POLAR SCIENCE
FOR PLANET EARTH

An Introduction to High Performance Computing 2021

Paul Sumption
pasump@bas.ac.uk

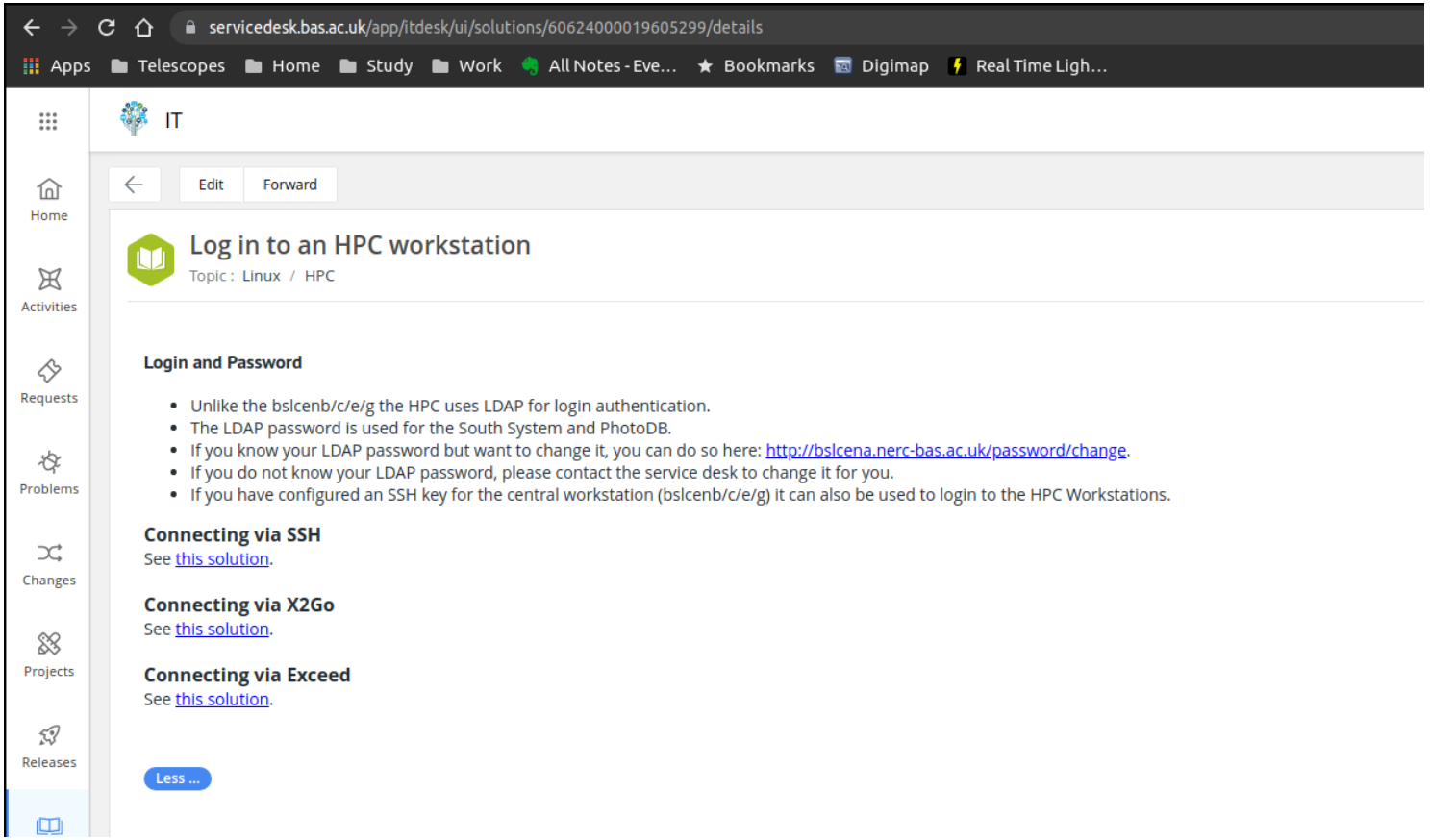
January 4, 2022

Exercise 1: Login to a BAS HPC Workstation

