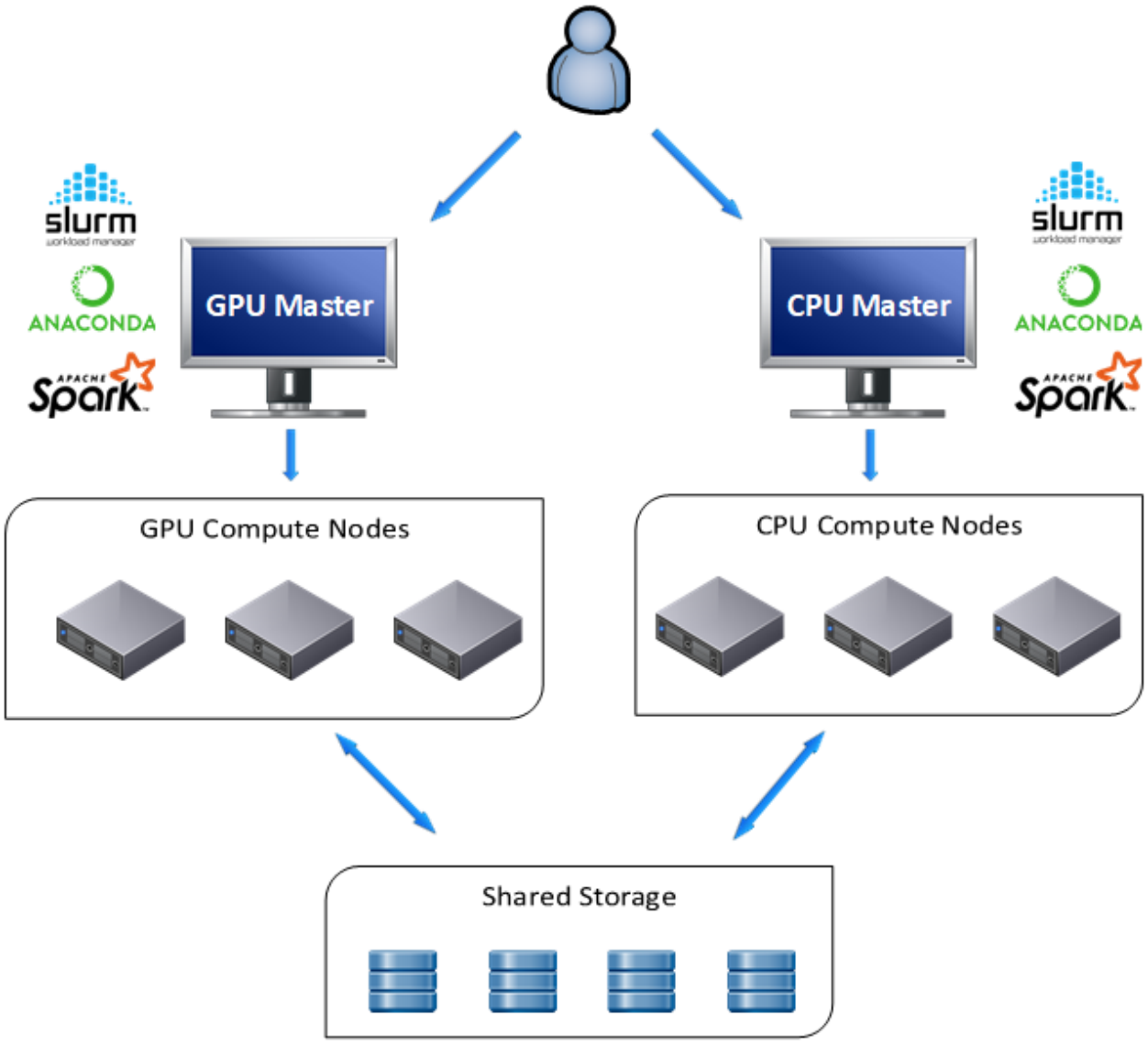
BGU ISE-CS-DT GPU Cluster User Guide

15/05/2022

Contents

Abstract

This document is located on the cluster in the '/storage' directory and being updated from time to time. Please make sure you have the most recent version. Another guide is located here: gpu.bgu.ac.il



BGU ISE, DT and CS departments have two Slurm clusters – a GPU cluster and a CPU cluster. The cluster is a job scheduler and resource manager. It consists of a manager node and several compute nodes.

The manager node is a shared resource used for launching, monitoring, and controlling jobs and should **NEVER** be used for computational purposes.

The compute nodes are powerful Linux machines, installed with GPUs.

The user connects to the manager node and launches jobs that are executed by a compute node.

A job is allocation of compute resources such as RAM memory, cpu cores, gpu, etc. for a limited time. A job may consist of job steps which are tasks within a job.

In the following pages, *Italic* writing is saved for Slurm command line commands.

Use

- Make sure you got admission to the cluster by your IT team.
- Ssh to the Manager Node: gpu.bgu.ac.il (132.72.44.112)
- Use your BGU user name and password to login to the manager node. The default path is to your home directory on the storage.
- Python users: create your virtual environment (Conda Create Environment) on the manager node.
- If you copy files to your home directory, don't forget about file permissions. E.g. files that need execution permissions, do: chmod +x <path to file>
- Remember that the cluster is a shared resource. Currently, users are trusted to act with responsibility in regards to the cluster usage – i.e. release unused allocated resources (with *scancel*), not allocate more than needed resources, erase unused files and datasets, etc. Please release the resources even if you are taking a few hours break from interactively using them.
- Anaconda3 is already installed on the cluster.
- Should you need tensorflow-gpu package, please do not use *pip install* to install. Rather use: *conda install -c anaconda tensorflow-gpu*
- Please read thoroughly, the following page or two. If you are clueless about Linux, Conda and the rest of the environment then use the Step by Step Guide for First Use page.

Submitting a Job

In order to submit a job, type:

sbatch <your batch file name>

- Conda users: Make sure you submit the job while virtual environment deactivated on the manager node ('conda deactivate')!
- Jupyter and IDE users: make sure you release the resources, after you were done, by using 'scancel <job_id>' (Cancel Jobs).

Jupyter: After submitting the job, open the output file, copy the 2nd token (https://132.72...) and paste it into your web browser's address bar.

