

# Using Bede

## Contents

- Registering
- Login
- Multi Factor Authentication
- Weekly "at risk" period for login nodes
- X2GO
- Acknowledging Bede
- File Storage
- Node Architectures and Partitions
- Running Jobs
- Grace-Hopper Pilot

Bede is running Red Hat Enterprise Linux 8 and access to its computational resources is mediated by the Slurm batch scheduler.

## Registering

Access to the machine is based around projects:

- For information on how to register a new project, please see [the N8CIR website](#).
- To create an account to use the system:
  - Identify an existing project, or register a new one.
  - Create an EPCC SAFE account and login to the SAFE system at [safe.epcc.ed.ac.uk](https://safe.epcc.ed.ac.uk)
  - Once there, select "Project->Request access" from the web interface and then register against your project
  - An account will then be created for you on the Bede. This is a manual process so will not be immediate. You should be notified once your account has been created and is ready to be used.

# Login

Bede offers SSH and X2GO services running on host `bede.dur.ac.uk` (which fronts the two `ppc64le` login nodes, `login1.bede.dur.ac.uk` and `login2.bede.dur.ac.uk`). SSH or X2GO should be used for all interaction with the machine (including shell access, file transfer and graphics).

## ! Important

Please use an SSH client and **not** X2GO the first time you ever log in to Bede, or immediately after a password reset. Your MFA token will be generated, but X2GO will not show it to you.

## ⚠ Warning

Login nodes are subject to a weekly "at risk" period every Tuesday from 08:00 to 10:00 (from 2025-02-25). See [Weekly "at risk" period for login nodes](#) for more details.

The login nodes are shared between all users of the service and therefore should only be used for light interactive work, for example: downloading and compiling software, editing files, preparing jobs and examining job output. Short test runs using their CPUs and GPUs are also acceptable.

Most of the computational power of the system is accessed through the batch scheduler, and so demanding applications should be submitted to it (see "Running Jobs").

# Multi Factor Authentication

When connecting to the login nodes, the system has to validate your identity. In order to help keep accounts secure, Bede employs a technique called Multi Factor Authentication (MFA), where something you know (your password), is combined with something you have (an app on a mobile device, such as your phone).

When you first login to Bede, or after a password reset, you should use SSH to connect to the machine (rather than x2go) and you will be prompted to:

1. Change your password
2. Import an MFA token into an app on your mobile device

Instead of *Password*, to login after this you will be prompted for:

- **First Factor** - this is your password

 [latest](#) ▼

- **Second Factor** - this is a one-time password code provided by the token in your app

Mobile apps that can be used to import an MFA token include Microsoft Authenticator, Google Authenticator, and other TOTP authentication clients. When the token is generated on Bede, it will display a QR code, which can then be imported into the app (e.g. select  to add an account and then *Scan a QR code* or *Other account* to scan the QR code).

Please be aware that this MFA token is only for Bede. You may also need separate tokens to access SAFE and other services, such as those provided by your home institution. Bede's token should be compatible with any MFA app you may already be using.

If you are unable to use MFA, for example because of accessibility reasons, please contact [support](#) for options.

**If you have lost your password or MFA token, please use EPCC's SAFE system to request a password reset for your Bede login account, which we normally aim to process within a working day.**

If you are finding that you are having to use your password and a MFA token too many times, we have provided some tips on [how to reduce the frequency that MFA is required](#).

## Weekly "at risk" period for login nodes

A weekly "at risk" period has been introduced for Bede's login nodes, every Tuesday morning from 8am to 10am (from 2025-02-25) for regular maintenance tasks.

This will sometimes involve interruptions, such as a reboot, in which case logged-in users will be warned on their terminal screen shortly before they are disconnected.

During this time:

- *Login sessions* and so *interactive jobs* may be ended.
- *File transfers* to (or from) Bede may be ended.
- Batch jobs and compute nodes will continue to run as normal.

