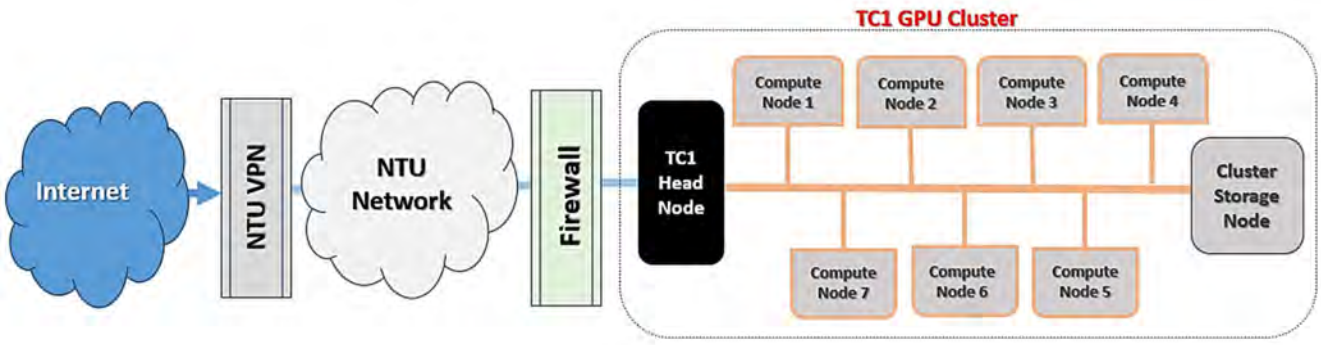
CCDS GPU Cluster (TC1) - USER GUIDE

Introduction



In TC1 GPU cluster, there are two diverse types of nodes for diverse types of tasks.
The initial node you log in to is the **Head Node**, serving as the main access point for the cluster.

The GPU cards for computation are in those Compute Nodes, NOT in the Head Node.
The users are **not allowed** to compile and run code on the Head Node. The users must create the job script to submit the computation request (knowns as **non-interactive job**) to SLURM (**Simple Linux Utility for Resource Management**) job scheduler, for the system to process in the allocated Compute Nodes.

The available resources for the user to utilise for computation is limited by the assigned **QoS** (SLURM-Quality of Service).

Client Tools to access TC1 Head Node

Name of Program	Description & Purpose	Where to download
PuTTY	**[SSH Client for Windows]** **PuTTY** is an open-source software as SSH and Telnet Client for *Windows Platform*. **SSH** (Secure Socket Shell) is a network protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network.	The installer is available online and can be easily located using any search engine. **Tip:** Download the portable edition "putty.exe" – no need to install, ready for use *http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html* **Other Online References:** *https://www.ssh.com/academy/ssh/putty*
	[SSH Client for MAC user] Beside ssh access via terminal, you may install other SSH Clients to facilitate the connection. **Online References:** *https://www.ssh.com/academy/ssh/putty/mac* *https://www.puttygen.com/download-putty* *https://termius.com/free-ssh-client-for-mac-os*	
WinSCP	**[SFTP Client for Windows]** **WinSCP** (Windows Secure Copy) is a free and open-source SFTP, FTP, WebDAV and SCP client for Microsoft Windows **SFTP** (SSH File Transfer Protocol) is an encrypted or secure file transfer protocol.	The official download site for WinSCP: *https://winscp.net/eng/index.php* **Installation tip:** Select "*Explorer*" for user interface style **Other Online References:** *https://winscp.net/eng/docs/guide_install* *https://www.puttygen.com/winscp*

FileZilla Client	**[SFTP Client for MAC]**	Download Site for FileZilla: *https://filezilla-project.org/index.php* **Other online References of using SFTP on MAC:** *https://lemp.io/how-to-connect-to-sftp-mac-os/* *https://beebom.com/how-to-use-mac-terminal-ftp-sftp-client/* *https://tipsmake.com/use-terminal-on-mac-as-ftp-or-sftp-client*

This User Guide is only providing general information for your kick-start in TC1, assuming that you already have the basic knowledge of accessing the Linux system via command-line interface:

- Connecting to a Linux server via SSH
- Moving files to a Linux server from your local computer, and vice versa
- Executing Linux commands at command-line

If you want in-depth information, you may visit those suggested **Online References** (*URL*) or search the internet for the required information.

Online References for Basic Linux Commands:
https://serverdale.com/en/linux-commands
https://centoshelp.org/resources/commands/linux-system-commands/

Online References on how to redirect or pipe the program output to a file:
https://tldp.org/LDP/intro-linux/html/sect_05_01.html
https://linuxconfig.org/how-to-pipe-output-to-a-file-on-linux
https://www.geeksforgeeks.org/ways-to-save-python-terminal-output-to-a-text-file/

Workflow for the user:

1. Establish your first access via SSH. Your home directory will only be created after your first login
2. Uploading all the necessary job script, coding, dataset from your host machine to the cluster's storage (your home directory) via SFTP client
3. Setup and configure your Conda environment(s) with required applications for computation
4. Verify your conda environment in the Compute Node via Jupyter *[Reference @ Using CUDA in TC1, page 11 onwards]*.
5. *The GPU cards are NOT in the Head Node*. Do not attempt to execute command to acquire the GPU information and CUDA status in the Head Node e.g., "*nvidia-smi*", "*nvcc --version*" etc. Do not attempt to execute your coding in the Head Node
6. Submitting the non-interactive job using SLURM scheduler, for your coding to execute in assigned Node *[Reference @ Guideline for Job Submission, page 16 onwards]*.
7. Copy the job output back to your host machine, via SFTP client

Logging into the cluster

The CCDS GPU cluster is only accessible **within NTU network**
For off-campus access, the user must connect via NTU Virtual Private Network (VPN) *[https://vpngate.ntu.edu.sg]*

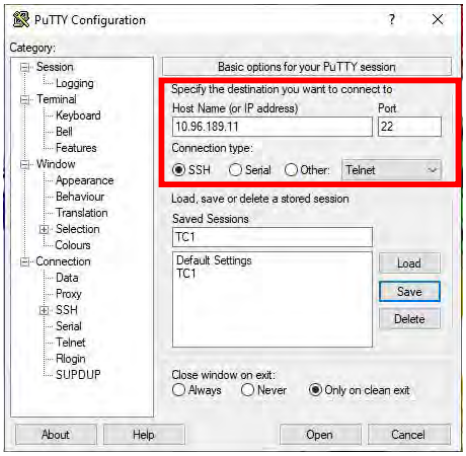
Other Online References:
https://www3.ntu.edu.sg/cits2/ras/for_cms_ref/GlobalProtectVPN.pdf
https://vpngate.ntu.edu.sg/global-protect/getsoftwarepage.esp

Hostname of TC1 Head Node	**CCDS-TC1**
IP Address	**10.96.189.11**
Login Credential	Your NTU Network Account ID ***(in Lower Case)*** & Password

SSH via PuTTY

1. Launch PuTTY.

   Under the Host Name, enter the IP Address [10.96.189.11], Port Number 22.

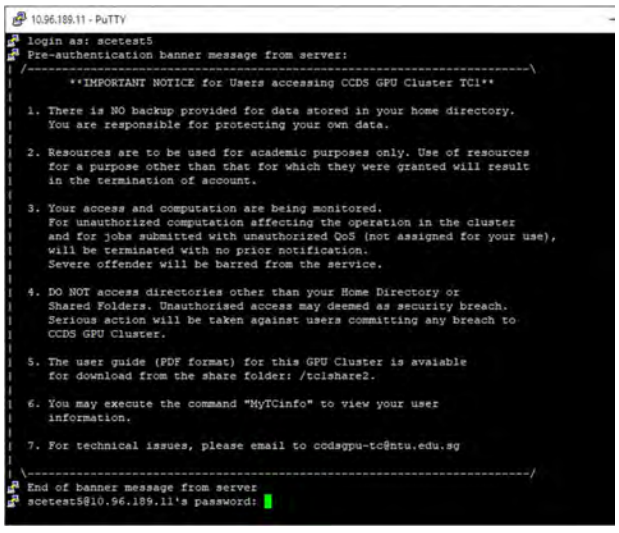


2. Examples of command-line access via your LINUX or MAC Terminal:

   >> *ssh -l <Your Username in Lower Case> 10.96.189.11*

   >> *ssh -p 22 <Your Username in Lower Case>@10.96.189.11*

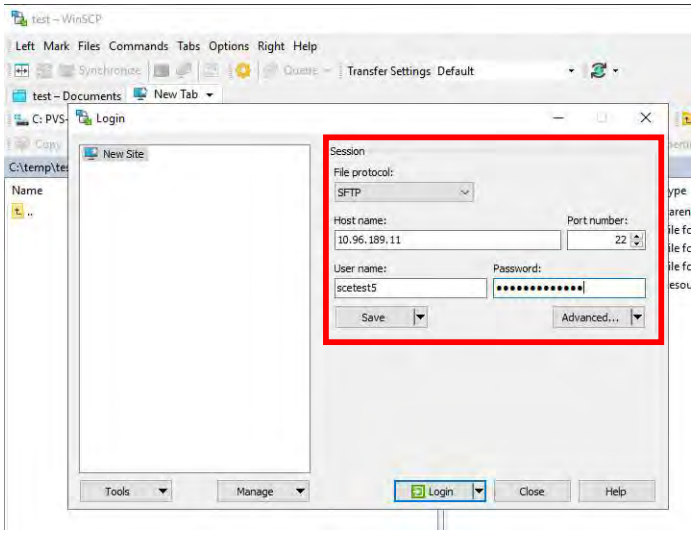
3. On your first login, you will be asked to accept the host key.
   Do click "**yes**" to continue, and you should get a terminal window, where you will be prompted for your credentials. Log in with your NTU Network Account ID (*in Lower Case*) and password.



Best practice to exit the SSH session: type "**exit**" or press the keys **Ctrl + D**

SFTP via WinSCP/FileZilla – For file transfer between your local host machine and TC1 Head Node

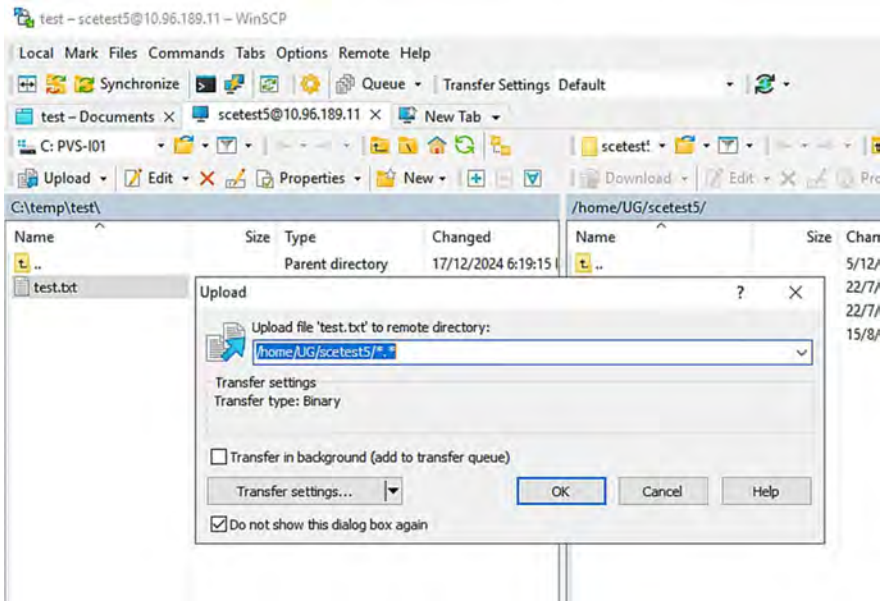
1. In this example, the client tool using for file transfer is WinSCP.
2. Launch WinSCP. Check that the File Protocol is set to SFTP.
3. Under the Host Name, key in the IP Address [10.96.189.11]. Ensure that Port is 22. Key in your NTU Network Account ID (in lower case) and password for the Username and Password field.