Batch File

The batch file should look like the following:

```
#!/bin/bash

### sbatch config parameters must start with #SBATCH and must precede any other command. to ignore just add another # - like ##SBATCH

#SBATCH --partition main                    ### specify partition name where to run a job. Any node: 'main'; NVidia 2080: 'rtx2080'; 1080: 'gtx1080'

#SBATCH --time 0-10:30:00                   ### limit the time of job running. Make sure it is not greater than the partition time limit (7 days)!! Format: D-H:MM:SS

#SBATCH --job-name my_job                   ### name of the job. replace my_job with your desired job name

#SBATCH --output my_job-id-%J.out           ### output log for running job - %J is the job number variable

#SBATCH --mail-user=user@post.bgu.ac.il     ### user's email for sending job status notifications

#SBATCH --mail-type=BEGIN,END,FAIL          ### conditions for sending the email. ALL,BEGIN,END,FAIL, REQUEU, NONE

#SBATCH --gpus=1            ### number of GPUs (can't exceed 8 gpus for now) allocating more than 1 requires the IT team permission

##SBATCH --tasks=2                  # 2 processes – use for processing of few programs concurrently in a job (with srun). Use just 1 otherwise


### Print some data to output file ###

echo "SLURM_JOBID"=$SLURM_JOBID

echo "SLURM_JOB_NODELIST"=$SLURM_JOB_NODELIST

nvidia-smi -L


### Start your code below ####

module load anaconda            ### load anaconda module

source activate my_env          ### activate a conda environment, replace my_env with your conda environment

jupyter lab    ### this command executes jupyter lab – replace with your own command e.g. 'python my.py my_arg'
```

There is an example sbatch file located on cluster here: /storage/sbatch_gpu.example

Should you need the IT team's support, in your request remember to state the job id and include the sbatch file and the output file.

Allocating Resources

Since the resources are expensive and in high demand, you should make use of just **1 GPU per job**.

You should use the **minimum possible RAM**. If your code makes use of 30G then by all means, do NOT ask for 50G! (to get an idea of how much RAM was in use, when the job completes, use: ***sacct -j <jobid> --format=JobName,MaxRSS***).

You can only load about 11G to most of the GPUs and 24G to the advanced ones. If your code makes use of 60G then revise it. Do not allocate more than 60G!

4-6 CPUs are sufficient to serve the GPU. Do not allocate more than 6 CPUs per GPU.

If your job does not require a GPU then submit it to the CPU cluster.

Interactive vs Non-Interactive Use

There are two ways to work with the cluster. Non-interactive way like fire-and-forget, the user specifies the code to be executed and the code shall be executed on a compute node without the user interfering the run. The output to terminal will be directed to a file.

Working interactively with the cluster requires the ssh session to be opened constantly. Once the session is closed it won't be possible to be renewed. This mode is used when working with Jupyter notebooks or IDEs such as pyCharm.

Information about the compute nodes

sinfo shows cluster information.

sinfo -Nel

NODELIST – name of the node

S:C:T - sockets:cores:threads

List of My Currently Running Jobs

squeue --me

Cancel Jobs

scancel <job id>

scancel --name <job name>

Cancel All Pending jobs for a Specific User

scancel -t PENDING -u <user name>

Running Job Information

Use *sstat*. Information about consumed memory:

sstat -j <job_id> --format=MaxRSS,MaxVMSize

scontrol show job <job_id>

Complete Job Information

sacct -j <jobid>

```
sacct -j <jobid> --
format=JobName,MaxRSS,AllocTRES,State,Elapsed,Start,ExitCode,DerivedExitcode,
Comment
```

MaxRSS is the maximum memory the job needed.

GPU Resources Usage

sres

Use it to help you make up your mind about which partition to use, when cluster is used at nearly full capacity.

Advanced Topics

Jupyter Lab

Installation

conda install jupyterlab

Make the Conda Environment Available in Notebook's Interface

Conda activate your Jupyter installed environment and type:

python -m ipykernel install --user --name <conda environment> --display-name "<env name to show in web browser>"

e.g.:

python -m ipykernel install --user --name my_env --display-name "my best env"

Don't forget to choose the right kernel while in the notebook.

Release Job Resources from Within Jupyter After Code Has Finished Running

Add the following 3 lines at the end of your code to make the code release the job resources when done:

```
Import os
job_cancel_str="scancel " + os.environ['SLURM_JOBID']
os.system(job_cancel_str)
```

How to add ipywidgets to Jupyter Lab

Create a Conda environment with ipywidgets and install as lab extension:

conda create -n tqdm -c conda-forge jupyterlab nodejs widgetsnbextension ipywidgets -y

conda activate tqdm

pip install tqdm==4.32.1 ipykernel

python -m ipykernel install --user --name tqdm --display-name "Python (tqdm)"

jupyter labextension install @jupyter-widgets/jupyterlab-manager

conda deactivate

Tensorboard

Run:

tensorboard --port=9989 --logdir logs/fit

If port 9989 occupied, choose another random port of the range 9980-9999.

Open your web browser here: http://132.72.X.Y:9989/ (change X, Y to form your compute node address).

If you cannot access Tensorboard then add, as another parameter: *--bind-all*

Working with Notebooks

Working with notebooks is interactive. If you closed your browser tab while a notebook's cell is running, it keeps running on the cluster, but you will lose the output. On the other hand, it is not always possible to leave your browser open. The simple solutions to that are either to write variable values and results into a file or to run the code as a python script instead of using a notebook.

In Ipython 6.0 and higher you can use %capture cell magic to save all output to file. Use the following line as the very first line of the cell: *%%capture cap_out*

Then, in order to save to variable, on the cell's last line: *var = cap_out.stdout*

If you rather print to file then: *with open('cap_output.txt', 'w') as f:*

f.write(cap_out.stdout)

When you come back to the notebook you can print the content of var or *cap_out.show()* in another cell.

High Priority Jobs

Some users have the right to prioritize their jobs, when the required resources for their job are not available. When you give your job high priority, while submitting it, it might preempt another running job or several running jobs. Also, the prioritized resources are limited and shared among the group users. If a user asks to prioritize a job and the prioritization resources rights are exhausted by the group users, then the job will be pending even though there may be available cluster resources.

The use of high priority jobs is disabled in 'main' partition. User should use another partition.

sbatch --qos=<high priority group user name> <batch file name>

high priority group user name is usually your instructor's user name.

Allocate Extra RAM/CPUs

If you are sure that your job requires more than the default 24G RAM per gpu:

In your sbatch file:

To override default ram:

#SBATCH --mem=48G

If you believe that your job requires more than 58G please contact the IT team.

If you are sure that your job requires more than the default allocation number of cpus:

To override default cpu number

#SBATCH --cpus-per-task=16

Working with the Compute Node SSD Drive

You may want to use the compute node local drive for fast access to data. /scratch directory on the compute node is intended for that.

Add to the sbatch script file:

#SBATCH --tmp=100G ### Asks to allocate enough space on /scratch

Then in users code section:

export SLURM_SCRATCH_DIR=/scratch/${SLURM_JOB_USER}/${SLURM_JOB_ID}

cp /storage/.img $SLURM_SCRATCH_DIR ### copy data TO node's local storage*

mkdir $SLURM_SCRATCH_DIR/testtttt

…

some user code

…

cp -r $SLURM_SCRATCH_DIR $SLURM_SUBMIT_DIR ### copy back final results to user home or other accessible location

When job has finished, canceled or failed, ALL data in $SLURM_SCRATCH_DIR is erased! This temp folder lives with running jobs only!

