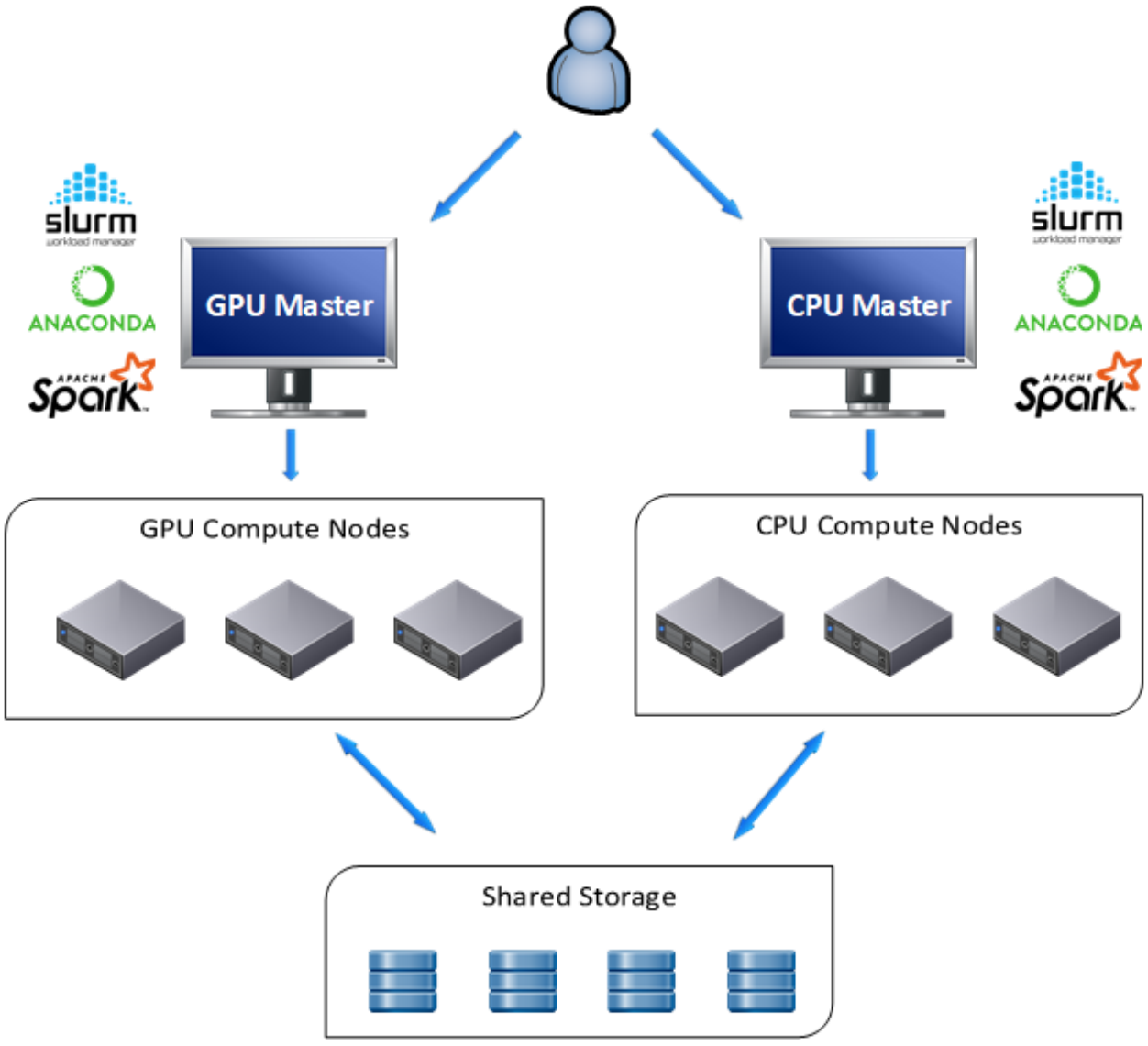
BGU ISE-CS-DT CPU Cluster User Guide

4/4/2024

Contents

Abstract

This document is located here and is 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. Slurm is job scheduler and resource manager used in most of the greatest super computers. The cluster consists of a manager node (also called master node) and several compute nodes (view above illustration).

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, by SSH, 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 reserved for Slurm CLI commands.

