



THE UNIVERSITY  
*of* EDINBURGH

# Demos for the GPU teaching cluster and PyTorch workflow

Machine Learning Systems

Week 2 Q&A Lecture

Yinsicheng Jiang, Yeqi Huang, Luo Mai

# Last week

## Async Kernel Dispatcher – What is a Kernel in PyTorch?

A kernel in PyTorch is a user-defined low-level function that processes tensor data by performing operations like element-wise computation and matrix transformations.

### 1. Define a Custom Autograd Function (Python):

- Implement the forward and backward methods to define the operation and its gradient, respectively.
- This is ideal for prototyping.

```
import torch

class MyCustomKernel(torch.autograd.Function):
    @staticmethod
    def forward(ctx, input):
        ctx.save_for_backward(input)
        return input * 2 # Example: Simple scaling operation

    @staticmethod
    def backward(ctx, grad_output):
        input, = ctx.saved_tensors
        return grad_output * 2 # Gradient of the operation
```

### 2. Write a C++/CUDA Extension for Performance:

- Define the kernel in C++/CUDA and compile it as an extension.
- This approach is suited for performance-critical applications.

### 3. Optional: Use PyTorch JIT for Just-in-Time Compilation:

- For operations that don't need full C++/CUDA customization but require some performance improvements, PyTorch's torch.jit can optimize the kernel directly in Python.

```
#include <torch/extension.h>

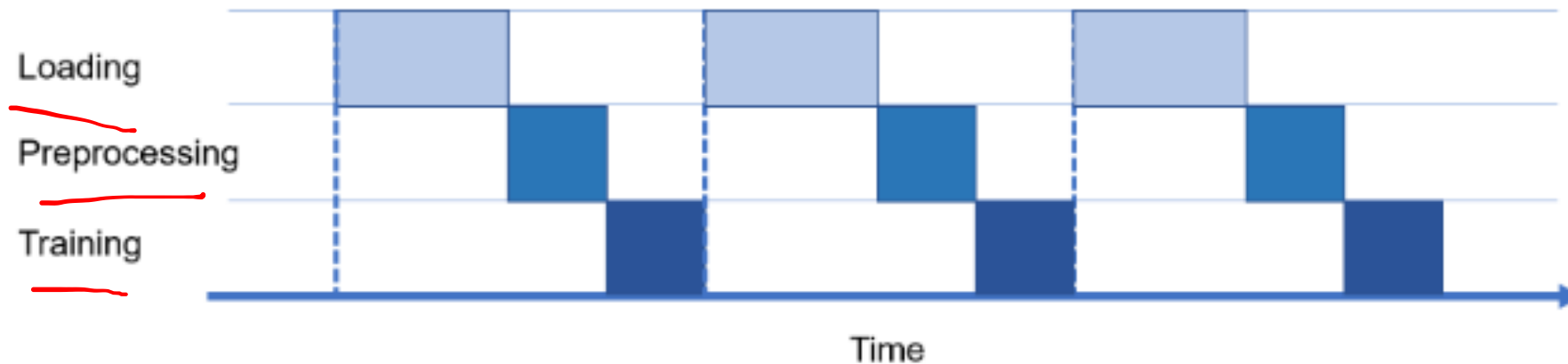
torch::Tensor my_custom_kernel(torch::Tensor input) {
    return input * 2; // Example operation
}

PYBIND11_MODULE(TORCH_EXTENSION_NAME, m) {
    m.def("my_custom_kernel", &my_custom_kernel, "My Custom Kernel");
}
```

# Last week

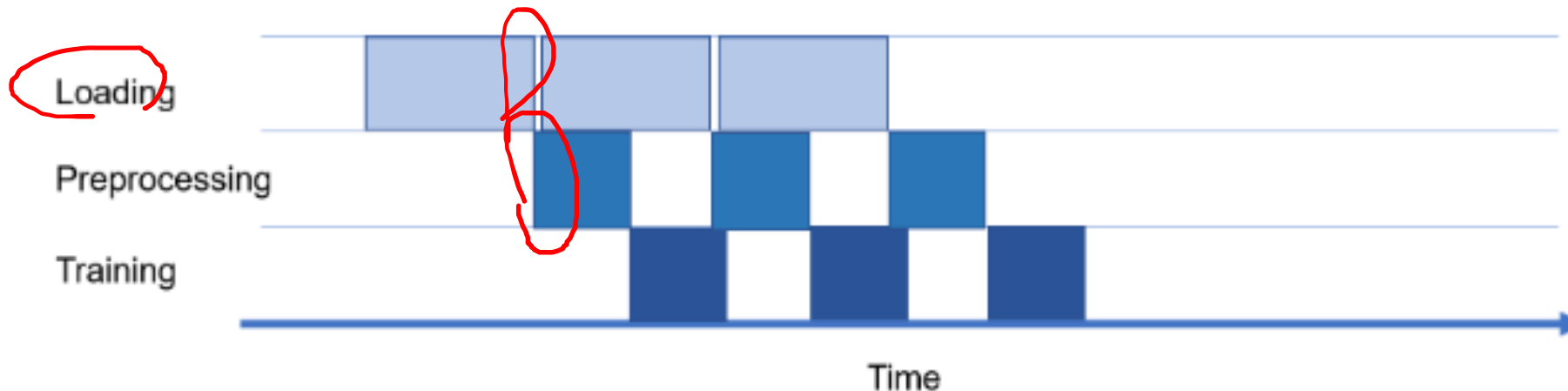
## Asynchronously scheduling of Kernels

Synchronous kernel execution mechanism:



The main thread is blocked for waiting the previous workload.

