The HEC

HEC = High End Computing

This presentation is based on the guide provided by the HEC of which this contains more details if wanted:

https://answers.lancaster.ac.uk/display/ISS/High+End+Computing+%28HEC%29+help

Presentation created by: Andrew Moore

Funded by: UCREL research centre

What is the HEC?

- Consists of:
  - 1 login node -> The computer that is used when you login. This is slow and should not be used for any tasks other than monitoring or assigning jobs really.
  - CPU nodes -> Various 16 core nodes with either 64 or 128GB of memory. Various 40 core nodes with 192GB of memory.
  - GPU nodes -> 2
- All computers/nodes are Linux based:
  - Good resource for learning Linux: https://robinlong-tutorials-linux.readthedocs.io/en/latest/introduction.html

What is the HEC?

- CPU Nodes:
  - Various 16 core nodes with between 64 and 128GB of memory.
  - In total 9,900 cores and 50TB of memroy.
- 2 GPU nodes each containing:
  - 3 Nvidia V100 32GB, 32 CPU cores, 192GB memory.

Get access?

- First you need to get a login, ask your PI/Supervisor to apply for an account. See here for more details: https://answers.lancaster.ac.uk/display/ISS/Get+access+to+the+HEC
- Once you have a login access via ssh using Lancaster login:

```
ssh username@wayland.hec.lancaster.ac.uk
```

- See here for more details for login (windows): https://answers.lancaster.ac.uk/display/ISS/Logging+in+to+the+HEC

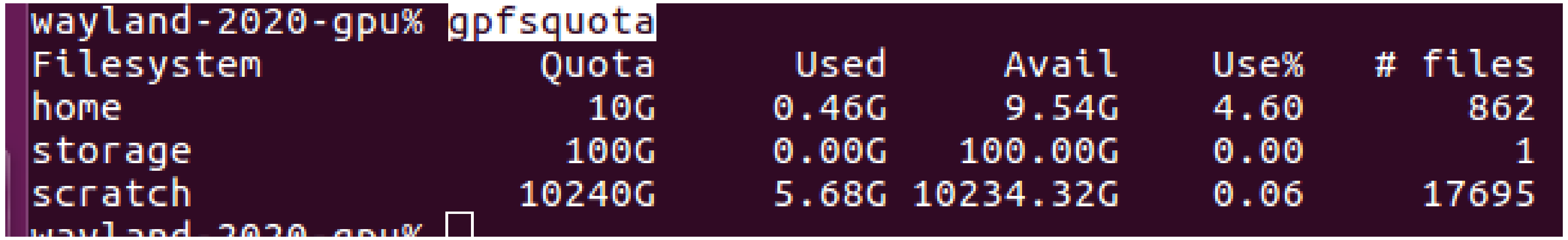
File store/File quota

- Home -> 10GB -> Backup nightly -> Permanent -> $HOME
- Storage -> 100GB -> No Backup -> Permanent -> $global_storage
- Scratch -> 10TB -> No Backup -> Deleted after 4 weeks -> $global_scratch
- Temp -> Unlimited -> No Backup -> Only exists when the job is running -> $TMPDIR

NOTE: $TMPDIR environment variable only exists when the job is running all others exist on the login node

File store/File quota

- To check the amount of storage used run: `gpfsquota`:

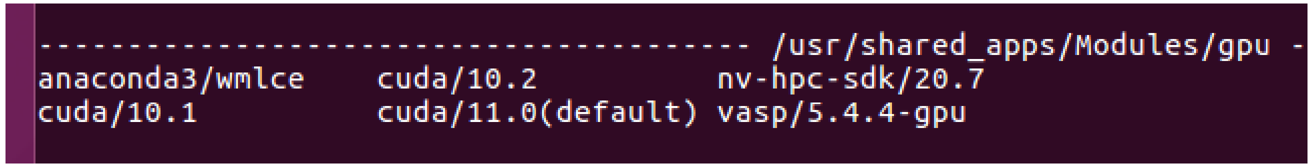


```
wayland-2020-gpu% gpfsquota
Filesystem              Quota       Used      Avail      Use%   # files
home                      10G      0.46G      9.54G      4.60       862
storage                  100G      0.00G    100.00G      0.00         1
scratch                10240G      5.68G  10234.32G      0.06     17695
```