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

Use

- Make sure you got admission to the cluster by your IT team.
- Ssh to the Manager Node: 132.72.64.80
- 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 in the CLI ('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=2              # 2 processes – use for processing of few programs concurrently in a job  (with srun). Use just 1 otherwise

#############  Following lines will be executed by a compute node    ########################

### 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. For jupyter write:  jupyter-lab
```

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

MaxRSS is the maximum memory the job needed.

Advanced Topics

Jupyter Lab

You may Run Jupyter Lab in Udocker . OR

Installation

conda install jupyterlab

Make the Conda Environment Available in Notebook's Interface

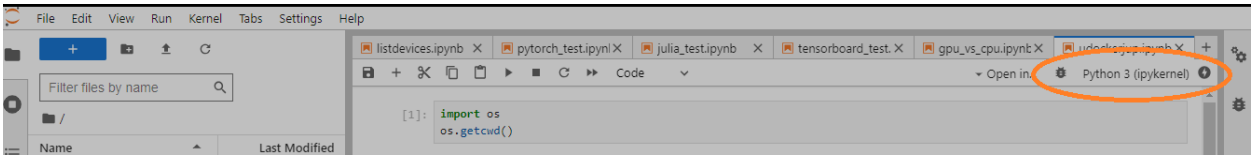
In your conda environment:

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.



Launch Jupyter Lab

Launch jupyter by submitting a job. The sbatch file should like the example Batch File, only replace the last line with: jupyter-lab

About 45 seconds after the job has started running, open the output file and look for a url containing the ip substring "132.72". Copy the whole url and paste it in your web browser's address bar.

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)
```

Tensorboard

No Jupyter

1. Run program and generate logs in my_log_dir (or any other directory)
2. Wait for program run to end
3. ssh to the compute node
4. conda activate my_environment
5. tensorboard --bind_all --logdir=my_log_dir
6. Wait for output. Copy paste link to web browser

In Jupyter

1. In one of the first cells, load the tensorboard extension: *%load_ext tensorboard*
2. After cells generated log files, write in a cell: *!tensorboard --bind_all --logdir=my_log_dir*
3. Wait for output. Copy paste link to web browser

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.

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

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!

Sending Arguments to sbatch File

```
Launch job with command line arguments:
sbatch --export=ALL,var1='1',var2='hello' my_sbatch_file.sbatch

In the sbatch file, use with '$' e.g.:
echo $var2
```

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 the termination of any previously launched jobs, sharing the same job name and user.

Examples

Command Line Example

```
$ sbatch job1.sh
12344
$ sbatch --dependency=afterok:112344 job2.sh
```

