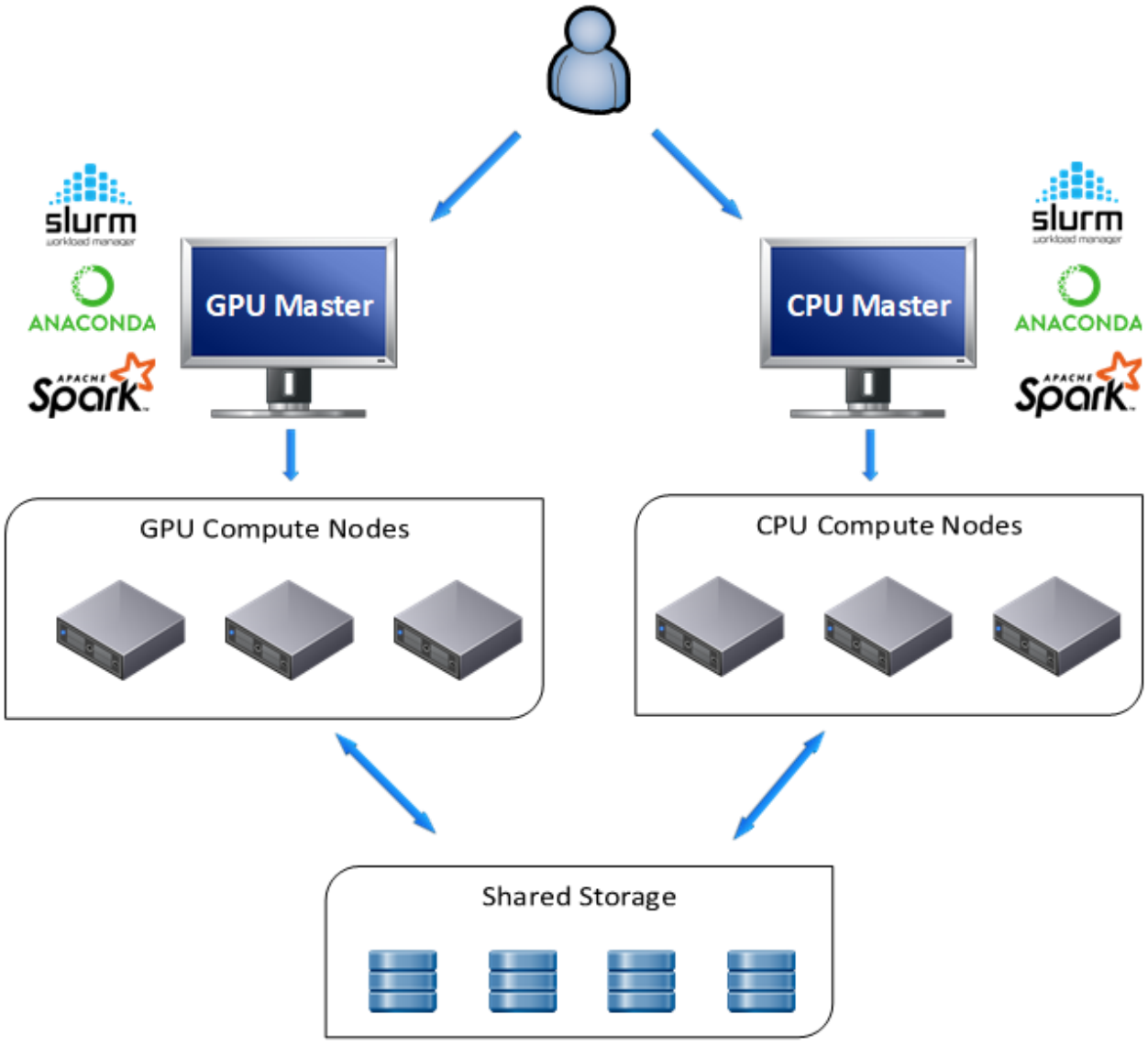
BGU ISE-CS-DT Cpu Cluster User Guide

23/05/2021

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.



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.

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

A job is allocation of compute resources such as RAM memory, cpu cores, 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.

You may find abundance of information regarding Slurm, on the web.

Use

- Ssh to the Manager Node: 132.72.65.199
- 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 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.
- 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

sbatch <batch file name>

- Conda users: Make sure you submit the job while virtual environment deactivated on the manager node ('conda deactivate')!