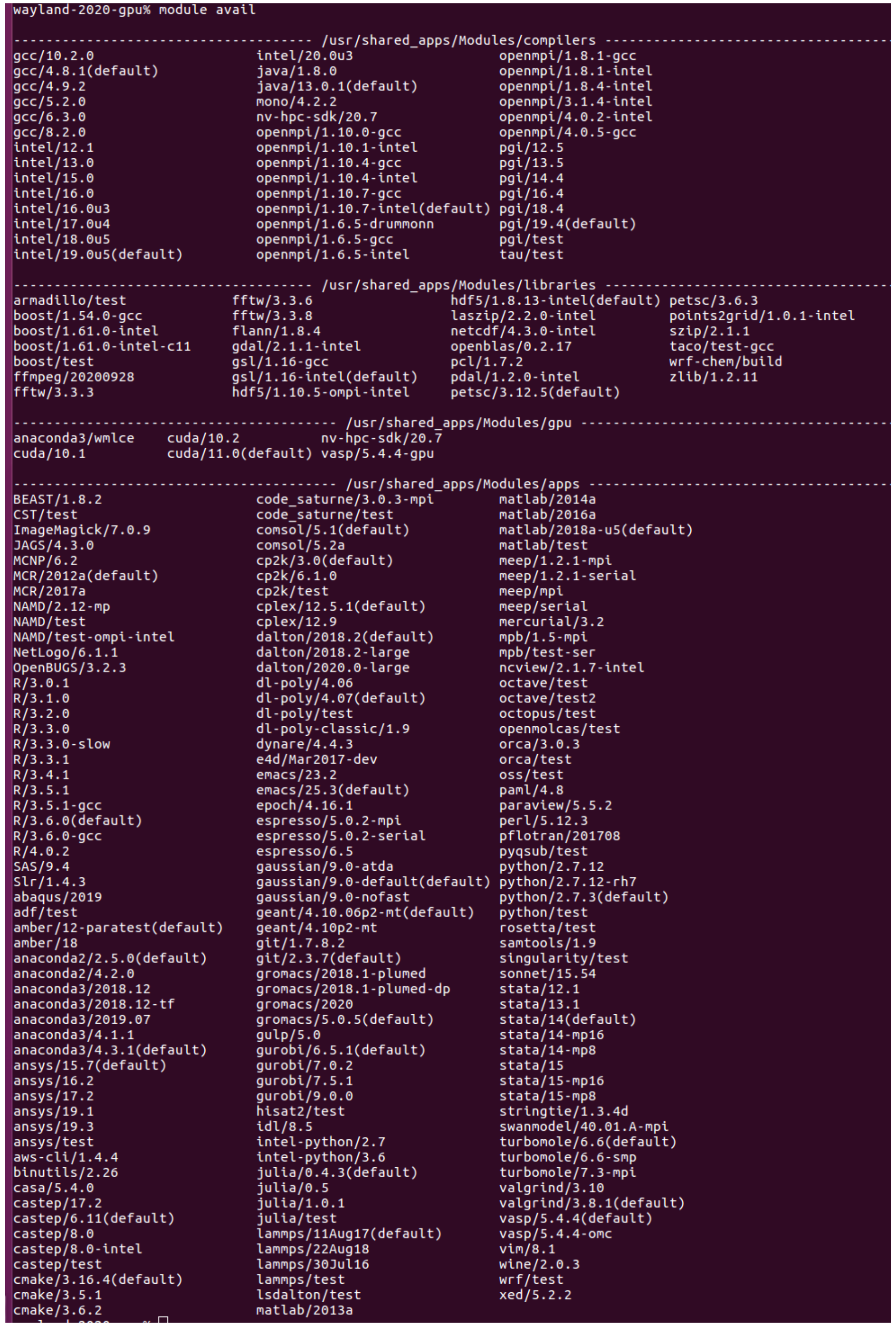
- The 10TB of scratch area is really useful. However files will be deleted end of day if last modified time is 4 weeks old, this point is really important as a lot of files you may have downloaded will likely have a last modified time of more than 4 weeks.

Installing Software

- Pre-installed software: `module avail`
- If you would like custom software installed on the HEC (including Conda environments) request it through https://helpcentre.lancaster.ac.uk before creating your own custom installation.