Asynchronous kernel execution mechanism:



The main thread becomes **unblocked** after the initial Loading kernel and immediately begins scheduling kernels asynchronously as threads are released.

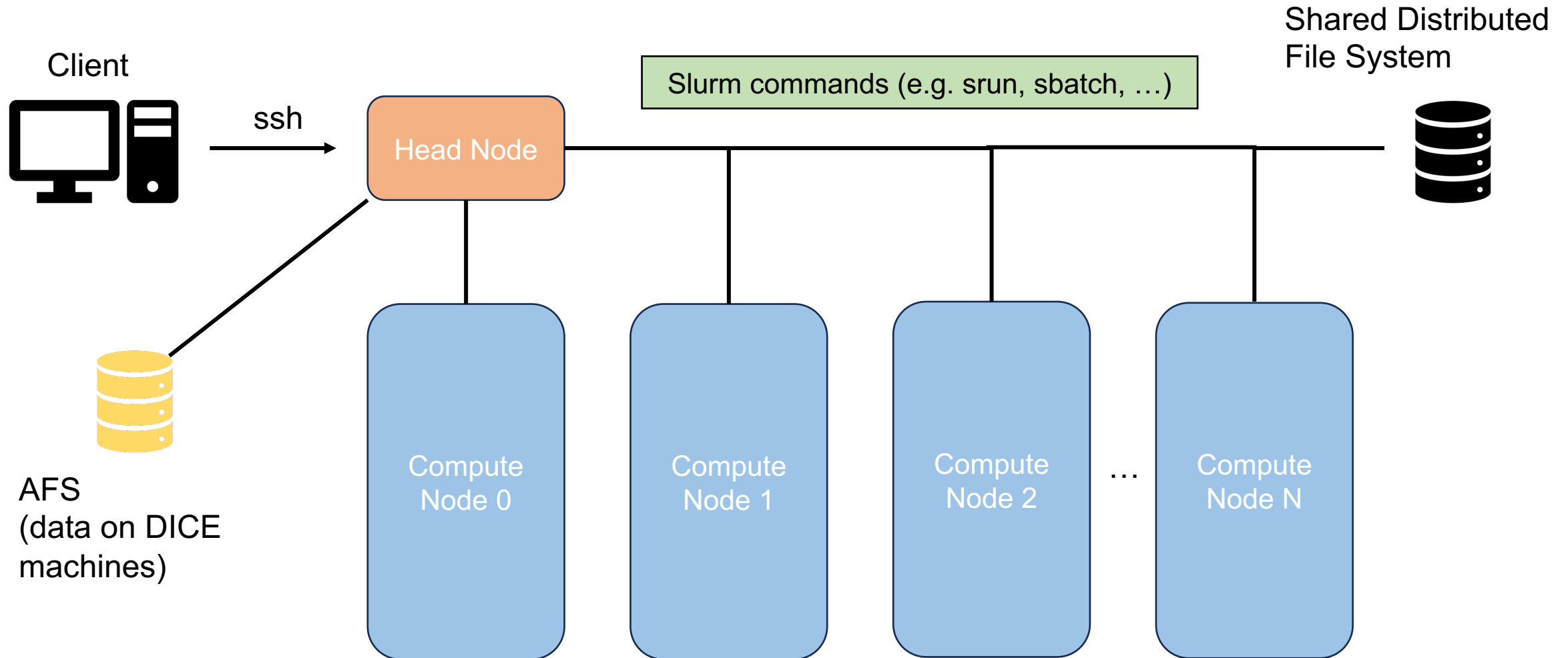
# Last week

## Suggestions:

- Slower teaching pace:
  - We will make every effort to ensure clarity and adjust our speaking pace to help students follow along more easily.
- Offline reading
  - Yes, we are currently preparing it and plan to announce it by next Tuesday.
- Where are the course materials?
  - All teaching materials will be in the “Course Schedule and Materials” section on **Learn** of MLS course.
- Grouping deadline
  - By the end of week 3 - 31 Jan, 2025

# Teaching GPU Cluster

## Teaching Cluster Overview



# Teaching GPU Cluster

## Basic Info:

The teaching cluster has over 120 GPUs in 20 servers (landonia[01-25]) with:

- 100 NVIDIA GTX 1060 6GB GPUs,
- 8 NVIDIA RTX A6000 48GB GPUs,
- a small number of NVIDIA TITAN-X 12GB GPUs.