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 so ##SBATCH

#SBATCH --partition main                ### specify partition name where to run a job. main - 7 days time limit

#SBATCH --time 0-01:30:00               ### limit the time of job running. Make sure it is not greater than the partition time limit!! 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 ### users email for sending job status notifications

#SBATCH --mail-type=BEGIN,END,FAIL      ### conditions when to send the email. ALL,BEGIN,END,FAIL, REQUEU, NONE

##SBATCH --cpus-per-task=6      # 6 cpus per task – use for multithreading, usually with --tasks=1

##SBATCH --tasks=4              # 4 processes – use for multiprocessing


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

echo "SLURM_JOBID"=$SLURM_JOBID

echo "SLURM_JOB_NODELIST"=$SLURM_JOB_NODELIST


### Start you code below        ####

module load anaconda            ### load anaconda module

source activate my_env          ### activating Conda environment, environment must be configured before running the job

python my.py my_arg             ### execute python script – replace with your own command
```

Example sbatch file location on cluster: `/storage/sbatch_cpu.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 the **minimum possible RAM**. If your code makes use of 30G then by all means, do NOT ask for 50G! (to tell how much RAM was used, when the job completes, use: ***sacct -j <jobid> --format=JobName,MaxRSS***).

Same goes for the number of CPUs. There is no need to allocate more cores than threads.

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 -l -u$USER

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,State,Elapsed,Start,ExitCode,DerivedExitcode,Comment
```

MaxRSS is the maximum memory the job needed.

Advanced Topics

Allocate Extra RAM/CPUs

In your sbatch file:

To override default ram:

#SBATCH --mem=58G

To override default cpu number

#SBATCH --cpus-per-task=16

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. 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=%a_your_output.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.

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

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 (just make sure GPU=0):

```
##########################################
# USER MODIFIABLE PARAMETERS:
 PART=main     # partition name
 TASKS=8       # 8 cores
 TIME="2:00:00" # 2 hours
GPU=0              # Make sure this is 0
 QOS=normal          # QOS Name
##########################################
```

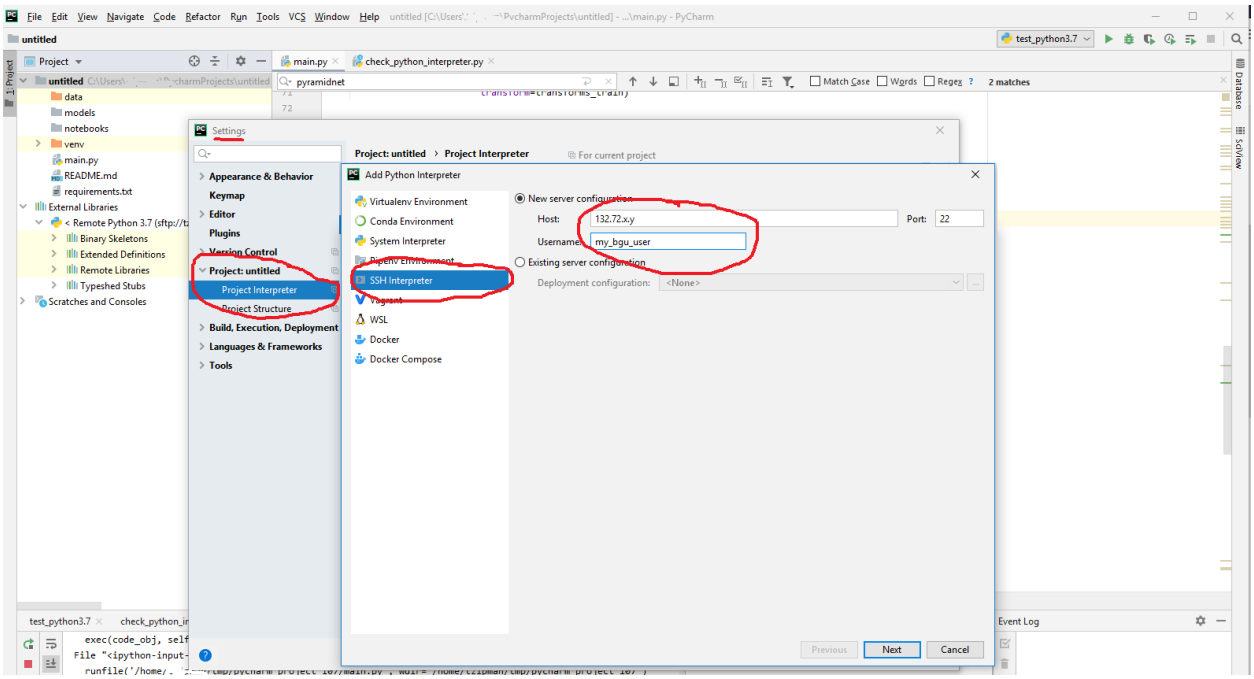
Run the script by typing: . /pycharm.sh

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'

