

Stats HPC GPU Cluster User Guide

PyTorch & TensorFlow

This guide explains how to use the system-wide PyTorch and TensorFlow Conda environments on the `srf_gpu_01` cluster.

Overview

The Department of Stats HPC has three clusters:

- Research groups: swan
- Shared CPU cluster: `srf_cpu_01`
- Shared GPU cluster: `srf_gpu_01`

At present, two preconfigured Conda environments are available on the **`srf_gpu_01`** cluster:

- `/opt/conda/envs/pytorch-2025a` (PyTorch, GPU-enabled)
- `/opt/conda/envs/tensorflow-2025a` (TensorFlow, GPU-enabled)

Connecting to the Stats HPC

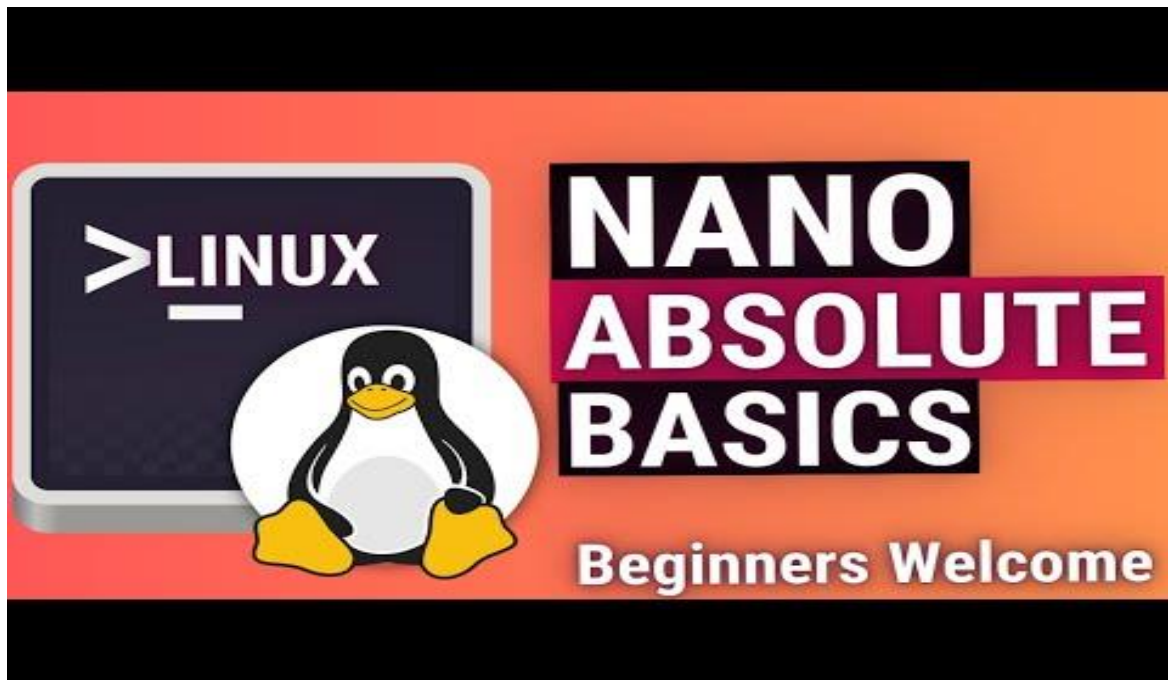
To copy your files to the Stats HPC and to connect to the HPC, please check the ***Intro HPC & Linux*** presentation slides, especially *p.8-11*.

Running your test PyTorch Slurm job on the `srf_gpu_01` cluster

Please copy the file **`test_pytorch_gpu.sbatch`** and/or **`test_tensorflow_gpu.sbatch`** to the `slurm-hn02` login node as shown in the ***Intro to HPC & Linux*** presentation slides. These are Slurm job files. Please open them and review.

You will need to use a command-line interface (CLI) text editor to open the files. If you have never used a Linux CLI text editor, nano is a great option for beginners. Here is a YouTube video that does a great job of introducing you to nano:

<https://www.youtube.com/watch?v=g2PU--TctAM>



Once you've opened the PyTorch or TensorFlow Slurm job file, you will see a number of #SBATCH lines. These are Slurm job parameters that will define how Slurm is going to run your job on the cluster. You can edit these to your requirements. For now, I recommend you only change the '--mail-user' parameter to your email address and submit the Slurm job (see below).