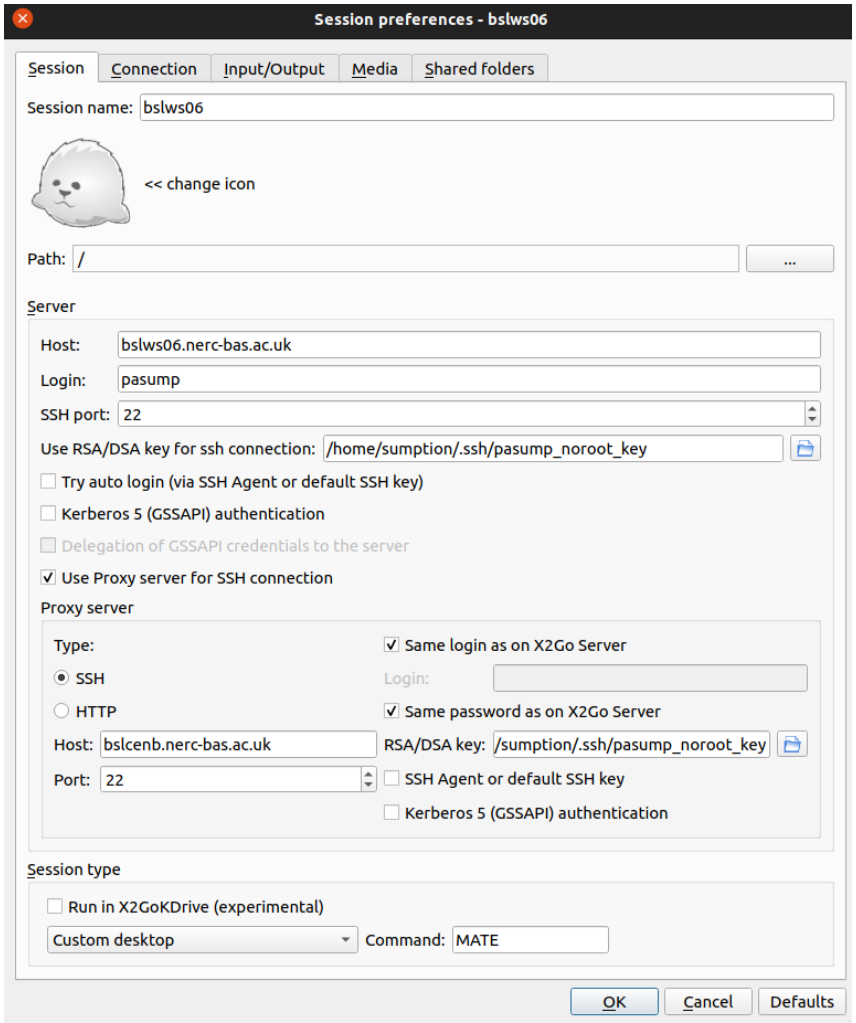
- Login to any of the workstations: bslws01...bslws12
- Configure X2go (it runs on Mac, Linux and Windows)
- X2go will also allow us to use a graphical desktop in our later exercises