Script Example

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=6       # 6 cores
 TIME="2:00:00" # 2 hours
GPU=0              # Make sure this is 0
 QOS=normal          # QOS Name
##########################################
```

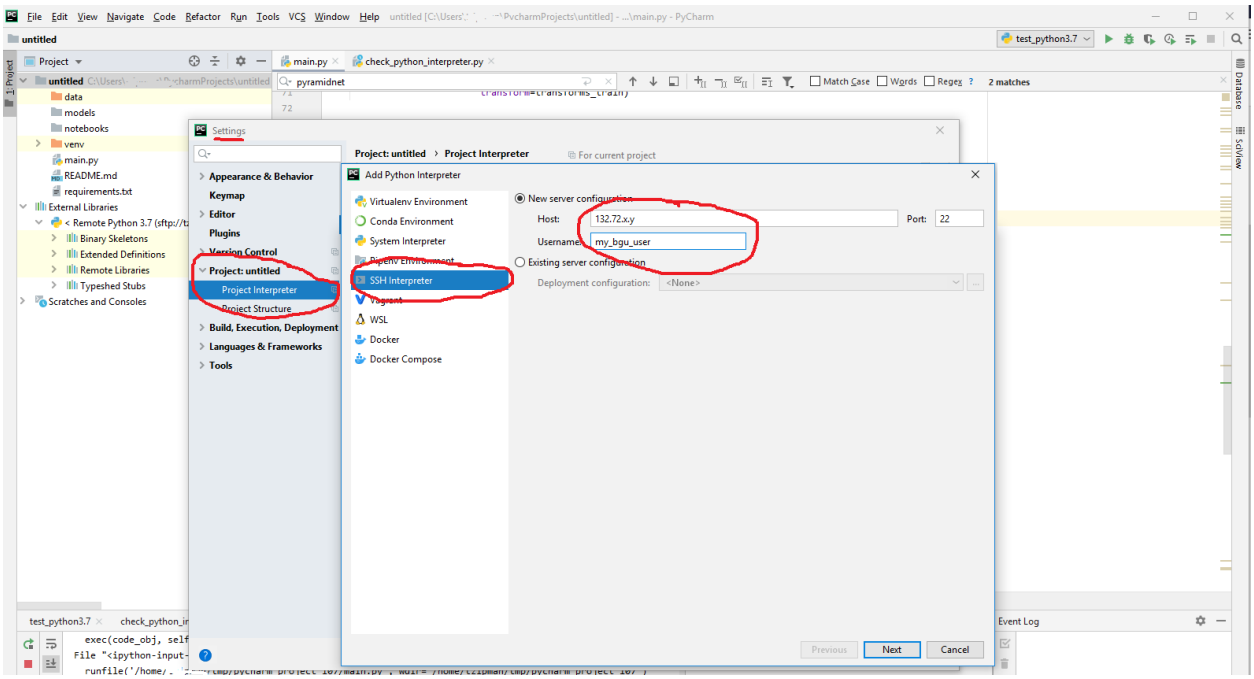
Run the script by typing: . /pycharm.sh

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'

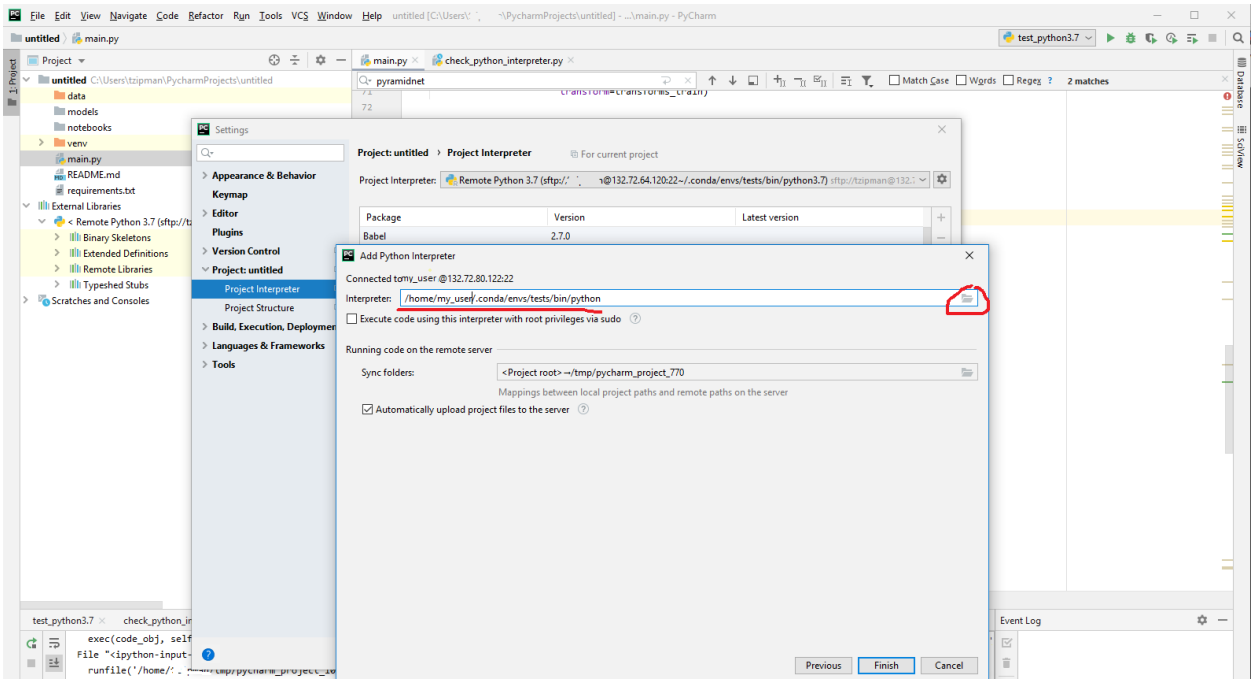


On the left hand side choose SSH Interpreter. Under 'New server configuration' fill in the compute node's ip address and your BGU user name. Click Next. **Do not fill in the master node's ip address!**

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=main # partition name

TASKS=6 # 6 cores

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

GPU=0 # Make sure this is 0

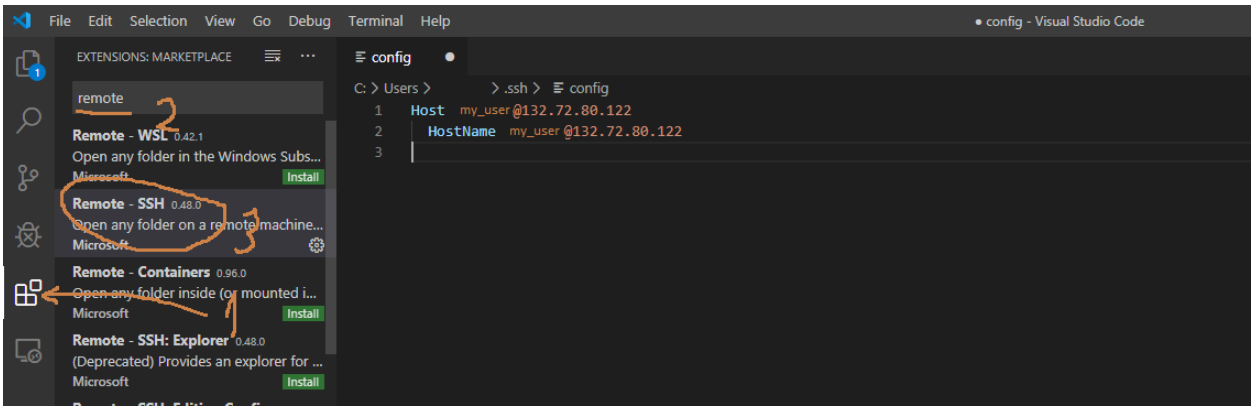
QOS=normal # QOS Name

##

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.