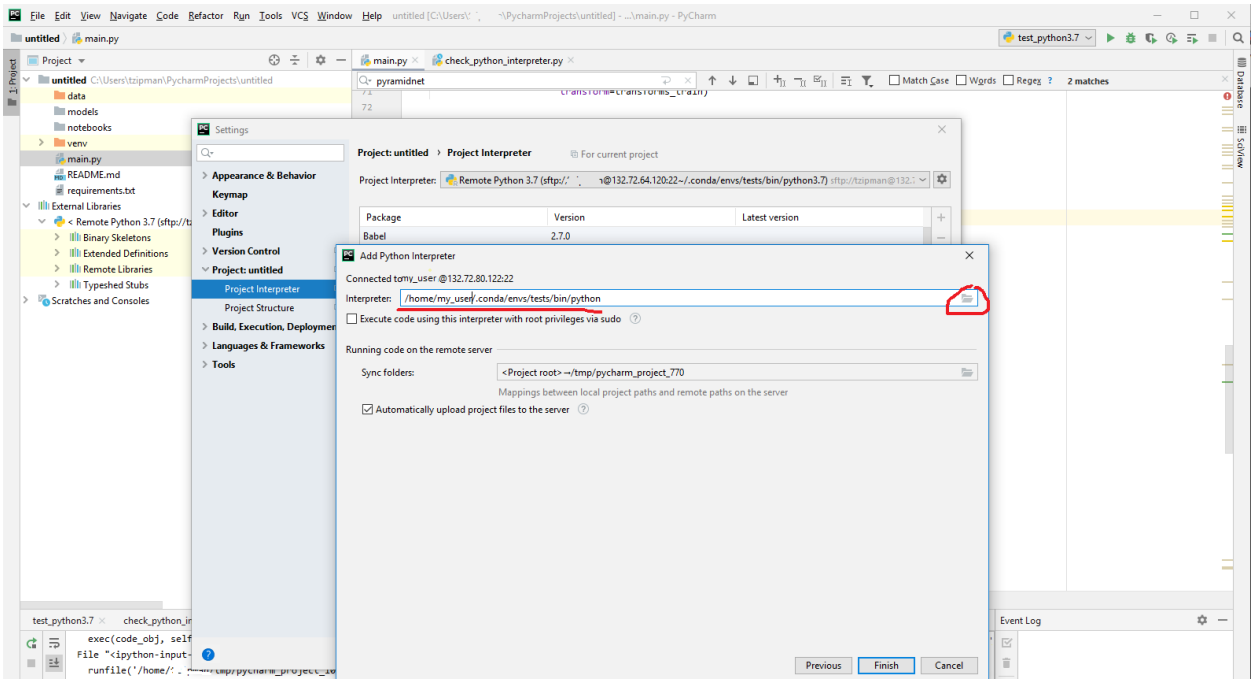


On the left hand side choose SSH Interpreter. Under ‘New server configuration’ fill in the compute node’s ip address and your BGU user name. Click Next.

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).

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.

