, sorting, and formatting options)
squeue	reports the state of jobs (it has a variety of filtering, sorting, and formatting options), by default, reports the running jobs in priority order followed by the pending jobs in priority order

All Slurm commands have extensive help through their man pages e.g.:

```
man sbatch
```

shows you the help pages for the [sbatch](#) command.

In addition to the above commands, the table below gives two more commands that can be used in special cases, e.g. to obtain an interactive session, such as used in the Machine Learning examples. The commands are

Command	Description
<code>salloc</code>	allocate resources for a job in real time (typically used to allocate resources and spawn a shell, in which the <code>srun</code> command is used to launch parallel tasks)
<code>srun</code>	used to submit a job for execution in real time

Please note: `srun` can be used to launch application into execution from within submission scripts. The success of this in the case of MPI distributed applications depends on the MPI software stack having been build with support for PMI (Process Management Interface).

Preparing a submission script

A submission script is a `bash` script that

- describes the processing to carry out (e.g. the application, its input and output, etc.) and
- requests computer resources (number of GPUs, amount of memory, etc.) to use for processing.

The simplest case is that of a job that requires a single node with the following requirements:

- the job uses 1 node,
- the application is a single process,
- the job uses a single GPU,
- the job will run for no more than 10 hours,
- the job is given the name “job123” and
- the user should be emailed when the job starts and stops or aborts.

Supposing the application run is called `myCode` and takes no command line arguments, the following submission script runs the application in a single job:

```
#!/bin/bash
# set the number of nodes
#SBATCH --nodes=1
# set max wallclock time
#SBATCH --time=10:00:00
# set name of job
#SBATCH --job-name=job123
# set number of GPUs
#SBATCH --gres=gpu:1
# mail alert at start, end and abortion of execution
#SBATCH --mail-type=ALL
# send mail to this address
#SBATCH --mail-user=name.surname@domain.extension
# run the application
myCode
```

We will now break down each of the commands.

- `#!/bin/bash`: This sets the script as a Linux bash script
- For Slurm, the lines starting with `#SBATCH` are directives that request job scheduling resources. (Note: it is important that you put all the directives at the top of a script, before any other commands; any `#SBATCH` directive coming after a bash script command is ignored!)
- `#SBATCH -nodes`: This sets the number of compute nodes requested for the job; in the case of the above example, we have only requested 1.
- `#SBATCH -error`: This set where the standard error file should be stored and specifies the name of the file. This can be left as default and will be stored in your home directory or can be set by including the full path to the directory after `SBATCh --error`.
- `#SBATCH -output`: This is the same principle as the standard error but allows the user to specify where the standard output is to be stored.
- `#SBATCH --job-name`: To be used to name the job you are running so that it can be distinguished for other work.
- `#SBATCH -time=T`: This sets the maximum walltime (where T has format H:M:S) that the job can run for before it is automatically killed by SLURM (your job may well finish before this time).
- `#SBATCH -gres`: This requests specific resources, primarily on JADE2 this will be requesting the number of GPUs the job requires, in the example we are requesting 1 gpu.
- `#SBATCH -partition`: This sets the partition you wish the job to run on, each partition has a separate queue with different specifications (maximum wall time and running job limit).
- `#SBATCH --mail-user=<email_address>`: An email notification is sent if an address is specified. The notification options can be set with `#SBATCH --mail-type=<type>`, where `<type>` may be `BEGIN`, `END`, `FAIL`, `REQUEUE` or `ALL` (for any change of job state).
- `myCode`: In the example the job is only required to run the application called `myCode`. This can be expanded to set various different operations required as part of the job and is to be written in normal Linux `bash` script. The job starts in the same folder where it was submitted (unless an alternative path is specified), and with the same environment variables (`modules`, etc.) that the user had at the time of the submission. In this example, this final part only involves invoking the `myCode` application executable.

Here is an example of a `bash` submission script which invokes a Python script named `tftestmultigpu.py`:

```
#!/bin/bash
# set the number of nodes
#SBATCH --nodes=1
# set max wallclock time
#SBATCH --time=01:00:00
# set name of job
#SBATCH --job-name=TF_TEST_MULTIGPU
# set number of GPUs
#SBATCH --gres=gpu:8
# run the application
python3 tftestmultigpu.py
```

Submitting jobs with the command `sbatch`

Once you have a submission script ready (e.g. called `submit.sh`), the job is submitted to the execution queue with the command `sbatch script.sh`. The queuing system prints a number (the job id) almost immediately and returns control to the Linux prompt. At this point the job is in the submission queue.