Job Arrays

Job array feature allows you to run identical version of your script with different environment variables. This is useful for parameter tuning or averaging results over seeds.

To use job array, add the following line to your Slurm batch file:

#SBATCH --array=1-10 ### run parallel 10 times

Adding this will run your script 10 times in parallel, actually creating 10 jobs where each job gets the requested resources, e.g., if you requested 6 CPUs then each job shall get 6 CPUs. The environment variable SLURM_ARRAY_TASK_ID for each run will have different values (from 1 to 10 in this case). You can then set different parameter setting for each parallel run based on this environment variable.

Remember to change #SBATCH --output=your_output.out to #SBATCH --output=output_file_name_%A_%a.out, so the output of each parallel run be directed to a different file. %a will be replaced by the corresponding SLURM_ARRAY_TASK_ID for each run. %A will be replaced by the master job id.

To get the above SLURM_ARRAY_TASK_ID variable in python:

```
Import sys

jobid = sys.getenv('SLURM_ARRAY_TASK_ID')
```

In R:

```
task_id <- Sys.getenv("SLURM_ARRAY_TASK_ID")
```

Send Name of an Input File to Each Task

For example, if the input files end in .txt

*file=$(ls *.txt | sed -n ${SLURM_ARRAY_TASK_ID}p)*

myscript -in $file

Read a Line from an Input File for Each Task

SAMPLE_LIST=($(<input.list))

SAMPLE=${SAMPLE_LIST[${SLURM_ARRAY_TASK_ID}]}

Email Notifications

If you would like to receive an email for each task in the array, rather than just for the whole job:

#SBATCH --mail-type=BEGIN,END,FAIL,ARRAY_TASKS

Limiting the Number of Simultaneously Running Tasks from the Job Array

For example, to limit the number of simultaneously running tasks from a 15 jobs job array to 4:

```
#SBATCH --array=0-15%4
```

Job Dependencies

Job dependencies are used to defer the start of a job based on other job's condition.

sbatch --dependency=after:<other_job_id> <sbatch_script> ### start job after other_job started
sbatch --dependency=afterok:<other_job_id> <sbatch_script> ### start job after other_job ends with ok status. E.g. sbatch --dependency=afterok:77:79 my_sbatch_script.sh -> start after both job 77 and 79 have finished
sbatch --dependency=singleton ### This job can begin execution after any previously launched jobs, sharing the same job name and user, have terminated

CUDA Version Selection

CUDA drivers are installed on all compute nodes.

To load a specific version, e.g. 9.0, use the following line in your sbatch file:

module load cuda/9.0

available versions:

cuda/7.0 cuda/7.5 cuda/8.0 cuda/9.0 cuda/9.1 cuda/9.2 cuda/10.0 cuda/10.1 cuda/10.2 cuda/11.0 cuda/11.1 cuda/11.2 cuda/11.3 cuda/11.4

IDEs

pyCharm

Make sure you have pyCharm Professional installed (free for students/academy people).

Create an interactive session:

Ssh to Slurm and copy the script file /storage/pycharm.sh to your working Slurm directory.

You can modify the following lines at the beginning of the file:

```
##########################################
# USER MODIFIABLE PARAMETERS:
 PART=main     # partition name
 TASKS=6       # 6 cores
 TIME="2:00:00" # 2 hours
 GPU=1               # 1 GPU
 QOS=normal          # QOS Name
##########################################
```

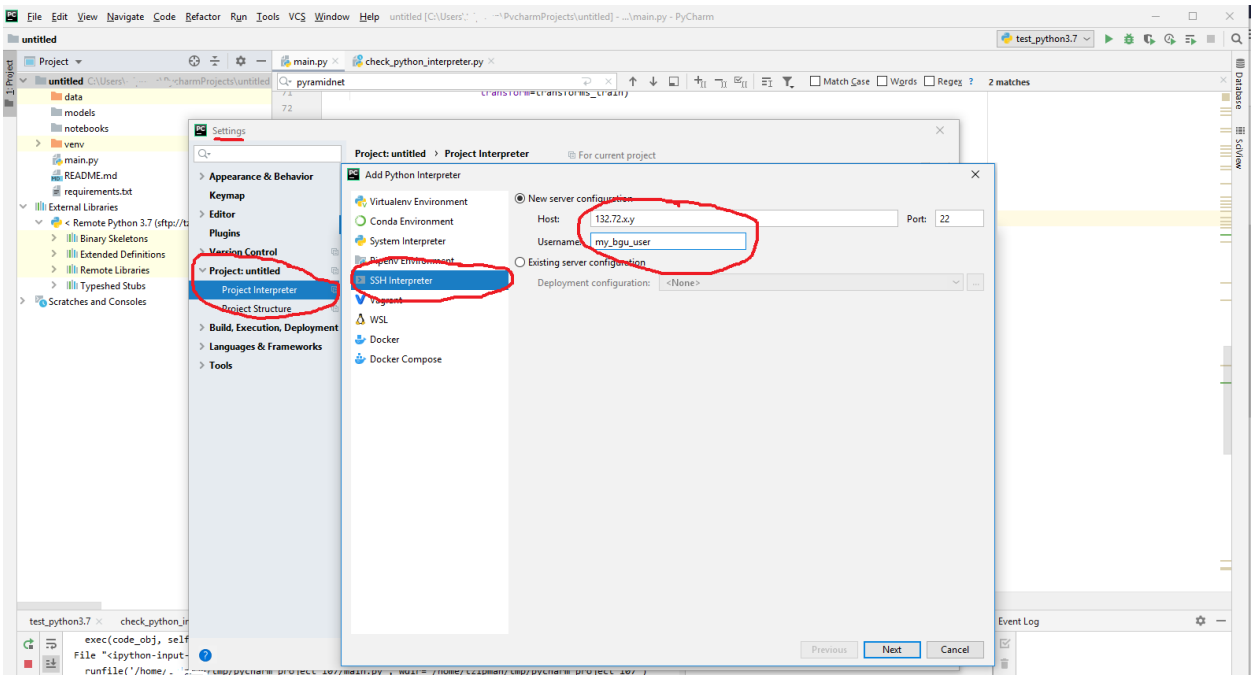
Run the script by typing: . /pycharm.sh

The output lists the node's ip address and the job id.

Open pyCharm.