4. Ensure that you are in the correct home directory before transferring your files over.
   You can verify the pathname of your home directory with following command in SSH session:
   >> *pwd*

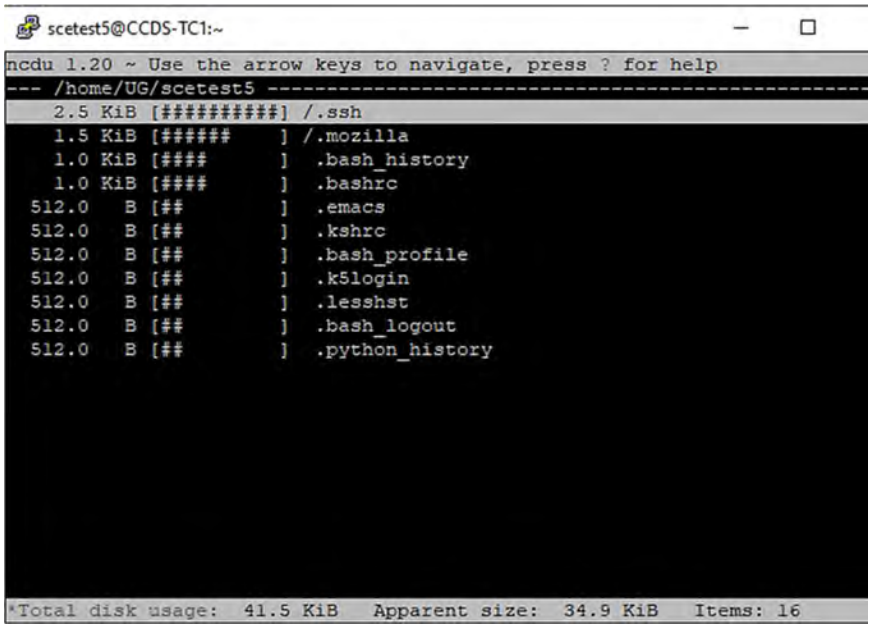
```
[scetest5@CCDS-TC1:0 ~]$ pwd
/home/UG/scetest5
[scetest5@CCDS-TC1:0 ~]$
```

   Mouse-click on the file that you want transfer and drag to the remote windows.
   The *Upload-Window* will pop out to transfer the file from your host machine to your home directory.
   The *Download-Window* will pop out to transfer file from your home directory to selected local disk in your host machine.



Maintain Data in Your Home Directory

1. Your data and computation are bounded within your assigned home directory and share folder (if any).
2. All user directories are set with disk quota limit. The users may have disk quota of at least 100GB, depending on their usage assignment.
3. The user may encounter following issues when the quota usage reached more than 98%:
   - Unable to login
   - Unable to compute data
   - Unable to save in new data, or having data lost
4. You should do regular housekeeping, ensure there is sufficient free space for your computation.
5. Command to verify the disk usage at command line: ***ncdu***



Press the key "***q***" to abort disk scanning or exit.

Online Reference on ncdu: *https://ostechnix.com/check-disk-space-usage-linux-using-ncdu/*

6. Best practices to ensure enough disk space for your computation:
   - Regularly check on the disk usage for your home directory
   - Transfer and backup your data to your personal device
   - Remove unwanted data in your home directory
   - *Do not remove those system directories and files, which naming with a "**.**" in front, such as ".bash_profile", ".bashrc", ".config" etc*
   - The purpose of the system folder "*.conda*" is to store the packages and environments that you have setup and installed for your computation. *Remove those unwanted environments to free the disk space.*

Conda Package and Environment

The module package tool is available in TC1, allowing users to easily configure their environment based on the application needed. As this GPU cluster TC1 is shared by users of different undergrad courses and projects in CCDS, there may be applications that are not relevant to you.

To view the available share applications, apply the modules under ***/cm/shared/modulefiles***
>> module avail

```
[scetest5@CCDS-TC1:1 ~]$ module avail
------------------------------------------------------------ /cm/local/modulefiles ------------------------------------------------------------
boost/1.81.0          cm-scale/cm-scale.module  cmsh              gcc/13.1.0        mariadb-libs  openldap        shared
cluster-tools/10.0    cm-setup/10.0             docker/26.1.5     ipmitool/1.8.19   module-git    python3         slurm/slurm/23.02.8
cm-bios-tools         cmd                       dot               lua/5.4.6         module-info   python39
cm-image/10.0         cmjob                     freeipmi/1.6.14   luajit            null          rocm-smi/4.3.0

------------------------------------------------------------ /cm/shared/modulefiles ------------------------------------------------------------
anaconda              cuda/12.4             gcc/13.3.0                        hwloc2/2.8.0          mvapich2/gcc/64/2.3.7       python/3.9.23
blas/gcc/64/3.11.0    cuda/12.5             gcc/14.1.0                        iozone/3.494          netcdf/gcc/64/gcc/64/4.9.2  python/3.10.18
bonnie++/2.00a        cuda/12.6             gdb/13.1                          lapack/gcc/64/3.11.0  netperf/2.7.0               python/3.11.13
cm-pmix3/3.1.7        cuda/12.9             globalarrays/openmpi/gcc/64/5.8   miniconda/py39        openblas/dynamic/(default)  python/3.12.11
cm-pmix4/4.1.3        default-environment   hdf5/1.14.0                       miniconda/py312       openblas/dynamic/0.3.18     python/3.13.5
cuda/11.8             gcc/11.5.0            hdf5_18/1.8.21                    miniconda/py313       openmpi/gcc/64/4.1.5        ucx/1.10.1
cuda/12.2             gcc/12.4.0            hwloc/1.11.13                     mpich/ge/gcc/64/4.1.1 openmpi4/gcc/4.1.5
[scetest5@CCDS-TC1:1 ~]$
```

Description	Command
To view the available share applications	***>> module avail***
To view the description of a module	***>> module show <module_name>***
To load the selected module for operation	***>> module load <module_name>***
To list loaded module	***>> module list***
To unload the selected module	***>> module unload <module_name>***
To unload all loaded modules	***>> module purge***
To know more about the command "module"	***>> module --help***

```
[scetest5@CCDS-TC1:1 ~]$ module load slurm
[scetest5@CCDS-TC1:1 ~]$ module list
Currently Loaded Modulefiles:
 1) slurm/slurm/23.02.8
[scetest5@CCDS-TC1:1 ~]$
```

Note: The module "slurm" should be loaded upon every login. If no, ensure to load it manually.
This module is required for you to apply those SLURM commands to submit and manage jobs.

Setup Your own Conda Environment

In TC1, the user has no right to execute Sudo and install application to the system.

You may download and install the Anaconda or Miniconda in your home directory for operation.

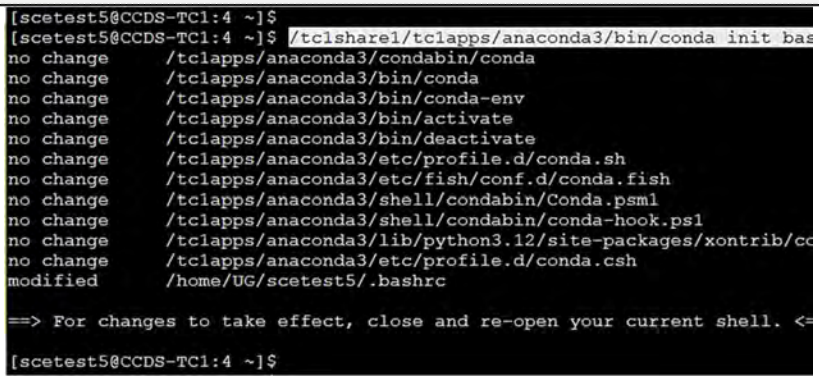
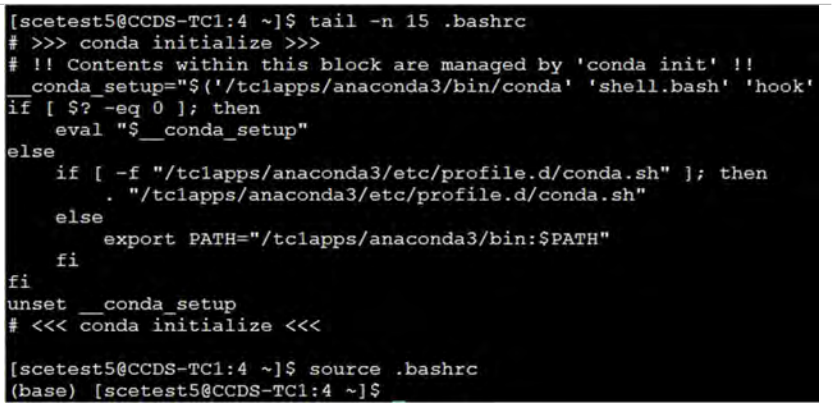
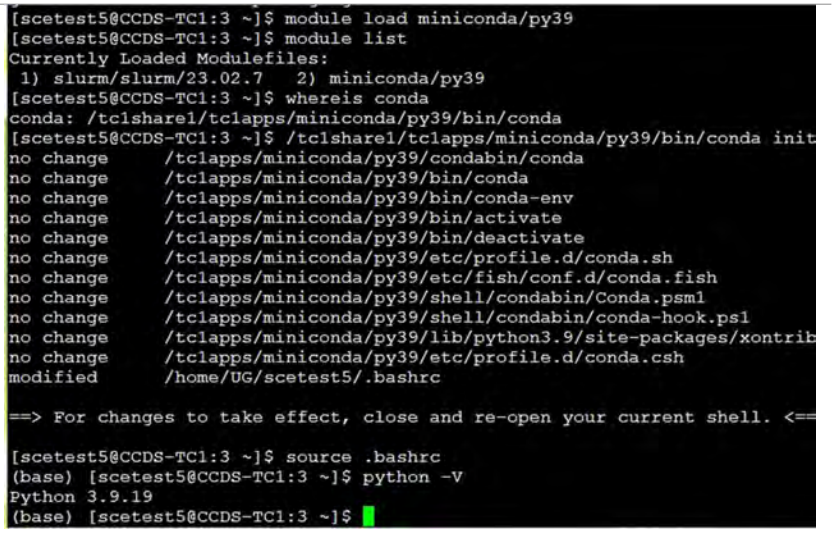
Online References:
https://www.anaconda.com/products/distribution
https://docs.conda.io/en/latest/miniconda.html

OR apply the available Anaconda in TC1:

Description	Command	Remark

To load the module of anaconda available in TC1	***>> module load anaconda***	`[scetest5@CCDS-TC1:0 ~]$ module load anaconda`
		`[scetest5@CCDS-TC1:0 ~]$ module list`
		`Currently Loaded Modulefiles:`
		`1) slurm/slurm/23.02.7  2) anaconda`
		`[scetest5@CCDS-TC1:0 ~]$`

To locate the actual path of conda to execute	***>> whereis conda***
	OR
	>> which conda
		`[scetest5@CCDS-TC1:4 ~]$ which conda`
		`/tc1apps/anaconda3/bin/conda`
		`[scetest5@CCDS-TC1:4 ~]$`
		`[scetest5@CCDS-TC1:4 ~]$ whereis conda`
		`conda: /tc1share1/tc1apps/anaconda3/bin/conda`
		`[scetest5@CCDS-TC1:4 ~]$`

For first-time running, to add the *conda initialize script* into the system file [*~/.bashrc*] This way, upon your sequence login, the base conda module will be loaded automatically. You may see "***(base)***" appearing at the command prompt.	***>> conda init bash***	
For the change to take effect, you may exit (logout) from your SSH session, and then login again. Or in your current login, execute the command "***source***" to apply the change immediately. The "***(base)***" will be appearing at the command prompt, when the conda base module is being loaded for your use.	***>> source .bashrc***	
To view the information of the loaded **anaconda**	***>> conda info***	
To list the applications available in the base environment	***>> conda list***	
To unload or exit from the conda module	***>> conda deactivate***	
You may also apply the above steps to load the modules of miniconda available in TC1. Do execute the command "*conda deactivate*" to exit from existing conda environment (if any) and loaded miniconda module, before executing the command to load another conda module Example to load the miniconda installed with Python 3.9 ***>> module purge*** ***>> module load miniconda/py39***		

You are **not allowed** to install any packages in the base environment. If you find those packages in the base environment do not meet your requirement, you may setup own conda environment in the home directory and install the require packages for your operation.

Commands to manage the Conda environment:

Description	Examples of Command Execution
To create a new Conda environment naming as "*TestEnv*"	*>> conda create -n TestEnv*
Activate the environment "*TestEnv*"	*>> conda activate TestEnv*
To exit from the environment "*TestEnv*"	*>> conda deactivate*
List the environment created in your home directory	*>> conda env list*
No command to rename the conda environment Resolving by cloning the existing env "OldEnv" into another name "NewEnv", then remove the unwanted env	*>> conda create -n NewEnv --clone OldEnv* *>> conda env remove -n OldEnv*
To export (backup) the configuration of the environment into a *.yml* file.	*>> conda env export > TestEnv.yml*
To create or restore the environment from an existing *.yml* file, naming as "Env002"	*>> conda env create -n Env002 -f TestEnv.yml*
To update the content of an existing *.yml* file, with option "*prune*" to remove the outdated configuration and dependencies	*>> conda env update --prefix ./TestEnv --file TestEnv.yml --prune*
To remove the environment "*TestEnv*"	*>> conda env remove -n TestEnv*

Online Reference on managing the conda environment:
https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#creating-an-environment-with-commands

Commands to manage the system variables in the environment:

Description	Examples of Command Execution
List the variables in the environment	*>> conda env config vars list*
To set a new variable in the environment "*my_var1*"	*>> conda env config vars set my_var1=value*
To unset (remove) the variable "*my_var1*"	*>> conda env config vars unset my_var1*
Always re-activate the environment after adding a new variable and removing a variable	*>> conda activate TestEnv*

```
[scetest5@CCDS-TC1:3 ~]$ conda activate TestEnv
(TestEnv) [scetest5@CCDS-TC1:3 ~]$ conda env config vars list
(TestEnv) [scetest5@CCDS-TC1:3 ~]$ conda env config vars set my_var1=10
To make your changes take effect please reactivate your environment
(TestEnv) [scetest5@CCDS-TC1:3 ~]$ conda deactivate
[scetest5@CCDS-TC1:3 ~]$ conda activate TestEnv
(TestEnv) [scetest5@CCDS-TC1:3 ~]$ conda env config vars list
my_var1 = 10
```

```
(TestEnv) [scetest5@CCDS-TC1:3 ~]$ conda env config vars unset my_var1
To make your changes take effect please reactivate your environment
(TestEnv) [scetest5@CCDS-TC1:3 ~]$ conda deactivate
[scetest5@CCDS-TC1:3 ~]$ conda activate TestEnv
(TestEnv) [scetest5@CCDS-TC1:3 ~]$ conda env config vars list
(TestEnv) [scetest5@CCDS-TC1:3 ~]$
```

Commands to manage the package in the environment

The user must always create a new environment to install the package for own use.