Once the job submitted, it will sit in a pending state until the resources have been allocated to your job (the length of time your job is in the pending state is dependent upon a number of factors including how busy the system is and what resources you are requesting). You can monitor the progress of the job using the command `squeue` (see below).

Once the job starts to run you will see files with names such as `slurm-1234.out` either in the directory you submitted the job from (default behaviour) or in the directory where the script was instructed explicitly to change to.

Job partitions on JADE2

Partitions are Slurm entities defined by the system administrators that allow the separation and control of jobs according to their characteristics. Each partition has a number of compute nodes associated with it, as well as properties that control job placement. A job can be submitted to be executed by a particular partition, and if no partition is specified, the default one is selected.

There are three partitions on JADE, which are:

Partition name	Description
<code>big</code>	Partition dedicated to jobs that occupy an entire node, i.e. 8 GPUs
<code>small</code>	Partition dedicated to jobs that utilise a single GPUs each.
<code>devel</code>	Partition dedicated to testing.

The partitions have the following limits for submitted jobs:

Partition name	Partition Size	Job Walltime limit	Running Job limit
<code>big</code>	30 nodes	24 hours	5 Jobs
<code>small</code>	30 nodes	6 days	8 Jobs
<code>devel</code>	3 nodes	1 hour	1 Job

The default partition is `big`. Information on these partitions can be obtained with the commands `sinfo -a` or `scontrol show partition=small`.

Submitting to a particular partition can be done by specifying the partition as an argument to `sbatch`, e.g. `sbatch -p devel sub.sh`, or by directly supplying a request for that partition in the submission script, e.g. `#SBATCH --partition=devel`.

The `devel` partition should be used to check your submission script works correctly and that your application starts to execute without errors.

Upon reaching the per user running job limit for a partition, any further jobs submitted to that same partition by the same user will be shown as state Pending (PD) with the Reason set as QOSMaxJobsPerUserLimit.

Monitoring jobs with the command `squeue`

`squeue` is the main command for monitoring the state of systems, groups of jobs or individual jobs.

The command `squeue` prints the list of current jobs. The list looks something like this:

OUTPUT:

```
| JOBID PARTITION   NAME     USER ST      TIME  NODES NODELIST(REASON)
| 2497   devel      srun     bob  R       0:07    1 dgk119
| 2499     big      test1    mary R       0:22    4 dgk[201,204]
| 2511   small      test2    steve PD      0:00    4 (Resources)
```

The first column gives the job ID, the second the partition where the job was submitted, the third the name of the job (specified by the user in the submission script) and the fourth the user ID of the job owner. The fifth is the status of the job (**R** = running, **PD** = pending, **CA** = cancelled, **CF** = configuring, **CG** = completing, **CD** = completed, **F** = failed). The sixth column gives the elapsed time for each particular job. Finally, there are the number of nodes requested and the nodelist where the job is running (or the cause that it is not running).

Some useful command line options for `squeue` include:

- `-u` for showing the status of all the jobs of a particular user, e.g. `squeue -u bob`;
- `-l` for showing more of the available information;
- `-j` for showing information regarding a particular job ID, e.g. `squeue -j 7890`;
- `--start` to report the expected start time of pending jobs.

Read all the options for `squeue` on the main page `squeue` using the command `man squeue`, including how to personalize the information to be displayed.

Deleting jobs with the command `scancel`

Use the `scancel` command to delete a job, e.g. `scancel 1121` to delete job with ID 1121. Any user can delete their own jobs at any time, whether the job is pending (waiting in the queue) or running. A user cannot delete the jobs of another user. Normally, there is a (small) delay between the execution of the `scancel` command and the time when the job is dequeued and killed.

Environment variables

At the time a job is launched into execution, Slurm defines multiple environment variables, which can be used from within the submission script to define the correct workflow of the job. A few useful environment variables are the following:

- `SLURM_SUBMIT_DIR`, which points to the directory where the `sbatch` command is issued;
- `SLURM_JOB_NODELIST`, which returns the list of nodes allocated to the job;

- `SLURM_JOB_ID`, which is a unique number Slurm assigns to a job.

In most cases, `SLURM_SUBMIT_DIR` does not have to be used, as the job lands by default in the directory where the Slurm command `sbatch` was issued.

`SLURM_SUBMIT_DIR` can be useful in a submission script when files must be copied to/from a specific directory that is different from the directory where `sbatch` was issued.

`SLURM_JOB_ID` is useful to tag job specific files and directories (typically output files or run directories) in order to identify them as produced by a particular job. For instance, the submission script line

```
myApp &> $SLURM_JOB_ID.out
```

runs the application `myApp` and redirects the standard output (and error) to a file whose name is given by the job ID. Note: the job ID is a number assigned by Slurm and differs from the character string name given to the job in the submission script by the user.