offline_training.py

```
import os
import sys

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

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 (just make sure GPU=0):

##

 # USER MODIFIABLE PARAMETERS:

PART=short # partition name

TASKS=8 # 8 cores

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

GPU=0 # Make sure this is 0

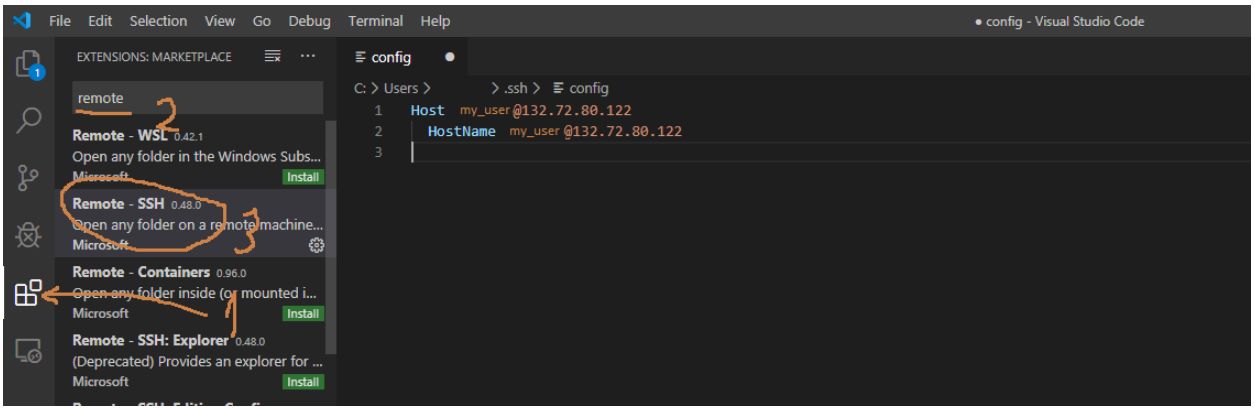
QOS=normal # QOS Name

##

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.

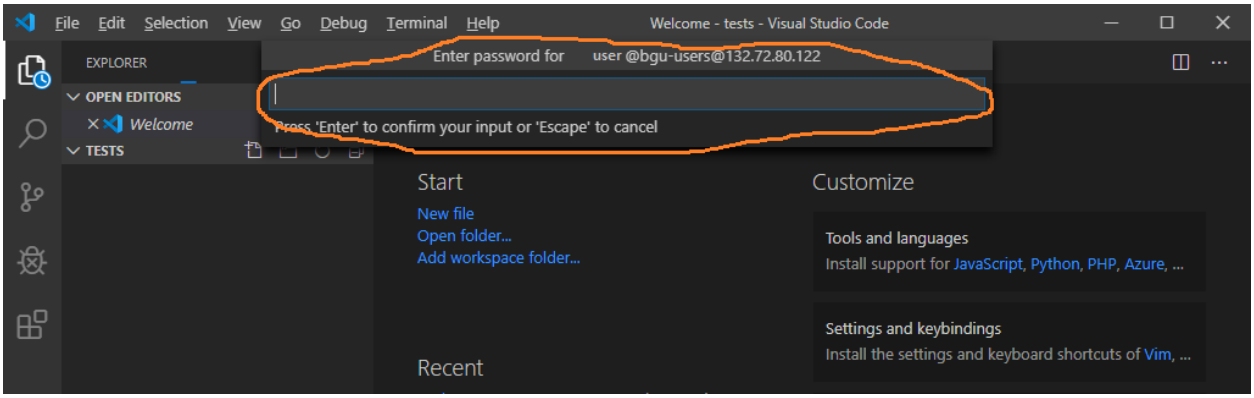


Install the Python package, if needed.

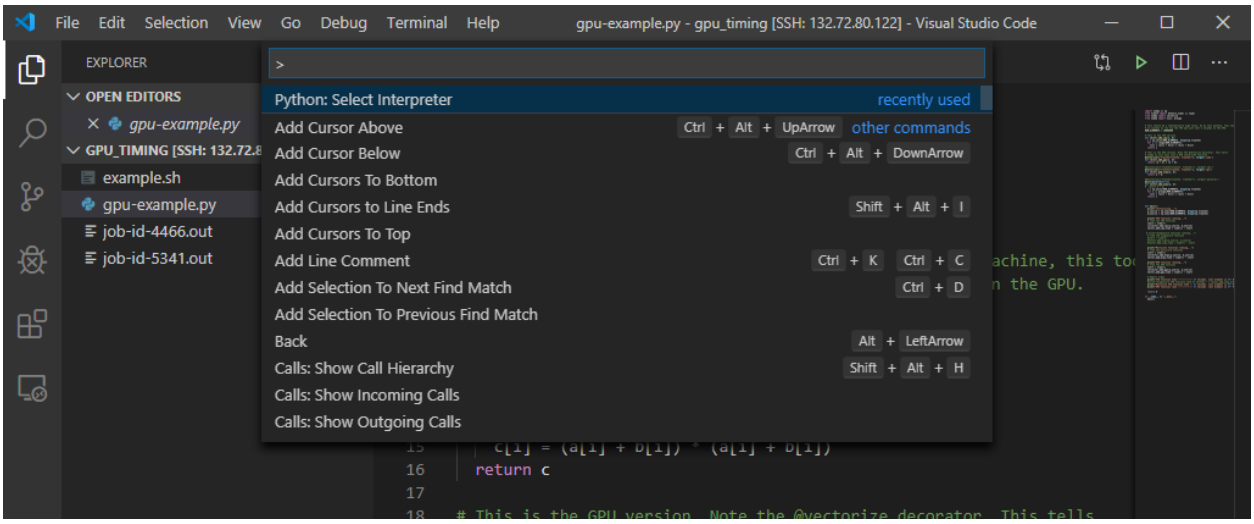
Press the green button (><) on the most lower left side of the window (under the 'settings' button).

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

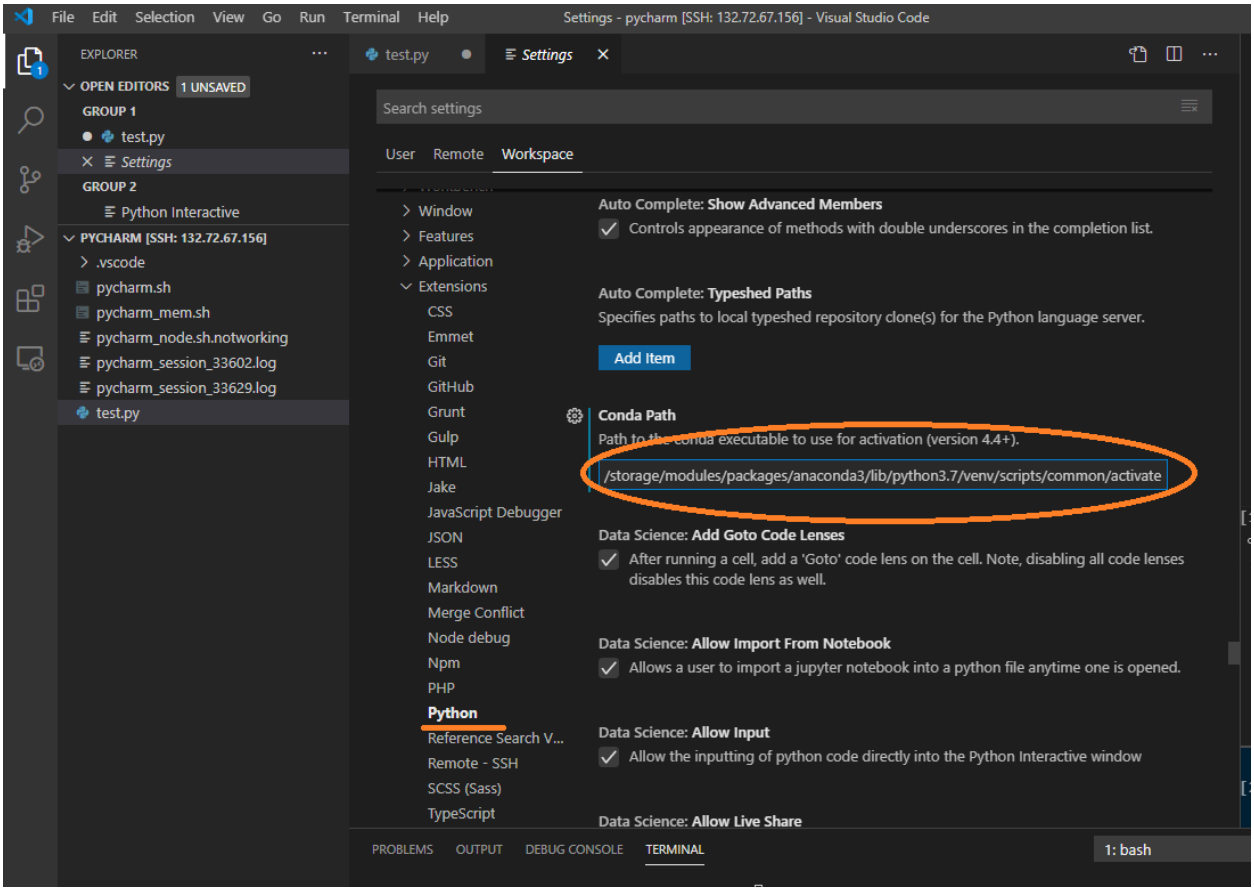
A new window opens. Enter your BGU password, when prompted.



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"]
        }
    ]
```

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`

Appendix

Step by Step Guide for First Use

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 132.72.65.199 , 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 <whatever package you need>
7. conda deactivate
8. Copy the sbatch file (job launching file) by typing (do not forget the dot at the end!):

   cp /storage/sbatch_cpu.example .

9. Edit the file using nano editor: *nano sbatch_cpu.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. 'python my.py my_arg' is the program to run on the compute node. You may use another command instead.
13. Press '<ctrl>+x', then 'y' and '<Enter>' to save and leave the file.
14. Launch a new job: *sbatch sbatch_cpu.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.

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>
```

Deap

Create Environment

1. conda create -n DEAP -y python=3
2. conda activate DEAP
3. conda install -c cyclus java-jdk
4. pip install deap
5. conda deactivate

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'

- 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.

Errors

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

  Use: pip install wget

- 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.

- 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>
…
```

- 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.

- 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