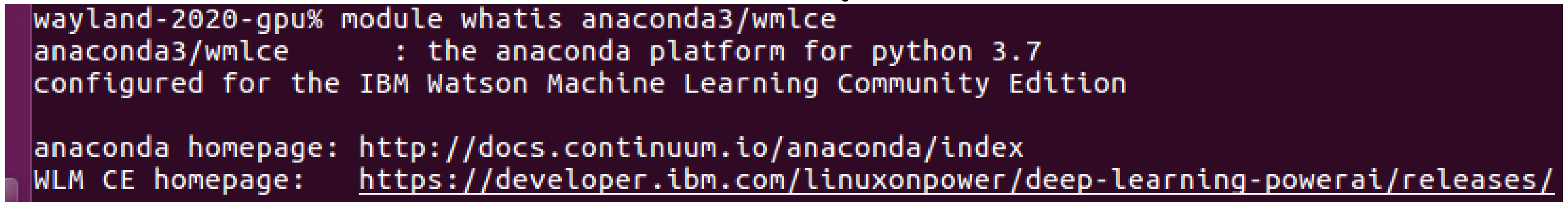
```
---------------------------------------- /usr/shared_apps/Modules/gpu -
anaconda3/wmlce    cuda/10.2          nv-hpc-sdk/20.7
cuda/10.1          cuda/11.0(default) vasp/5.4.4-gpu
```



```
wayland-2020-gpu% module avail

------------------------------------ /usr/shared_apps/Modules/compilers -----------------------------------
gcc/10.2.0                    intel/20.0u3                  openmpi/1.8.1-gcc
gcc/4.8.1(default)            java/1.8.0                    openmpi/1.8.1-intel
gcc/4.9.2                     java/13.0.1(default)          openmpi/1.8.4-intel
gcc/5.2.0                     mono/4.2.2                    openmpi/3.1.4-intel
gcc/6.3.0                     nv-hpc-sdk/20.7               openmpi/4.0.2-intel
gcc/8.2.0                     openmpi/1.10.0-gcc            openmpi/4.0.5-gcc
intel/12.1                    openmpi/1.10.1-intel          pgi/12.5
intel/13.0                    openmpi/1.10.4-gcc            pgi/13.5
intel/15.0                    openmpi/1.10.4-intel          pgi/14.4
intel/16.0                    openmpi/1.10.7-gcc            pgi/16.4
intel/16.0u3                  openmpi/1.10.7-intel(default) pgi/18.4
intel/17.0u4                  openmpi/1.6.5-drummonn        pgi/19.4(default)
intel/18.0u5                  openmpi/1.6.5-gcc             pgi/test
intel/19.0u5(default)         openmpi/1.6.5-intel           tau/test

------------------------------------ /usr/shared_apps/Modules/libraries -----------------------------------
armadillo/test             fftw/3.3.6                 hdf5/1.8.13-intel(default) petsc/3.6.3
boost/1.54.0-gcc           fftw/3.3.8                 laszip/2.2.0-intel         points2grid/1.0.1-intel
boost/1.61.0-intel         flann/1.8.4                netcdf/4.3.0-intel         szip/2.1.1
boost/1.61.0-intel-c11     gdal/2.1.1-intel           openblas/0.2.17            taco/test-gcc
boost/test                 gsl/1.16-gcc               pcl/1.7.2                  wrf-chem/build
ffmpeg/20200928            gsl/1.16-intel(default)    pdal/1.2.0-intel           zlib/1.2.11
fftw/3.3.3                 hdf5/1.10.5-ompi-intel     petsc/3.12.5(default)

--------------------------------------- /usr/shared_apps/Modules/gpu ---------------------------------------
anaconda3/wmlce    cuda/10.2          nv-hpc-sdk/20.7
cuda/10.1          cuda/11.0(default) vasp/5.4.4-gpu

--------------------------------------- /usr/shared_apps/Modules/apps --------------------------------------
BEAST/1.8.2                   code_saturne/3.0.3-mpi        matlab/2014a
CST/test                      code_saturne/test             matlab/2016a
ImageMagick/7.0.9             comsol/5.1(default)           matlab/2018a-u5(default)
JAGS/4.3.0                    comsol/5.2a                   matlab/test
MCNP/6.2                      cp2k/3.0(default)             meep/1.2.1-mpi
MCR/2012a(default)            cp2k/6.1.0                    meep/1.2.1-serial
MCR/2017a                     cp2k/test                     meep/mpi
NAMD/2.12-mp                  cplex/12.5.1(default)         meep/serial
NAMD/test                     cplex/12.9                    mercurial/3.2
NAMD/test-ompi-intel          dalton/2018.2(default)        mpb/1.5-mpi
NetLogo/6.1.1                 dalton/2018.2-large           mpb/test-ser
OpenBUGS/3.2.3                dalton/2020.0-large           ncview/2.1.7-intel
R/3.0.1                       dl-poly/4.06                  octave/test
R/3.1.0                       dl-poly/4.07(default)         octave/test2
R/3.2.0                       dl-poly/test                  octopus/test
R/3.3.0                       dl-poly-classic/1.9           openmolcas/test
R/3.3.0-slow                  dynare/4.4.3                  orca/3.0.3
R/3.3.1                       e4d/Mar2017-dev               orca/test
R/3.4.1                       emacs/23.2                    oss/test
R/3.5.1                       emacs/25.3(default)           paml/4.8
R/3.5.1-gcc                   epoch/4.16.1                  paraview/5.5.2
R/3.6.0(default)              espresso/5.0.2-mpi            perl/5.12.3
R/3.6.0-gcc                   espresso/5.0.2-serial         pflotran/201708
R/4.0.2                       espresso/6.5                  pyqsub/test
SAS/9.4                       gaussian/9.0-atda             python/2.7.12
Slr/1.4.3                     gaussian/9.0-default(default) python/2.7.12-rh7
abaqus/2019                   gaussian/9.0-nofast           python/2.7.3(default)
adf/test                      geant/4.10.06p2-mt(default)   python/test
amber/12-paratest(default)    geant/4.10p2-mt               rosetta/test
amber/18                      git/1.7.8.2                   samtools/1.9
anaconda2/2.5.0(default)      git/2.3.7(default)            singularity/test
anaconda2/4.2.0               gromacs/2018.1-plumed         sonnet/15.54
anaconda3/2018.12             gromacs/2018.1-plumed-dp      stata/12.1
anaconda3/2018.12-tf          gromacs/2020                  stata/13.1
anaconda3/2019.07             gromacs/5.0.5(default)        stata/14(default)
anaconda3/4.1.1               gulp/5.0                      stata/14-mp16
anaconda3/4.3.1(default)      gurobi/6.5.1(default)         stata/14-mp8
ansys/15.7(default)           gurobi/7.0.2                  stata/15
ansys/16.2                    gurobi/7.5.1                  stata/15-mp16
ansys/17.2                    gurobi/9.0.0                  stata/15-mp8
ansys/19.1                    hisat2/test                   stringtie/1.3.4d
ansys/19.3                    idl/8.5                       swanmodel/40.01.A-mpi
ansys/test                    intel-python/2.7              turbomole/6.6(default)
aws-cli/1.4.4                 intel-python/3.6              turbomole/6.6-smp
binutils/2.26                 julia/0.4.3(default)          turbomole/7.3-mpi
casa/5.4.0                    julia/0.5                     valgrind/3.10
castep/17.2                   julia/1.0.1                   valgrind/3.8.1(default)
castep/6.11(default)          julia/test                    vasp/5.4.4(default)
castep/8.0                    lammps/11Aug17(default)       vasp/5.4.4-omc
castep/8.0-intel              lammps/22Aug18                vim/8.1
castep/test                   lammps/30Jul16                wine/2.0.3
cmake/3.16.4(default)         lammps/test                   wrf/test
cmake/3.5.1                   lsdalton/test                 xed/5.2.2
cmake/3.6.2                   matlab/2013a
```