Job arrays

An array job allows users to run the same job a number of times with different inputs, using a single submission command and submission script. Clearly, this is a lot quicker than manually submitting the job multiple times and allows for easier management of running and completed jobs as well as maximising results throughput.

In order to submit array jobs, users must prepare the appropriate submission script template, keeping in mind that it will be used by all the jobs in the array, i.e. each job will require the same amount of resources, wall-time etc. Note that users should add to the submission script a way to provide, or generate, the appropriate simulation/input parameters needed by the program executable to run each simulation in the array.

Here are a few examples:

```
# submit a job array with index values between 0 and 7
$ sbatch --array=0-7 sub.sh
```

```
# submit a job array with index values of 1, 3, 5 and 7
$ sbatch --array=1,3,5,7 sub.sh
```

```
# submit a job array with index values between 1 and 7 with a step size of 2 (i.e. 1,
3, 5 and 7)
$ sbatch --array=1-7:2 sub.sh
```

The index values are used by Slurm to initialise two environment variables when the job launches into execution. These variables are

- `SLURM_ARRAY_JOB_ID`, set to the first job ID of the array and
- `SLURM_ARRAY_TASK_ID`, set to the job array index value.

To give an example, suppose you submit an array of three jobs using the submission command `sbatch --array=1-3 sub.sh`, which returns:

`OUTPUT: Submitted batch job 10`

Then, the environment variables in the three jobs will be

Job array index	Variables
1	<code>SLURM_ARRAY_JOB_ID=10; SLURM_ARRAY_TASK_ID=1</code>
2	<code>SLURM_ARRAY_JOB_ID=10; SLURM_ARRAY_TASK_ID=2</code>
3	<code>SLURM_ARRAY_JOB_ID=10; SLURM_ARRAY_TASK_ID=3</code>

The above environment variables can be used within the submission script to define what each individual job within the array does. To take a simple example, suppose each job in the array uses a single GPU and takes the input from a file that is identified by the same index as the job. The submission script could look like this:

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --job-name=test
#SBATCH --time=00:30:00
#SBATCH --gres=gpu:1
myCode --input "file_${SLURM_ARRAY_TASK_ID}.inp"
```

The application `myCode` will then use the `file_1` as the input for the first job in the array `file_2` as the input for the second job in the array and so forth.

The module tool

The GNU/Linux operating system makes extensive use of the working environment, which is a collection of individual environment variables. An environment variable is a named object in a shell that contains information used by one or more applications; two of the most used such variables are `$HOME`, which defines a user's home directory name, and `$PATH`, which represents a list paths to different executables. A large number of environment variables are already defined when a shell is open but the environment can be customised, either by defining new environment variables relevant to certain applications or by modifying existing ones (e.g. adding a new path to `$PATH`).

`module` is a Software Environment Management tool, which is used to manage the working environment in preparation for running the applications installed on JADE2. By loading the module for a certain installed application, the environment variables that are relevant for that application are automatically defined or modified.

Useful commands

The module utility is invoked by the command `module`. This command must be followed by an instruction of what action is required and by the details of a pre-defined module.

The utility displays a help menu by doing:

`module help`

The utility displays the available modules by issuing the command:

`module avail`

OUTPUT:

```
----- /jmain02/apps/lmod/lmod/modulefiles/Core -----
lmod/7.6      settarg/7.6   use.dev      use.own      use.viz

----- /jmain02/apps/Modules/modulefiles/packages -----
autodock-gpu/1.5.2      gromacs/2021.2 (D)   plumed/2.6.2
emacs/27.1              mdtraj/1.9.5      pytorch/1.9.0
ffmpeg/4.2.2           namd/2.14         sox/14.4.2
gnuplot/5.2.8          namd/3.0-alpha7 (D)  tensorflow/2.3.1
gromacs/2020.3         nano/5.3          tensorflow/2.7.0 (D)
gromacs/2020.4-plumed-2.6.2  opencv/4.2.0    tmux/3.2a      (D)
gromacs/2021.1         openmm/7.5.0     vim/8.2

----- /jmain02/apps/Modules/modulefiles/other -----
cuda/10.1              gcc/9.1.0         python/anaconda3
cuda/10.2              hdf5/1.10.7       python/3.8.6   (D)
cuda/11.1-gcc-9.1.0    netcdf/4.7.4     tmux/3.2a
cuda/11.2              (D)  openmpi/4.0.5-gcc-9.1.0
```

Where:

D: Default Module

Use "module spider" to find all possible modules.

Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".

or displays only the information related to a certain software package, e.g.:

`module avail tensorflow`

OUTPUT:

```
----- /jmain02/apps/Modules/modulefiles/packages -----
tensorflow/2.3.1      tensorflow/2.7.0 (D)
```

Where:

D: Default Module

Use "module spider" to find all possible modules.

Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".

The `avail` instruction displays all the versions available for the installed applications, and shows which version is pre-defined as being the default. A software package is loaded with the `load` or the `add` instructions, e.g.:

`module load tensorflow`

If no version is specified, the default version of the software is loaded. Specific versions, other than the default can be loaded by specifying the version, e.g.:

```
module load tensorflow/2.7.0
```

OUTPUT:

```
Python 3.8.6 is now loaded in your environment.
```

```
    CUDA-11.2 loaded
```

```
    TensorFlow-2.7.0 with Python3 loaded.
```

The modules that are already loaded by users in a session are displayed with the command:

```
module list
```

A module can be “unloaded” with the unload or rm instructions, e.g.:

```
module unload tensorflow
module load gromacs/2020.3
```

Lastly, all modules loaded in a session can be “unloaded” with a single command::

```
module purge
```

Best practices

`module` can be used to modify the environment after login, e.g. to load the Portland compilers in order to build an application. However, most frequent usage will be to load an already built application, and the best way to do this is from within the submission script. For example, assuming a job uses the NAMD molecular dynamics package, the submission script contains:

```
module purge
module load namd
```

Using Containerised Applications

On entering the container the present working directory will be the user’s home directory: `/home_directory`

Any files you copy into `/home_directory` will have the same userid as normal and will be available once exiting the container. The local disk space on the node is available at: `/local_scratch/$USERID`

Any data will be lost once the interactive session is ended. There are two ways of interacting with the containerised applications.

Docker Containers

Listing available containers

To list the containers and version available on the system do:

```
-bash-4.2$ containers
```

OUTPUT:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nvcr.io/nvidia/tensorflow	20.11-tf2-py3	98a7952f7f9c	5 weeks ago	11.6GB
nvcr.io/nvidia/pytorch	20.11-py3	ae35b2b3cad1	5 weeks ago	13.2GB
nvcr.io/hpc/namd	3.0-alpha3-singlenode	205dd048d054	5 months ago	1.24GB
nvcr.io/hpc/gromacs	2020.2	c8a188675719	5 months ago	570MB
nvcr.io/nvidia/caffe	20.03-py3	aa6834df762b	9 months ago	4.82GB
nvcr.io/nvidia/caffe2	18.08-py3	e82334d03b18	2 years ago	3.02GB
nvcr.io/nvidia/torch	18.08-py2	889c9b4d3b71	2 years ago	3.06GB

All the applications in containers can be launched interactively in the same way using 1 compute node at a time. The number of GPUs to be used per node is requested using the `gres` option. To request an interactive session on a compute node the following command is issued from the login node:

```
strun --gres=gpu:2 --pty /jmain02/apps/docker/pytorch 20.11-py3
```

This command will show the following, which is now running on a compute node:

OUTPUT:

```
=====
== PyTorch ==
=====

NVIDIA Release 20.11 (build 17345815)
PyTorch Version 1.8.0a0+17f8c32

Container image Copyright (c) 2020, NVIDIA CORPORATION. All rights reserved.

Copyright (c) 2014-2020 Facebook Inc.
Copyright (c) 2011-2014 Idiap Research Institute (Ronan Collobert)
Copyright (c) 2012-2014 Deepmind Technologies (Koray Kavukcuoglu)
Copyright (c) 2011-2012 NEC Laboratories America (Koray Kavukcuoglu)
Copyright (c) 2011-2013 NYU (Clement Farabet)
Copyright (c) 2006-2010 NEC Laboratories America (Ronan Collobert, Leon Bottou, Iain Melvin, Jason Weston)
Copyright (c) 2006 Idiap Research Institute (Samy Bengio)
Copyright (c) 2001-2004 Idiap Research Institute (Ronan Collobert, Samy Bengio, Johnny Mariethoz)
Copyright (c) 2015 Google Inc.
Copyright (c) 2015 Yangqing Jia
Copyright (c) 2013-2016 The Caffe contributors
All rights reserved.

NVIDIA Deep Learning Profiler (dlprof) Copyright (c) 2020, NVIDIA CORPORATION. All rights reserved.

Various files include modifications (c) NVIDIA CORPORATION. All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the underlying project or file.