Description	Examples of Command Execution
Create and activate a new conda environment	*>> conda create -n TestEnv* *>> conda activate TestEnv*
Search for available packages. For this example, search for available python to install, then install the required version.	*>> conda search python* *>> conda install python=3.11*
To search and install available package from third-party channel The conda package manager usually installs the package from the official default channels. You must specify the channel in the command if to search and install package from third-party channel ***conda-forge*** is one of the third-party channels, providing latest conda packages. ***Online References:*** *https://docs.conda.io/projects/conda/en/latest/user-guide/concepts/channels.html* *https://ostechnix.com/enable-conda-forge-channel-for-conda-package-manager/*	# To search for a package in 3rd party channel "conda-forge" *>> conda search -c conda-forge <name of package>* # To install the package from the selected third-party channel *>> conda install -c conda-forge <name of package>* *>> conda install conda-forge::<name of package>*

List the installed packages	***>> conda list***
Install selected version of the package *# The installation will replace the existing version (if any) in the environment for operation*	***>> conda install python=3.10***
Remove a package	***>> conda uninstall python=3.9***

Online References:
https://docs.anaconda.com/anaconda/user-guide/tasks/install-packages/
https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-pkgs.html#installing-packages

IMPORTANT NOTE:

Advise to install the packages using ***conda install***. Only use ***pip install*** if the required package is not available through conda.
This is to prevent package incompatibility.

Remove unwanted Conda environment, to free usage space in your home directory

1. Exit from the conda environment >> *conda deactivate*
2. List the conda environment in your home directory >> *conda env list*
3. Only remove unwanted environment in your home directory
   >> conda env remove –name <Name of ENV to remove>

4. You may also remove the Deactivated conda environment by its folder under *.conda/envs*, e.g.
   >> rm -Rf ~<your home directory>/.conda/envs/<folder>

```
[scetest5@CCDS-TC1:3 ~]$ conda env list
# conda environments:
#
TestEnv                  /home/UG/scetest5/.conda/envs/TestEnv
base                     /tc1apps/anaconda3

[scetest5@CCDS-TC1:3 ~]$ rm -Rf /home/UG/scetest5/.conda/envs/TestEnv
```

Using CUDA in TC1

CUDA (Compute Unified Device Architecture) is a software framework developed by NVIDIA to expand the capabilities of GPU acceleration. In TC1, you may opt to apply the *System installed CUDA* (*Nvidia Cudatoolkit*) or *Conda CUDA* (*Cuda toolkit installed in your own conda environment*) for computation.

The *NVIDIA GPU cards compatible with CUDA* are in the Compute Nodes [***TC1Nxx***], not in TC1 Head Node [***CCDS-TC1***].
Do not attempt to execute commands to verify on the GPU and CUDA in TC1 Head Node.

The access to the System-installed CUDA in those Compute Nodes is made available as modules, for the user to load in the job script for execution.

Command to list the available CUDA modules:
>> module avail | grep cuda

```
(base) [scetest5@CCDS-TC1:0 ~]$ module avail | grep cuda
anaconda             cuda/12.5            gdb/13.1                          mi
blas/gcc/64/3.11.0   cuda/12.6            globalarrays/openmpi/gcc/64/5.8   mi
bonnie++/2.00a       cuda/12.9            hdf5/1.14.0                       mi
cuda/11.8            gcc/12.4.0           hwloc2/2.8.0                      ne
cuda/12.2            gcc/13.3.0           iozone/3.494                      ne
cuda/12.4            gcc/14.1.0           lapack/gcc/64/3.11.0              op
(base) [scetest5@CCDS-TC1:0 ~]$
```

Example to apply one of the above CUDA for your computation into your job script:
module load cuda/12.9

The system-installed CUDA available in TC1 may not be applicable to all users' operation.
You may create own conda environment, to install the required CUDA toolkit or other CUDA related packages, for your computation.

Online References:
https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html
https://pytorch.org/get-started/previous-versions/
https://www.tensorflow.org/install/source#tested_build_configurations
https://anaconda.org/anaconda/cudatoolkit

To install Cuda in own conda environment…

Example to search for available version of cuda toolkit to install from NVIDIA
>> conda search -c nvidia cuda-toolkit

Example to search for available version of Cuda to install from Anaconda
>> conda search cudatoolkit

Example to install the select version for operation
>> conda install -c nvidia cuda-toolkit=<version>

Example to install NVIDIA CUDA Complier (nvcc)
>> conda install -c nvidia cuda-nvcc=<version>

How to verify whether your Conda Environment complying to your requirement?

1) ***Example to verify CUDA version installed in your own conda environment*** ***@ TC1 Head Node***
   >> conda activate <Name of the Conda Environment>
   >> conda list | grep cuda

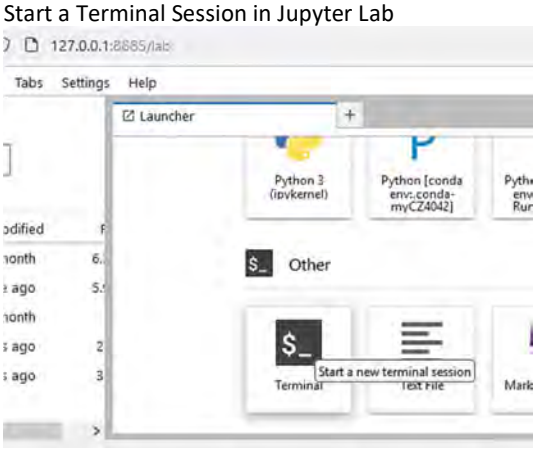
2) ***Example to verify your CUDA and Conda Environment*** ***@ one of the Compute Nodes***
   a) Create and submit the job to launch ***Jupyter-lab***
      (Jupyter setup and launching details are available in section " **Shared Resources** " and " **Miscellaneous** ")

- Sample of the job script to launch Jupyter Lab from the share conda environment "*Jupyter*"

```
#!/bin/sh
#SBATCH --partition=UGGPU-TC1
#SBATCH --qos=normal
#SBATCH --mem=10G
#SBATCH --ntasks-per-node=4
#SBATCH --gres=gpu:1
#SBATCH --nodes=1
#SBATCH --nodelist=TC1N04
#SBATCH --time=360
#SBATCH --job-name=J1
#SBATCH --output=output_%x_%j.out
#SBATCH --error=error_%x_%j.err

module load anaconda
source activate /tc1apps/2_conda_env/Jupyter
jupyter-lab --ip=$(hostname -i) --port=8882
```

- Start a Terminal Session in Jupyter Lab



b) Example to list and activate the available conda environments in your home folder
In the terminal, command to list own conda environments >> ***conda env list***
Command to load the selected environment >> ***conda activate <Name of the environment>***

```
scetest5@TC1N04:~/tc1-job:
(base) [scetest5@TC1N04 tc1-jobs]$ conda env list
# conda environments:
#
RunJupyter               /home/UG/scetest5/.conda/envs/RunJupyter
cuda-python              /home/UG/scetest5/.conda/envs/cuda-python
myCZ4042                 /home/UG/scetest5/.conda/envs/myCZ4042
numba-test               /home/UG/scetest5/.conda/envs/numba-test
tf-gpu                   /home/UG/scetest5/.conda/envs/tf-gpu
base                  *  /tc1apps/anaconda3

(base) [scetest5@TC1N04 tc1-jobs]$ conda activate numba-test
(numba-test) [scetest5@TC1N04 tc1-jobs]$
```

c) Example to launch the shared conda environment, installed with ***Pytorch, Pytorch-Cuda and Cuda-nvcc***

```
(base) [scetest5@TC1N04 ~]$ ll /tc1apps/2_conda_env/
total 4
drwxr-xr-x 19 root root  295 Aug  1 15:14 CZ4042_v4
drwxr-xr-x 17 root root  271 Aug  1 15:07 CZ4042_v5
drwxr-xr-x 30 root root 4096 Jan 28 12:16 CZ4042_v6
drwxr-xr-x 13 root root  195 Aug  1 14:13 Jupyter
(base) [scetest5@TC1N04 ~]$ conda activate /tc1apps/2_conda_env/CZ4042_v6
(/tc1apps/2_conda_env/CZ4042_v6) [scetest5@TC1N04 ~]$ conda list | grep cuda
cuda-cudart               11.8.89                       0    nvidia
cuda-cupti                11.8.87                       0    nvidia
cuda-libraries            11.8.0                        0    nvidia
cuda-nvcc                 11.8.89                       0    nvidia
cuda-nvrtc                11.8.89                       0    nvidia
cuda-nvtx                 11.8.86                       0    nvidia
cuda-runtime              11.8.0                        0    nvidia
cuda-version              12.8                          3    nvidia
pytorch                   2.5.1           py3.12_cuda11.8_cudnn9.1.0_0    pytorch
pytorch-cuda              11.8                 h7e8668a_6    pytorch
pytorch-mutex             1.0                         cuda    pytorch
(/tc1apps/2_conda_env/CZ4042_v6) [scetest5@TC1N04 ~]$ python
Python 3.12.3 | packaged by Anaconda, Inc. | (main, May  6 2024, 19:46:43) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> print(torch.cuda.is_available())
True
>>> print(torch.version.cuda)
11.8
>>> quit()
(/tc1apps/2_conda_env/CZ4042_v6) [scetest5@TC1N04 ~]$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Sep_21_10:33:58_PDT_2022
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
(/tc1apps/2_conda_env/CZ4042_v6) [scetest5@TC1N04 ~]$
```

Verify Cuda via python and pytorch

d) Example to verify the System-installed CUDA in the Compute Node
In the terminal, command to list the available cuda >> ***module avail | grep cuda***
Command to verify the cuda version being loaded for use >> ***nvcc -V***

```
scetest5@TC1N04:~/tc1-job: ×  +

(numba-test) [scetest5@TC1N04 tc1-jobs]$ module list
Currently Loaded Modulefiles:
 1) anaconda
(numba-test) [scetest5@TC1N04 tc1-jobs]$ module avail | grep cuda
anaconda <L>        cuda/12.6                    hdf5/1.14.0              mpich/ge/gcc/64/4.1.1
blas/gcc/64/3.11.0  cuda/12.9                    hdf5_18/1.8.21           mvapich2/gcc/64/2.3.7
cuda/11.8           gcc/13.3.0                   lapack/gcc/64/3.11.0     openblas/dynamic/0.3.18
cuda/12.2           gcc/14.1.0                   miniconda/py39           openmpi/gcc/64/4.1.5
cuda/12.4           gdb/13.1                     miniconda/py312          openmpi4/gcc/4.1.5
cuda/12.5           globalarrays/openmpi/gcc/64/5.8  miniconda/py313      python/3.9.23
(numba-test) [scetest5@TC1N04 tc1-jobs]$ module load cuda/12.4
(numba-test) [scetest5@TC1N04 tc1-jobs]$ module list
Currently Loaded Modulefiles:
 1) anaconda   2) cuda/12.4
(numba-test) [scetest5@TC1N04 tc1-jobs]$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2024 NVIDIA Corporation
Built on Thu_Mar_28_02:18:24_PDT_2024
Cuda compilation tools, release 12.4, V12.4.131
Build cuda_12.4.r12.4/compiler.34097967_0
(numba-test) [scetest5@TC1N04 tc1-jobs]$ nvidia-smi
Mon Aug 25 15:40:31 2025
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 575.57.08                                                        |
|-----------------------------------------+------------------------+----------+
| 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  Tesla V100-PCIE-32GB           Off |   00000000:86:00.0 Off |                    0 |
| N/A   47C    P0             38W /  250W |       0MiB /  32768MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI              PID   Type   Process name                  GPU Memory |
|        ID   ID                                                          Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
(numba-test) [scetest5@TC1N04 tc1-jobs]$
```

*The command " **nvidia-smi** " will only display information of the GPU being assigned to you.*

e) Example to launch the conda environment, installed with *Cuda-Python*
Online Reference: https://nvidia.github.io/cuda-python/latest/
 # The listing shows the related cuda applications being installed together with Cuda-Python

```
scetest5@TC1N04:~/tc1-job: ×  +

(base) [scetest5@TC1N04 tc1-jobs]$ conda activate cuda-python
(cuda-python) [scetest5@TC1N04 tc1-jobs]$ conda list | grep cuda-python
# packages in environment at /home/UG/scetest5/.conda/envs/cuda-python:
cuda-python               11.8.0          py39h3fd9d12_0    nvidia
(cuda-python) [scetest5@TC1N04 tc1-jobs]$ conda list | grep cuda-nvcc
cuda-nvcc                 12.4.131             h02f8991_0
cuda-nvcc-dev_linux-64    12.4.131             h4ee8466_0
cuda-nvcc-impl            12.4.131             h99ab3db_0
cuda-nvcc-tools           12.4.131             h99ab3db_0
cuda-nvcc_linux-64        12.4.131             he92618c_0
(cuda-python) [scetest5@TC1N04 tc1-jobs]$ python -V
Python 3.9.19
(cuda-python) [scetest5@TC1N04 tc1-jobs]$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2024 NVIDIA Corporation
Built on Thu_Mar_28_02:18:24_PDT_2024
Cuda compilation tools, release 12.4, V12.4.131
Build cuda_12.4.r12.4/compiler.34097967_0
(cuda-python) [scetest5@TC1N04 tc1-jobs]$ which nvcc
~/.conda/envs/cuda-python/bin/nvcc
(cuda-python) [scetest5@TC1N04 tc1-jobs]$
```

SLURM User Guide

SLURM (*Simple Linux Utility for Resource Management*) is a software package for submitting, scheduling, and monitoring jobs on large computer clusters.