Why we are doing this:

- The timely application of security updates is increasingly important. The new "at risk" period will help us keep Bede and your work secure.
- The login nodes fill up with mislaid user processes over time, increasing load and slowing down your work. The new "at risk" policy will help improve Bede's interactive performance
- The new "at risk" period will reduce the number of routine emails we send to your Inbox.

# X2GO

Running graphical applications over the Internet using an ordinary SSH client can be quite slow. If you need to run such programs on Bede, you may wish to consider X2GO as an alternative.

Instructions for installing the X2GO client on Windows, Mac and Linux are on the [X2GO website](#).

## Warning

Please use an SSH client and **not** X2GO the first time you ever log in to Bede, or immediately after a password reset. Your MFA token will be generated, but X2GO will not show it to you.


Once you have installed and launched the X2GO client, provide Bede's connection details by clicking on the *Session -> New session...* menu item to open the *Session preferences - New session* window (this may happen automatically when you first run X2GO). Enter the following details:

- Session name: `bede.dur.ac.uk`
- Host: `bede.dur.ac.uk`
- Login: *your Bede username*
- Session type: *select Published applications from the drop down menu*

When you are ready to login, click on the `bede.dur.ac.uk` name in the session selection window, enter your username and click Ok. You will be prompted for your password (or `First Factor` and `Second Factor` if MFA is enabled) similar to a SSH login.

X2GO should now connect to Bede and show some details such as the *Session Id*. Once you see this, please pay attention to the row of four pale blue icons inside the grey box. These icons are:

- Circular icon - *open published applications window*
- Folder icon - *configure exporting a local directory to Bede*
- Pause icon - *disconnect from your X2GO session, leaving programs running, allowing you to reconnect to it later. If you use this feature, you will need to make sure that all your sessions use the same login node. To do this, follow the instructions above to create an X2GO session, but instead of using the address `bede.dur.ac.uk`, give the address of one of the login nodes explicitly (either `login1.bede.dur.ac.uk` or `login2.bede.dur.ac.uk`)*
- Terminate icon - *close your X2GO session, stopping all running programs*

At this point, Click on the circular icon to open the *Published Applications* window, select *System -> MATE Terminal* and click Start. If there are no applications to select please click on the  `latest` dropdown menu. In the *Published Applications* window, wait for 5 seconds, then open click on the circular icon again.

A terminal window should open, giving you command line access to Bede. Use it to interact with the service and launch applications, including graphical ones, in the usual way.

## Acknowledging Bede

All work that makes use of the Bede should properly acknowledge the facility wherever the work is presented.

We provide the following acknowledgement text, and strongly encourage its use:

"This work made use of the facilities of the N8 Centre of Excellence in Computationally Intensive Research (N8 CIR) provided and funded by the N8 research partnership and EPSRC (Grant No. EP/T022167/1). The Centre is co-ordinated by the Universities of Durham, Manchester and York."

Acknowledgement of Bede provides data that can be used to assess the facility's success and influences future funding decisions, so please ensure that you are acknowledging where appropriate.

## File Storage

Each project has access to the following shared storage:

- Project home directory ( `/projects/<project>` )
  - Intended for project files to be backed up
  - Modest performance
  - A default quota of `20GB`
- Project Lustre directory ( `/nobackup/projects/<project>` )
  - Intended for bulk project files not requiring backup
  - Fast performance
  - No quota limitations

By default, files created within a project area are readable and writable by all other members of that project.

In addition, each user has:

- Home directory ( `/users/<user>` )
  - Intended for per-user configuration files to be backed up
  - Modest performance

- A default quota of `20GB`

Please note that, as access to Bede is driven by project use, no personal data should be stored on the system.

Current utilisation and limits of a user's home directory can be found by running the `quota` command. Similar information can be found for the project home directory using the `df -h /projects/<project>` command.

To examine how much space is occupied by a project's Lustre directory, a command of the form `du -csh /nobackup/projects/<project>` is required. As `du` will check each and every file under the specified directory, this may take a long time to complete. We plan to develop the service and provide this information in a more responsive format in the future.