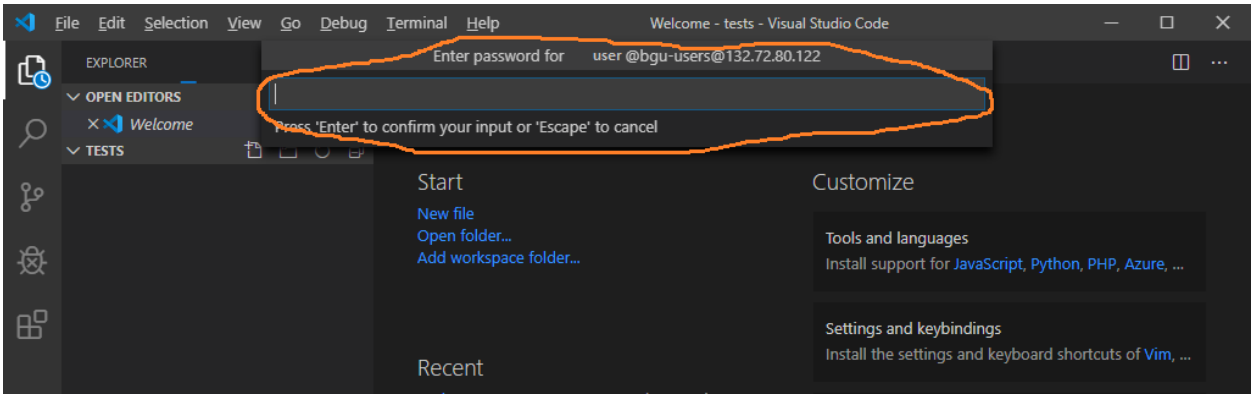


Install the Python package, if needed.

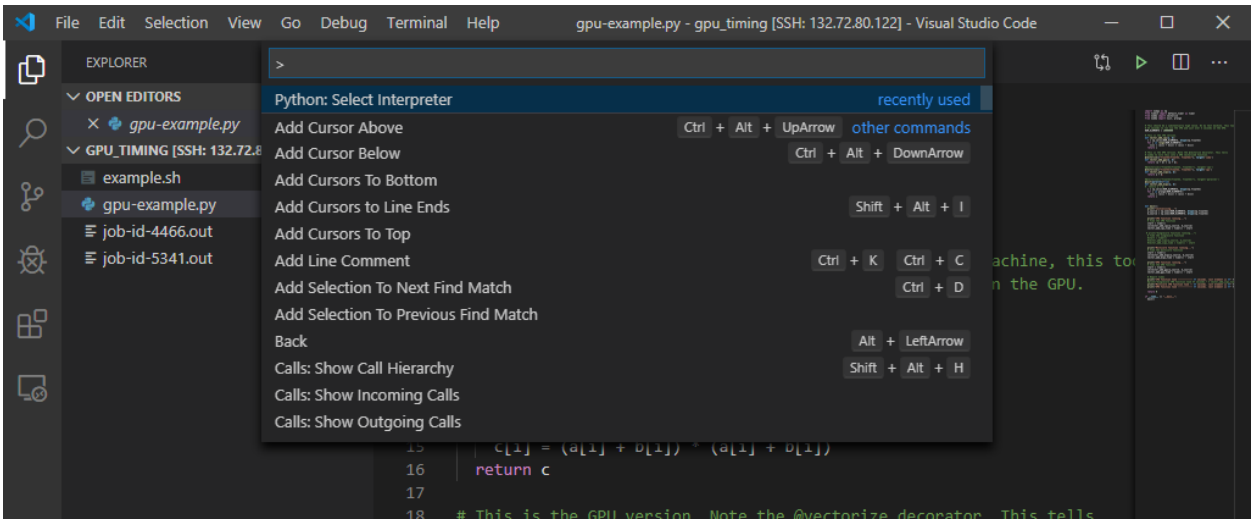
Press the green button (><) on the bottom left corner of the window (under the 'settings' button).

On the middle upper side of the window, choose "Remote – SSH: Connect to host…" and enter <your_BGU_user>@<compute_node_ip_address> . **Do not fill in the master node's ip address!**

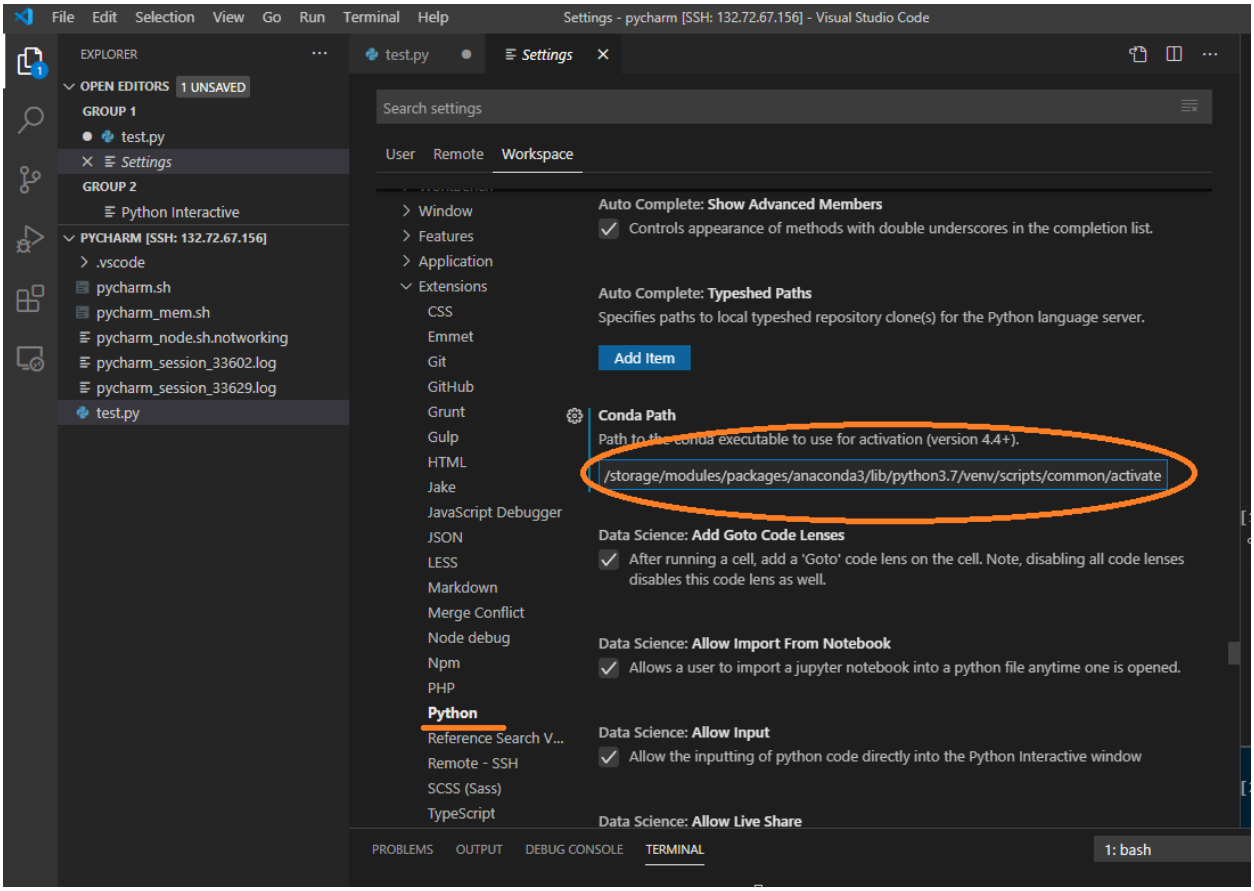
A new window opens. Enter your BGU password, when prompted.



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, you may have to: 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:

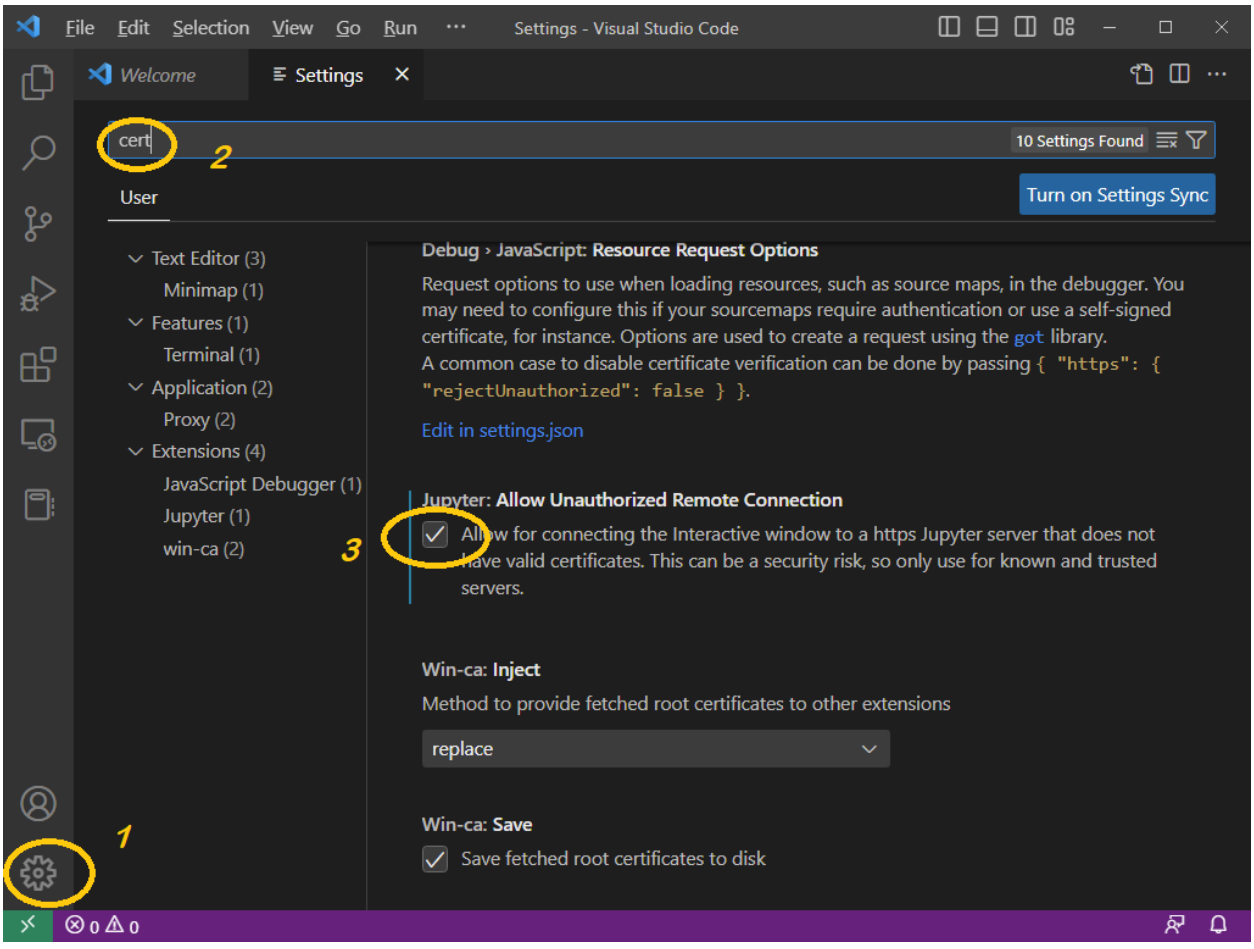
```
    "configurations": [
        {
            "name": "Python: Current File",
```

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

Run Jupyter Notebook