Exercise 1: X2go solution

Follow the service desk solution to setup your client



Exercise 1: X2go session preferences

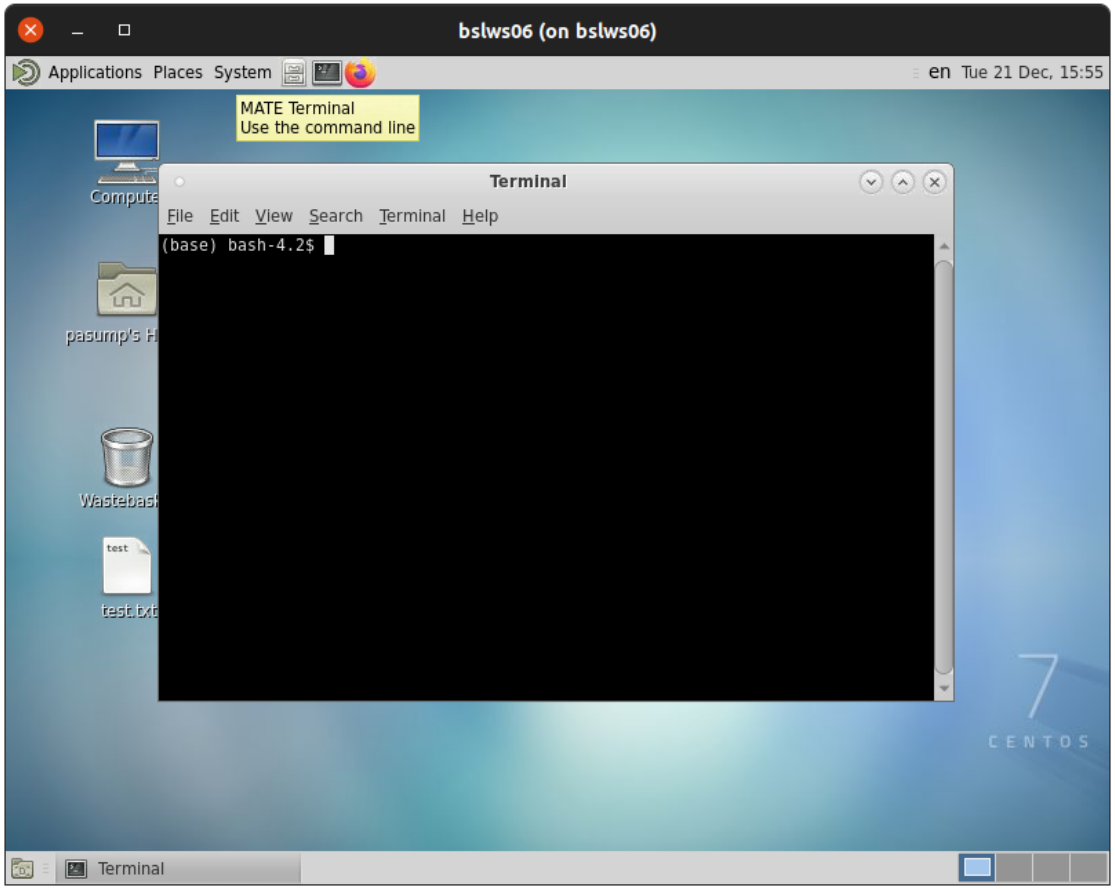


- Follow the instructions in the solution for your operating system
- Configure a session preference for a workstation
- My example uses ssh keys, you can just use passwords

Exercise 1: X2go session launch

- Click on a session and launch it
- You should see a MATE desktop
- Demonstration
- Troubleshooting

Exercise 2: Simple command line operations



- Click on the terminal icon
- A terminal window will open
- We can now interact with our 'shell'

Exercise 2: Simple command line operations

(a) List your current directory (folder) using **ls -al**. Use **df -h** to see the various filesystems, their sizes and their current total usages. You check the hostname of the system using – **hostname**, and **w** to find out who else is using it.

(b) Examine your personal filesystem quotas with the command **myquota**.

You should see a quota on /users, and a quota on /data/hpcdata as you have two homes.

Exercise 3: File transfer

Before attempting exercise 3:

- `http://ictdocs.nerc-bas.ac.uk/wiki/index.php/HPC:User_Guide_Exercises`
- Download the file **exercises.tgz** to your desktop.
- You will need an sftp client on your laptop to then transfer this file to the cluster.
- Linux and Mac users - open a terminal, you can use sftp from the command line
- Windows users - download winsftp

Exercise 3: File transfer

- Use SFTP to transfer the file **exercises.tgz** to your home directory

  Hints: The command is **sftp**. Use the same remote host, username and password as in the previous exercise.

  Use **mkdir exercises** to create a directory then **cd exercises** to change into the target directory, then **put exercises.tgz** to transfer the file from your desktop to the target directory on the cluster. Use **quit** to close the connection.

Optionally, copy the file over again using **rsync**.

Exercise 3: File transfer (ctd)

- Switch back to the SSH session you created in the previous exercise. Verify that the file is now present by using **ls**.

  Hints: Do **ls -al ~/exercises/**. Note that you can often reduce typing by pressing **TAB**.

- Unpack the tar archive to create an exercise subdirectory.

  Hints: Do **cd ~/exercises/** then **tar -zxvf exercises.tgz**.

Exercise 5: Modules and Compilers

- Go to the **exercises** directory of your cluster account.

  Hints: Firstly you may need to review Exercise 1 in order to reconnect to your cluster account. At the remote command prompt, change to the exercises directory (**cd ~/exercises**).