## Node Architectures and Partitions

As described on the [Hardware](#) page, Bede contains a mix of nodes using 2 CPU architectures and 3 models of NVIDIA GPU. Software must be compiled for each CPU architecture, and not all software is available, provided or compatible with each architecture.

Bede's original nodes contain Power 9 CPUs (`ppc64le`), with Nvidia Volta and Turing architecture GPUs (`SM_70` & `sm_75`). Jobs in the `gpu`, `test` and `infer` partitions will run on `ppc64le` architecture nodes.

The newer Grace Hopper open pilot include [NVIDIA Grace Hopper Superchips](#) which are composed of an ARM CPU (`aarch64`) and an NVIDIA Hopper GPU (`sm_90`). Jobs in the `ghlogin`, `gh` and `ghctest` partitions will run on the `aarch64` architecture nodes.

## Connecting to the `ghlogin` node

To get an interactive login-session on a Grace-Hopper node in the `ghlogin` partition, you must connect to Bede's regular login nodes as usual via `ssh` / `x2go`.

Once connected, the `ghlogin` command can be used to request an interactive session on the `ghlogin` node. The login environment includes shared (unrestricted) access to the Hopper GPU, and by default will provide 4 CPU cores and 16GB of RAM for 8 hours. Use additional `srn` style flags to request a different duration or resources. You must provide your project account.

```
# Request a default login session 4 cores, 16GB, 8 hours
ghlogin -A <project>
# Request 4 hours with 8 cores and 24GB of memory
ghlogin -A <project> --time 4:00:00 -c 8 --mem 24G
```



This will provide shell access to the login environment, which is a single Grace Hopper superchip. Access is mediated by slurm and you'll have a default of 4 cores and 1GB RAM for 8 hours (amend by adding srun style flags to the `ghlogin` command). Access to the GPU in the login environment is currently unrestricted.

## Running Jobs

Access beyond the login nodes should only be done through the Slurm batch scheduler, by packaging your work into units called jobs.

A job consists of a shell script, called a job submission script, containing the commands that the job will run in sequence. In addition, some specially formatted comment lines are added to the file, describing how much time and resources the job needs.

Resources are requested in terms of the type of node, the number of GPUs per node and the number of nodes required. The job will then receive the corresponding fraction of the node.

For example:

- a job requesting `2` GPUs on a `gpu` node (containing 4 V100 GPUs) will receive `2/4` of the CPU cores (`20` / `40` cores) and memory (`256` / `512GB`).
- a job requesting `1` GPU on a `gh` node (containing `1` GH200) will receive the full `72` cores and `480GB` of memory.

Higher Priority access for short test jobs requesting up to `2` nodes (8x V100 GPUs or 2x GH200 GPUS) for up to `30` minutes, to allow experimentation especially for jobs trying to make use of Bede's architecture for multi-GPU, multi-node use.

There are a number of example job submission scripts below.

## Requesting resources

Batch jobs for the `gpu`, `infer` and `test` partitions should be submit from the `ppc64le` login nodes via `sbatch` or `srun`.

To submit a job to the `gh` and `ghctest` partitions, you can use `sbatch` or `srun` as normal from within a `ghlogin` session. Alternatively, use the `ghbatch` or `ghrun` commands from a Bede login node.



## Part of, or an entire node

ppc64le

aarch64

Example job script for programs written to take advantage of a GPU or multiple GPUs on a single ppc64le node:

```
#!/bin/bash

# Generic options:

#SBATCH --account=<project> # Run job under project <project>
#SBATCH --time=1:0:0        # Run for a max of 1 hour

# Node resources:
# (choose between 1-4 gpus per node)

#SBATCH --partition=test    # Choose either "gpu", "test" or "infer" partition
#SBATCH --nodes=1           # Resources from a single node
#SBATCH --gres=gpu:1        # One GPU per node (plus 25% of node CPU and RAM)

# Run commands:

nvidia-smi # Display available gpu resources
nproc      # Display available CPU cores

# Place other commands here

echo "end of job"
```