To avoid SSL certificate error, do the following:

1. Press the cogwheel then select 'Settings', start writing 'cert', scroll all the way down and tick 'Jupyter: Allow Unauthorized Remote Connection'.
2. Edit cluster file ~/.vscode/settings.json -> add the line: "http.systemCertificates": true
3. Add the following line line at the end of cluster file ~/.bashrc:
   export NODE_TLS_REJECT_UNAUTHORIZED='0'



Docker

Running Docker containers on the cluster may be done using either the preferred Apptainer or UDOCKER.

Apptainer

- Create an interactive job using the following script: /storage/pycharm.sh
- Ssh to the compute node that was allocated to you
- Download your desired container and create a .sif file:
  apptainer build --force --sandbox my_container.sif docker://my_container_page/my_container
- Run the container (the last argument is a command. In this example, it starts a bash shell):
  apptainer exec my_container.sif /bin/bash

You may bind a folder to the container's folder by adding the following switch:
--bind host1:containers1,host2:container2

UDOCKER

UDOCKER shall be installed in a Conda environment.

UDOCKER is not a full Docker replacement and usage is currently **limited to pulling and running containers**. The actual containers should be built using Docker and dockerfiles.

UDOCKER is not a full Docker replacement and usage is currently **limited to pulling and running containers**. The actual containers should be built using Docker and dockerfiles.

Installation

- Installation for Python3 udocker. For Python2: remove the "python=3.8" string in the next line.

conda create -n udocker_env python=3.8

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

# pull image, create container and '--rm' remove 'hello-world' container after run

udocker run hello-world
```