## Usage:

Access through **ssh** and use **SLURM** commands.

1. Access the cluster's head node via:

- Connect to the Informatics VPN or use the DICE machine and then run: `ssh <YOUR_UUN>@mlp.inf.ed.ac.uk`
- You can remotely connect to your DICE machine using the command **below** if you are connected to the Informatics VPN or the eduroam Wi-Fi. Once you have accessed the DICE machine, run the command mentioned **above**.

`ssh <YOUR_UUN>@ssh.inf.ed.ac.uk`

- You can find information about how to connect to Informatics VPN at:  
<https://computing.help.inf.ed.ac.uk/openvpn>

# Teaching GPU Cluster

## Usage (continue):

After successfully connecting to the head node, your terminal will display the following:

```
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-130-generic x86_64)

This is a cluster head node please do not run compute intensive processes here
this node is intended to provide an interface to the cluster only
Please nice any long running processes
-----
Looking for tips? https://auth.computing.help.inf.ed.ac.uk/cluster-tips
Jan 2025: Note that some GRES have changed to swap dashes to underscores.

Last login: Mon Jan 13 15:45:11 2025 from openvpn-125-012.inf.ed.ac.uk
[uhrtred]s2020153: █
```

## 2. Run your code with SLURM command:

There are two ways you can run your code:

- Interactive job - allow you to interact directly with the allocated compute resources, good for debugging and testing code.
- Batch job - run the task automatically based on a pre-written script (.sh file) without interaction with the compute node and constant user attention, good for long-running tasks.

# Teaching GPU Cluster

## Usage (continue):

E.g. Let's say you want to run the code utilizing 1 GPU.

Interactive jobs: `srun --gres=gpu:1 --pty bash`

After executing this command, run “nvidia-smi”, and you will see output similar to the following:

```
((base) [landonia19]s2020153: nvidia-smi
Mon Jan 13 17:48:21 2025

+-----+
| NVIDIA-SMI 550.127.08                  Driver Version: 550.127.08          CUDA Version: 12.4         |
+-----+-----+
| GPU   Name                               Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf              Pwr:Usage/Cap |      Memory-Usage | GPU-Util  Compute M. |
|                               |                      |              MIG M. |
+-----+-----+
|    0  NVIDIA GeForce GTX 1060  6GB      Off   | 00000000:03:00.0 Off |                  N/A |
| 24%    30C    P8              5W / 120W | 0MiB / 6144MiB |      0%      Default |
|                               |                      |                  N/A |
+-----+-----+

+-----+
| Processes:                               |
|  GPU   GI    CI          PID    Type    Process name                  GPU Memory |
|          ID    ID                                   |             Usage |
+-----+-----+
| No running processes found                |                    |
+-----+
```



# Teaching GPU Cluster

## Usage (continue):

E.g. Let's say you want to run the code utilizing 1 GPU.

Batch jobs: `sbatch --gres=gpu:1 test.sh`

After executing this command, you will see “Submitted batch job [ID]”.

The output file will be saved as `slurm-[ID].out` by default in the directory where you call sbatch. You can set the output directory and file names by using the `--output` option with the sbatch command.

test.sh:

```
#!/bin/bash
```

```
echo I love Machine Learning Systems.  
pwd
```

**You must include this**



```
[uhtred]s2020153: sbatch --gres gpu:1 test.sh  
Submitted batch job 1945143  
[uhtred]s2020153: cat slurm-1945143.out  
I love Machine Learning Systems.  
/home/s2020153
```

# Teaching GPU Cluster

## Usage (continue):

Since servers are equipped with different types of GPUs, you can specify the type of GPU you wish to use:

E.g. You want to have 1 NVIDIA Titan X GPU:

```
srun --gres=gpu:titan_x:1 --pty bash
```

```
sbatch --gres=gpu:titan_x:1 test.sh
```

```
[uhtred]s2020153: srun --gres=gpu:titan_x:1 --pty bash
srun: job 1944884 queued and waiting for resources
srun: job 1944884 has been allocated resources
[landonia08]s2020153: nvidia-smi
Mon Jan 13 15:46:28 2025
```

NVIDIA-SMI 550.127.08				Driver Version: 550.127.08			CUDA Version: 12.4		
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute	M.		
0	NVIDIA GeForce GTX TITAN X	Off	00000000:07:00.0	Off			N/A		
22%	27C	P8	16W / 250W	0MiB / 12288MiB	0%	Default	N/A		

You are allowed to request a maximum of 8 GTX 1060 GPUs, 4 Titan X GPUs, 1 Titan X Pascal GPU, or 2 A6000 GPUs at a time.