Installing Software

- `module whatis anaconda3/wmlce`:



```
wayland-2020-gpu% module whatis anaconda3/wmlce
anaconda3/wmlce       : the anaconda platform for python 3.7
configured for the IBM Watson Machine Learning Community Edition

anaconda homepage: http://docs.continuum.io/anaconda/index
WLM CE homepage:   https://developer.ibm.com/linuxonpower/deep-learning-powerai/releases/
```

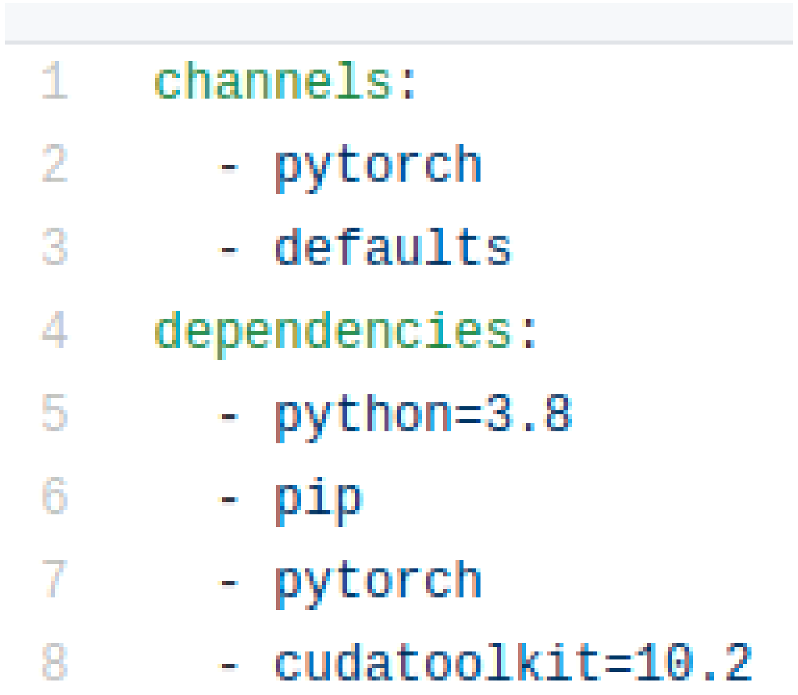
- `module add anaconda3/wmlce`
- `module list` -> lists all software currently being used.
- More details about what that package is doing run: `module show anaconda3/wmlce`

Anaconda3/wmlce package

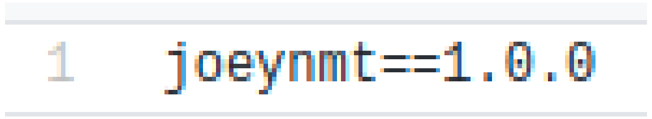
- `conda --version` -> conda 4.8.2 came out 24/1/2020
- `conda list` shows what packages have been installed
- `source activate wmlce_env` will use all of the packages that is associated with the `wmlce_env` environment. This includes Tensorflow etc.

Your own conda environment

- Still need to use the anaconda3/wmlce, as we need conda.
- We need to specify what we want to install via an environment file and a Python requirements file.



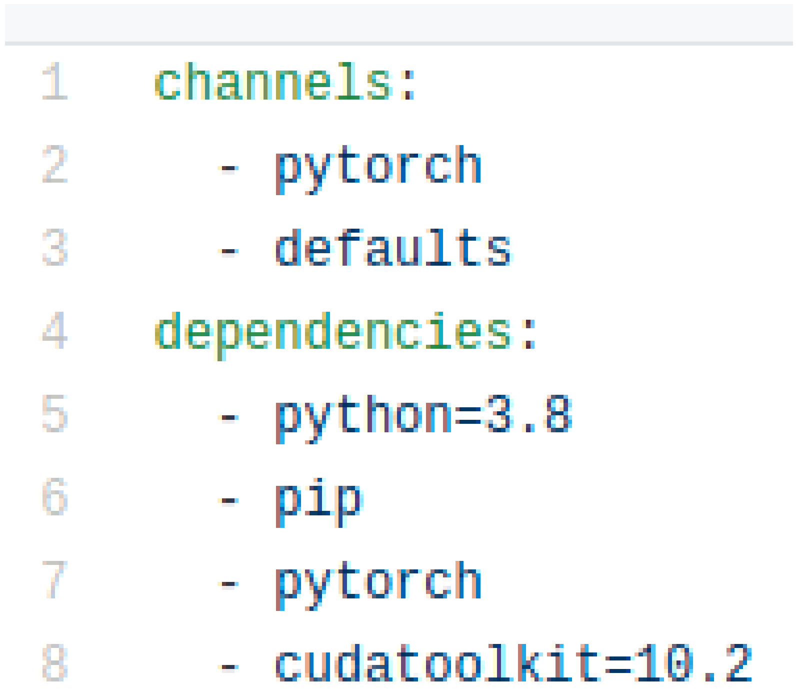
```
channels:
   - pytorch
   - defaults
dependencies:
   - python=3.8
   - pip
   - pytorch
   - cudatoolkit=10.2
```



```
joeynmt==1.0.0
```