Go to settings->Project->Project Interpreter

On the upper right hand corner next to Project Interpreter: press the settings icon, choose 'add'

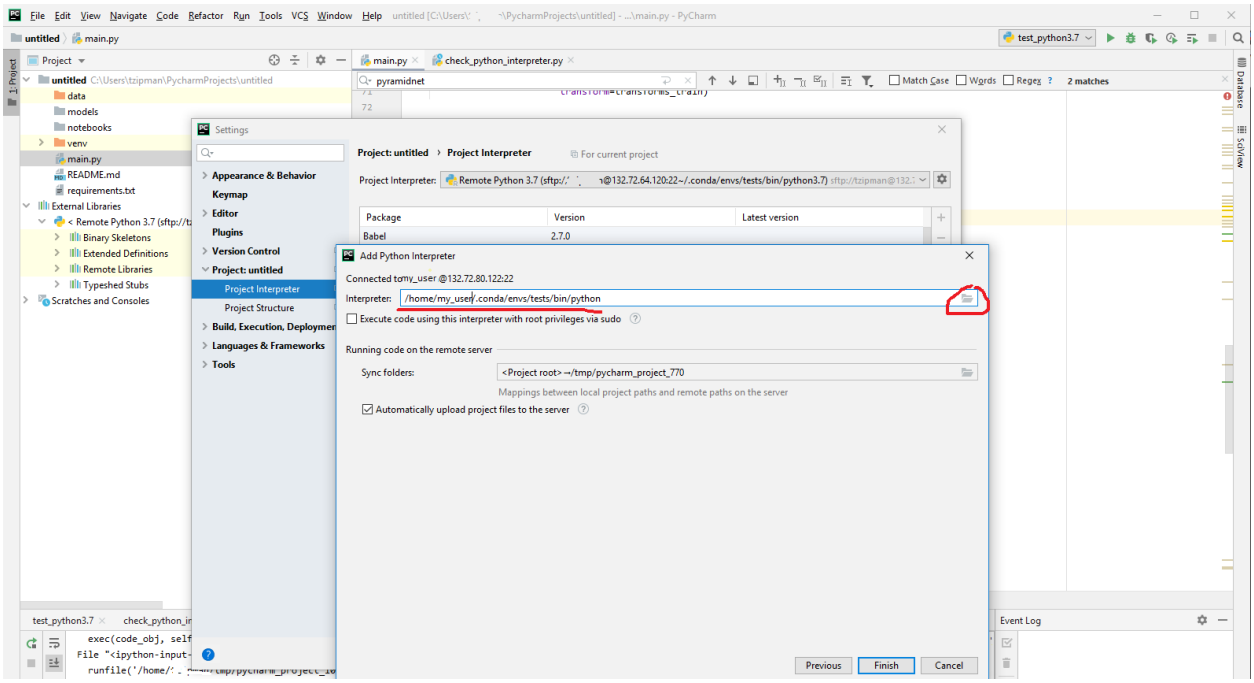


On the left hand side choose SSH Interpreter. Under ‘New server configuration’ fill in the compute node’s ip address and your BGU user name. Click Next.

You might get a message about the authenticity of the remote host, asking if you want to continue connecting. Click ‘yes’.

Enter your BGU password. Click Next.

In the ‘Interpreter:’ line, enter the path to the desired interpreter. You can find your environments interpreters under /home/<your_user>/.conda/envs/<your_environment>/bin/python.



Click Finish.

Give pyCharm some time to upload files to the cluster (upload status is shown on the status bar).

- If you don't want to upload your files to the compute node each time you connect to a new compute node, you can map the sync folder to your cluster home directory or subdirectory. You may also opt not to automatically sync files. These are done in the bottom part of the 'Add python interpreter' box that is illustrated above.

Make phCharm Continue Running Script When Sessions is Disconnected

The offline_training.py script in the frame below, launches another script with arguments:

train.py -- size 192

The output be redirected into result.txt file.

```
import os
import sys

os.system("nohup bash -c '" +
          sys.executable + " train.py --size 192 >result.txt" +
          "' &")
```

offline_training.py

The result.txt file may be found on the compute node. Once you run offline_training.py In Pycharm, on the 'Python Console' pane, the next line should show up:

runfile('/tmp/pycharm_project_<some_number>/<your_offline_running_file>.py', wdir='/tmp/pycharm_project_<some_number>')

The path is the path to the folder in the compute node where your local files are synced. Ssh to the compute node and you may find result.txt at that path.

Remember that once the job ends, that folder is erased!

Visual Studio Code

Create an interactive session:

Ssh to Slurm and copy the script file /storage/pycharm.sh to your working Slurm directory.

You can modify the following lines at the beginning of the file:

##

 # USER MODIFIABLE PARAMETERS:

PART=main # partition name

TASKS=6 # 6 cores

TIME="2:00:00" # 2 hours

GPU=1 # 1 GPU

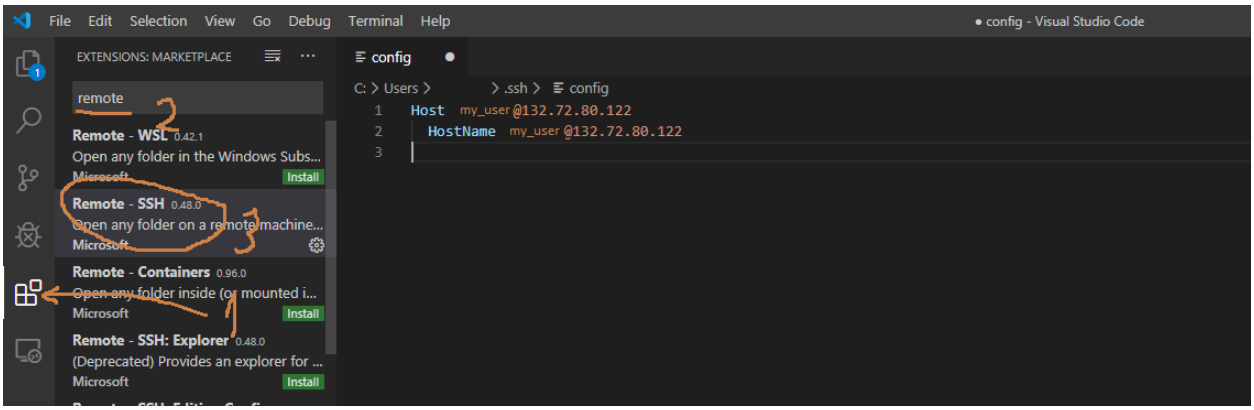
QOS=normal # QOS Name

##

Run the script by typing: . /pycharm.sh

The output lists the newly allocated compute node's ip address and the job id.

Assuming you have VS Code installed and a supported OpenSSH client installed, install the 'Remote – SSH' pack.