Explanation of #SBATCH Options

--job-name	Job name shown in queues/logs.
--mail-user	Email address to where you want to receive Slurm notifications.
--mail-type	Events to notify in the Slurm email notifications (BEGIN, END, FAIL).
--cluster	Which cluster you want to run the Slurm job (srf_gpu_01 or srf_cpu_01).
--partition	Partition/queue (e.g., standard-gpu).
--gres=gpu:1	The number of GPU you're requesting, eg 1 GPU.
--cpus-per-task	The number of CPU cores you want assigned per task.
--mem	How much memory allocation you're requesting (e.g., 4G).
--time	Maximum runtime/timelimit (HH:MM:SS).

--output

Where you want Slurm to output/write the Slurm job log file (%j = job ID).

The PyTorch test Slurm job

```
1 #!/bin/bash
2 #SBATCH --job-name=test_pytorch_gpu
3 #SBATCH --mail-user=dayo.ntwari@stats.ox.ac.uk
4 #SBATCH --mail-type=BEGIN,END,FAIL
5 #SBATCH --cluster=srf_gpu_01
6 #SBATCH --partition=standard-gpu
7 #SBATCH --odelist=swangpu24.cpu.stats.ox.ac.uk
8 #SBATCH --gres=gpu:1
9 #SBATCH --cpus-per-task=2
10 #SBATCH --mem=4G
11 #SBATCH --time=00:10:00
12 #SBATCH --output=myslurmlogs/test_pytorch_gpu_%j.out
13
14 # Load the Conda module
15 module load conda
16
17 # Source conda.sh for non-interactive shell
18 source /opt/conda/etc/profile.d/conda.sh
19
20 # Activate PyTorch environment
21 conda activate /opt/conda/envs/pytorch-2025a
22
23 # Display GPU information
24 echo "Running on host: $(hostname)"
25 echo "CUDA_VISIBLE_DEVICES: $CUDA_VISIBLE_DEVICES"
26 nvidia-smi
27
28 # Run a PyTorch CUDA test
29 python - <<'EOF'
30 import torch
31 print("PyTorch version:", torch.__version__)
32 print("CUDA available:", torch.cuda.is_available())
33 if torch.cuda.is_available():
34     print("Using GPU:", torch.cuda.get_device_name(0))
35     x = torch.rand(1000, 1000, device="cuda")
36     y = torch.rand(1000, 1000, device="cuda")
37     print("Tensor sum (GPU):", (x + y).sum().item())
38 else:
39     print("Running on CPU.")
40 EOF
41
42 sleep 300
```

After defining the Slurm job parameters in the #SBATCH lines, follow the Conda and PyTorch part of the Slurm job.

l.14 loads the cluster's system-wide Conda module and then sources its config in l.18.

After that, l21 activates your PyTorch environment inside your Conda. If you are new to using Conda and PyTorch in this way, I recommend you do not change l.15-21. Ideally,

the system Conda + PyTorch is sufficient for your needs. If it's missing anything or not working as expected, please let me know.

l.23-25 are self-explanatory, I think. ``echo`` is the print command for Linux. And l.26 the `nvidia-smi` command, is a common way for you to query information about the GPUs on the cluster. You will find the output of all these commands in your Slurm job's log file (see below).

And l.29-40 is just a basic PyTorch script to check whether PyTorch detects the GPU. Later on, you would either paste your PyTorch script into the Slurm job file, or just replace l.29-40 with the path to the Python file, eg

```
python my-pytorch-script.py
```

The `sleep` command in l.42 is not necessary for you to use, going forward. I just included it, to allow you the time to view your job running in the queue. The PyTorch script I used here is too basic to require much computing power, and so the cluster would complete the job before you had the time to run the `squeue` command (see below). That is why I've included ``sleep 300``, which means "Do nothing, and just wait for 300 seconds."

Submitting the Slurm job

On the login node (eg **slurm-hn02**) type the Slurm batch job submission command into the terminal:

For **PyTorch**:

```
sbatch test_pytorch_gpu.sbatch
```

For **TensorFlow**:

```
sbatch test_tensorflow_gpu.sbatch
```

When you press ENTER, if everything went well, the terminal will return a message like:

*Submitted batch **job 9616** on cluster **srf_gpu_01***

Each job is assigned a unique job ID by Slurm. In the above example, the job ID is 9616.