In TC1, you are **not allowed** to compile and run your code on the Head Node. You must create a job script to submit your computation request (**non-interactive jobs**) to SLURM Scheduler, for the system to process in allocated Compute Nodes.

The resources for GPU computation are limited by the **QoS** (SLURM-Quality of Service) assigned to you.

Command to view the QoS assigned to the user
>> sacctmgr show user <username> withassoc format=user,qos

 # You may copy and paste the above command to the command prompt in your SSH session window for execution
 # Replace the text "<username>" with your own username

```
[scetest5@CCDS-TC1:3 ~]$ sacctmgr  show user scetest5 withassoc format=user,qos
      User                  QOS
---------- --------------------
  scetest5               normal
[scetest5@CCDS-TC1:3 ~]$
```

 # The above example shown the user "scetest5" can use the assigned QoS "normal" for GPU computation

---

Command to show the resources configured for the QoS:
>> sacctmgr -P show qos <Name of QoS> withassoc format=name,MaxTRESPU,MaxJobsPU,MaxWall
>> sacctmgr show qos <Name of QoS> withassoc format=name%+15,MaxTRESPU%+40,MaxJobsPU%+10,MaxWall%+10

 # %+<value> is to add value to expand the viewing field

```
[scetest5@CCDS-TC1:1 ~]$ sacctmgr -P show qos normal withassoc format=name,MaxTRESPU,MaxJobsPU,MaxWall
Name|MaxTRESPU|MaxJobsPU|MaxWall
normal|cpu=20,gres/gpu=1,mem=64G|2|06:00:00
[scetest5@CCDS-TC1:1 ~]$ sacctmgr show qos normal withassoc format=name%+15,MaxTRESPU%+40,MaxJobsPU%+10,MaxWall%+10
           Name                                  MaxTRESPU  MaxJobsPU    MaxWall
--------------- ---------------------------------------- ---------- -----------
         normal                cpu=20,gres/gpu=1,mem=64G          2    06:00:00
[scetest5@CCDS-TC1:1 ~]$
```

Example showing the Resources assigned to above QoS "**normal**":

1. MaxTRESPU (*Maximum Trackable RESources Per User*):
   cpu (*Maximum number of CPUs/CORES for the user to deploy*) = 20
   gres/gpu (*Maximum number of GPU for the user to deploy*) = 1
   mem (*Maximum size of Memory for the user to deploy*) = 64G
2. MaxJobsPU (*Maximum number of Jobs Per User, can run at a given time*) = 2
3. MaxWall (*Maximum Wall clock time per user, to run the job, DD-HH:MM:SS*) = 6-hours

---

The SLURM job script is required, to specify the necessary resources, application, and path to execute your code. Then, submit the job script to SLURM at command line.

Online References on creating the SLURM job script:
https://slurm.schedmd.com/quickstart.html
https://www.carc.usc.edu/user-information/user-guides/hpc-basics/slurm-templates
https://svante.mit.edu/use_slurm.html

Common SLURM commands for you to manage the job submission:

Command	Definition (#) & Use Examples (>>)
scontrol	*# View the details of a running job, based on its jobid and showing the id of GPU card in the node running the job, under the line "Nodes=TC1N…"* **>> scontrol show -d jobid <jobid>**

```
NodeList=TC1N01
BatchHost=TC1N01
NumNodes=1 NumCPUs=2 NumTasks=2 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
ReqTRES=cpu=2,mem=10G,node=1,billing=2,gres/gpu=1
AllocTRES=cpu=2,node=1,billing=2,gres/gpu=1
Socks/Node=* NtasksPerN:B:S:C=2:0:*:* CoreSpec=*
JOB_GRES=gpu:1
  Nodes=TC1N01 CPU_IDs=36-37 Mem=0 GRES=gpu:1(IDX:0)
MinCPUsNode=2 MinMemoryNode=10G MinTmpDiskNode=0
Features=(null) DelayBoot=00:00:00
OverSubscribe=OK Contiguous=0 Licenses=(null) Network=(null)
```

	# View the status of the nodes in the Partition "CCDSGPU_UG" **>> scontrol show node TC1-[01-07]** *You can reference the resource status of the node, based on the information at the lines "CfgTRES" and "AllocTRES".* *The values shown under "AllocTRES" are the resources being allocated to the running jobs in the node.*

sinfo	*# Display state of the compute nodes in respective partition*
	>> sinfo -N -l
	`NODELIST NODES PARTITION STATE CPUS S:C:T MEMORY TMP_DISK WEIGHT AVAIL_FE REASON`
	`TC1N01 1 UGGPU-TC1* mixed 72 2:18:2 386590 0 1 location none`
	`TC1N02 1 UGGPU-TC1* idle 72 2:18:2 386590 0 1 location none`
	`TC1N03 1 UGGPU-TC1* idle 72 2:18:2 386590 0 1 location none`
	`TC1N04 1 UGGPU-TC1* idle 72 2:18:2 386590 0 1 location none`
	`TC1N05 1 UGGPU-TC1* mixed 72 2:18:2 386590 0 1 location none`
	`TC1N06 1 UGGPU-TC1* mixed 72 2:18:2 386590 0 1 location none`
	`TC1N07 1 UGGPU-TC1* mixed 72 2:18:2 386590 0 1 location none`
	For STATE not showing "idle" or "mix", may indicate the node under NODELIST has been out of service

sacct	*# To view your own job history* **>> sacct --format=jobid,jobname%+15,qos,alloctres%+50,nodelist,start,elapsed,state,reason%+20** *# To view all job history for the day, option "-a"* **>> sacct -a --format=user,jobid,jobname%+15,qos,alloctres%+50,nodelist,start,elapsed,state,reason%+20** *# Job viewing, excluding those lines with "batch..."* **>> sacct -a --format=user,jobid,jobname%+15,qos,alloctres%+50,nodelist,start,elapsed,state,reason%+20 | grep -wv batch** *# List the jobs executed by specific user <username>* **>> sacct -u <username>**
sbatch	*# Submit the job script to the job queue for execution by SLURM in the background* *# Once submitted the job, user may exit from the SSH session, with no consequence.* **>> sbatch job.sh**
scancel	*# The jobid can be obtained by the command "squeue"* *# Cancel a job base on its jobid* **>> scancel <jobid>**
squeue	*# Display the jobs in the job queue* **>> squeue** *# List more available information with option 'la'* **>> squeue -la** *# List the running jobs submitted by the user <username>* **>> squeue -u <username>**
seff	*# Slurm Job Efficiency Report* *# Display the statistics of resource being utilized by the completed job, allow to review on the resource assignment for the job* **>> seff <jobid of completed job>**

Guideline for Job Submission

1. **DO NOT execute your coding on TC1 Head Node [CCDS-TC1]**
   The visible impacts are lagging or failed SSH access, freeze data surfing and service suspension, affecting other users accessing TC1.
   For the user process found executing coding and occupying high CPU and Memory usage in the Head Node will be terminated with no prior notification. Repeated offenders will be banned from TC1.

2. **Only apply the QoS assigned to you**
   The job submitted with unauthorized QoS (QoS not assigned for your use) will be terminated by the system with no prior notification.

3. **Apply the QoS resources within the limit in your job script**

Execute the customized script/command to view your account and actual QoS value assigned
>> MyTCinfo

```
[scetest5@CCDS-TC1:1 ~]$ MyTCinfo
------- User Info of scetest5 in TC1 ----------------------
uid=30720(scetest5) gid=30720(scetest5) groups=438(ollama),441(docker),1003(tc1conda),
Home Directory in TC1= /home/UG/scetest5
========User Info of scetest5 in SLURM DB=========================================
          User              Cluster                                QOS
-------------- ------------------- ------------------------------
       scetest5               slurm                         normal

===QOS Info=======================================================================
          Name                        MaxTRESPU       MaxSubmit          MaxWall
-------------- ----------------------------- --------------- ---------------
        normal      cpu=20,gres/gpu=1,mem=64G               2        06:00:00
     Partition
--------------
 UGGPU-TC1

-----End-----------------------------------------------------
[scetest5@CCDS-TC1:1 ~]$
```

In your job script, ensure to apply the job flag with stated value below or equal to (**<=**) the values in the assigned QoS.

MaxSubmit	The maximum of number of jobs allow to submit at a time **<= 2** *You may submit two jobs to run at a time*
MaxWall	Total computation of your running jobs must not exceed the maximum wall time **<= 6-hr**
MaxTRESPU	Total amount of resources allocated in your two job scripts must not exceed the maximum values. *For example: No of CPU/cores <=20, Memory <=64GB and No of GPU Card=1, as stated in QoS assigned.*

Note: The QoS resource values may subject to change due to support change for every school semester

 # Other customized scripts/commands are available under /tc1share2/tc-scripts #

```
[scetest5@CCDS-TC1:1 ~]$ ll /tc1share2/tc-scripts/
total 161
-rw-r--r-- 1 root root   92 Oct 23  2024 1_The_scripts_can_only_run_in_HeadNode
-rwxr-xr-x 1 root root  563 Mar  7 09:43 MyJobHistory
-rwxr-xr-x 1 root root 2014 Jul 19  2024 MyTCinfo
-rwxr-xr-x 1 root root 2166 Mar  7 09:42 TC1JobHistory
-rwxr-xr-x 1 root root 2463 Mar 10 12:43 TC1RunningJob
-rwxr-xr-x 1 root root 2022 Oct 23  2024 TCluserinfo
[scetest5@CCDS-TC1:1 ~]$
```

Example to list the current running job in TC1

```
[scetest5@CCDS-TC1:1 ~]$ TC1RunningJob
Tue Jul 22 18:03:19 2025
NODELIST   NODES  PARTITION       STATE CPUS    S:C:T MEMORY TMP_DISK WEIGHT AVAIL_FE REASON
TC1N01         1 UGGPU-TC1*      mixed 72     2:18:2 386590        0      1 location none
TC1N02         1 UGGPU-TC1*      mixed 72     2:18:2 386590        0      1 location none
TC1N03         1 UGGPU-TC1*       idle 72     2:18:2 386587        0      1 location none
TC1N04         1 UGGPU-TC1*       idle 72     2:18:2 386587        0      1 location none
TC1N05         1 UGGPU-TC1*       idle 72     2:18:2 386587        0      1 location none
TC1N06         1 UGGPU-TC1*       idle 72     2:18:2 386587        0      1 location none
TC1N07         1 UGGPU-TC1*       idle 72     2:18:2 386587        0      1 location none
-------------------------------------------------------------------------------
Note: Each Node is equipped with x3 GPU cards and each job can only utilize x1 GPU.
Thus, Max number of Job can compute with GPU in each Node = 3
GPU Card Sequence: [1]-IDX:0, [2]-IDX:1, [3]-IDX:2
-------------------------------------------------------------------------------
######= [ TC1N01 ]@ 2025-07-22-18-03  =##############################################
# User | JobID | JobName | QoS | ReqTRES | State | Start | Elapsed | NodeList #
-------------------------------------------------------------------------------
```

4. Create/Edit your SLURM job script

You may use the text editor available on your Desktop *(e.g., Notepad in Windows, Sublime Text in Mac, gedit in Linux OS and etc*) to create the job script. Ensuring the job script is saved with file extension "***.sh*** ". Then, transfer to your home directory in the GPU Cluster to execute.

