Compute Canada

A shared computing resources for use by researchers across Canada.

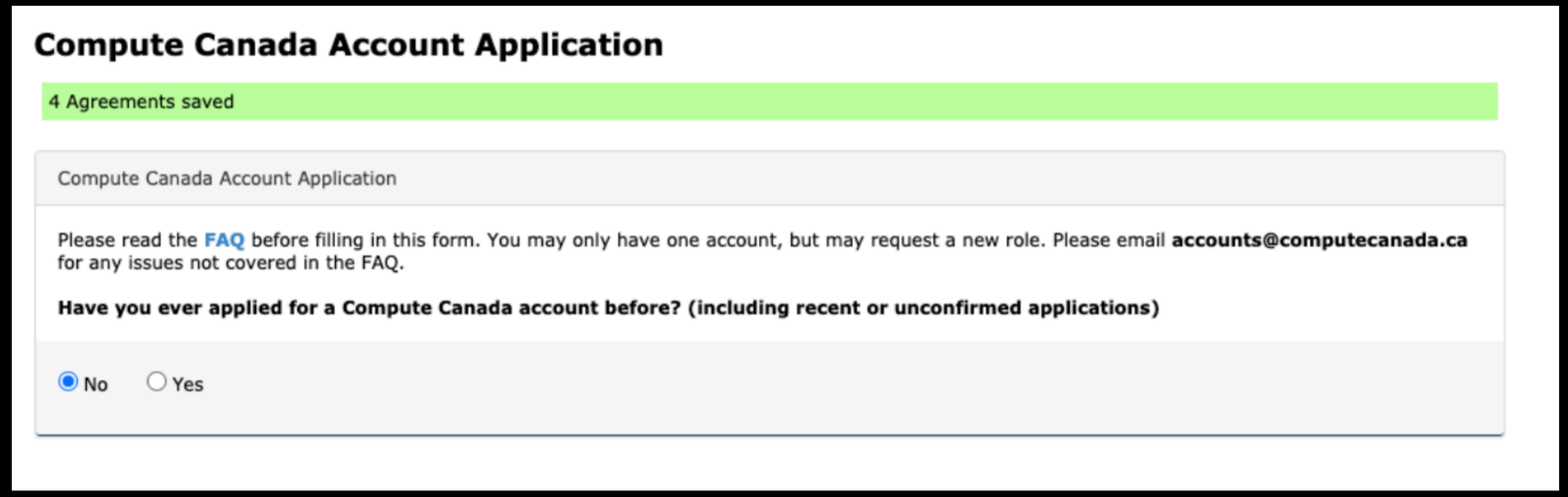
sign up process

1. Registration Link: https://ccdb.computecanada.ca/security/login

sign up process

2. Accept the required agreements

3. Answer if you applied before.



sign up process

- Use uvic email id for signup process
- Sponsor identifier for neil is gtq-965-aa
- submit application and wait for approvals from Neil and also Compute canada staff (typically done within a business day)

logging In

- After receiving an email saying account is activated
- login using “ssh <u>username@cedar.computecanada.ca</u>"

Folder Structure

- Nearline - long term storage, mainly for keeping energy consumption to zero, it is slower to access, as it is not intended for active projects.
- projects - recommended storage for active projects
- scratch - temporary, fast storage for data during job execution or quick experiments, it is deleted every month.

- there should be a folder by your username in def-nernst directory inside projects file system.
- recommended place for project files - “cd projects/def-nernst/username”

```
================================================================================
Welcome to Cedar!

For information see: https://docs.computecanada.ca/wiki/Cedar
Email support@computecanada.ca for assistance and/or to report problems.
================================================================================

[[rpudari@cedar5 ~]$ ls
 nearline  projects  scratch
[[rpudari@cedar5 ~]$ cd projects
[[rpudari@cedar5 projects]$ ls
 def-nernst
[[rpudari@cedar5 projects]$ cd def-nernst
[[rpudari@cedar5 def-nernst]$ ls
 akoenzen  nernst  roshan  rpudari  satt-replication.ipynb  zanelib1
[[rpudari@cedar5 def-nernst]$ cd rpudari
 [rpudari@cedar5 rpudari]$
```

DON’T BE THE PERSON WHO DOES THESE

- Limit the usage of login node to file transfers and job scheduling.
- Prefer editing your source files locally over Compute Canada.
- DO NOT use the login system to run ANY scripts, always use job scheduler.
  - this makes the resource busy and others won’t be able to login.

File Transfer

- Globus - recommended by CC, gives best performance, third party.
- Rsync - slower and best for small files.
- SFTP - easiest, and reasonably fast. (faster inside uvic network)
  - login - “sftp username@cedar.computecanada.ca"
  - cd to the directory and use get and put to transfer both ways.

shell scripts for jobs

- use #SBATCH to pass all the required arguments for the job.
  - there are many more optional arguments which can be found in the documentation.

```
[[rpudari@cedar5 rpudari]$ cat process.sh
 #!/bin/bash
 #SBATCH --account=def-nernst
 #SBATCH --time=0:0:05
 #SBATCH --mail-user=rpudari@uvic.ca
 #SBATCH --mail-type=BEGIN
 #SBATCH --mail-type=END
 #SBATCH --cpus-per-task=1
 #SBATCH --mem=10G

 echo hello

 [rpudari@cedar5 rpudari]$
```