Viewing the Slurm job queue

Now that your job is running, you can view its status in the job queue:

```
squeue --long -M srf_gpu_01
```

```
slurm-hn02{dntwari}% squeue --long -M srf_gpu_01
Fri Jul 25 12:44:39 2025
CLUSTER: srf_gpu_01

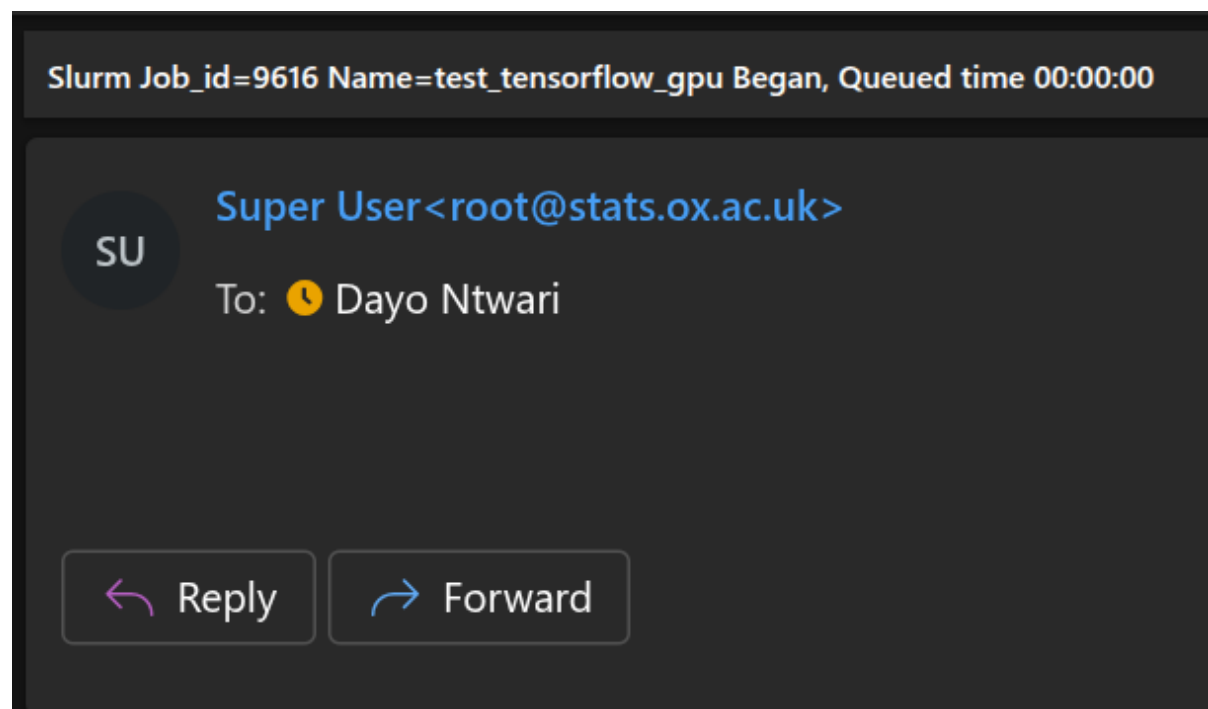
```