Your own conda environment

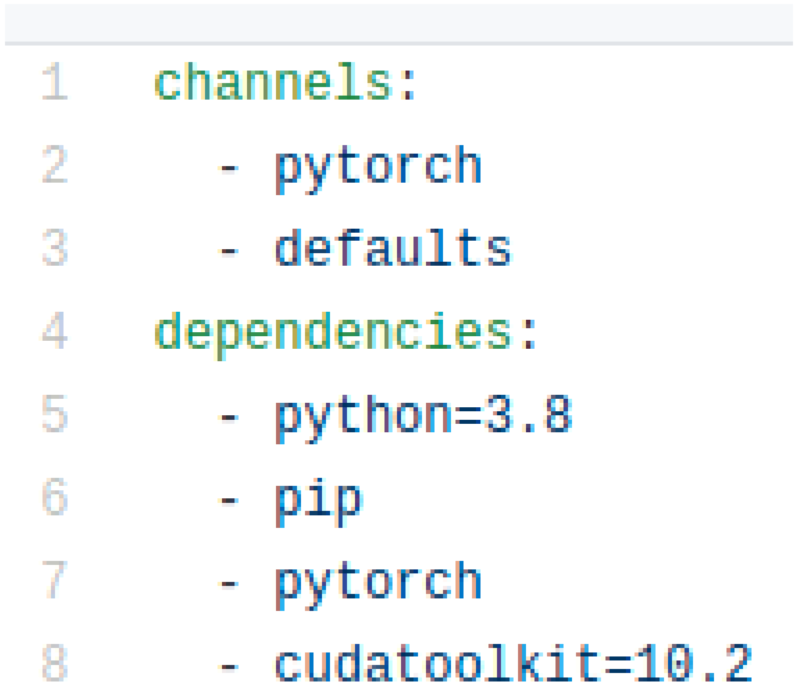
- `conda install pytorch cudatoolkit=10.2  -c pytorch`



```yaml
channels:
    - pytorch
    - defaults
dependencies:
    - python=3.8
    - pip
    - pytorch
    - cudatoolkit=10.2
```

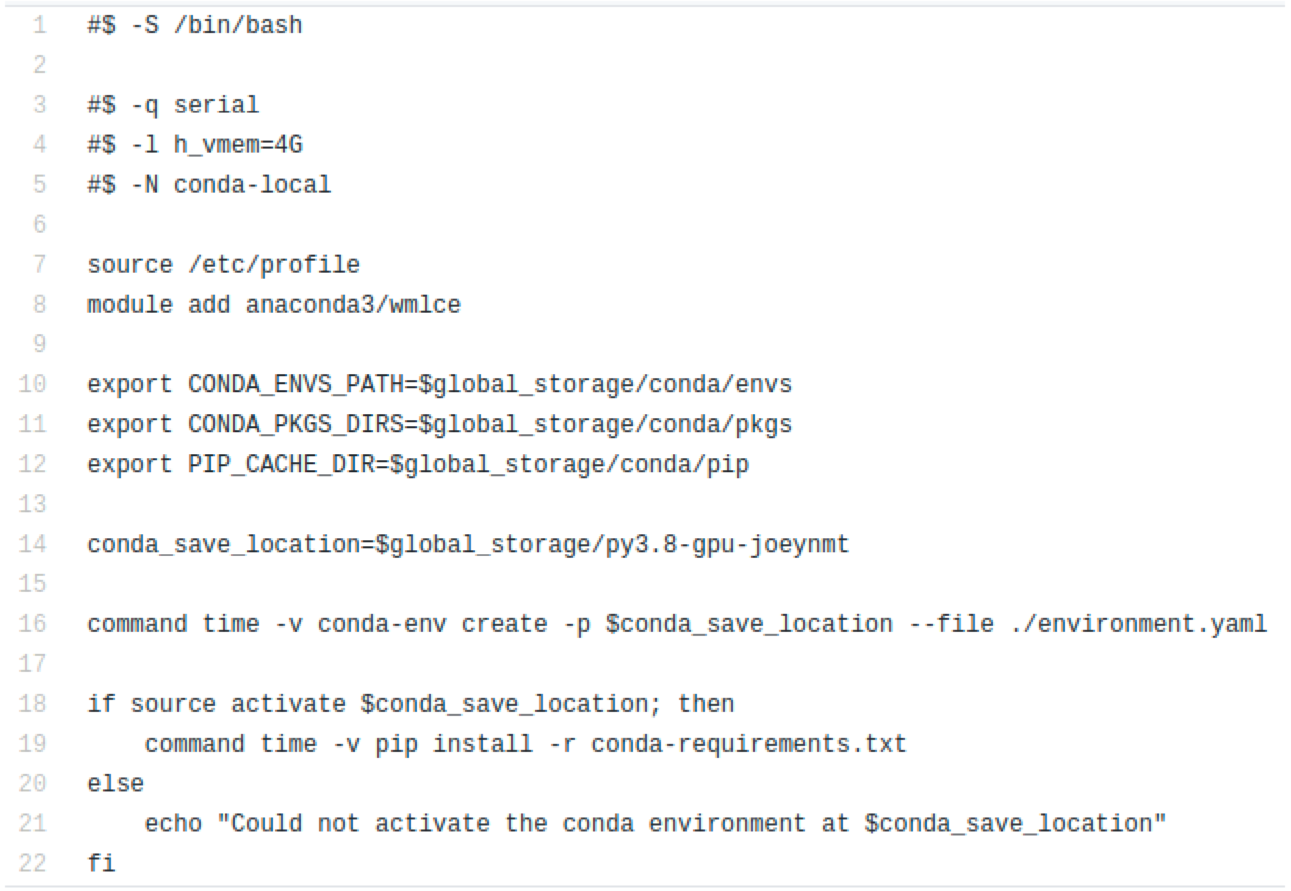
Your own conda environment

- The defaults/main channel for Linux and others:



```
channels:
  - pytorch
  - defaults
dependencies:
  - python=3.8
  - pip
  - pytorch
  - cudatoolkit=10.2
```

