

# Introduction Lecture and GPU sources

---

*Summary of the GPU resources available to Imperial Students taking Deep Learning module*

*Author: Anton Zhitomirsky*

## Contents

<b>1</b>	<b>CSG Guides</b>	<b>2</b>
<b>2</b>	<b>Basic SSH into the labmachine</b>	<b>2</b>
2.1	Setting up the public key to not type in password . . . . .	2
2.2	Setting up the tunnel from shell1 – another machine . . . . .	2
2.3	Checking if resource being used . . . . .	2
<b>3</b>	<b>SLURM [reference]</b>	<b>2</b>
3.1	SSH into the gpu cluster . . . . .	2
3.2	Using CUDA . . . . .	3
3.3	Template Example bash submission script . . . . .	3
3.4	Scheduling job . . . . .	3
3.5	Creating a SLURM job to act as a VM . . . . .	3
3.5.1	In Terminal 1 . . . . .	4
3.5.2	In Terminal 2 . . . . .	4
3.5.3	In Jupyter . . . . .	4
<b>4</b>	<b>SageMaker</b>	<b>4</b>
<b>5</b>	<b>PaperSpace</b>	<b>4</b>
<b>6</b>	<b>Python virtual environment</b>	<b>4</b>
6.1	Where to store data . . . . .	4
6.1.1	Why do you need a python virtual environment? . . . . .	5
6.1.2	Creating a Python Virtual Environment in /vol/bitbucket . . . . .	5
6.1.3	Why activate and deactivate? . . . . .	5

# 1 CSG Guides

CSG source

## 2 Basic SSH into the labmachine

### 2.1 Setting up the public key to not type in password

Copy the public key on your machine over to the authorized hosts on the other machine to avoid typing in your password every time you ssh. [\[source\]](#)

```
1 ssh-copy-id -i ~/.ssh/id_ed25519.pub <usrname>@<host>
```

### 2.2 Setting up the tunnel from shell1 – another machine

Create an alias on your home machine for the shell you'll use [\[source1\]](#) [\[source2\]](#)

```
1 Match host shell*
2     Hostname %h.doc.ic.ac.uk
3     User az620
```

To begin tunnelling through with ssh [\[source\]](#) and find your favourite [\[workstation\]](#) and set up tunnel with [\[source3\]](#)

```
1 Match host texel*
2     Hostname %h.doc.ic.ac.uk
3     User az620
4     ProxyJump shell1
```

### 2.3 Checking if resource being used

```
1 nvidia-smi && top htop
```

## 3 SLURM [\[reference\]](#)

The department has a pool of GPUs for deep learning tasks. To reduce complexity of scheduling, [slurm](#) is a system that schedules tasks into these resources.

### 3.1 SSH into the gpu cluster

In the `~/.ssh/config` file, add the line below and ssh into it to schedule jobs. This should only be used for scheduling jobs into the slurm scheduler.

```
1 # should be either 2 or 3
2 Match host gpucluster*
3     Hostname %h.doc.ic.ac.uk
4     User az620
5     ProxyJump shell1
```

Then submit a pre-existing script using the `sbatch` command. The output will be stored, by default, in the root of your `~/` directory, with the filename `slurm20-{xyz}.out`.

See Section [6](#) for guide on python virtual environment.


## 3.2 Using CUDA

Many jobs will make use of the [Nvidia CUDA toolkit](#), multiple versions of which are at /vol/cuda. If there ever appears a need to use this toolkit then in the submission bash script add:

```
1 ./vol/cuda/12.0.0/setup.sh # if you're using version 12.0.0
```

Care that you choose a version with which your tensorflow or pytorch are compatible with.

## 3.3 Template Example bash submission script

 This example assumes you have followed the previous steps and installed a python environment (using virtualenv, extra lines may be needed using miniconda, check the example script further below) as directed. Please adjust paths if you have an existing python environment, or if you already load your environment in ~/.bashrc (note: sbatch does not load ~/.bashrc, source it as per example script) . Do not uncomment #SBATCH lines, keep them as below, make sure the #SBATCH directives are directly after #!/bin/bash

Remember to make the submission script<sup>1</sup>. executable with `chmod +x <script_name>.sh`

```
1 #!/bin/bash
2 #SBATCH --gres=gpu:1
3 #SBATCH --mail-type=ALL # required to send email notifications
4 #SBATCH --mail-user=<your_username> # required to send email notifications
5 export PATH=/vol/bitbucket/${USER}/myenv/bin/:$PATH
6 # the above path could also point to a miniconda install
7 # if using miniconda, uncomment the below line
8 # source ~/.bashrc
9 source activate
10 source /vol/cuda/12.0.0/setup.sh
11 /usr/bin/nvidia-smi
12 uptime
```