**NOTE: Please allocate resources according to your specific requirements. Avoid over-allocating to ensure resources are available for others!**

# Teaching GPU Cluster

## Usage (continue):

Other useful SLURM commands:

Check all available GPU types: `scontrol show node | grep gpu`

```
Gres=gpu:titan_x_pascal:1(S:0),gpu:titan_x:2(S:0),gpu:gtx_1060:2(S:1)
CfgTRES=cpu=12,mem=96000M,billing=12,gres/gpu=5
Gres=gpu:gtx_1060:8(S:0-1)
CfgTRES=cpu=12,mem=96000M,billing=12,gres/gpu=8
```

Check current SLURM job status: `squeue`

```
[[uhtred]s2020153: squeue
      JOBID PARTITION    NAME    USER  ST       TIME  NODES NODELIST(REASON)
      1563085 General_U collect_ s2263903 PD       0:00      1 (BadConstraints)
      1944830 PGR-Stand ilcc_pre s2148449 PD       0:00      1 (Resources)
```

Cancel a job: `scancel <job_id>`


For more details about the school cluster and running SLURM commands, please refer to:  
<https://computing.help.inf.ed.ac.uk/teaching-cluster>.

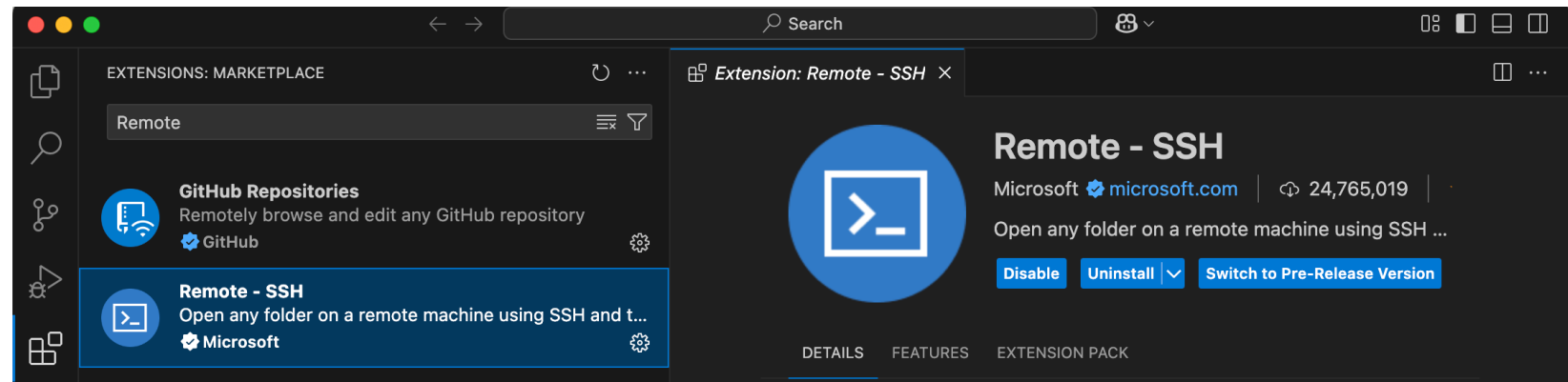
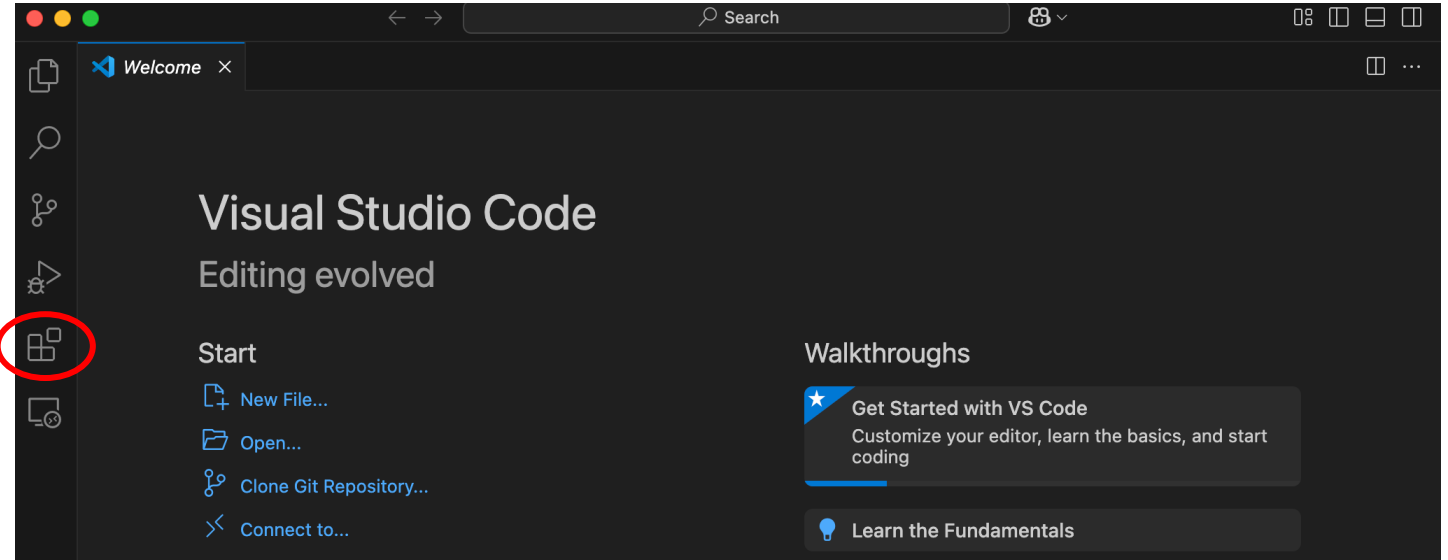
**IMPORTANT:** Please read the **Files and Backups** section and **GPU cluster tips** in this link carefully, as it contains crucial information about file storage and can help prevent data loss.

