#### 1. [How to login to MeluXina machine](https://docs.lxp.lu/first-steps/quick_start/)

* 1.1 [Please take a look if you are using Windows](https://docs.lxp.lu/first-steps/connecting/)
* 1.2 [Please take a look if you are using Linux/Mac](https://docs.lxp.lu/first-steps/connecting/)

#### 2. Use your username to connect to MeluXina

* 2.1 For exmaple the below example shows the user of u100490
* $ ssh u100490@login.lxp.lu -p 8822  
  ### or  
  $ ssh meluxina

#### 3. Once you have logged in

* 3.1 Once you have logged in, you will be in a default home directory
* [u100490@login02 ~]$ pwd  
  /home/users/u100490
* 3.2 After that go to project directory (Nvidia Bootcamp activites).
* [u100490@login02 ~]$ cd /project/home/p200117  
  [u100490@login02 p200117]$ pwd  
  /project/home/p200117

#### 4. And please create your own working folder under the project directory

* 4.1 For example, here it is user with u100490:
* [u100490@login02 p200117]$ mkdir $USER  
  ### or   
  [u100490@login02 p200117]$ mkdir u100490

#### 5. Now it is time to move into your home directory

* 5.1 For example, with user home directory u100490
* [u100490@login02 p200117]$cd u100490

#### 6. Now it is time to copy the folder which has examples and source files to your home directory

* 6.1 For example, with user home directory u100490
* [u100490@login03 u100490]$ cp -r /project/home/p200117/CUDA .  
  [u100490@login03 u100490]$ cd CUDA/  
  [u100490@login03 CUDA]$ pwd  
  /project/home/p200117/u100490/CUDA  
  [u100490@login03 CUDA]$ ls -lthr  
  total 20K  
  -rw-r-----. 1 u100490 p200117 51 Mar 13 15:50 module.sh  
  drwxr-s---. 2 u100490 p200117 4.0K Mar 13 15:50 Vector-addition  
  drwxr-s---. 2 u100490 p200117 4.0K Mar 13 15:50 Unified-memory  
  ...  
  ...

#### 7. Untill now you are in the login node, now its time to do the dry run test

* 7.1 Reserve the interactive node for running/testing CUDA applications
* $ salloc -A p200117 --res training\_part1 --partition=gpu --qos default N 1 -t 01:00:00
* ??? “check if your reservation is allocated” [u100490@login03 ~]$ salloc -A p200117 --res training\_part1 --partition=gpu --qos default N 1 -t 01:00:00 salloc: Pending job allocation 296848 salloc: job 296848 queued and waiting for resources salloc: job 296848 has been allocated resources salloc: Granted job allocation 296848 salloc: Waiting for resource configuration salloc: Nodes mel2131 are ready for job
* 7.2 You can also check if got the interactive node for your computations, for example here with user u100490:

[u100490@mel2131 ~]$ squeue -u u100490  
 JOBID PARTITION NAME USER ACCOUNT STATE TIME TIME\_LIMIT NODES NODELIST(REASON)  
 304381 gpu interact u100490 p200117 RUNNING 0:37 01:00:00 1 mel2131

#### 8. Now we need to check simple CUDA application, if that is going to work for you:

* 8.1 Go to folder Dry-run-test

[u100490@login03 CUDA]$ cd Dry-run-test/  
[u100490@login03 Dry-run-test]$ ls   
Hello-world.cu module.sh

#### 9. Finally we need to load the compiler to test the GPU CUDA codes

* 9.1 We need a Nvidia HPC SDK compiler for compiling and testing CUDA code

$ module load OpenMPI/4.1.4-NVHPC-22.7-CUDA-11.7.0  
### or  
$ source module.sh

* ??? “check if the module is loaded properly” ``` [u100490@mel2131 ~]$ module load OpenMPI/4.1.4-NVHPC-22.7-CUDA-11.7.0 [u100490@mel2131 ~]$ module list
* Currently Loaded Modules:  
  1) env/release/2022.1 (S) 6) numactl/2.0.14-GCCcore-11.3.0 11) libpciaccess/0.16-GCCcore-11.3.0 16) GDRCopy/2.3-GCCcore-11.3.0 21) knem/1.1.4.90-GCCcore-11.3.0  
  2) lxp-tools/myquota/0.3.1 (S) 7) CUDA/11.7.0 12) hwloc/2.7.1-GCCcore-11.3.0 17) UCX-CUDA/1.13.1-GCCcore-11.3.0-CUDA-11.7.0 22) OpenMPI/4.1.4-NVHPC-22.7-CUDA-11.7.0  
  3) GCCcore/11.3.0 8) NVHPC/22.7-CUDA-11.7.0 13) OpenSSL/1.1 18) libfabric/1.15.1-GCCcore-11.3.0  
  4) zlib/1.2.12-GCCcore-11.3.0 9) XZ/5.2.5-GCCcore-11.3.0 14) libevent/2.1.12-GCCcore-11.3.0 19) PMIx/4.2.2-GCCcore-11.3.0  
  5) binutils/2.38-GCCcore-11.3.0 10) libxml2/2.9.13-GCCcore-11.3.0 15) UCX/1.13.1-GCCcore-11.3.0 20) xpmem/2.6.5-36-GCCcore-11.3.0  
    
  Where:  
   S: Module is Sticky, requires --force to unload or purge  
  ```

#### 10. Please compile and test your CUDA application

* For example, Dry-run-test

// compilation  
$ nvcc -arch=compute\_70 Hello-world.cu -o Hello-World-GPU  
  
// execution  
$ ./Hello-World-GPU  
  
// output  
$ Hello World from GPU!  
 Hello World from GPU!  
 Hello World from GPU!  
 Hello World from GPU!

#### 11. Similary for the hands-on session, we need to do the node reservation:

$ salloc -A p200117 --res training\_part1 --partition=gpu --qos default N 1 -t 01:00:00

* ??? “check if your reservation is allocated” [u100490@login03 ~]$ salloc -A p200117 --res training\_part2 --partition=gpu --qos default N 1 -t 01:00:00 salloc: Pending job allocation 296848 salloc: job 296848 queued and waiting for resources salloc: job 296848 has been allocated resources salloc: Granted job allocation 296848 salloc: Waiting for resource configuration salloc: Nodes mel2131 are ready for job

#### 12. We will continute with our Hands on exercise

* 12.1 For example Hello World example, we do the following steps:

[u100490@mel2063 CUDA]$ pwd  
/project/home/p200117/u100490/CUDA  
[u100490@mel2063 CUDA]$ ls  
[u100490@mel2063 CUDA]$ ls  
Dry-run-test Matrix-multiplication Profiling Unified-memory  
Hello-world module.sh Shared-memory Vector-addition  
[u100490@mel2063 CUDA]$ source module.sh  
[u100490@mel2063 CUDA]$ cd Hello-world  
// compilation  
[u100490@mel2063 CUDA]$ nvcc -arch=compute\_70 Hello-world.cu -o Hello-World-GPU  
  
// execution  
[u100490@mel2063 CUDA]$ ./Hello-World-GPU  
  
// output  
[u100490@mel2063 CUDA]$ Hello World from GPU