Your own conda environment



```
#$ -S /bin/bash

#$ -q serial
#$ -l h_vmem=4G
#$ -N conda-local

source /etc/profile
module add anaconda3/wmlce

export CONDA_ENVS_PATH=$global_storage/conda/envs
export CONDA_PKGS_DIRS=$global_storage/conda/pkgs
export PIP_CACHE_DIR=$global_storage/conda/pip

conda_save_location=$global_storage/py3.8-gpu-joeynmt

command time -v conda-env create -p $conda_save_location --file ./environment.yaml

if source activate $conda_save_location; then
    command time -v pip install -r conda-requirements.txt
else
    echo "Could not activate the conda environment at $conda_save_location"
fi
```

- Serial – single CPU node
- -l – 4GB memory
- -N – name of job

- File can be found here

Useful commands for running jobs

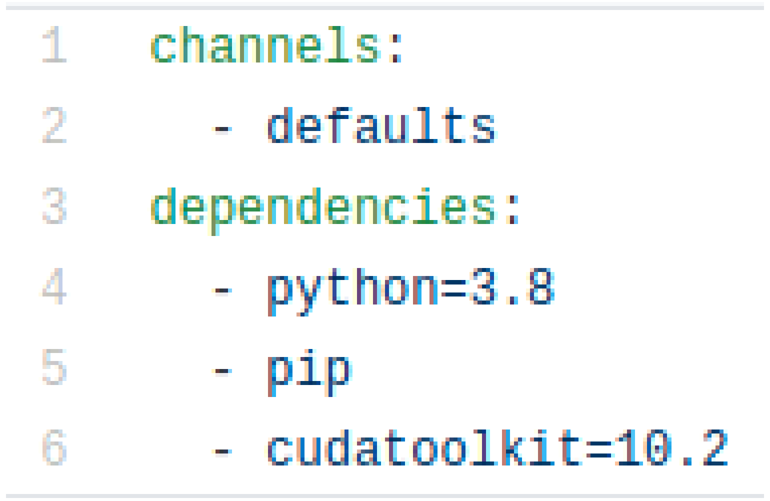
- Jobs can be run with the `qsub` command e.g. `qsub install.com`
- Check number of free slots `qslots` a more detailed view `qslots -v`
- Check status of jobs `qstat`
- Check CPU/Memory usage of jobs `qtop -u USERNAME`
  e.g. `qtop -u moorea`
- Check amount of resources used and are allowed to used (only applicable to CPU nodes) `qquota` LIMITED to 350 cores and 1.64TB memory

GPU Example – NER tagging using SpaCy

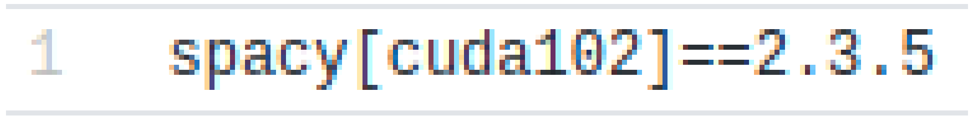
- 2 GPU nodes each containing:
  - 3 Nvidia V100 32GB, 32 CPU cores, 192GB memory.
- This is broken up mainly into GPUs e.g. 6 GPUs of which if you want to use multiple GPUs a hardware limit of 3 is applied.
- LIMITATION -> Only allowed to use a GPU node for 12 hours, but could use 3 GPUs for 12 hours ~ 1 GPU for 36 hours.

GPU Example – NER tagging using SpaCy

- Install the required Conda and Python packages:

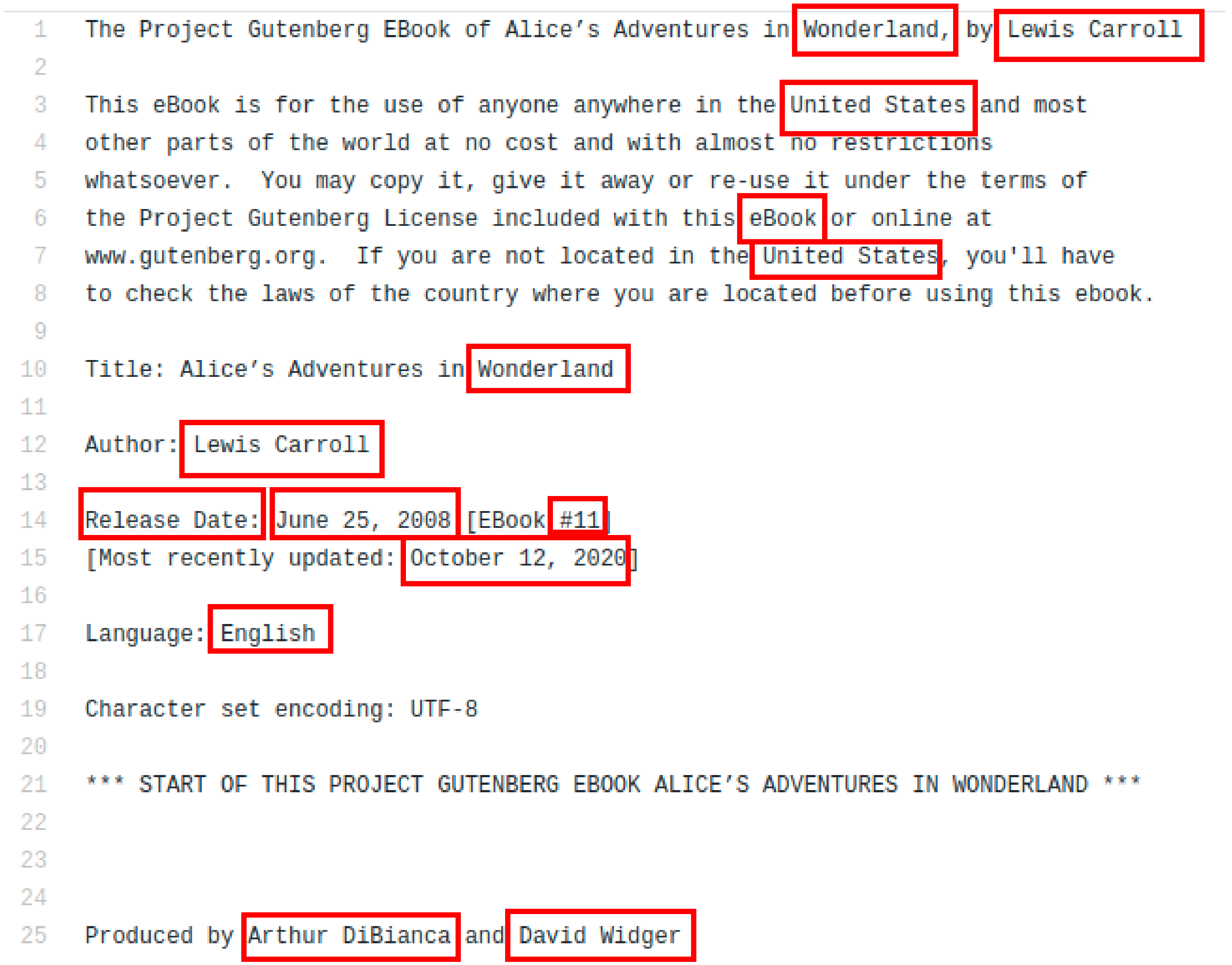


```
channels:
  - defaults
dependencies:
  - python=3.8
  - pip
  - cudatoolkit=10.2
```

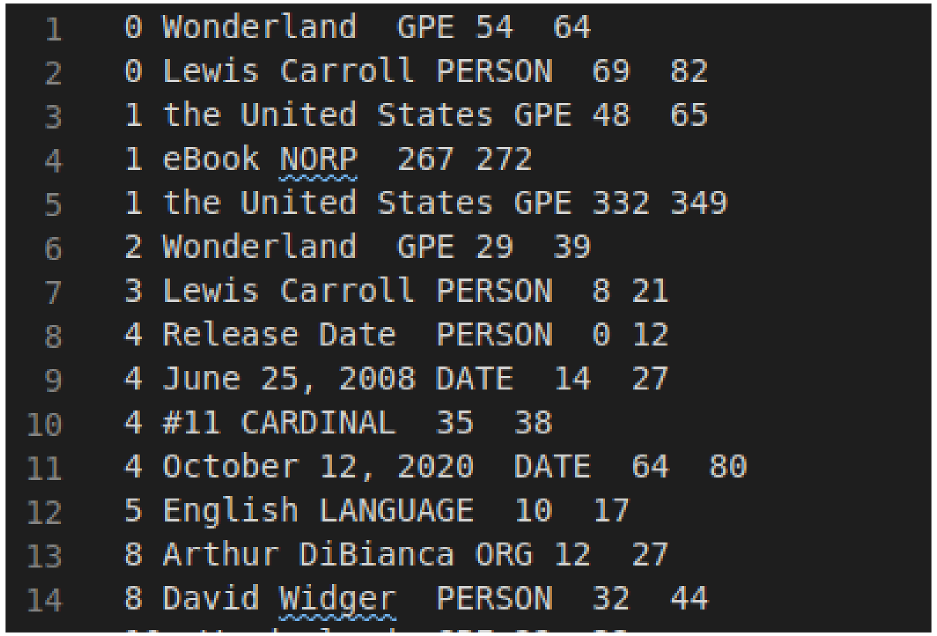


```
spacy[cuda102]==2.3.5
```

GPU Example – NER tagging using SpaCy



```
The Project Gutenberg EBook of Alice’s Adventures in Wonderland, by Lewis Carroll

This eBook is for the use of anyone anywhere in the United States and most
other parts of the world at no cost and with almost no restrictions
whatsoever.  You may copy it, give it away or re-use it under the terms of
the Project Gutenberg License included with this eBook or online at
www.gutenberg.org.  If you are not located in the United States, you'll have
to check the laws of the country where you are located before using this ebook.

Title: Alice’s Adventures in Wonderland

Author: Lewis Carroll

Release Date: June 25, 2008 [EBook #11]
[Most recently updated: October 12, 2020]

Language: English

Character set encoding: UTF-8

*** START OF THIS PROJECT GUTENBERG EBOOK ALICE’S ADVENTURES IN WONDERLAND ***



Produced by Arthur DiBianca and David Widger
```



```
0 Wonderland   GPE 54   64
0 Lewis Carroll PERSON   69   82
1 the United States GPE 48   65
1 eBook NORP  267 272
1 the United States GPE 332 349
2 Wonderland   GPE 29   39
3 Lewis Carroll PERSON   8 21
4 Release Date   PERSON   0 12
4 June 25, 2008 DATE   14   27
4 #11 CARDINAL   35   38
4 October 12, 2020   DATE   64   80
5 English LANGUAGE   10   17
8 Arthur DiBianca ORG 12   27
8 David Widger   PERSON   32   44
```