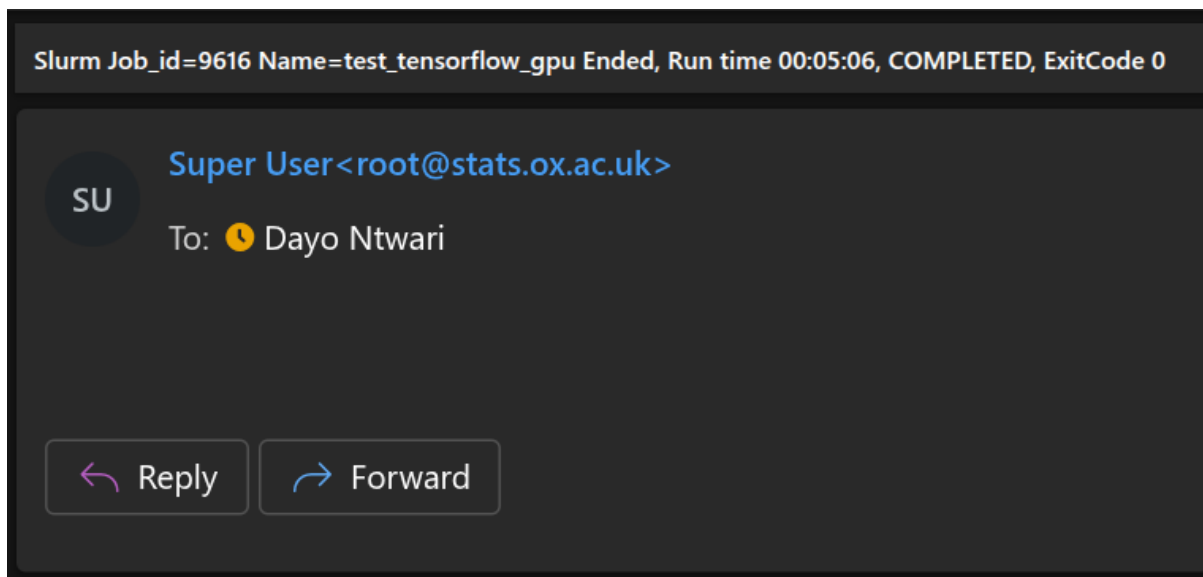
JOBID	PARTITION	NAME	USER	STATE	TIME	TIME_LIMI	NODES	NODELIST(REASON)
9615	high-bigb	alphadoc	prat	RUNNING	12:41	7-00:00:00	1	zizgpu06.cpu.stats.ox.ac.uk
9592	high-opig	guidedoc	buttensc	PENDING	0:00	14-00:00:00	1	(Resources)
8659	high-opig	4_a3a	ifashe	RUNNING	5-16:55:25	14-00:00:00	1	nagagpu04.cpu.stats.ox.ac.uk
8813	high-opig	2_magic	ifashe	RUNNING	4-20:28:42	14-00:00:00	1	nagagpu04.cpu.stats.ox.ac.uk
8779	high-opig	1_magic	ifashe	RUNNING	5-13:32:49	14-00:00:00	1	nagagpu04.cpu.stats.ox.ac.uk
9566	high-opig	refined_	broster	RUNNING	20:20:41	14-00:00:00	1	nagagpu04.cpu.stats.ox.ac.uk
9616	standard-	test_ten	dntwari	RUNNING	0:49	10:00	1	swangpu24.cpu.stats.ox.ac.uk
9271	standard-	alphadoc	prat	RUNNING	2-20:23:06	7-00:00:00	1	greystrich.stats.ox.ac.uk

```
slurm-hn02{dntwari}%
```

In the screenshot you can see Slurm job 9616 running on the srf_gpu_01 cluster. If you see something similar for your job, congratulations, you've just submitted your first Slurm job to the Department of Statistics HPC!

If you set your email in the Slurm job file, you will have received two emails from the HPC. One at the beginning of the job, when you submitted the job and Slurm ran it on the HPC, and one at the end of the job, when the job completed.





Reviewing your job results

Once this test Slurm job has completed, you can open (eg with nano) the Slurm job log file you defined in the job file. In this example, it's located in the folder *myslurmlogs/* and is called *test_pytorch_gpu_9616.out* <-- is the Slurm job ID I received when I ran the job at the time of writing this tutorial. Your individual Slurm job IDs will be different and unique.

```
$ nano ~/myslurmlogs/test_pytorch_gpu_.....out
```

Output of test_pytorch_gpu_9616.out:

```
Running on host: swangpu24.cpu.stats.ox.ac.uk
CUDA_VISIBLE_DEVICES: 0
Fri Jul 25 13:09:28 2025
```

NVIDIA-SMI 575.57.08				Driver Version: 575.57.08				CUDA Version: 12.9			
GPU	Name	Perf	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr. ECC	GPU-Util	Compute M.	MIG M.	
Fan	Temp		Pwr:Usage/Cap		Memory-Usage						
0	NVIDIA RTX 6000 Ada Gene...		On	00000000:0D:00.0	Off		Off				
30%	30C	P8	21W / 300W	1MiB / 49140MiB		0%	Default			N/A	

```

Processes:
GPU  GI  CI          PID  Type  Process name                      GPU Memory
   ID  ID                                     Usage
=====
No running processes found
PyTorch version: 2.5.1
CUDA available: True
Using GPU: NVIDIA RTX 6000 Ada Generation
Tensor sum (GPU): 1000099.875

```

The first two lines in the above screenshots are the outputs of:

l.23 # Display GPU information

l.24 echo "Running on host: \$(hostname)"

l.25 echo "CUDA_VISIBLE_DEVICES: \$CUDA_VISIBLE_DEVICES"

And then from “Fri Jul 25...” to “No running processes found..” you have the output of the nvidia-smi command, which is showing the GPU information for the cluster node this Slurm job ran on. Which in this case was swangpu24.cpu.stats.ox.ac.uk

And the remaining lines are from PyTorch.

If you need similar guides or software installations on the HPC, please let me know at ithelp@stats.ox.ac.uk