## 3.4 Scheduling job

- `ssh gpucluster2`
- `cd /vol/bitbucket/${USER}`
- `sbatch <path to executable>.sh`
- `less slurm-XYZ.out` saves the output.
- `squeue -l` for queue of running jobs
- `scancel <job ID>` to delete slurm job

## 3.5 Creating a SLURM job to act as a VM

```
1 ssh texel10 # DON'T ssh into a gpucluster machine, it won't work
2 cd /vol/bitbucket/az620
3 python -m venv /vol/bitbucket/az620/dlenv
4 source /vol/bitbucket/az620/dlenv/bin/activate
5 pip3 install ipykernel
6 pip3 install jupyterlab
7 pip3 install jupyterhub
8 # Add CUDA to the path from /vol/cuda/{version}
```

<sup>1</sup>example can be found at /vol/bitbucket/shared/slurmseg.sh

### 3.5.1 In Terminal 1


```
1 ssh gpucluster2
2 salloc --gres=gpu:1
3 queue | grep az620
```

### 3.5.2 In Terminal 2

If you have defined a config entry for `shell*` then replace accordingly with `az620@shell13.doc.ic.ac.uk`  
`↳ shell13.`

```
1 ssh -t -L 10001:localhost:10001 az620@shell13.doc.ic.ac.uk "/vol/linux/bin/slurm.sshjob.sh -g -w ~/ -p 10001 -e /
  ↳ vol/bitbucket/{USER}/dlenv"
```

`http://localhost:10001/lab?token={from terminal}`

 remember to: `scancel <your job id>` back in Terminal 1

### 3.5.3 In Jupyter

```
1 ssh gpucluster1
2 salloc --gres=gpu:1
3 queue
4 # in VS Code:
5 ssh -A -J {USER}@shell3.doc.ic.ac.uk {USER}@kingfisher.doc.ic.ac.uk
```

## 4 SageMaker

Access [SageMaker](#) and follow instructions in slides

## 5 PaperSpace

 Remember to close the notebook when finished

Access [Paperspace](#)

After access granted, you can access [this](#) and use paperspace as your location of the kernel to execute super fast.

## 6 Python virtual environment

### 6.1 Where to store data

Data is stored in the `/vol/bitbucket/${USER}`. It can be created with `mkdir -p ...` if it doesn't exist. You should create personal folders here. Otherwise, if you store data in the home directory you risk going over the quota limit [\[source\]](#)<sup>2</sup>. `/vol/bitbucket` should only be used for temporary storage of material that can be regenerated (downloads or compilations). To see where you are storing disk space use `/vol/linux/bin/usage` (disk space) and `/vol/linux/bin/nfiles` (number of files and directories) [\[source\]](#).

<sup>2</sup>you can check disk quota with `quota -Q`

### 6.1.1 Why do you need a python virtual environment?

Python isn't good at dependency management. pip places all external packages into site-packages/ in the base Python installation. By creating a virtual environment you avoid system pollution (since these packages mix with system-relevant packages causing unexpected side effects)

### 6.1.2 Creating a Python Virtual Environment in /vol/bitbucket

👉 Use a lab PC to prepare the Python environment; don't use the gpucluster machine for this. You may encounter 'out of space' errors.

Its advised that you create Python virtual environments on /vol/bitbucket. Steps taken from [\[source\]](#)

1. create /vol/bitbucket/\${USER}

2. create virtual environment

```
1 export PENV=/vol/bitbucket/${USER}/myenv
2 python3 -m virtualenv $PENV
3 ls -al $PENV
```

3. activate your VE:

```
1 source $PENV/bin/activate
2 which pip
3 [should say: /vol/bitbucket/${USER}/myenv/bin/pip]
4 which python
5 [should say: /vol/bitbucket/${USER}/myenv/bin/python]
6 which python3
7 [should say: /vol/bitbucket/${USER}/myenv/bin/python3]
```

4. install packages and verify with `which <module name>` that it is saved under /vol/bitbucket/\${USER}/myenv/

5. Or prepare a package requirements file `requirements.txt` which specifies a list of packages (optionally with specific version constraints) and install with `pip install -r requirements.txt`

6. once you're done working with the virtual environment you can deactivate it with `deactivate`

7. each time you login to a specific machine redo the activate stage with `source /vol/bitbucket/${USER}/myenv/` (this can be appended to the end of the `~/.bash_profile` in home dir.)

### 6.1.3 Why activate and deactivate?

Activating it gives us a new path for the python executable because, in an active environment, the \$PATH environment variable is slightly modified.