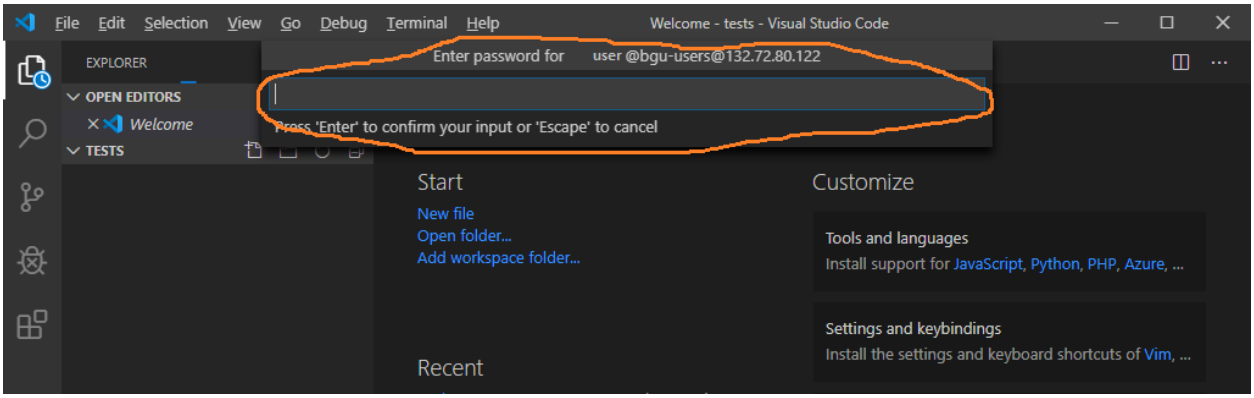


Install the Python package, if needed.

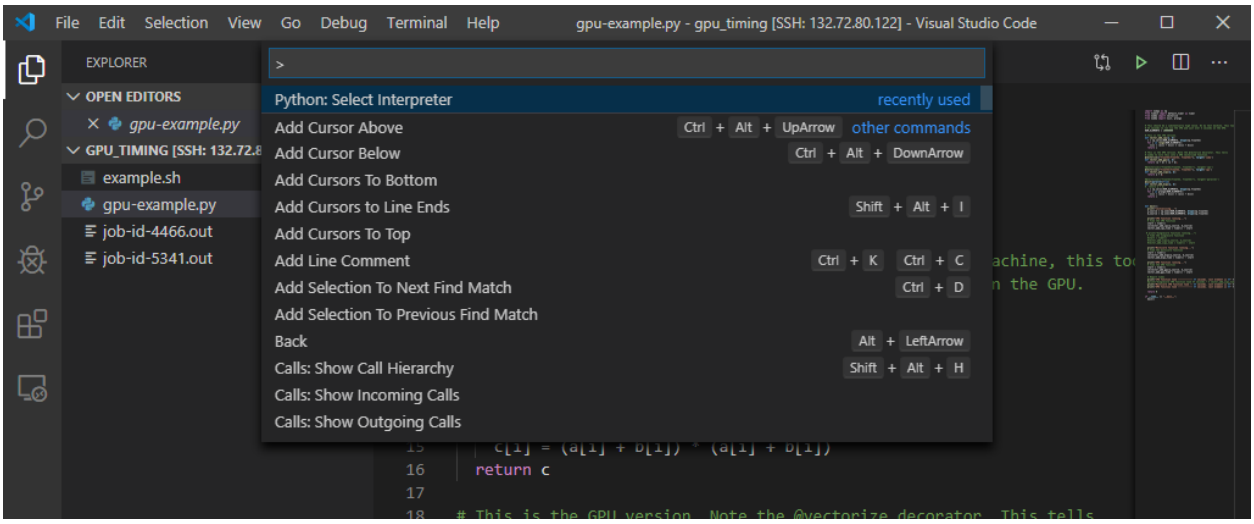
Press the green button (><) on the most lower left side of the window (under the 'settings' button).

On the middle upper side of the window, choose "Remote – SSH: Connect to host…" and enter <your_BGU_user>@<compute_node_ip_address>

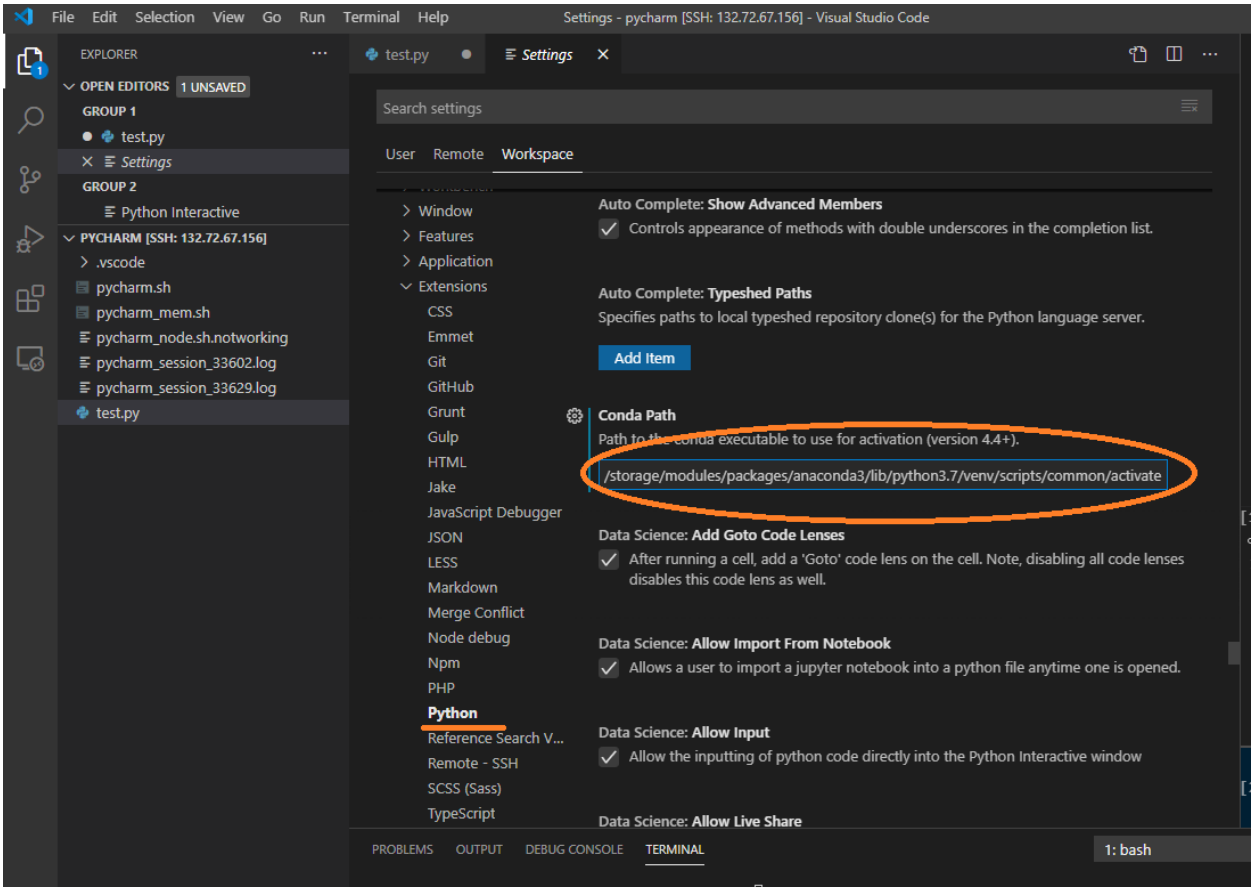
A new window opens. Enter your BGU password, when prompted.