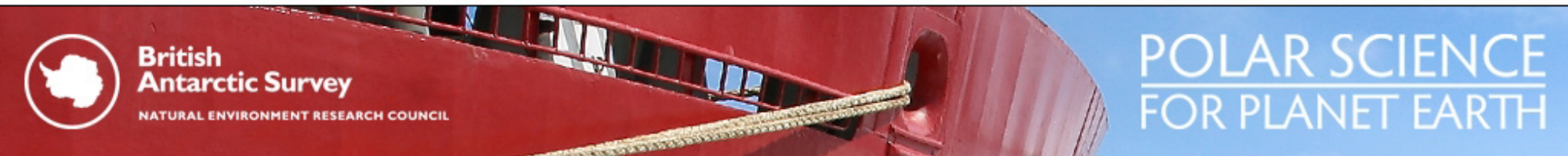
- Try to compile the **hello.c** program using the default **gcc** compiler (it will fail because there is a deliberate bug).

  Hints: **gcc hello.c -o hello**

Exercise 5: Modules and Compilers

- To fix the problem, open the **hello.c** file in an editor (e.g. **gedit**, **nano**, **emacs**).

  Hints: Launch gedit in the background by doing **gedit&**. A gedit window should appear. Remove the word **BUG**, save the file and recompile. Do **./hello** to run the program.



Exercise 5: Modules and Compilers (ctd)

- The default version of gcc on the HPC cluster is 4.8.5. Compile hello.c again with **hpc/gcc/7.2.0**.

  Hints: module av, module load, then **gcc hello.c -o hello2**

- Launch the Matlab GUI. Note this should work from either the SSH command-line or remote desktop sessions.

  Hints: **module load matlab** then run: **matlab&**

- Quit Matlab and launch it again without the graphical desktop interface. This is the way to launch it inside a batch job.

  Hints: **matlab -nodisplay -nojvm -nosplash**

Exercise 6: Submitting Jobs (Matlab)

- Submit a job which will run **matlab** on the **file.m** command file (which contains just the Matlab **ver** command).

  Hints:

  1. Load the matlab module at the place indicated in the file **job_script** in your exercises directory.
  2. Set the value of application to **¨matlab -nodesktop -nosplash -nojvm¨**
  3. Set the value of options to **¨-r file¨**
  4. Submit the job with **sbatch job_script**. The jobid is then printed.
  5. Watch the job in the queue with **squeue**.
  6. After it has disappeared, open the output file **slurm-jobid.out** in your editor. It should contain a list of licensed Matlab features from the ver command.

Exercise 7: Submitting Jobs (serial or threaded application)

- Submit a job which will run a copy of your hello program on 1 cpu.

Hints:

1. Edit the script **job_script** in your exercises directory. Set:
   #SBATCH –nodes=1
   #SBATCH –ntasks=1
   application="./hello"
2. Submit the job with **sbatch job_script**. The jobid is then printed.
3. Watch the job in the queue with **squeue**.
4. After it has disappeared, open the output file **slurm-jobid.out** in your editor. There should be exactly one "Hello, World!" message.

Exercise 7: Submitting Jobs (serial or threaded application)

- Experiment with varying the number of nodes and tasks.
- Note you will need to launch the application with **srun** to actually use more than 1 cpu.

Exercise 8: Submitting Jobs (R)

- R jobs may be serial, threaded, or even MPI parallel depending on the packages used. Submit a job which will run the trivial script **hello.r** program on 1 cpu.

  Hints:

  1. Edit the script **job_script** in your exercises directory. Set:
     #SBATCH –nodes=1
     #SBATCH –ntasks=1
     application="Rscript"
     options="hello.r"
  2. Submit the job with **sbatch job_script**. The jobid is then printed.

- Repeat this using a different version of R.

Exercise 9: Array Jobs

- Submit your last job in the form of an array with indices 1-64. Use -H with sbatch to mark the array as held (so that it won't run immediately).

  Hints:
  1. Use **sbatch -H --array=1-64 job_script**
  2. Use **squeue -u userid** to see your array job. Note that **-r** reports each array element individually.

Exercise 9: Array Jobs

- Release array element 1 and allow it to run. Then release the others.

  Hints:

  1. Use **scontrol release ${SLURM_ARRAY_JOB_ID}_1**
  2. Use **squeue -u userid** again to watch what happens.
  3. Release the others with
     scontrol release ${SLURM_ARRAY_JOB_ID}
     i.e. use the array id to release the entire array.
  4. When all the jobs complete you should have 64 slurm-${SLURM_ARRAY_JOB_ID}_N.out files saying hello from various cpus on possibly multiple nodes.