## Multiple nodes (MPI)

Example job script for programs using MPI to take advantage of multiple CPUs/GPUs across one or more machines, via `bede-mpirun`:

ppc64le

aarch64



```
#!/bin/bash

# Generic options:

#SBATCH --account=<project> # Run job under project <project>
#SBATCH --time=1:0:0       # Run for a max of 1 hour

# Node resources:

#SBATCH --partition=gpu    # Choose either "gpu", "test" or "infer" partition
#SBATCH --nodes=2         # Resources from two nodes
#SBATCH --gres=gpu:4      # Four GPUs per node (plus 100% of node CPU and RA

# Run commands:

bede-mpirun --bede-par 1ppc <mpi_program>

echo "end of job"
```

The `bede-mpirun` command takes both ordinary `mpirun` arguments and the special `--bede-par <distrib>` option, allowing control over how MPI jobs launch, e.g. one MPI rank per CPU core or GPU.

The formal specification of the option is: `--bede-par <rank_distrib>[:<thread_distrib>]` and it defaults to `1ppc:1tpt`

Where `<rank_distrib>` can take `1ppn` (one process per node), `1ppg` (one process per GPU), `1ppc` (one process per CPU core) or `1ppt` (one process per CPU thread).

And `<thread_distrib>` can take `1tpc` (set `OMP_NUM_THREADS` to the number of cores available to each process), `1tpt` (set `OMP_NUM_THREADS` to the number of hardware threads available to each process) or `none` (set `OMP_NUM_THREADS=1`)

Examples:

```
# - One MPI rank per node:
bede-mpirun --bede-par 1ppn <mpirun_options> <program>

# - One MPI rank per gpu:
bede-mpirun --bede-par 1ppg <mpirun_options> <program>

# - One MPI rank per core:
bede-mpirun --bede-par 1ppc <mpirun_options> <program>

# - One MPI rank per hwthread:
bede-mpirun --bede-par 1ppt <mpirun_options> <program>
```

 [latest](#) ▼

**Note**

On `aarch64`, the `--1ppt` option is synonymous with `--1ppc` due to the absence of hardware SMT.

## Maximum Job Runtime

Partitions on Bede have default job times, and maximum job times that can be requested:

Partition Name	Default Job Time	Maximum Job Time
<code>infer</code>	<code>01:00:00</code>	<code>2-00:00:00</code>
<code>gpu</code>	<code>01:00:00</code>	<code>2-00:00:00</code>
<code>test</code>	<code>00:15:00</code>	<code>0-00:30:00</code>
<code>ghlogin</code>	<code>01:00:00</code>	<code>0-08:00:00</code>
<code>gh</code>	<code>01:00:00</code>	<code>2-00:00:00</code>
<code>ghtest</code>	<code>00:15:00</code>	<code>0-00:30:00</code>

Where, for example, `2-00:00:00` means ‘two days, zero hours, zero minutes, and zero seconds’. These job time limits affect what will and won’t be accepted in the `--time` field of your job script: `--time` values above the partition maximum will result in your job submission being rejected.

## Job Allocation Limits

There are currently no limits on the total or concurrent number of running jobs per project or per user. There are also no software-defined limits on the size of jobs (memory, cores, nodes), but jobs must be able to run on the available hardware.

Jobs are scheduled subject to Slurm’s [Multifactor Priority Plugin](#), which considers factors such as age, job size and fair-use when selecting jobs for execution.

## Grace-Hopper Pilot

Bede contains 6 NVIDIA Grace-Hopper nodes.

 [latest](#) ▼

Each Grace-Hopper node contains a single [Grace Hopper Superchip](#), containing one 72-core 64-bit ARM CPU and one 96GB Hopper GPU with NVLink-C2C providing 900GB/s of bidirectional bandwidth between the CPU and GPU. Further details are listed on the [Hardware](#) page.