# Teaching GPU Cluster

## How to write codes in the teaching cluster?

### VSCode!

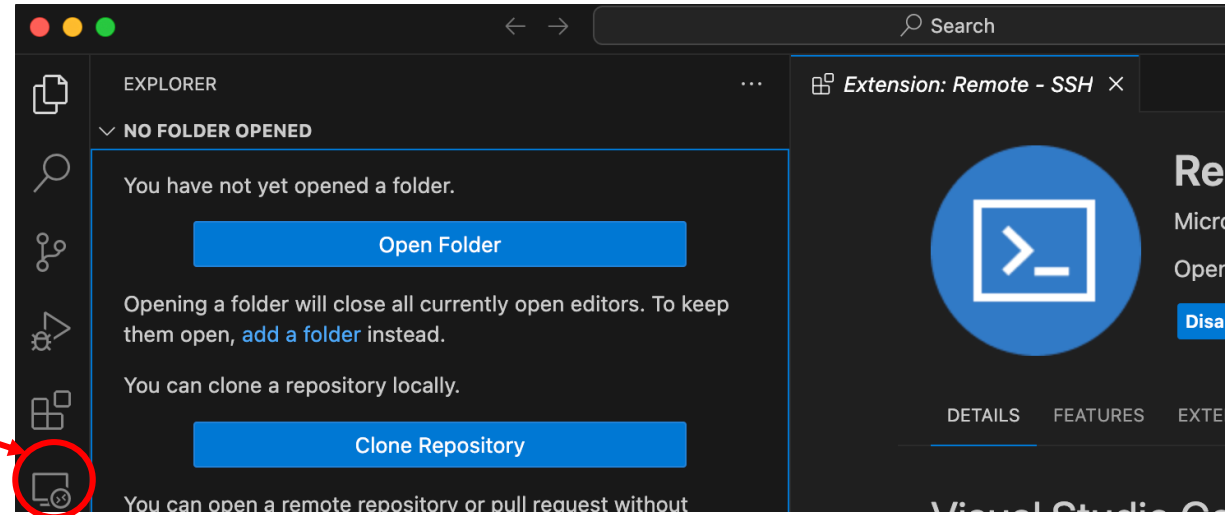
1. Open your VScode
2. Go to Extensions 
3. Search for the keyword "Remote" and install the "Remote - SSH" extension.