Common encounter - > Incorrect file format

In Linux System, the file extension may not determine the actual data format and type for the file. The user has to verify by the Linux command "***file***" to know the actual file type for a file

Online Reference: *https://www.linuxfordevices.com/tutorials/linux/file-command-in-linux*

Whenever you transferred the script file from your host machine to TC1, it is advisable to examine the content of the file in TC1 before executing. The common issue encountered by the users in TC1 is unknown characters being found in the job script file and thus failed to submit job.

The example below shows the unknown boxes replaced the characters "**- -**" in a job script.

```
#!/bin/sh
#SBATCH □□partition=SCSEGPU_UG
#SBATCH □□qos=normal
#SBATCH □□gres=gpu:1
#SBATCH □□nodes=1
#SBATCH □□mem=10G
#SBATCH □□job□name=run1
#SBATCH □□output=output_%x_%j.out
#SBATCH □□error=error_%x_%j.err

module load anaconda
source activate testenv
jupyter□notebook □□ip=$(hostname □i) □□port=8886
```

For this case, you have to edit the file in TC1, to correct those unknown characters.

In TC1 Head Node, you may use those linux command line editors to edit your job script: ***vi***, ***nano*** or ***ne***

Online References on Linux Command-line Editors:
https://www.guru99.com/the-vi-editor.html
https://www.geeksforgeeks.org/nano-text-editor-in-linux/
https://itsfoss.com/command-line-text-editors-linux/

The following are the basic job flags to apply in the job script

*The flags with * must specify in the job script*

Resource	Flag Syntax	Description
Partition*	--partition=UGGPU-TC1	# Specify the node group for your job execution, defines as *Partition* # Command to list the available Partitions and allowed QoS in the Cluster ***>> scontrol show partition***
QoS*	--qos=normal	# Specify the QoS to apply for the job # The QoS must be one of them allow to execute in the Partition
GPU*	--gres=gpu:1	# Specify the use of GPUs on compute nodes. # The number of GPU card to use is corresponding to the assigned QoS # This option is to be in the job script, to deploy GPU Card for your execution # *You can omit this flag if confirmed not using the GPU Card for computation*
Memory*	--mem=8000M Or --mem=8G	# Specify on the memory to apply for the job. # The value is corresponding to the assigned QOS, must not exceed the assigned maximum memory size. # This option must be present in the job script, for the system to deploy the memory for the computation.
CPUs/cores	--ntasks-per-node=2	# ***Optional***: only add this flag if want to apply more than one core for the computation # Specify the number of "tasks" (cores) per node, for use with distributed parallelism. *Default value=1*
	--cpus-per-tasks=2	# ***Optional***: only add this flag if want to apply more than one core for the computation # Specify the number of CPU-cores to allocate to per task, for multi-threaded tasks. *Default value=1*

nodes*	*--nodes=1	# Number of compute nodes for the job # Max node number allows per job = 1
	--nodelist=TC1Nxx	# ***Optional***: you may specify the node (hostname: *TC1Nxx*) to run the job where xx: 01, 02, 03, 04, 05, 06 or 07 # Command to list those idle nodes and their hostname in the Partition ***>> sinfo -N | grep idle*** *# SLURM will auto deploy your job to any ready node if this flag is not set.* *# Advise to set this flag with an idle node only if your job requiring high CPU or memory.*
Time*	--time=60	# Specify a time limit for the job # Time format: *<min>* or *<min>:<sec>* or *<hr>:<min>:<sec>* or *<days>-<hr>:<min>:<sec>* or *<days>-<hr>*
Job name*	--job-name=MyJob	# Name of job
Output file*	--output=output_%j.out Or --output=output_%x_%j.out	# State the name of the file for standard output ***Filename patterns:*** ***%x: job name*** ***%j: job id, generated by SLRUM***
Error file*	--error=error_%j.err Or --error=error_%x_%j.err	# State the name of the file for error log – if any

The following is an example of a standard job script: job.sh

```
#!/bin/bash
#SBATCH --partition=UGGPU-TC1
#SBATCH --qos=normal
#SBATCH --nodes=1
#SBATCH --gres=gpu:1
#SBATCH --mem=8G
#SBATCH --time=60
#SBATCH --job-name=MyJob
#SBATCH --output=output_%x_%j.out
#SBATCH --error=error_%x_%j.err



module load cuda/12.2
module load anaconda
source activate TestEnv
python test.py
```

Remarks:
You may add necessary flags from the table above. Do note that if you exceed the limit for your QOS, the job will not run.
From this job script, 2 files will be generated upon running:
output_%x_%j.out
- *The standard output from running the code will be saved here*

error_%x_%j.err
- *Error log from job if any*

Load the necessary modules to run the code
Activate your own conda environment (if any)
Run your code

Question: Can I adjust the GPU Memory in the job script?
The GPU cards [NVIDIA Tesla V100 32GB] in TC1 are with fixed memory of 32GB, not possible to adjust in the job script.
Please search online for guide to utilize the desired GPU memory size in your coding.

Online References:
https://www.tensorflow.org/guide/gpu
https://wiki.ncsa.illinois.edu/display/ISL20/Managing+GPU+memory+when+using+Tensorflow+and+Pytorch

5. Submit your job script

Action to submit a job	Command
To submit your job script, where job.sh is the name of your job script	**sbatch job.sh**

The system should respond with a job ID:

```
[scetest5@CCDS-TC1:3 ~/jobscript]$ sbatch hellosumbit.sh
Submitted batch job 120
```

6. Avoid using the command "*srun*" to submit job

The command "***srun***" is to submit job at the command line for ***real time execution***. You have to maintain your SSH session until the entire process completed. The disconnection of your SSH session may kill the process and causing you to lose the control over the execution. Thus, for the jobs requiring more than an hour to compute, are advised to submit using the command "***sbatch***".

The command "***sbatch***" is to submit job for later execution, handling by SLURM in the background.
Once submitted the job using "***sbatch***", you may exit from your SSH session, with no consequence.

*To avoid high volume of SSH connections to TC1 Head Node, all users are advised to use the command "**sbatch**" for job submission. Then exit from the session, access later to see the result.*

7. Verify on the Node operational status in TC1

Command to verify on the node status:
>> sinfo
Online Reference for SLURM sinfo: *https://slurm.schedmd.com/sinfo.html*

```
NODELIST   NODES  PARTITION       STATE CPUS    S:C:T MEMORY TMP_DISK WEIGHT AVAIL_FE REASON
TC1N01         1 UGGPU-TC1*      mixed 72     2:18:2 386590        0      1 location none
TC1N02         1 UGGPU-TC1*      down* 72     2:18:2 386590        0      1 location Not responding
TC1N03         1 UGGPU-TC1*      mixed 72     2:18:2 386590        0      1 location none
TC1N04         1 UGGPU-TC1*      down* 72     2:18:2 386590        0      1 location Not responding
TC1N05         1 UGGPU-TC1*      mixed 72     2:18:2 386590        0      1 location none
TC1N06         1 UGGPU-TC1*      mixed 72     2:18:2 386590        0      1 location none
TC1N07         1 UGGPU-TC1*       idle 72     2:18:2 386590        0      1 location none
```

Verify the nodes of TC1 via the following common status showing under "***STATE***"

STATE	Description
idle	The node is alive and free to accept job
mixed	There are jobs running in that node
down	The SLRUM service in that node has been down
completing	The node is in the process to complete the final job computation
mixed+drain	There are jobs still running in the node. The 'draining' means there are failed tasks unable or still in progress to terminate. The result will be degrading in system performance and causing other running jobs in the node to slow down or hang. For this case, if the administrator resets the affected node, the existing running jobs in the affected node will be terminated. The affected jobs will join the PENDING queue for re-assignment.
draining	
drained	The node has failed task, unable to run any job. Requiring the administrator to verify and reset the affected node.

To know the overall status of the nodes in TC1, you may refer to those log files under ***/tc1share2/Log-TC1Status***
File format: ***ActiveNode_YYYY-MM-DD-HH-mm***
Example to view the status of the active node(s) (with running job) at specific hour of the day e.g., 3pm
>> more /tc1share2/Log-TC1Status/ActiveNode_2025-01-14-15-*

Example to view the log files of Active Node on 1st Jan 2025
>> more /tc1share2/Log-TC1Status/ActiveNode_2025-01-01-*

To view the overall node status in TC1
>> more /tc1share2/Log-TC1Status/TC1_ClusterInfo

8. Verify on the jobs processing in TC1

Command to verify the current jobs and their status in the queue:

>> squeue

***Online References for SLURM squeue**: https://slurm.schedmd.com/squeue.html*

Actions to check on job status	Command
Display the jobs in the scheduling queue	**squeue -la**
Display job history for a user <username>	**sacct -u <username>**
Show the detail for a running/pending job <jobid>	**scontrol show jobid <jobid>**

```
[scetest5@CCDS-TC1:3 ~]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               119 UGGPU-TC1 twogpute scetest6 PD       0:00      1 (QOSMaxGRESPerUser)
               120 UGGPU-TC1 hello-wo scetest5 PD       0:00      1 (QOSMaxWallDuration
               114 UGGPU-TC1       J1 scetest5  R      47:48      1 TC1N01
               116 UGGPU-TC1 twogpute scetest6  R       3:47      1 TC1N02
[scetest5@CCDS-TC1:3 ~]$
```

In the SLRUM job squeue, the job failed to run will show with ***ST*** (STATE) "*PD*" and reason coded under **NODELIST *(REASON)***. The job may fail due to following reasons:

- *The required resources stated in your job script have exceeded the limit set in QoS*
- *TC1 is unable to allocate the required resources for job execution*
- *The job being terminated half-way due to system failure or resource shortage*

The Resource in TC1 is referring to those GPU Nodes and their available CPU, Memory, and GPU Cards.
The resource allocation is first-come, first-served basis, based on the value stated in the job script. The jobs allocated with the required resources and with no coding issue, may show **STATE** "*R*" and executing in the assigned node stating under **NODELIST (REASON)**.

The SLURM in TC1 will verify the available resource to allocation at the time of job submission. If failed to allocate the requested node and resources, the **STATE** of the job will change to "PD".

Some common reasons for pending jobs associated to QoS failure (*exceeded any of the values stated for the QoS*):

- ***QOSMaxWallDurationPerJobLimit***
  Verify on the flag [**#SBATCH --time=xx**], ensure it does not exceed the *MaxWall* of the QoS
- ***QOSMaxJobsPerUserLimit***
  Verify on your total number of jobs in the queue, must not exceed the value stated in "**MaxJobsPU**".
  The subsequence job may go into the PENDING queue or being rejected.
- ***QOSMaxMemoryPerUser***, ***QOSMaxCpuPerUserLimit***, ***QOSMaxGRESPerUser***
  The total resource counts in [**#SBATCH --mem=x**, **#SBATCH --ntasks-per-node=x**] stated in your submitted job scripts using the same QoS, must not exceed the max value stated in "**MaxTRESPU**".
- ***QOSMaxGRESPerUser***
  One(x1) GPU card can only run one(x1) job. This error will appear if you set to use the GPU card in your subsequence job using the same QoS with **MaxTRESPU** stating "gres/gpu=1". If your computation does not require the GPU, you may omit the flag [**#SBATCH --gres=gpu:1**] in your job script.

For jobs unable to run or being terminated half-way due to *system failure or resource shortage*, they will go into PENDING queue, and showing following reason codes:

- ***Priority***
  The job is waiting with higher priority, will run eventually when the required node and resources are available
- ***Resources***
  The job is pending for available resources, will run eventually when obtains the allocation

The pending jobs will be removed from the job queue if the waiting time has exceeded the Time Limit (referenced to **MaxWall** stated in the QoS).