NOTE: Legacy NVIDIA Driver detected. Compatibility mode ENABLED.

groups: cannot find name for group ID 31196
I have no name!@085a165a38ee:/home_directory$
```

Note. The warnings in the last two lines can be ignored. To exit the container, issue the “exit” command. To launch the other containers the commands are:

```
srun --gres=gpu:8 --pty /jmain02/apps/docker/theano 18.08
srun --gres=gpu:4 --pty /jmain02/apps/docker/torch 18.08-py2
```

Batch Mode

There are wrappers for launching the containers in batch mode. For example, to launch the Torch application:

A Slurm batch script is used to launch the code, such as:

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH -J Torch
#SBATCH --gres=gpu:8
#SBATCH --time=01:00:00
/jmain02/apps/docker/torch-batch -c ./submit-char.sh
```

The output will appear in the slurm standard output file.

Each of the containerised applications has its own batch launching script:

```
/jmain02/apps/docker/torch-batch
/jmain02/apps/docker/caffe-batch
/jmain02/apps/docker/theano-batch
```

Singularity Containers

Singularity 3.7 is installed on compute nodes, it is not available on login nodes. When you build your container, within your own environment, your container you must have the following directories:

```
/tmp
/local_scratch
```

These will be mounted by the local node when your container executes. The `/tmp` & `/local_scratch` directory are the local RAID disks on the DGX node and should be used for building code or temporary files.

Unlike Docker containers, the home directory is the same as when you’re outside the container (e.g. `/jmain02/home/your_project/your_group/your_username`). You can use `cd ~` to get to your home directory and `echo $HOME` to print out your home location.

There are 2 scripts in the `/jmain02/apps/singularity/2.4/bin` directory that you can use to launch your container using Slurm:

```
singbatch
singinteractive
```

You call them with either

```
    singinteractive CONTAINER_FILE
# OR
    singbatch CONTAINER_FILE SCRIPT_TO_EXECUTE
```

You should use these scripts with Slurm. So for example with an INTERACTIVE session:

```
srun -I --pty -t 0-10:00 --gres gpu:1 -p small singinteractive
/jmain02/apps/singularity/singularity-images/caffe-gpu.img
```

If you want to run in batch mode, you should call `singbatch` (using `sbatch`) and provide a script to execute within the container.

You MUST respect the `CUDA_VISIBLE_DEVICES` variable within the container, as you can see ALL the GPUs in the container. Some of these GPUs may be in use by other users and Slurm has allocated you a specific ones/group & will set this variable for you. If you are familiar with Docker, it only shows you the GPUs have been allocated.

Slurm will clear out `/tmp` and `/local_scratch` once you exit the container, so make sure you copy anything back to your home directory if you need it! There is an example “caffe” image provided in [/jmain01/apps/singularity/singularity-images](#) if you wish to contribute an image for others to use, please submit an issue to the Github Issue tracker.

How and where to save your data on JADE2

We would like to make users aware that data stored on the JADE2 cluster is not backed up. Although JADE2 is designed for resilience and will be carefully administered by our Platforms and Infrastructure group, in the event of a catastrophic failure there is no guarantee data would be able to be retrieved from the machines. Users who have project critical data are encouraged to take regular local backups to avoid transferring large volumes of data from the system at once.

Data from jobs run on the JADE2 system should be stored in the user's `home` directory and will be stored there by default. Data can also be stored in the group or project shared directory, if data needs to be accessed by collaborators it should be stored in the shared directory.

The directory structure would be:

```
/jmain02/home/project/group/user
```

 for the user's home directory

And for shared directories:

```
/jmain02/home/project/shared  
/jmain02/home/project/group/shared
```

The simplest way to direct where the data would be stored is to specify this in the submission script using the `SBATCH` commands for standard error and standard output, etc. Should you wish to save data outside of the default location, for example in shared directories, you can specify the full path for the file.

Where required there is additional 'flash' storage that can be used in addition to the standard storage disks, this is only provided to specific groups who are approved by the JADE management consortium. If your project wishes to utilise this flash storage the Principle investigator of the project must submit a request via the Self-Service portal specifying which groups within that project are requesting access to the flash storage and details to explain why this is required. If this is approved the users within that group will have their accounts configured to access flash.

When storing data on flash storage the file path would be

```
/jmain02/flash/project/group/user  
/jmain02/flash/project/group/shared
```

In addition to these persistence storage options there is also temporary 'raid' storage which can be used to hold data required during simulations. This storage is only available when the job is running and will be removed by Slurm automatically once the job has been completed.

It should also be mentioned that `Git` version control system is installed on JADE2.

More information about SFTP File Transfer Protocol, and SFTP client and server software for all operating systems can be found at the following link:

<https://www.ssh.com/academy/ssh/sftp>.