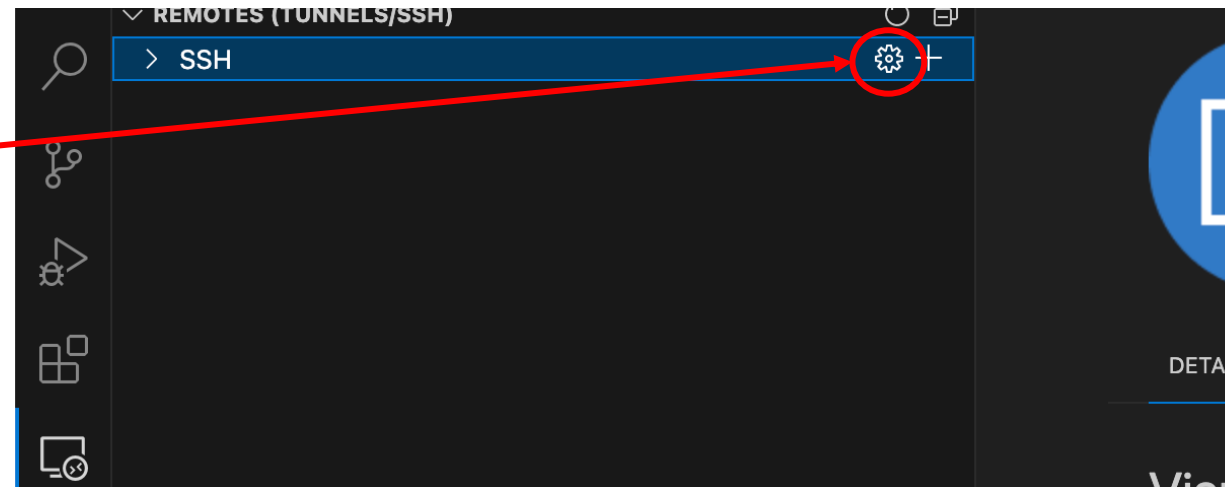
# Teaching GPU Cluster

## How to write codes in the teaching cluster? (Continue)

4. Once the extension is installed, the REMOTES icon will appear on the left sidebar.



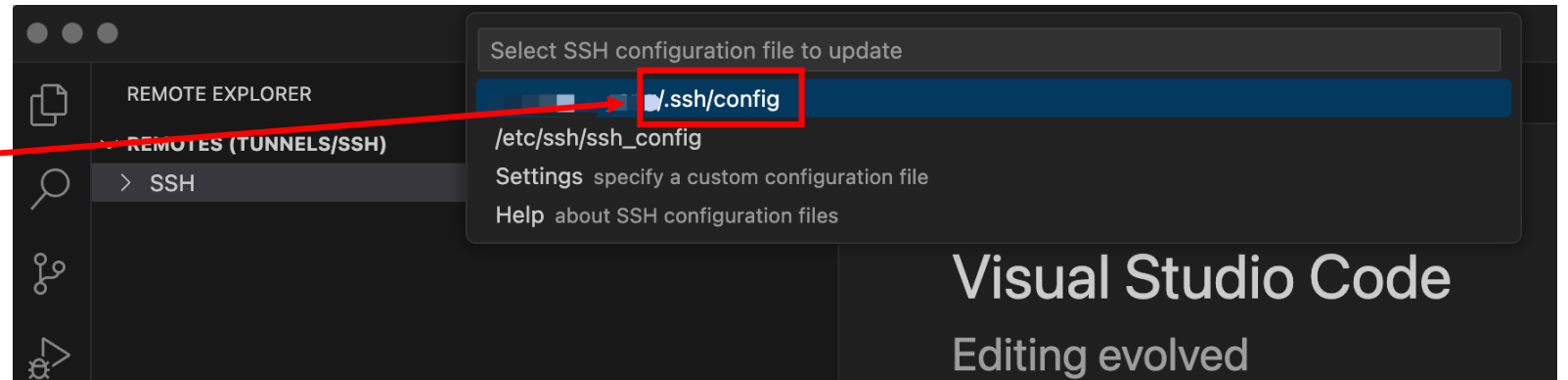
5. Click on the REMOTES icon, then select the Config icon located to the right of "SSH."