Advice for users with jobs hanging in the PENDING queue:

- *Cancel your pending jobs failed the QoS limit Check*
- *Review and modify the resource assignment (CPU and Memory) in your job script to acceptable value to facilitate SLURM to assign to available node*
- *Restrict your number of job submission*

Example to cancel the job	Command
To cancel a job, where 263 is your job id	scancel 263

Once the job execution is completed, it will be removed from the queue. You may see the **output files** for the computation result. If you did not specify a specific path for the output file in your job script, it will be created under the folder where you submitted the job.

```
[scetest5@CCDS-TC1:3 ~/tc1-jobs]$ ls
error_J1_114.err  J1.sh  output_J1_114.out
[scetest5@CCDS-TC1:3 ~/tc1-jobs]$
```

Based on the following flags set in the job script, two output files are being generated.
#SBATCH --output=output_%x_%j.out
#SBATCH --error=error_%x_%j.err

In above example, the two output files are
Output_MyJob_xxxx.out
Error_MyJob_xxxx.err
Commands to view the content of the file at command line: ***cat***, ***less***, ***more*** and ***tail***
Online Reference: *https://www.cyberciti.biz/faq/unix-linux-command-to-view-file/*

9. Know more about your submitted jobs

You may use the customised script to view your job history for the day, *with details on applied QoS, allocated resources (cpu, gpu, memory assigned for the job), node assigned, job-submit time, start time, elapsed time, state, and reason*
>> MyJobHistory

```
[scetest5@CCDS-TC1:3 ~]$ MyJobHistory
-- List Job History for [ scetest5 ]----------------------------------------------------
| User | jobid | jobname | qos | allocResources | start | elapsed | nodelist | state | reason
  scetest5      104 twogputestj+ q_ug3x24              billing=2,cpu=2,gres/gpu=3,node=1 202
  scetest5      105 twogputestj+   normal              billing=2,cpu=2,gres/gpu=1,node=1 202
  scetest5      107 hello-world+   normal                        billing=6,cpu=6,node=1 202
  scetest5          114           J1   normal          billing=2,cpu=2,gres/gpu=1,node=1 202
  scetest5      120 hello-world+   normal
```

 # Know the CPU and Memory being utilized by your Completed Job #
Command to display the resource utilization statistics for your completed job, let you review and deploy the right number of resources (CPU/cores and Memory) in your job script, for subsequence computation
>> seff <job id>

```
[scetest6@CCDS-TC1:0 ~]$ seff 115
Job ID: 115
Cluster: slurm
User/Group: scetest6/scetest6
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 4
CPU Utilized: 00:00:00
CPU Efficiency: 0.00% of 00:00:24 core-walltime
Job Wall-clock time: 00:00:06
Memory Utilized: 1.32 MB
Memory Efficiency: 0.00% of 16.00 B
[scetest6@CCDS-TC1:0 ~]$
```

 # Examples of the utilization showing high usage on CPU and Memory

```
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 00:02:16
CPU Efficiency: 144.68% of 00:01:34 core-walltime
Job Wall-clock time: 00:01:34
Memory Utilized: 2.54 GB
Memory Efficiency: 25.40% of 10.00 GB
```

```
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 00:44:59
CPU Efficiency: 103.05% of 00:43:39 core-walltime
Job Wall-clock time: 00:43:39
Memory Utilized: 278.29 GB
Memory Efficiency: 434.83% of 64.00 GB
```

 # The user should review own coding and adjust the values of CPU and Memory in the job script to enhance the performance in subsequence computation

 # Know the actual GPU Card being assigned to your Running job #
*Using the command **"scontrol", with option "-d"***
>> scontrol -d show jobid <jobid of running job>

Look for the line "JOB_GRES=gpu:x "

```
NodeList=TC1N04
BatchHost=TC1N04
NumNodes=1 NumCPUs=4 NumTasks=4 CPUs/Task=1 ReqB:S:C:T=0:0:*:*
ReqTRES=cpu=4,mem=10G,node=1,billing=4,gres/gpu=1
AllocTRES=cpu=4,node=1,billing=4,gres/gpu=1
Socks/Node=* NtasksPerN:B:S:C=4:0:*:* CoreSpec=*
JOB_GRES=gpu:1
  Nodes=TC1N04 CPU_IDs=0-1,36-37 Mem=0 GRES=gpu:1(IDX:0)
MinCPUsNode=4 MinMemoryNode=10G MinTmpDiskNode=0
Features=(null) DelayBoot=00:00:00
```

 # In each Compute Node, the GPU Cards are identified by SLURM as *IDX:0*, *IDX:1* and *IDX:2*
 # The above example showing the job is executing on GPU Card 1 in the node TC1N04

 # Know the utilization rate of the GPU Card assigned to your Running job #

The actual GPU utilization information in TC1 requiring the root access to retrieve from the active nodes.
The information is being retrieved using the command "nvidia-smi" from respective Compute node

Queries Option	Description
name	Official Product Name of the GPU Card
pci.bus_id	PCI Bus ID
driver_version	The version of the installed NVIDIA Display Driver
pstate	Current Performance state of the GPU Card, ranging from P0 (maximum performance) to P12 (minimum performance)
power.limit	Maximum Power [Watt] can be draw by the GPU
power.draw	Power [Watt] being draw by the GPU for operation
** **temperature.gpu**	Core GPU Temperature, in Degrees C
memory.total [MiB]	Total installed GPU Memory
memory.used [MiB]	Total Memory being allocated by the active contexts
* **utilization.gpu [%]**	Percent of time over the past second during which one or more kernels was executing on the GPU
* **utilization.memory [%]**	Percent of time over the past second during which global (device) memory was being read or written

**** Utilization rates report how busy the GPU is over time, and determine how much the assigned job has been using the GPU.***

```
TC1N01: name, pci.bus_id, driver_version, pstate, power.limit [W], power.draw [W], temperature.gpu, memory.total [MiB],
TC1N01: Tesla V100-PCIE-32GB, 00000000:37:00.0, 565.57.01, P0, 250.00 W, 145.42 W, 72, 32768 MiB, 3594 MiB, 95 %, 30 %
TC1N01: Tesla V100-PCIE-32GB, 00000000:86:00.0, 565.57.01, P0, 250.00 W, 35.20 W, 29, 32768 MiB, 1 MiB, 0 %, 0 %
TC1N01: Tesla V100-PCIE-32GB, 00000000:D8:00.0, 565.57.01, P0, 250.00 W, 35.65 W, 35, 32768 MiB, 1 MiB, 0 %, 0 %
```

***** Optimal Temperature of the GPU card should stay below 85 ˚C. If the temperature hits 90 ˚C and with GPU Utilization Rate stays at 100%, vital action must be executed to terminate the computation in the affected node.***

In TC1, the usage records are being captured at scheduled timing and keep under the following folders:

S/N	Log Directory and File format	Remark
1	**/tc1share2/Log-RunHistory** Log file: Running_YYYY-MM-DD-HH-mm	Records of running job detected for past 24H, showing the actual GPU being utilized by the job at 5-mins interval
2	**/tc1share2/Log-JobHistory** Log file: JobHistory_YYYY-MM-DD-HH-mm	Records of jobs being computed for past 48H
3	**/tc1share2/Log-TC1Status** Log file 1: TC1_ClusterInfo Log file 2: ActiveNode_YYYY-MM-DD-HH-mm	Records of the overall resource utilization in TC1 for past 28H

You may use the command "***more***" to view those files for the day e.g.,

>> more /tc1share2/Log-JobHistory/JobHistory_2025-01-09-*

 # Do your part to enhance the efficiency of your computation and reduce resource wastage

The job execution and resource assignment are first-come, first-served basis. The resources are allocated according to the flags and values stated in your job script. For the jobs submitted after you, the resource will be allocating based on the balance (leftover). Thus, please help to allocate and utilize the resource reasonably for your computation.

Most users often begin with predefined resource allocations for their tasks, aiming to strike a balance between performance and efficiency. However, this can lead to two extreme outcomes: degradation in performance if resource utilization exceeds 100%, or wastage if utilization remains below 10% of the allocated resources.

The resource utilization statistics (*retrieve using command " **seff** "*) enable you to know the actual amount of CPU and memory being deployed for your computation, allowing you to verify and set reasonable values in the job script to enhance subsequence computation.
The log files in the above Log Directories "*/tc1share2/Log-* *" enable you to know whether you have been utilized the GPU reasonably for your past computation.

Shared Resources

There are share conda environments ready under ***/tc1apps/2_conda_env/*** .

S/N	Name of Shared Conda Environment	Apps installed	Remarks
1	/tc1apps/2_conda_env/CZ4042_v4	Python 3.10 Pytorch 2.4 Pytorch-Cuda 12.4 Cudnn 8.9 Tensorflow 2.9 Cuda-toolkit 11.8	**Configured Conda Environment Variable** *(For Tensorflow 2.9)* TF_ENABLE_ONEDNN_OPTS=0 >> conda env config vars set TF_ENABLE_ONEDNN_OPTS=0
2	/tc1apps/2_conda_env/CZ4042_v5	Python 3.11.4 Pytorch 2.0.1 Pytorch-Cuda 11.7 Jupyter	
3	/tc1apps/2_conda_env/CZ4042_v6	Python 3.12.3 Pytorch 2.5.1 Pytorch-Cuda 11.8 Jupyter	*This is the latest Conda Environment created for the course SC4001/CZ4042* The exported YML file "*SC4001.yml*" is available under ***/tc1apps/3_conda_yml/***. You may copy the yml file and setup similar conda environment in own home directory with following commands, naming as "*mySC4001*" >> cd >> cp /tc1apps/3_conda_yml/SC4001.yml . >> conda env create -f SC4001.yml -n mySC4001
4	/tc1apps/2_conda_env/Jupyter	Python 3.12 Jupyter	Jupyter lab + notebook

To launch Jupyter lab from the share environment [/tc1apps/2_conda_env/Jupyter] , you may input the following lines in your job script for job submission. Upon the Jupyter being launched, setup the *SSH Tunneling* for the access.

```
module load anaconda
source activate /tc1apps/2_conda_env/Jupyter
jupyter-lab --ip=$(hostname -i) --port=8881
```

Note for Jupyter Users

Jupyter is a web-based computing platform to facilitate the user on code development, analyse and debugging.

*(You may refer to the next section "**Miscellaneous**" for exemplary guides on Jupyter setup and features to access)*

Usually, the user may exit the SSH session after submitted the job via the command SBATCH. Then, access later to retrieve the job result.
The Jupyter user must setup and maintain the ***SSH Tunneling*** in the SSH session for the web access and can only exit the session after terminating the Jupyter service in the Web Browser or cancel the job manually.

The default port for Jupyter is 8888. You may change to other port number if it has been in-used by other users in TC1.
You may verify whether your desired port number is being in-used via one of the following commands in TC1 Head Node:
*>> **netstat -tp** | grep <port number>*

```
[xxxxxxx@CCDS-TC1:tcljobs]# netstat -tp | grep 888
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 CCDS-TC1.cm.clust:36072 TC1N02:8885             ESTABLISHED -
tcp        0      0 CCDS-TC1.cm.clust:36004 TC1N02:8885             ESTABLISHED -
tcp        0      0 CCDS-TC1.cm.clust:36040 TC1N02:8885             TIME_WAIT   -
tcp        0      0 CCDS-TC1.cm.clust:36048 TC1N02:8885             TIME_WAIT   -
tcp        0      0 CCDS-TC1.cm.clust:36018 TC1N02:8885             ESTABLISHED -
tcp        0      0 CCDS-TC1.cm.clust:38328 TC1N02:8885             ESTABLISHED -
tcp        0      0 CCDS-TC1.cm.clust:36024 TC1N02:8885             TIME_WAIT   -
tcp        0      0 CCDS-TC1.cm.clust:36056 TC1N02:8885             TIME_WAIT   -
[xxxxxxx@CCDS-TC1:tcljobs]#
```

If your port number is not found in the output, then it means that port number is free for use.

For the applications to execute for own computation, you should setup and install in own conda environment via command-line.
When your coding is ready for computation, you should execute directly via the job script submission, not launching via Jupyter.

Miscellaneous

SSH Tunneling

The GPU Cluster is only accessible via SSH connection. For running *Jupyter Lab or Notebook*, the user must setup the SSH tunnel for the access.

SSH tunneling (also known as ***SSH port forwarding***) is a method of creating an encrypted SSH connection between a client and a server through which services port can be relayed.

Exemplary guide to create own conda environment to launch Jupyter:

- *Using Putty in Windows Desktop to access TC1, setup own conda environment for Jupyter.*
- *Submit the job script to launch Jupyter, obtain the access info from the error log file.*
- *Setup the SSH Tunnel in PUTTY, then copy and paste the URL to the web browser to access.*

1. **SSH to GPU Head Node and setup a conda environment in your home directory**

Description	Execution
Load the anaconda module	# Execute at command line **>> module load anaconda**
Create and activate the new conda environment	**>> conda create -n RunJupyter** **>> conda activate RunJupyter**
Install the required package for Jupyter lab and notebook ***Online References:*** *https://jupyter.org/install* *https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html* *https://www.dataquest.io/blog/jupyter-notebook-tutorial/*	**>> conda install conda-forge::jupyter** **>> conda install -c conda-forge nb_conda_kernels**
Deactivate/Exit the conda environment	**>> conda deactivate**

2. **Create the job script and submit**

Description	Execution
Create the job script to run Jupyter. For this example, the job script is named as "***run1.sh***"	*# Sample of the job script* *#!/bin/sh* *#SBATCH --partition=UGGPU-TC1* *#SBATCH --qos=normal* *#SBATCH --mem=10G* *#SBATCH --gres=gpu:1* *#SBATCH --time=360* *#SBATCH --nodes=1* *#SBATCH --job-name=run1* *#SBATCH --output=output_%x_%j.out* *#SBATCH --error=error_%x_%j.err* *module load anaconda* *source activate RunJupyter* *<HERE add in the Command to launch Jupyter Lab or Notebook>*
Command to launch Jupyter-Notebook, *where XXXX is your desired port number*	*jupyter-notebook --ip=$(hostname -i) --port=XXXX*
Command to launch Jupyter-Lab, *where XXXX is your desired port number*	*jupyter-lab --ip=$(hostname -i) --port=XXXX*
Submit the job script	*# Execute at command line* **>> sbatch run1.sh**

3. **Verify the access information in the error log file**

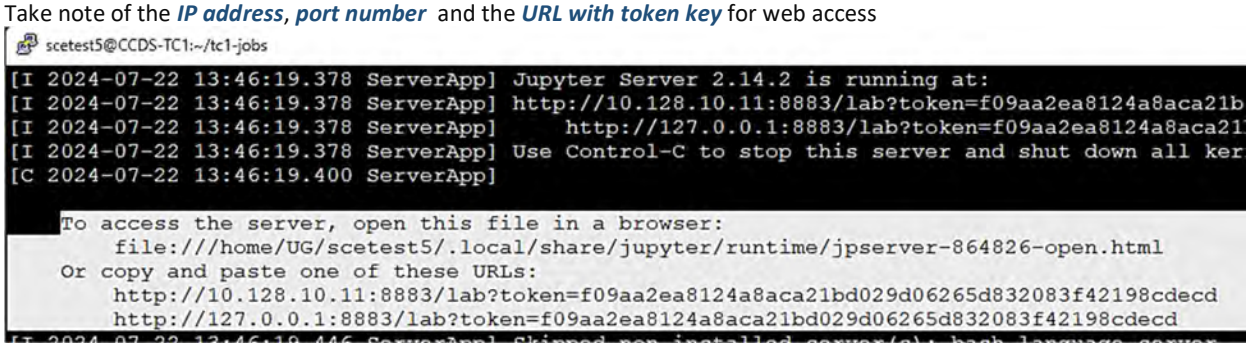
The error log file is being generated under the folder where you executed the job script: error_run1_xxxx.err.