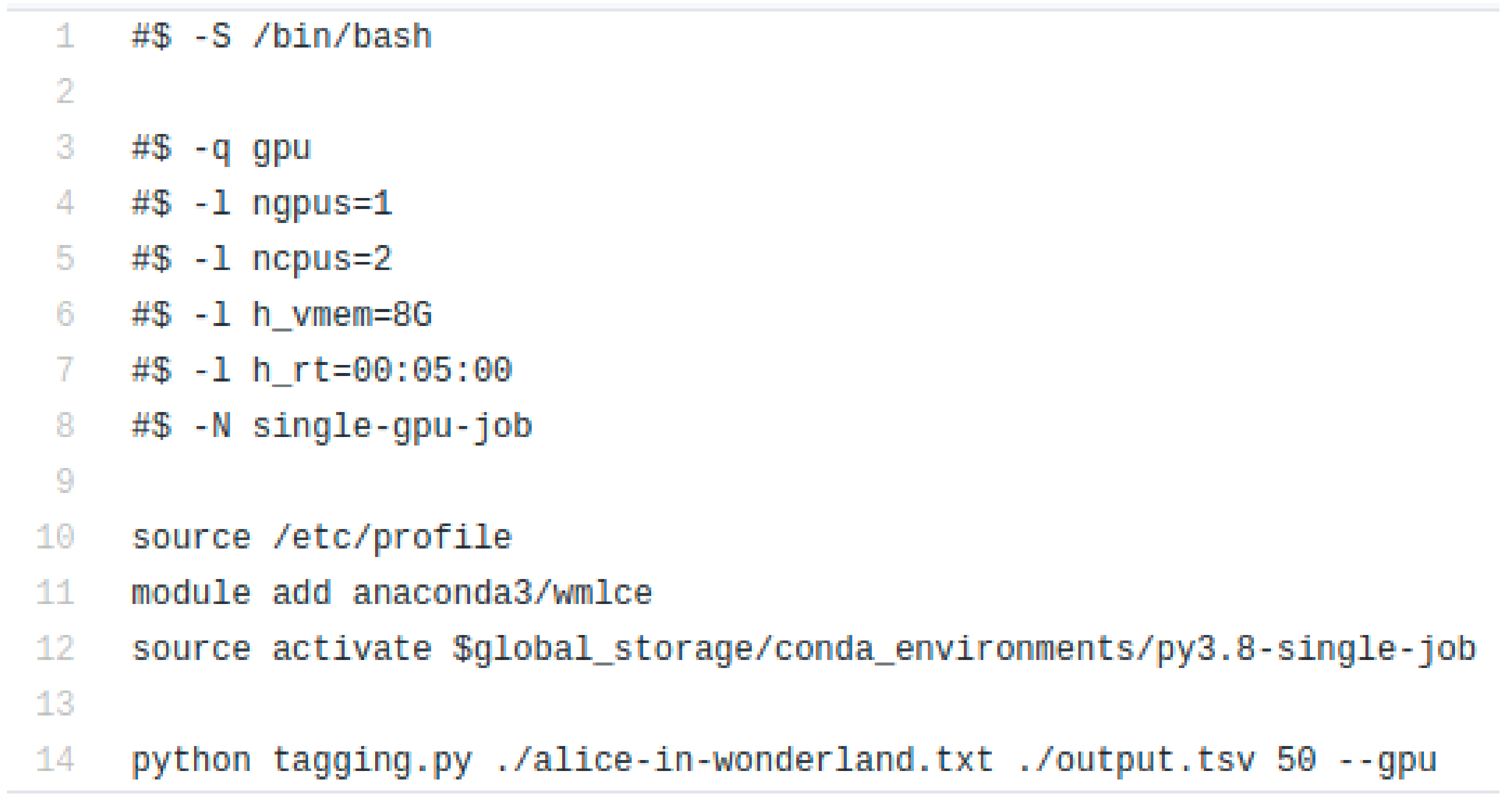
GPU Example – NER tagging using SpaCy

`tagging.py` takes 4 arguments:

1. Input text file to tag
2. File to save the TSV data too
3. Batch Size – 50 in this case
4. Use GPU or not

```
python tagging.py ./alice-in-wonderland.txt ./output.tsv 50 --gpu
```

GPU Example – NER tagging using SpaCy



```
#$ -S /bin/bash

#$ -q gpu
#$ -l ngpus=1
#$ -l ncpus=2
#$ -l h_vmem=8G
#$ -l h_rt=00:05:00
#$ -N single-gpu-job

source /etc/profile
module add anaconda3/wmlce
source activate $global_storage/conda_environments/py3.8-single-job

python tagging.py ./alice-in-wonderland.txt ./output.tsv 50 --gpu
```

- GPU– queue
- 1 GPU
- 2 CPUS
- 8GB RAM
- Job has upto 5 minutes to run

- File can be found here

GPU Example – NER tagging using SpaCy

Time taken to process Alice in Wonderland with SpaCy's small English NER model. Contains 881 paragraphs.

	Batch Size	
Hardware	50	1000
CPU only	1.8 seconds	1.1 seconds
GPU	12 seconds	0.6495 seconds

- For more NLP based HEC examples see: https://github.com/apmoore1/HEC

Thanks for listening, any questions?

Presentation created by: Andrew Moore
Twitter: @apmoore94

Funded by: UCREL research centre