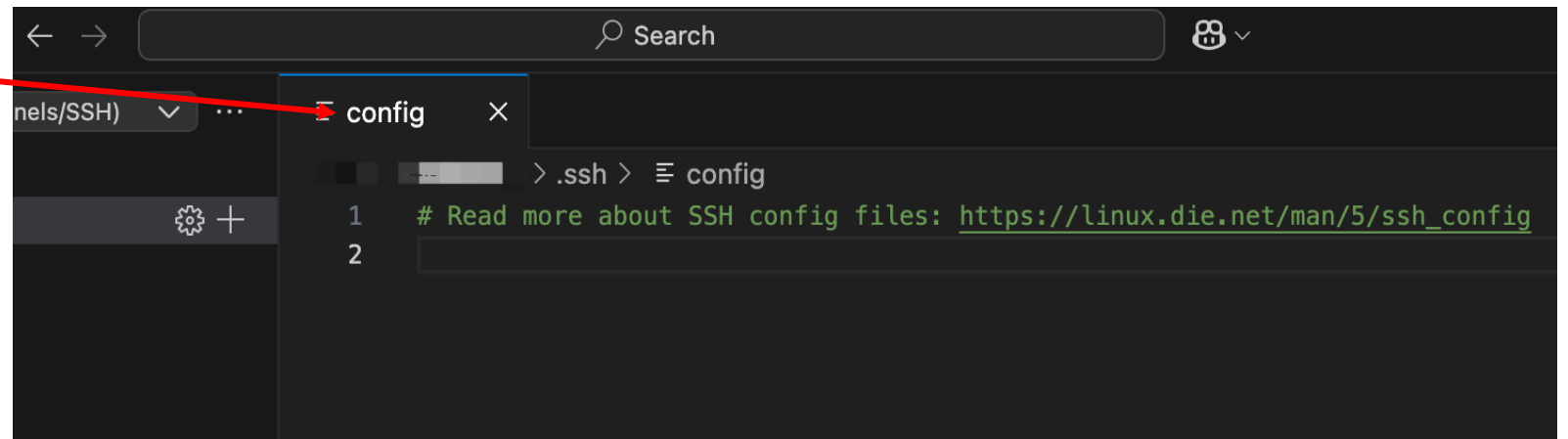
# Teaching GPU Cluster

## How to write codes in the teaching cluster? (Continue)

6. After clicking the Config icon, choose the configuration file with a path ending in `.ssh/config`.



7. You will see a page like this.



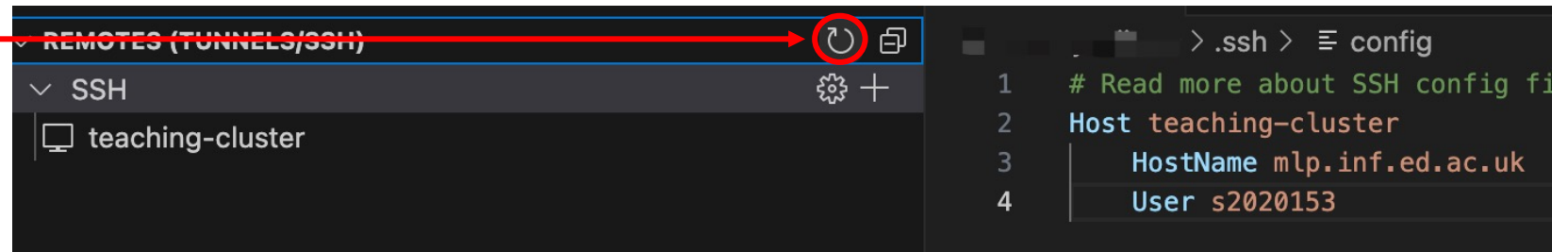
# Teaching GPU Cluster

## How to write codes in the teaching cluster? (Continue)

8. In this page, write the SSH configuration text and **save it**:

```
Host teaching-cluster  
    HostName mlp.inf.ed.ac.uk  
    User <YOUR_UUN>
```

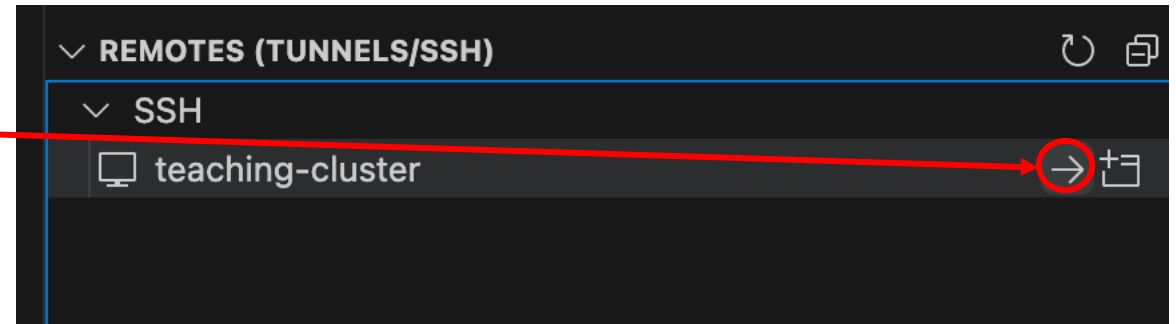
9. Refresh the REMOTES page, and a screen similar to this will appear.



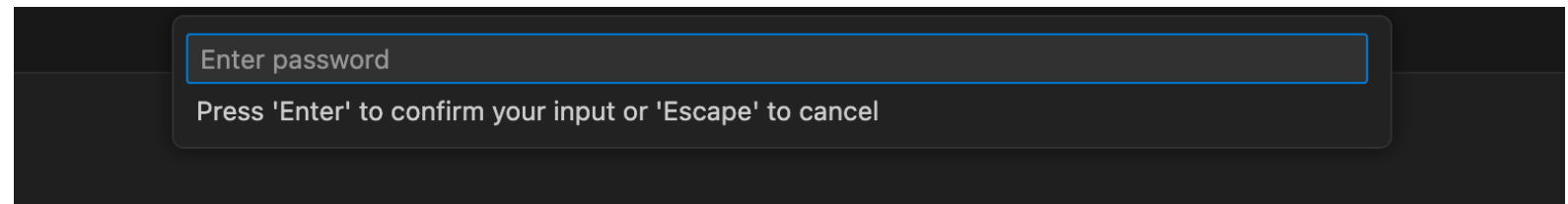
# Teaching GPU Cluster

## How to write codes in the teaching cluster? (Continue)

10. **Connect to the Informatics VPN or Use the DICE machine**, then click the right arrow icon to connect to the teaching cluster's head node:



11. Enter the password of your DICE account

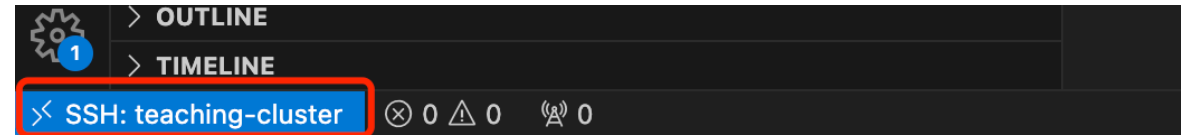




# Teaching GPU Cluster

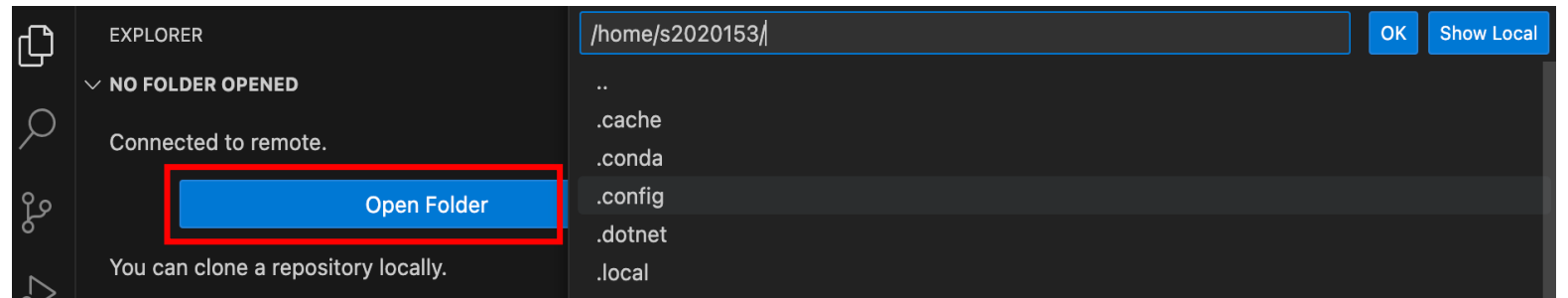
## How to write codes in the teaching cluster? (Continue)

12. If this screen appears, congratulations! You have successfully connected to the teaching cluster's head node.



13. Go to your home directory by clicking "Open Folder", and select your home directory (/home/<YOUR\_UUN>). Click OK.

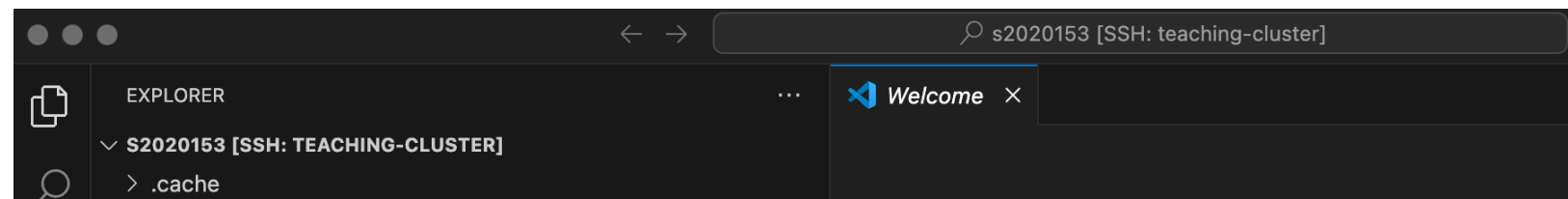
If password required, just type your password of the DICE account again.



# Teaching GPU Cluster

## How to write codes in the teaching cluster? (Continue)

14. If you can see this screen, you are now ready to write your code in the teaching cluster.

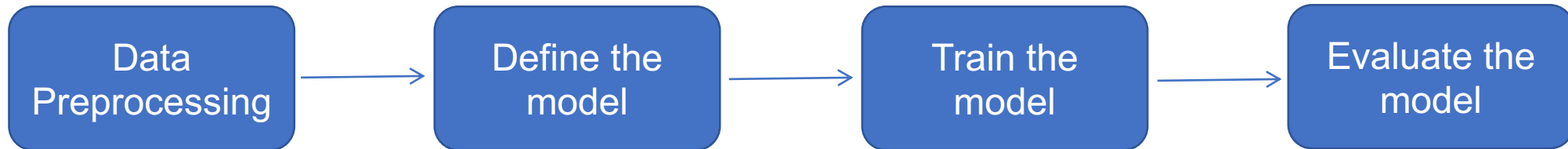


### IMPORTANT NOTES:

1. Any changes you make in VSCode, such as writing code, creating or deleting files and folders, will automatically synchronize with the cluster. That's why we recommend it 😊. **Do not forget to save your changes**
2. Only install your environment and write your code on the head node. Actions such as Git operations and similar tasks should also be performed on the head node, as they will not work on the compute nodes.
3. Only run your code on the compute nodes using `srun` or `sbatch`, as the head node has limited computing resources and does not have GPUs. Running `torch.cuda.is_available()` on the head node will return `False`.

If you have further issues about accessing the teaching cluster, please contact the Computing Support Team:  
<https://computing.help.inf.ed.ac.uk/>

# Pytorch Workflow



Our demo showcases the following steps:

- Generate a binary classification dataset using scikit-learn.
- Split the dataset into training and testing partitions.
- Convert the training and testing sets into TensorDataset objects.
- Load the datasets into train\_dataloader and test\_dataloader, respectively.
- Implement a simple 2-layer MLP model.
- Train the model using the training set.
- Evaluate the model on the testing set.

The demo can be found in the pytorch-demo/ folder of the repository:

<https://github.com/ed-aisys/edin-mls-25-spring>.

Please refer to the README file within the pytorch-demo/ directory for detailed instructions.

# Q&A





THE UNIVERSITY  
*of* EDINBURGH

Thank you very much