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.

Run Jupyter Lab in Udocker

- After installing a udocker conda environment as explained above, copy file /storage/udocker_jup.sbatch to your directory
- Modify the environment name and the docker image as desired.
- Run: sbatch udocker_jup.sbatch
- Open the output file and grab the port number from the first lines. Grab the token from the last lines of the file (it may take some time before the image is downloaded and the token displayed. For the example image it took 15 minutes)
- Replace the port number of the token with the port number you grabbed.
- Replace the hostname with its IP address (you can ping to it to have its IP address displayed)
- Paste the modified token in your favorite web browser's address bar and hit <Enter>

Matlab

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

module load matlab

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

Make sure your ssh terminal supports x11 forwarding!

Send Matlab script to run as batch by headless Matlab:

srun --nodes=1 --mem=24G --cpus-per-task=4 --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

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

Activate your conda environment.

conda install -c conda-forge dotnet-sdk

Use

- Use ./pycharm.sh script (see *pyCharm*) to allocate a compute node.
- ssh to the compute node
- Activate environment: *conda activate <your dotnet environment name>*
- Create a new dotnet project: *dotnet new console -o myApp*
- *cd myApp*
- Run: *dotnet run*

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 132.72.64.80 , 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>

   In most cases it is advisable to use 'conda install' rather than 'pip install'

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
```

Remove Environment

```
conda env remove --name myenv
```

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 --load-cookies /tmp/cookies.txt https://docs.google.com/uc?export=download&confirm=$(wget --quiet --save-cookies /tmp/cookies.txt --keep-session-cookies --no-check-certificate 'https://docs.google.com/uc?export=download&id=<YOUR FILE ID>' -O- | sed -rn 's/.*confirm=([0-9A-Za-z_]+).*/\1\n/p')&id=<YOUR FILE ID>
```

where <YOUR FILE ID> is the alphanumeric long string that shows when you right click the file in Chrome and view the file's link.

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.

- ### 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.

- ### How to profile python memory usage?

  https://www.pluralsight.com/blog/tutorials/how-to-profile-memory-usage-in-python

- ### Using Bert consumes a lot of RAM and either OOM errors occur or a lot of RAM needs to be allocated.

  Follow the following links to learn how to reduce memory consumption when working with Bert:

github.com/google-research/bert#out-of-memory-issues

stackoverflow.com/questions/59617755/training-a-bert-based-model-causes-an-outofmemory-error-how-do-i-fix-this

- I need java version 11, but conda install openjdk, installs a version called 1.8

  If you are good with java 21: *conda install -c conda-forge openjdk*

  If not, the following will install java 11 in your home folder:

  Create a conda environment and pip install install-jdk==0.1.0

  In python:

  ```
  import jdk

  jdk.install('11')
  ```

  For using this java version, you have to modify environment variables (e.g. version 11.0.22):

  ```
  export JAVA_HOME="/home/<your username>/.jdk/jdk-11.0.22+7"

  export PATH=$JAVA_HOME/bin:$PATH
  ```

Errors

- Running Jupyter in VS Code I get the following error: Failed to start the Kernel. request to https://some.ip.and:port/api/sessions?a_number failed, reason: self signed certificate. View Jupyter log for further details.