Use the command " ***tail*** " to view the access info available at the bottom end of the error log file

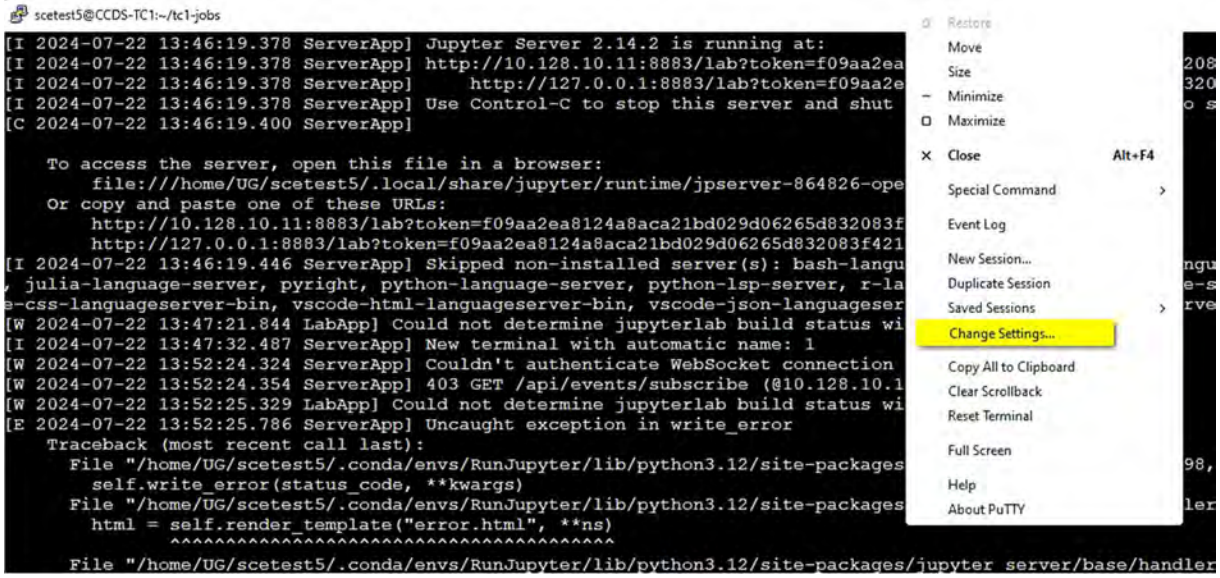
>> tail error_run1_xxxx.err

Online Reference: *https://www.geeksforgeeks.org/tail-command-linux-examples/*

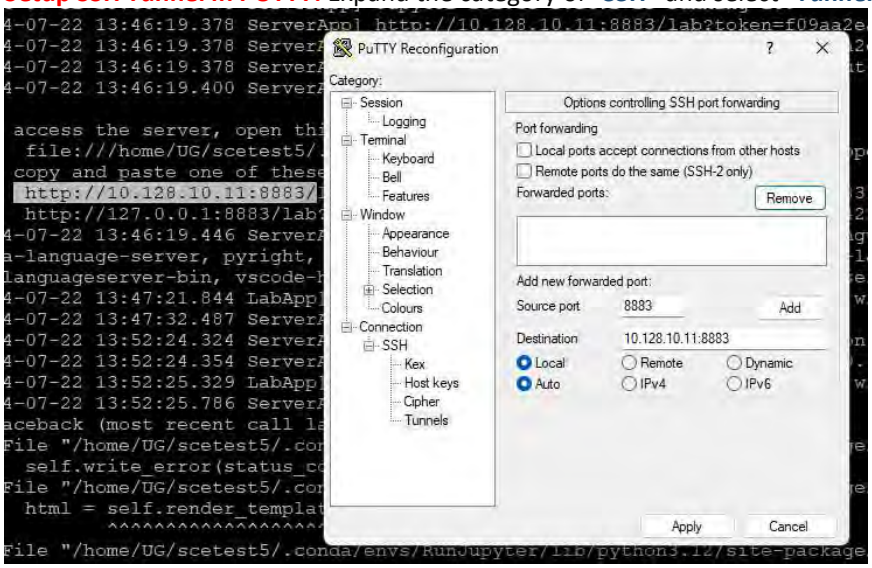
Take note of the ***IP address***, ***port number*** and the ***URL with token key*** for web access



4. ***Mouse-Right-Click*** on your current SSH session (*top bar*) and select "***Change Settings…***"



5. **Setup SSH Tunnel in PUTTY:** Expand the category of "***SSH***" and select "***Tunnels***"



Take note of the IP address and port number showing in the first URL "http://10.128.10.11:8883/tree... ".
Input the info to following fields accordingly
***Source port**: 8883*
***Destination**: 10.128.10.11:8883*

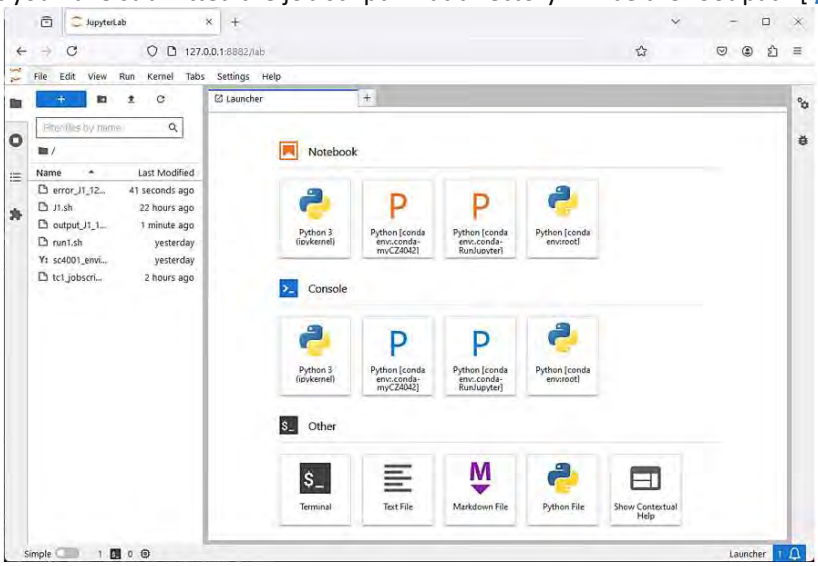
*Click on the button "**Add**" to add the new forwarded port. The added info should appear in the box of "Forwarded ports".*
*Then, click on "**Apply**" for the change to take effect*

6. You may **access the second URL in your local machine**

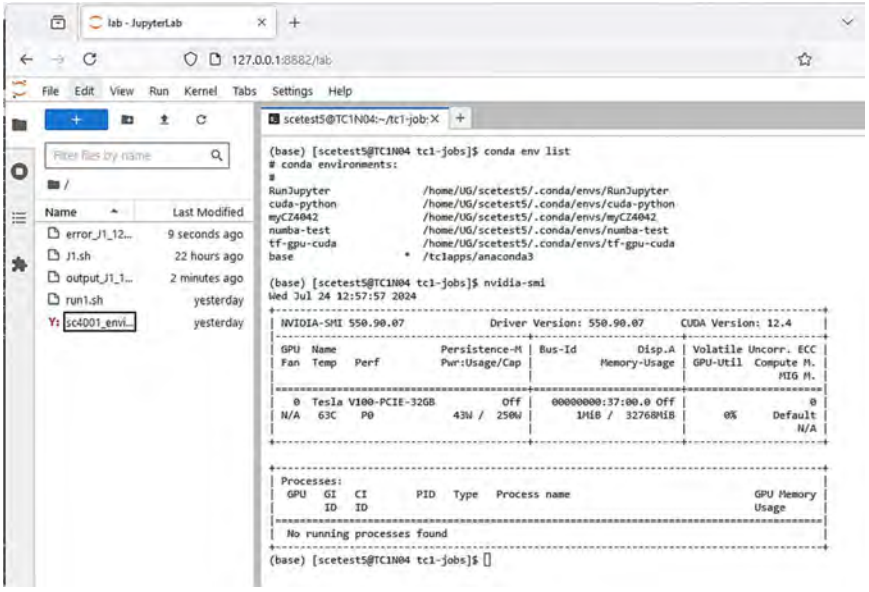
```
To access the server, open this file in a browser:
    file:///home/UG/scetest5/.local/share/jupyter/runtime/jpserver-864826-open.html
Or copy and paste one of these URLs:
    http://10.128.10.11:8883/lab?token=f09aa2ea8124a8aca21bd029d06265d832083f42198cdecd
    http://127.0.0.1:8883/lab?token=f09aa2ea8124a8aca21bd029d06265d832083f42198cdecd
```

Copy and paste the *second* URL to your web browser for access *http://127.0.0.1:8883/lab?token=.....*.

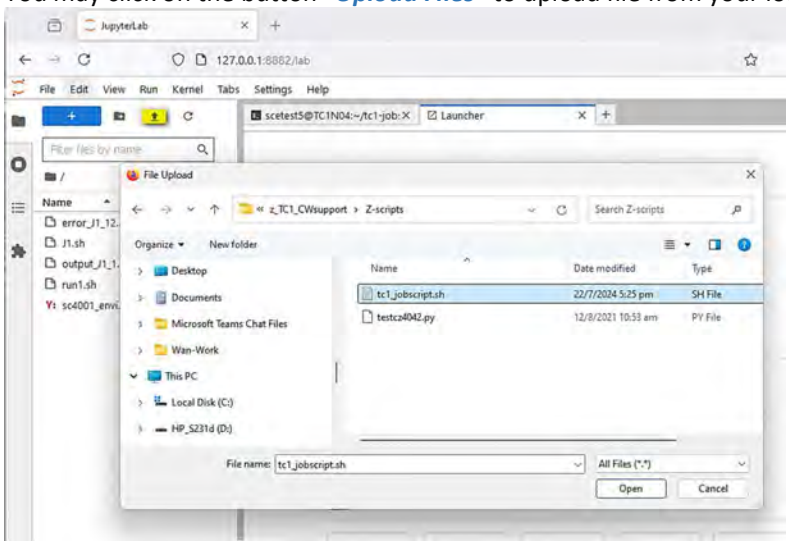
The display in your browser, upon successful launching, will be explorer-view, showing the files in the directory that you have submitted the job script. That directory will be the root path [**/**] for your access in Jupyter.



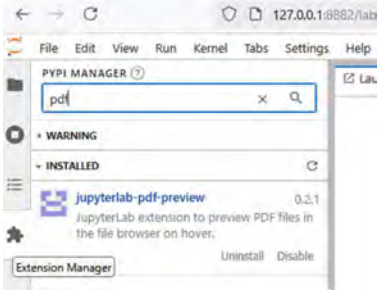
You may click on the Button "***Terminal***" under section "*Other*" to execute at command line.
You may launch (activate) your other conda environments to test in the terminal.



