Practical 4 - CSF Exercise

Overview

- Connect to CSF via RVDS
- Run a serial and parallel python job on CSF
- View the results on the CSF & iCSF

Note

- Although perfectly acceptable, it's not necessary to use the RVDS in order to connect to the CSF. Due to the CSF being a batch system and not designed for interactive use, persistent connections are not required. Therefore, using the Terminal app (Linux or Mac OSX) or a Terminal emulator application for Windows, e.g. Mobaxterm will suffice. For the sake of convenience, we will continue to use the RVDS.
- On the CSF the Linux man command shows manual pages for a command, e.g. enter man ssh to see the manual pages for ssh (use space for next page,q to quit)
- On the CSF the up and down arrows can be used to scroll through previously entered commands. Press tab while entering a command or filename to auto-complete it (this saves a lot of typing – very handy!)
- It is usually possible to paste into a terminal window, e.g. by using the middle mouse button (sometimes the right mouse button).

1.Connect to the CSF

- **Resume/Restart** the previously connected session to nyx* by opening the **X2go** software
- Open a new Terminal windows so you arrive at a **nyx*** - **Applications ---> System Tools ---> Terminal (MATE Terminal)** enter the following in order to connect to the CSF3 login node.

```
       [username@nyx* ~]$ ssh -X username@csf.itservices.manchester.ac.uk
```

On-campus

- You will need to authenticate using your 2FA device.
- If you use the Duo mobile app as your 2FA device, Enter ‘1’ at the prompt and press Enter. Once the push notification has been received via the Duo mobile app ‘Approve’ the request in order to log on.

```
$ ssh -X username@csf.itservices.manchester.ac.uk
Password:
Enter a passcode or select one of the following options:

1. Duo Push to +XX XXXX XXX555

Passcode or option (1-1):1
```

OR

- If you use a Duo fob as your 2FA device, generate a passcode with the fob, type the passcode at the prompt and press Enter

```
$ ssh -X username@csf.itservices.manchester.ac.uk
Password:
Duo two-factor login for username

Passcode: 123
```

Off-campus

If logging in from off-campus you will first need to be connected to the University VPN (GlobalProtect)

2. A serial core job on CSF

- Change directory to the **RHUMCOMPUTE/practical4** directory remember – after today, for real job runs you should create folders in scratch and run jobs from there, when they finish copy important files to home/RDS.

```
[username@login2 [csf3] ~]$ cd ~/RHUMCOMPUTE/practical4
```

- In that directory, you should see a sample jobscript entitled '**pythonExample_jobscript.txt**'. Let's take a closer look at the job script using a text editor called gedit (this is a default app, no modulefile to load):

```
[username@hlogin2 [csf3] ~]$ gedit pythonExample_jobscript.txt &
```

- You should hopefully recognise this as CSF job script. Let's make some changes.
    - Give the job a memorable same, e.g. **pythonjob_1_core**

```
      #$ -N pythonjob_1_core
```

    - Save the file

- Submit the job:

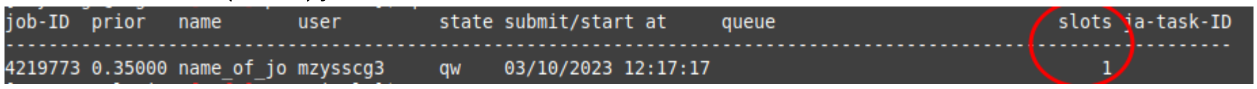
```
[username@login2 [csf3] ~]$ qsub pythonExample_jobscript.txt
```

It will provide you a jobID

- Let's monitor the job using the following command

```
[username@login2 [csf3] ~]$ qstat
```

- You should see the job either in the queue **(qw)** or running **(r)**. You should also notice that it has been assigned 1 core. This is because we have not specified a parallel environment, therefore by default it will run as a serial (1 core) job.

- 

```
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
4219773 0.35000 name_of_jo mzysscg3     qw    03/10/2023 12:17:17                                    1
```

3. Parallel jobs on CSF

- Now, let's submit a 4-core parallel job.
- Let's edit the jobscript (it's not unusual to reuse jobscripts).

```
[username@login2 [csf3] ~]$ gedit pythonExample_jobscript.txt &
```

- Let's make some changes.
    - Give the job a memorable name, e.g. **pythonjob_4_core**

```
      #$ -N pythonjob_4_core
```

  - We need to tell the batch system we want the job to run in a parallel environment, so let's add the following line to the jobscript

    ```
    #$ -pe smp.pe 4
    ```
  - Save the file
  - Submit the job to the batch system using the **`qsub`** command
  - Monitor the job using the **`qstat`** command
- If time permits submit an 8-core and 16-core job, using the same jobscript.
- Immediately edit the jobscript change the name of the job, so the output filenames are different.
- Edit the number of cores, then resubmit (don't need to wait for previous job to complete).

4. View the output

- Remember each job produces an error (**`jobname.eJobID`**) and output (**`jobname.oJobID`**) file, depending on the application it may produce more.
- Look at the output files produced - You should have two each (1x error & 1x output) for the serial (1-core), and 4 core jobs we submitted (8 in total)
- To view the files you can use the following command

  ```
  [username@login2 [csf3] ~]$ cat jobname.oJobID
  ```
- What can you tell from looking at the output files, does the time it takes the job to run reduce the more cores used?

5. At the end of today's course – Terminate your X2GO virtual desktop

- In exercise iCSF 1 we suspended the virtual desktop then resumed it, as though we were working on it later (e.g., from home) – see RVDS exercise 1 if unsure about this.
- If you really want to kill off everything running on your virtual desktop, click the Terminate icon in the X2GO control panel (see exercise 1) to shut down your virtual desktop and kill everything that is running there.
- The next time you connect to the RVDS you will get a brand---new virtual desktop – nothing will be running on it.