  Make sure you use the settings described in chapter *Run Jupyter Notebook*.

- 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>
…
```

- Working with udocker I get these errors when pulling an image. TypeError: unicode argument expected, got 'str', Error: manifest not found or not authorized

  Your udocker installation uses python2, instead of python3. Refer to *Docker Installation*.

- VSCode error: Could not establish connection to… The VS Code Server failed to start SSH

  This is an SSL certificate error on the client side. Make sure SSH package of VS Code is updated and follow the instructions described in chapter Run Jupyter Notebook.

- 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

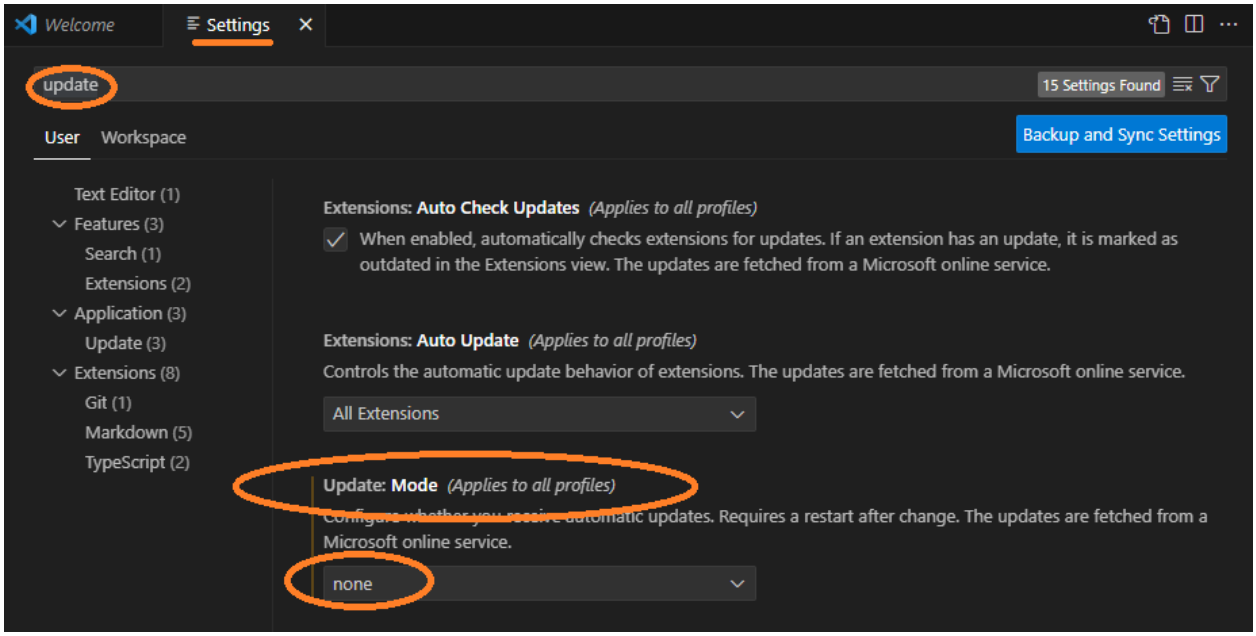
  These errors mean that the remote host key does not much to the key saved locally, anymore. The keys mismatch may be a result of reinstalling the remote host. If that is the reason go to

C:\Users\<your Windows user>\.ssh\ and change the names of the files. Windows will create new updated files instead.

- VSCode glibc errors when connecting to cluster

  New versions from 1.86 on require advanced glibc versions which are currently not available for the cluster. Uninstall VSCode and install VSCode 1.85 from here: update.code.visualstudio.com/1.85.2/win32-x64-user/stable

  Then, immediately, disable auto update by: File -> settings, type “update” and change “Update: Mode” value to “none”. See picture below.



- Cannot find kernels when running notebooks in VS Code after reinstalling VS Code

  Your profile may be corrupted. File->Preferences->Profile->Show Profile Content and inspect the packages for irregularities. You can also create a new profile in that menu.

- When using gfortran I get the following errors: undefined reference to `dsyev_' or undefined reference to `dgemm_'

  Write the command as follows: gfortran /home/<my username>/my_fortran_file.f90 -o file.output -L/home/<my username>/.conda/envs/<conda env name>/lib/ -llapack -lblas

  Make sure you have blas and lapack packages installed in your conda 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

- Got 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”

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