## Connecting to the `ghlogin` node

To get an interactive login-session on a Grace-Hopper node in the `ghlogin` partition, you must connect to Bede's regular login nodes as usual via `ssh` / `x2go`.

Once connected, the `ghlogin` command can be used to request an interactive session on the `ghlogin` node.

Please see [Connecting to the ghlogin node](#) for more information.

## Batch Jobs

To submit a job to a Grace Hopper compute node, you can use `sbatch` or `srun` as normal from within a `ghlogin` session. Alternatively, use the `ghbatch` or `ghrun` commands from a Bede login node.

Your job submission scripts should specify the `--partition=gh` or `--partition=ghtest`.

Further details and example batch job submission scripts are provided in the [Requesting resources](#) section above.

## Software availability

The Grace-Hopper nodes contain `aarch64` architecture CPUs, rather than the `ppc64le` architecture CPUs in the other Bede nodes. As such, software built for the `ppc64le` CPUs will not run on the `aarch64` CPUs and vice versa.

The Grace-Hopper nodes are also running a newer operating (Rocky 9) system than the `ppc64le` nodes (RHEL 8), which can impact software availability.

Use `module avail` from the `ghlogin` node to list centrally provided software modules for the grace-hopper nodes.

Documentation for [software provided centrally on Bede](#) now includes grace-hopper specific information.



Key differences to be aware of include:

- [CUDA](#)
  - Code should be compiled for Compute Capability `90` using CUDA 11.8+.
  - CUDA 11.7 available on `aarch64` nodes should embed PTX for Compute Capability `80` (i.e. `-gencode=arch=compute_80,code=compute_80`)
- [Singularity](#) / [Apptainer](#)
  - The Grace-Hopper nodes provide [Apptainer](#) rather than [Singularity CE](#) for container support and enables the use of `--fakeroot` to build containers on the aarch64 nodes directly.
  - Usage is broadly the same, however there are some differences as [documented by apptainer](#) and on the [Apptainer page](#)
  - Container files are large, consider setting `APPTAINER_CACHEDIR` to avoid filling your home directory quota
- [PyTorch](#)
  - Installation options for pre-built pytorch for `aarch64` platforms with CUDA support enabled is limited. As of July 2024, only `pip` installs of pytorch `2.4.0` or newer for `CUDA 12.4` are suitable. See [PyTorch](#) for more information.
  - NVIDIA provide [NGC Pytorch containers](#) which can be used instead, with pytorch installed into the default python environment.
    - These containers are large, so you will likely want to set your `APPTAINER_CACHEDIR` environment variable to avoid filling your home directory quota
      - e.g. `export APPTAINER_CACHEDIR=/nobackup/projects/<project>/$USER/apptainer-cache`
  - Or follow [PyTorch Building from Source instructions](#)
- [GCC](#) compiling with `--std=c++17` may emit psABI warnings. These can be suppressed via `--Wno-psabi`.
- [MPI](#)
  - The `openmpi` module is available, and CUDA support is enabled if you additionally load a CUDA module. The `bede-mpirun` command should be used to launch programs.
  - The `ppc64le` equipment has an MPI with optimised multi-node GPU communications provided by the `mvapich2-gdr` module. This is not available for the Grace Hopper equipment; however, we plan to provide a `mvapich-plus` module in future to provide this functionality. In the meantime, if this is of interest, please contact us.

## Bash environment

If you have modified your Bede user environment (`.bashrc`, or `.bash_profile`) to make software available by default (i.e. conda), you may need to modify your env environment variables or source scripts based on the CPU architecture.

You can check the CPU architecture in your `.bashrc` via:

```
# Get the CPU architecture
arch=$(uname -i)
if [[ $arch == "aarch64" ]]; then
    # Set variables and source scripts for aarch64
    export MYVAR=F00
elif [[ $arch == "ppc64le" ]]; then
    # Set variables and source scripts for ppc64le
    export MYVAR=BAR
fi
```