Ctrl+Shift+P for the Command Palette then choose ‘Python: Select Interpreter’ (start typing – it will show up) and choose the interpreter from your desired environment (~/.conda/envs/<environment>/bin/python).



To enable interactive work with notebook like cells, Ctrl+Shift+P for the Command Palette then choose ‘Preferences: Open Workspace Settings’ (start typing – it will show up) and click ‘Python’. Scroll down until you find ‘Conda Path’ and fill in ‘/storage/modules/packages/anaconda3/lib/python3.7/venv/scripts/common/activate’.



To solve an errata with finding the actual path of the python script add the following line to launch.json file:

```
"cwd": "${fileDirname}"
```

Refer to the following paragraph for instructions as to how to place it in the file and where.

Run/Debug with Arguments

Press the Debug symbol on the left vertical ribbon. Click 'create a launch.json file' on the left pane.

Open file launch.json and add another line within 'configurations like so (example for 4 arguments):

```
"args": ["–arg_name1", "value_1", "–arg_name2", "value_2"]
```

Here is an example:

```
    "configurations": [
        {
            "name": "Python: Current File",
```

```
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "console": "integratedTerminal",
            "cwd": "${fileDirname}",
            "args": ["cuda", "100", "exit"]
        }
    ]
```

Docker

Running Docker containers on the cluster may be done using UDOCKER. UDOCKER shall be installed in a Conda environment.

Installation

conda create -n udocker_env

conda activate udocker_env

conda install configparser

pip install udocker

Test

Test a tensorflow-gpu container. Copy paste the following at the end of an sbatch file:

```
module load anaconda
source activate udocker_env
udocker pull tensorflow/tensorflow:2.8.0rc0-gpu-jupyter   # pull image
udocker create --name=tf_gpu_jup28 tensorflow/tensorflow:2.8.0rc0-gpu-jupyter   # create and name container
udocker setup --nvidia tf_gpu_jup28    # setup GPU support
udocker run tf_gpu_jup28 nvidia-smi    # run container with command 'nvidia-smi'
# mount your code directory to container's /home directory and run your python code
udocker run -v /home/my_user/my_code_dir:/home tf_gpu_jup28 python3 /home/my_code.py
```

Once udocker environment is activated you can use:

udocker --help – info about commands and way of use

udocker run --help – help for the 'run' command. This can be done also with other commands

udocker ps – list your containers

udocker images – list your images

udocker rm <container name/id> - remove container

udocker rmi <image id> - remove image

There is no need to pull the image every time.

There is no need to create the container if it already exists. No need to setup Nvidia, for that container, either, if that was already done.

Matlab

Run Matlab GUI straight from terminal (no need for sbatch):

module load matlab

srun --x11 --nodes=1 --mem=24G --cpus-per-task=4 --gpus=1 --partition=main matlab -desktop -sd ~

For Matlab 2021A:

module load matlab/R2021A

srun --x11 --nodes=1 --mem=24G --cpus-per-task=4 --gpus=1 --partition=main --time=01:00:00 matlab -desktop -sd ~

Make sure your ssh terminal supports x11 forwarding!

Send Matlab script to execute as batch by headless Matlab:

srun --nodes=1 --mem=24G --cpus-per-task=4 --gpus=1 --partition=main matlab -nosplash -nodisplay -nodesktop -sd ~ -batch "my_matlab_script"

Headless Matlab may be run by sbatch as well.

- Cluster params:
  - --nodes – number of allocated cluster nodes (must be 1)
  - --mem=24G – memory allocation
  - --cpus-per-task – number of CPUs
  - --gpus – number of GPUs
  - --partition – partition name
- Matlab params:
  - -desktop – run matlab in GUI mode
  - -sd – matlab working directory (user's home directory)
- Check allocation in Matlab console

```
gpuDeviceCount

feature('numcores')
```

julia

First time installation:

```
julia -e 'using Pkg;Pkg.add("IJulia")'
```

The Julia kernel and Julia console will be available in Jupyter Notebook.

R

Command Line

- Create an R conda environment. E.g.: `conda create -n r_env r-essentials r-base`
- Copy the following file: /storage/pycharm_mem.sh
- Edit the first section of the above file to suit your demands.
- Execute the file: `./pycharm_mem.sh`
- Wait for the resources to be allocated.
- Copy the compute node's ip address from the script output.
- SSH to the compute node.
- Type: `conda activate r_env`
- Type: `R`

C#

Install In Conda Environment

conda install -c conda-forge dotnet-sdk

Use

- Use ./pycharm.sh script (see ***Error! Reference source not found.***) to allocate a compute node.
- ssh to the compte node
- Activate environment: *conda activate <your dotnet environment name>*
- Create a new dotnet project: *dotnet new console -o myApp*
- *cd myApp*
- Run: *dotnet run*

Fiji – Image Analysis Tool

You can read about Fiji here: https://fiji.sc/

Run Fiji on the cluster

srun --x11 --gpus=1 --partition=gtx1080 /storage/apps/Fiji/ImageJ-linux64

Make sure you use ssh terminal that supports X11 forwarding!

Appendix

Step by Step Guide for First Use of Python and Conda

1. Make sure you are connected through VPN or from within BGU campus.
2. Download a SSH terminal (https://mobaxterm.mobatek.net/download.html).
3. Open the SSH terminal and start a SSH session (port 22). The remote host is 132.72.44.112 (or gpu.bgu.ac.il). The username is your BGU username and the password is your BGU password.
4. Once logged into the cluster's manager node, create your Conda environment. E.g.:

   conda create -n my_env python=3.7

5. *conda activate my_env*
6. *pip install <whatever package you need>* or *conda install*… Should you need tensorflow-gpu package, please do not use *pip install* to install. Rather use: *conda install -c anaconda tensorflow-gpu*
7. conda deactivate
8. Copy the sbatch file (job launching file) by typing (do not forget the dot at the end!):

   cp /storage/sbatch_gpu.example .