Job Scheduler

- use “sbatch script_name.sh” to submit a job.
- you can view the list of all submitted jobs and their status using “sq”
- you will receive emails if you asked it to send notifications in SBATCH arguments

```
[[rpudari@cedar5 rpudari]$ sbatch process.sh
 Submitted batch job 26842290
[[rpudari@cedar5 rpudari]$ sq
           JOBID     USER      ACCOUNT           NAME  ST  TIME_LEFT NODES CPUS TRES_PER_N MIN_MEM NODELIST (REASON)
        26842290  rpudari def-nernst_c     process.sh  PD       1:00     1    1        N/A     10G  (Priority)
 [rpudari@cedar5 rpudari]$
```

Running Python Scripts

- make sure to use check documentation on which python versions are available, if you need a version not in the list, you need to write shell script to download and install your specific version.
  - run “module avail python” to get a list of versions available.

```
[[rpudari@cedar5 rpudari]$ module avail python

------------------------------------------------------------------------ Core Modules ------------------------------------------------------------------------
   ipython-kernel/2.7    ipython-kernel/3.7        ipython-kernel/3.10            python/3.7.7 (t)        python/3.8.10 (t,D:3.8)
   ipython-kernel/2.7    ipython-kernel/3.8 (D)    python/2.7.18       (t,2.7)    python/3.7.9 (t,3.7)    python/3.9.6  (t,3.9)
   ipython-kernel/3.6    ipython-kernel/3.9        python/3.6.10       (t,3.6)    python/3.8.2 (t)        python/3.10.2 (t)

  Where:
   t:        Tools for development / Outils de développement
   Aliases:  Aliases exist: foo/1.2.3 (1.2) means that "module load foo/1.2" will load foo/1.2.3
   D:        Default Module

Use "module spider" to find all possible modules and extensions.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".


[rpudari@cedar5 rpudari]$
```

Running Python Scripts

- scipy-stack is stack of popular python modules like Numpy, Scipy, pandas, matplotlib. (check documentation for complete list)
- it will automatically load the most recent compatible version for your python version. (you can also specify versions in requirements file)

```
[[rpudari@cedar5 rpudari]$ cat install.sh
#!/bin/bash

#SBATCH --account=def-nernst
#SBATCH --time=0:30:00
#SBATCH --mail-user=rpudari@uvic.ca
#SBATCH --mail-type=BEGIN
#SBATCH --mail-type=END
#SBATCH --cpus-per-task=4
#SBATCH --mem=30G

module load python/3.9
module load scipy-stack
virtualenv --no-download $SLURM_TMPDIR/env
source $SLURM_TMPDIR/env/bin/activate
pip install --no-index --upgrade pip
pip install -v --no-binary=all -r requirements.txt

python data_download.py
[rpudari@cedar5 rpudari]$
```

Running Python Scripts

- use virtualenv to create virtual environment
- If you omit the —no-index option, pip will search both PyPI and local packages (compute canada wheels), and use the latest version available
- you can use the —no-binary option, which tells pip to ignore pre-built packages entirely and compile latest versions from source.

```
[[rpudari@cedar5 rpudari]$ cat install.sh
#!/bin/bash

#SBATCH --account=def-nernst
#SBATCH --time=0:30:00
#SBATCH --mail-user=rpudari@uvic.ca
#SBATCH --mail-type=BEGIN
#SBATCH --mail-type=END
#SBATCH --cpus-per-task=4
#SBATCH --mem=30G

module load python/3.9
module load scipy-stack
virtualenv --no-download $SLURM_TMPDIR/env
source $SLURM_TMPDIR/env/bin/activate
pip install --no-index --upgrade pip
pip install -v --no-binary=all -r requirements.txt

python data_download.py
[rpudari@cedar5 rpudari]$
```

Running Python Scripts

- sample script for running pytorch on a single GPU

```
#!/bin/bash
#SBATCH --nodes 1
#SBATCH --gres=gpu:1 # request a GPU
#SBATCH --tasks-per-node=1
#SBATCH --cpus-per-task=1 # change this parameter to 2,4,6,... and increase "--num_workers" accordingly to see the effect on performance
#SBATCH --mem=8G
#SBATCH --time=0:05:00
#SBATCH --output=%N-%j.out
#SBATCH --account=<your account>

module load python # Using Default Python version - Make sure to choose a version that suits your application
virtualenv --no-download $SLURM_TMPDIR/env
source $SLURM_TMPDIR/env/bin/activate
pip install torch torchvision --no-index

echo "starting training..."
time python cifar10-gpu.py --batch_size=512 --num_workers=0
```

Caveats

- Outputs will be saved in “jobid.out” file in your directory, or you can also specify output file name in sbatch parameters in shell script
- File outputs such as csv or trained model pickle files will be saved in your directory.
- Pip has a tendency to go into infinite loop trying to find dependency version that satisfies every package requirements.
- DO NOT install Anaconda. Compute Canada is very slow with it and causes many conflicts.
- you can launch Jupyter notebooks, check documentation (I never tried it)
- Google is really bad in finding ccdb documentation pages, instead use compute canada’s built-in search function on top right of every page in documentation to find relevant information.