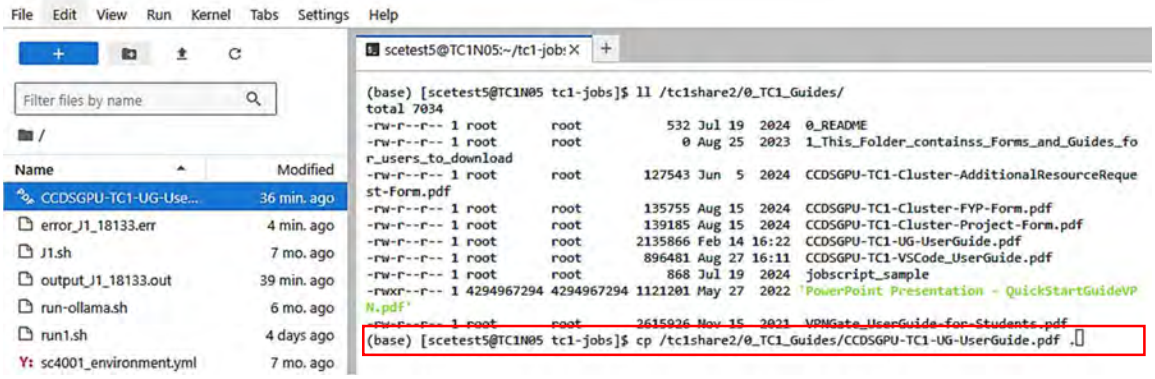
8. You may click on the button "***Upload Files***" to upload file from your local drive to the root path [**/**].



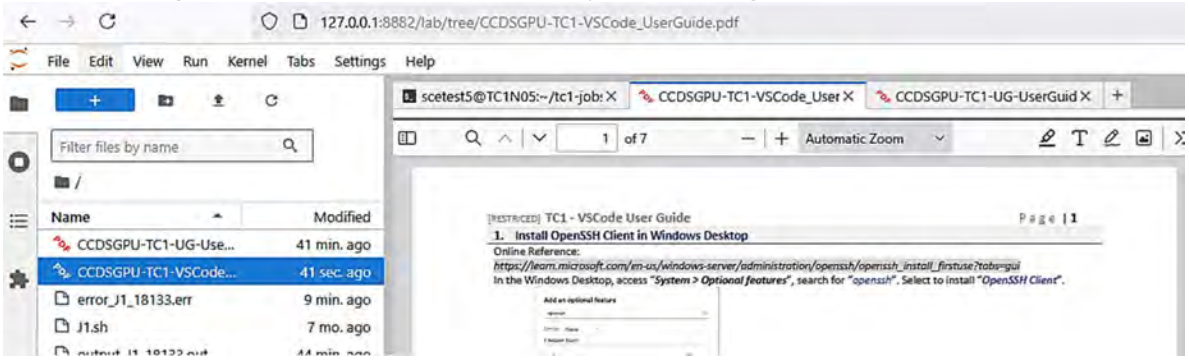
9. You may install the Extension "***jupyterlab-pdf-preview***" to view the PDF User Guides available in TC1.
   - Install the required extension under Extension Manager.



   - Copy the user guides available under /tc1share2/0_TC1_Guides to your root path [**/**] to view.



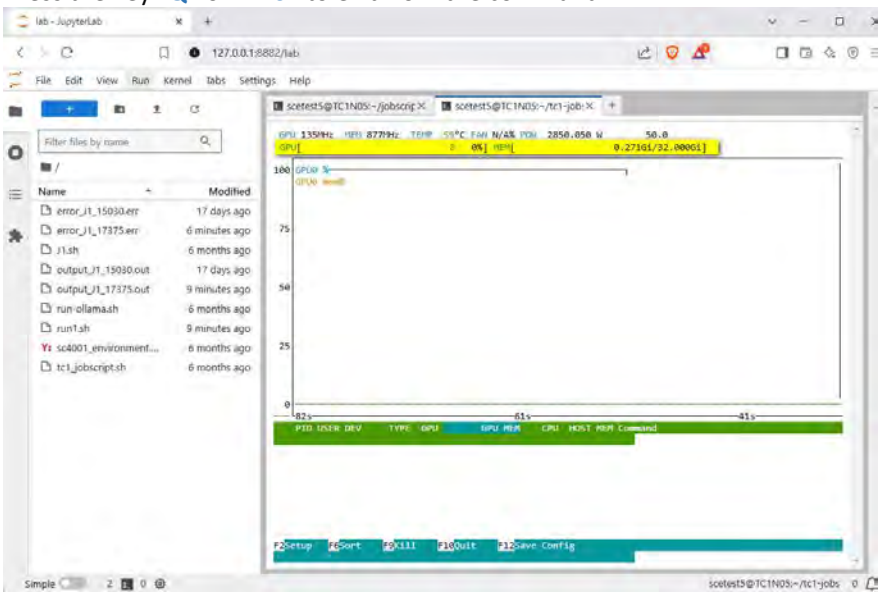
   - Then, mouse-right-click on the selected PDF file to open for viewing.



10. You may use the command "***nvtop***" to monitor the GPU real-time usage

Online reference: https://www.cyberciti.biz/hardware/nvtop-command-in-linux-to-monitor-nvidia-amd-intel-gpus/

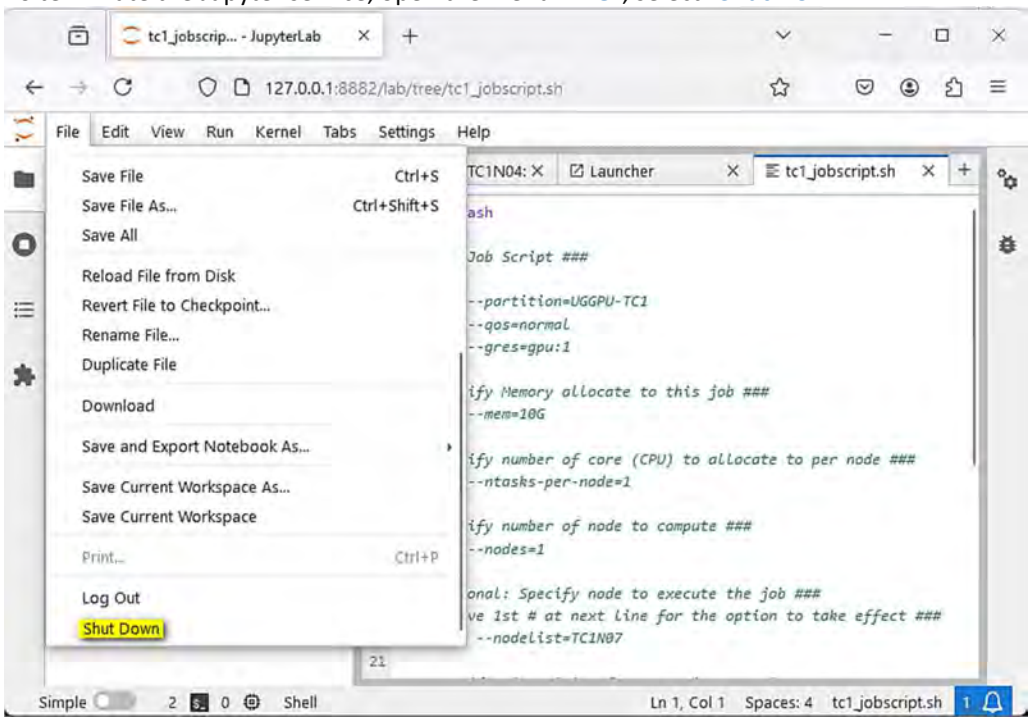
- Launch another terminal, execute the command "***nvtop***".
- You may monitor the ***GPU*** and ***MEM*** being utilized by your process execution.
- Press the key "***Q***" or "***F10***" to exit from the command.



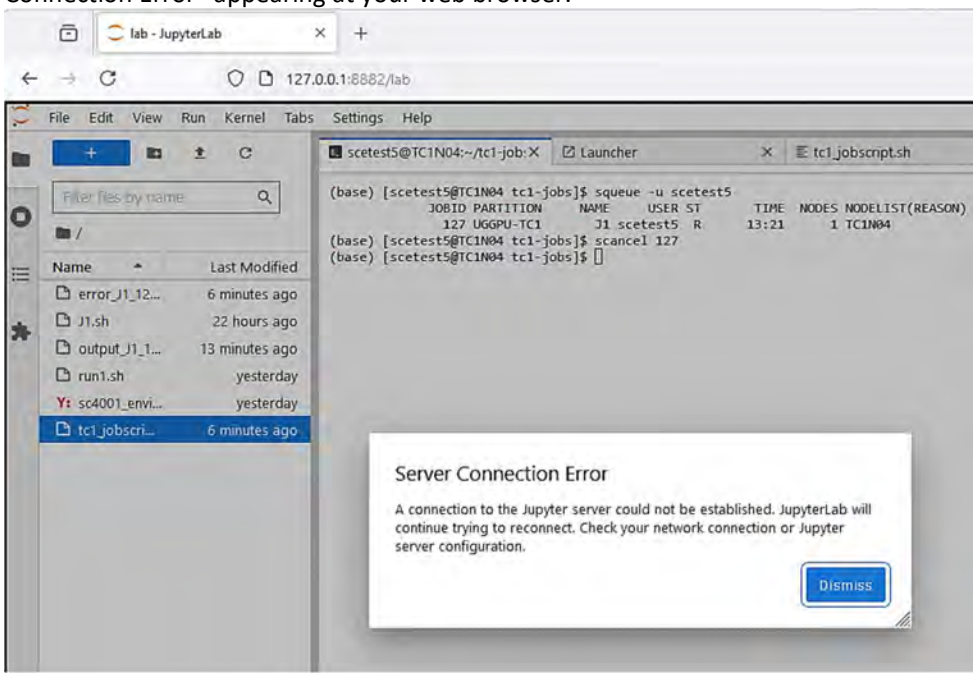
11. Other system commands that you may explore to verify on Node system status

S/N	Command + Option	Remark
1	***free -h***	Display summarized Memory and Swap info, in human readable format
2	***mpstat***	Report CPU related statistics
3	***top -i***	Show real-time view of running processes
4	***ps -u***	Display own processes

12. To terminate the Jupyter service, open the menu "**File**", select "**Shut Down**".



13. OR terminate by cancel the SLURM job in the Terminal. Once your job being cancelled, you may see "Server Connection Error" appearing at your web browser.



14. For best practice, please help to cancel your job when no longer in use. This is to release the resources (*the allocated CPU, Memory and GPU being assigned to your job*) back to the pool for other to use.

Note
For MAC and Linux user establishing the ssh session at command line, you may have to search online for command to setup SSH tunnelling at command line. OR you may install other SSH Clients to facilitate the connection:
https://www.ssh.com/academy/ssh/putty/mac
https://termius.com/free-ssh-client-for-mac-os

Other Resources for application

QoS

The user may apply additional QoS for…

- longer computation hour, selecting range: 8-hr, 12-hr, 24-hr, till max 48-hr
- computation with more resource allocation: CPUs, Memory, and GPU Card

Usage period: not more than 1-month

Method to apply:

- Must provide valid *Reason for the usage* and *Verification (latest coding and computation result executed in TC1 with the assigned QoS), to prove that the existing resource unable to support the user's requirement for computation*
- Send the request e-mail to CCDSGPU-TC-Support [***CCDSgpu-tc@ntu.edu.sg***]
- The administrator will evaluate, verify on the real requirement, and then assign the QoS accordingly
- The requestor will be notified, upon successful application

Additional Storage Space

The user may apply additional storage space for computation data, at least more than 100GB.
A share folder will be created in another storage to assign to the user.

Usage period: 4-months
Maximum usage quota: depending on the available free space and approval

Method to apply:

- Send the request e-mail to CCDSGPU-TC-Support [***CCDSgpu-tc@ntu.edu.sg***]
- Must provide the reason and verification for the usage request
- The user will be receiving the PFD form via e-mail, to complete for the application
- The user will be notified of the resource assignment, upon successful application

Terms and Conditions

- The application approval is subjecting to usage purpose, resource availability, cluster operational workload and administration criteria.
- The usage of the assigned resource will be monitored.
- The administrator claims the right to revoke the assigned resource if the usage has been verified underuse or impacting on the operation in the GPU Cluster.
- The assigned resource will be removed upon usage expiry, with no prior notification.

Important Notice

1. This User Guide is only providing general information for your kick-start in TC1. If you want in-depth information, you may visit those suggested Online References (*URL*) providing in the guide or search the internet for your required information.
2. There is NO backup provided for data stored in your home directory. You are responsible for protecting and maintaining your own data.
3. The resources in TC1 are to be used for academic purposes only. Use of resources for a purpose other than that for which they were granted will result in the termination of account.
4. Your access and computation are being monitored. For user computation impacting on the operation in TC1 and for jobs submitted with unauthorized QoS (NOT assigned for your use), will be terminated with no prior notification. Severe offender will be barred from the service.
5. DO NOT access directories other than your Home Directory and Shared folder. Unauthorised access to other directories or files, even for the purpose of "browsing", shall be deemed as a security breach. Serious action will be taken against users committing any breach to the CCDS GPU Cluster.
6. For issue regarding your coursework and project, please consult your Supervisor or TA (Teaching Assistant).
7. For technical issue, drop an email to CCDSGPU-TC-Support [***CCDSgpu-tc@ntu.edu.sg***]

~ End ~