9. Edit the file using nano editor: *nano sbatch_gpu.example*
10. You may change the job name by replacing my_job with your own string.
11. Go to the last lines of the file. 'source activate my_env': if needed, replace 'my_env' with your environment name that you have created on paragraph 4.
12. 'jupyter lab' is the program to run on the compute node – it will start a jupyter program that you may use (refer to Jupyter Lab paragraph). You may use another command instead of 'jupyter lab', such as 'python my_script.py my_arg'
13. Press '<ctrl>+x', then 'y' and '<Enter>' to save and leave the file.
14. Launch a new job: *sbatch sbatch_gpu.example*
15. You should, instantly, get the job id.
16. To see the status of your job(s) type *squeue --me*
17. Under 'ST' (state) column if the state is 'PD' then the job is pending. If the the state is 'R' then the job is running and you can look at the output file for initial results (jupyter results will take up to a minute to show): *less job-<job id>.out*
18. If you asked for jupyter, then copy the 2nd link (which starts with 'https://132.72.'). Copy the whole link, including the token, and paste it in the address bar of your web browser. Make the browser advance (twice) in spite of its warnings.

Example for Creating Tensorflow-gpu and Jupyter Lab Environment

- Create new environment named 'tfgpu_jup', with tensorflow-gpu: *conda create -n tfgpu_jup*
- Activate new env: *conda activate tfgpu_jup*
- Install packages: *conda install python=3.9 cudatoolkit cudnn*
- Install tensorflow: *conda install -c anaconda tensorflow-gpu=2.6*
- Install Jupyter Lab: *conda install -c conda-forge jupyterlab*
- Make the Conda Environment Available in Notebook's Interface: *python -m ipykernel install --user --name tfgpu_jup --display-name "tfgpu_jup"*
- Deactivate new environment: *conda deactivate*
- Submit a new job request: *sbatch sbatch_gpu.example*
- wait about a minute and open the output file. Copy the whole middle token (address 132.72.X.Y) and paste in your favorite web browser's address bar.
- Create a new notebook.
- Select tfgpu_jup kernel from the upper right corner of the notebook.

Conda

Viewing a list of your environments

```
conda env list
```

list of all packages installed in a specific environment

```
conda list
```

to see a not activated environment

```
conda list -n <my_env>
```

Activating / deactivating environment

```
source activate <my_env>
```

or (depends on conda version)

```
conda activate <my_env>
```

```
conda deactivate
```

Create Environment

```
conda create -n <my_env>
```

with specific python version

```
conda create -n <my_env> python=3.4
```

with specific package (e.g. scipy)

```
conda create -n <my_env> scipy
```

Or

```
conda create -n <my_env> python
conda install -n <my_env> scipy
```

with specific package version

```
conda create -n <my_env> scipy=0.15.0
```

with multiple packages

```
conda create -n <my_env> python=3.4 scipy=0.15.0 astroid babel
```

Update Conda

```
conda update conda
```

Compare Conda Environments

The following python (2) script compares 2 conda environments and can be found in '/storage' directory.

```
python conda_compare.py <environment1> <environment2>
```

Transfer Files

To / From Your PC

You can use WinSCP to transfer files, or if you use MobaXTerm then it has its own file browser.

In Windows, SISE department students can also map their cluster home directory like so:

1. connect to BGU vpn
2. open file explorer.
3. on the left pane right click "This PC".
4. on the menu that will open, click on "map network drive".
5. choose a drive letter or leave the default letter in place.
6. in the "Folder" field write down: \\132.72.65.201\usr_home\<your BGU username>
7. you can choose to check or uncheck "Reconnect at sign-in".
8. you must check "Connect using different credentials".
9. click on "Finish" button.
10. a new authentication window will open, in the Username field write down: bgu-users\<your BGU username>
11. in the Password field write down your BGU password.

Get a Public File

from AWS s3

Use wget:

```
wget --no-check-certificate --no-proxy  'https://<your bucket name>.s3.amazonaws.com/<path and name of file>'
```

from Google Drive

Use wget:

```
wget --no-check-certificate 'https://drive.google.com/uc?id=<file id> -O <file name to save>
```

where file-id is the alphanumeric long string that shows when you right click the file in Chrome and view the file's link. for example:

```
wget --no-check-certificate 'https://drive.google.com/uc?id=1A3Ef4gHdhsdjklmn6o' -O my_file.txt
```

FAQ

Usage

- ### Can I ssh the cluster when I am away from university?

  This can be done by using VPN.

- ### I uploaded files to the cluster, while logged in to the manager node, can a compute node find these files?

  The files were uploaded to the storage. **All** cluster nodes have access to your files on the. storage

- ### I need sudo to install library X / tool Y

  Install it in your conda environment, using 'conda install' or 'pip install'

- ### How to tell which GPU was allocated for the job?

  Write '*nvidia-smi -L*' either in the sbatch file or ssh to the compute node and run it there.

- ### Performance seems slow when using Tensorflow

  Make sure you have 'tensorflow-gpu' package in the Conda environment you use.

- ### Even though I installed tensorflow-gpu, it does not recognize any gpu.

  Please do not use *pip install* to install tensorflow-gpu. Rather use: *conda install -c anaconda tensorflow-gpu*

- ### Does a Jupyter notebook keep on running when I close the browser?

  Please refer to Working with Notebooks.

- ### When running Jupyter lab on my browser, kernel shows as disconnected?

  Try using another browser or reset browser to factory defaults. Sometimes browser add ons may prevent the browser from properly communicating with kernel.

- ### With Mac OS, when opening Jupyter lab on Chrome or Safari browsers, security settings prevent the notebook from loading.

  Use Firefox or Maxthon browsers.

- ### Is Git installed on the cluster?

  Git is installed on the manager node.

- ### Why is my job pending? What's the meaning of REASON?

  PartitionTimeLimit – the 'time' variable in your sbatch file is set to time limit greater than the partition's maximum possible time limit (usually 7 days).

  Resources – currently, the cluster has insufficient resources to fulfill your job.

Priority – job is queued behind higher priority jobs. You may have exceeded your group QoS priority resources – You can launch a job with no QoS priority or wait for QoS priority resource to be available.

QOSMaxJobsPerUserLimit – you have reached the maximum allowed concurrent jobs for the requested partition.

MaxGRESPerAccount – your requested high priority job exceeds the limit of concurrent gpus allocated to your account. Job is waiting for golden card to be available.

- In python app I print some run time info but they are buffered and being printed all at once.

  To use unbuffered print to output: python -u my_py_app.py

  u – unbuffered.

  Another option is to add the following line to your sbatch scripy:

  export PYTHONUNBUFFERED=TRUE

  Please note it has performance toll, so it is not advisable to use it when not debugging.

- My job requires a lot of RAM, what can I do?

  Find the reason for consuming so much RAM. For example if you are working with a pictures dataset and the preprocessing of the dataset consumes a lot of RAM, then use pointers to pictures for preprocessing the pictures themselves, if possible, as demonstrated here: https://www.kaggle.com/itslek/transfer-learning-keras-flowers-sf-dl-v1

Errors

- "RuntimeError: CUDA out of memory. Tried to allocate 448.00 MiB (GPU 0; 10.73 GiB total capacity; 9.64 GiB already allocated; 124.69 MiB free; 195.47 MiB cached)"

  Or "Resource exhausted: OOM when allocating tensor with shape…"

  The problem arises when trying to allocate more GPU memory than available (e.g. 10.73GiB for Nvidia 2080). Try to reduce the size of the batch you load to the GPU in order to fit the GPU's available memory.

  If you have done the above and still get this error, it may be related to the code grabbing more memory than you realize.

  Working with Tensorflow – Tensorflow grabs 95% of the memory as default (to avoid performance costly dynamic allocation), so when trying to allocate more memory (sometimes from another process) you will get this error.

  To configure Tensoflow to allocate growing amount of memory (flexible but performance costly):

config = tf.ConfigProto()
config.gpu_options.allow_growth = True
session = tf.Session(config=config, ...)

Or:
tf.config.experimental.set_memory_growth(physical_devices[0], True)

To configure Tensoflow to allocate another fixed size (e.g. 1/3) of fraction of memory per process:

gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.333)
sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))

Working with pyTorch – a) sometimes you leave a reference to a tensor(s) in Cuda. That will make cuda keep the memory for the tensor and if you are iterating then it accumulates (refer to responder 'samkellerhals' in https://github.com/pytorch/pytorch/issues/16417). b) Use *detach()* if appropriate (to remove the graph associated with the tensor if you don't use gradient descent). c) You can also use the garbage collector and cache emptying, in your code, as follows:

del variables
gc.collect()
torch.cuda.empty_cache()

In order to have a readable summary of allocation of memory in the gpu use:

torch.cuda.memory_summary(device=None, abbreviated=False)

More reading: https://pytorch.org/docs/stable/notes/faq.html

- I installed wget python package like so: 'conda install wget', and I get 'ModuleNotFoundError: No module named wget' error.

  Use: pip install wget

- In my conda environment I installed a package which is a python wrapper of binary code. Running a code that uses it, with Jupyter notebook was successful, but running the very same code from pycharm, failed with the message 'NotImplementedError: "…" does not appear to be installed or on the path, so this method is disabled. Please install a more recent version of … and re-import to use this method.'.

  This happens because with pycharm, the environment variable 'PATH' remains unchanged, unlike with Jupyter that when choosing conda environment, 'PATH' gets modified. The solution is to modify 'PATH' **prior** to **import**ing the wrapper package in your python code, as follows (replace <your_user> and <your_env> with yours):

```
import os
os.environ['PATH'] =
```

```
'/home/<your_user>/.conda/envs/<your_env>/bin:/storage/modules/packages/anacon
da3/bin:/storage/modules/bin:/storage/modules/packages/anaconda3/condabin:/usr
/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/storage/modules/packages/matlab
/R2019B/bin:/home/<your_user>/.local/bin:/home/<your_user>/bin'
import <python wrapper package>
…
```

- Using GPU 3090 with tensorflow, I get the following error: InternalError: CUDA runtime implicit initialization on GPU:0 failed. Status: device kernel image is invalid

  You need tensorflow 2.2.

- Using GPU 3090 with pytorch, I get the following error: NVIDIA GeForce RTX 3090 with CUDA capability sm_86 is not compatible with the current PyTorch installation.
  The current PyTorch install supports CUDA capabilities sm_37 sm_50 sm_60 sm_70.
  If you want to use the NVIDIA GeForce RTX 3090 GPU with PyTorch…

  Pip upgrade your torch version: *pip3 install torch==1.10.1+cu113 torchvision==0.11.2+cu113 torchaudio==0.10.1+cu113 -f https://download.pytorch.org/whl/cu113/torch_stable.html*

- VSCode error: windows remote host key has changed port forwarding is disabled

  OR: Could not establish connection to "x.x.x.x": Remote host key has changed, port forwarding is disabled.

  OR: visual studio code could not establish connection… the process tried to write to a nonexistent pipe

  These errors mean that the remote host key does not much to the key saved locally, anymore. The keys mismatch may be a result of reinstalling the remote host. If that is the reason go to C:\Users\<your Windows user>\.ssh\ and change the names of the files. Windows will create new updated files instead.

- NotImplementedError: Cannot convert a symbolic Tensor… to a numpy array

  Downgrade numpy version to 1.19: *pip install numpy==1.19.5*

- The sbatch file that activates Jupyter, no longer prints the 132.72… ip address to the output file

  You need to upgrade to jupyterlab version 3, as version 2 is no longer supported.

- When I run the Fiji srun command, I get the following message "srun: error: No DISPLAY variable set, cannot setup x11 forwarding."

  This happens when you use ssh terminal that does not support x11 forwarding. Use a terminal that does support it, such as MobaXterm.

- Output file shows: slurmstepd: error: _is_a_lwp: open() /proc/60830/status failed: No such file or directory

  It's a rare Slurm accounting error message that should not affect the job. Just ignore it.

- I get RuntimeError: CUDA error: no kernel image is available for execution on the device CUDA kernel errors might be asynchronously reported at some other API call,so the stacktrace below might be incorrect. For debugging consider passing CUDA_LAUNCH_BLOCKING=1

  This error is a result of incompatible pyTorch version. Update installed version, e.g.: `pip install torch==1.8.0+cu111 torchvision==0.9.0+cu111 torchaudio==0.8.0 -f https://download.pytorch.org/whl/torch_stable.html`

- I use Python, yet I get "srun: error: … Segmentation fault (core dumped)"

  This is typical error, showing when the Conda environment gets corrupted. Create a new environment.

- Got an error like this: libstdc++.so.6: version `GLIBCXX_3.4.26' not found

  If you already installed libgcc in your conda environment (like so: conda install libgcc), add the following line to your sbatch script right after the #SBATCH lines (replace 'username' and 'my_env' with yours):

  export LD_LIBRARY_PATH=/home/username/.conda/envs/my_env/lib:$LD_LIBRARY_PATH

- Using Ninja introduced the following errors "You're running a too old version of GCC. We need GCC 5 or later." and "You need C++14 to compile PyTorch"

  Add the following lines to your sbatch file, right after the '#SBATCH' lines:
  scl enable devtoolset-9 bash
  export LD_LIBRARY_PATH=/home/<your username>/.conda/envs/<your conda environment name>/lib:$LD_LIBRARY_PATH