Introduction to High Performance Computing (HPC)

QLS 612 - 2022
Brent McPherson, PhD

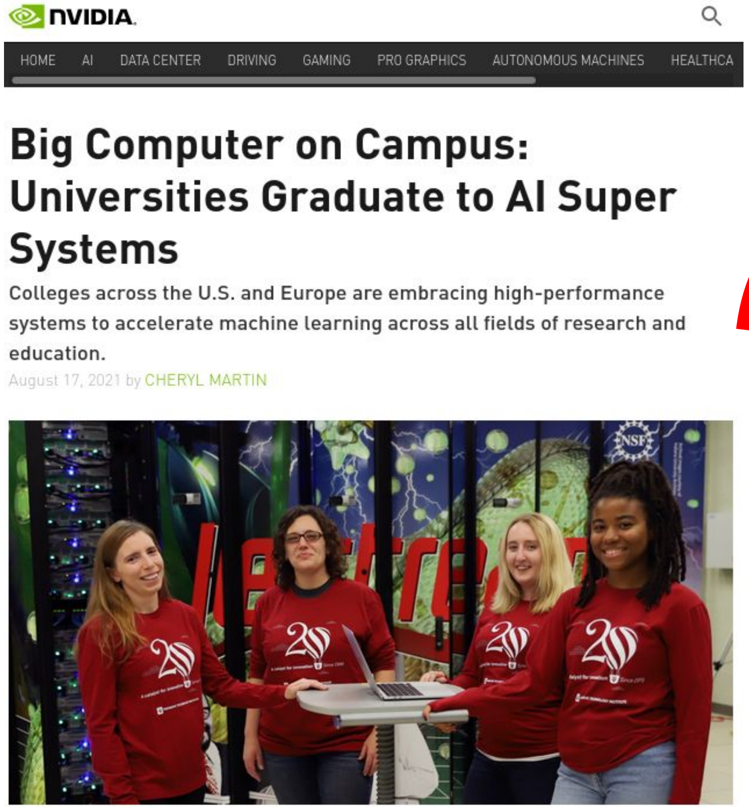
The goals of this lecture are to:

- Describe the structure of HPCs and how they differ from other traditional computers and common approaches.
    - Laptops / Desktops
- Introduce the concepts of using High Performance Computing (HPCs).
- How to begin effectively using HPC clusters in your work.

This will reference concepts covered in the “Introduction to the Terminal and Bash” and “Containerization” lectures.

Modern NeuroData Science Requires Computers

- No part of our research can progress without access to useful computing resources.
  - Data is acquired and stored on computers.
  - Data preparation and curation is performed on computers.
  - Analysis and modeling require computers.
  - Visualization and reporting or results require computers.
  - The distribution and reporting of results require computers.
- If at any point our work cannot continue in a digital space, it has effectively ended.
- In order to maximize our ability to perform innovative work in a field dependent on modern computational advances, a useful understanding of these machines is necessary.



Big Computer on Campus: Universities Graduate to AI Super Systems

Colleges across the U.S. and Europe are embracing high-performance systems to accelerate machine learning across all fields of research and education.

August 17, 2021 by CHERYL MARTIN

The Concepts of High Performance Computing (HPC)

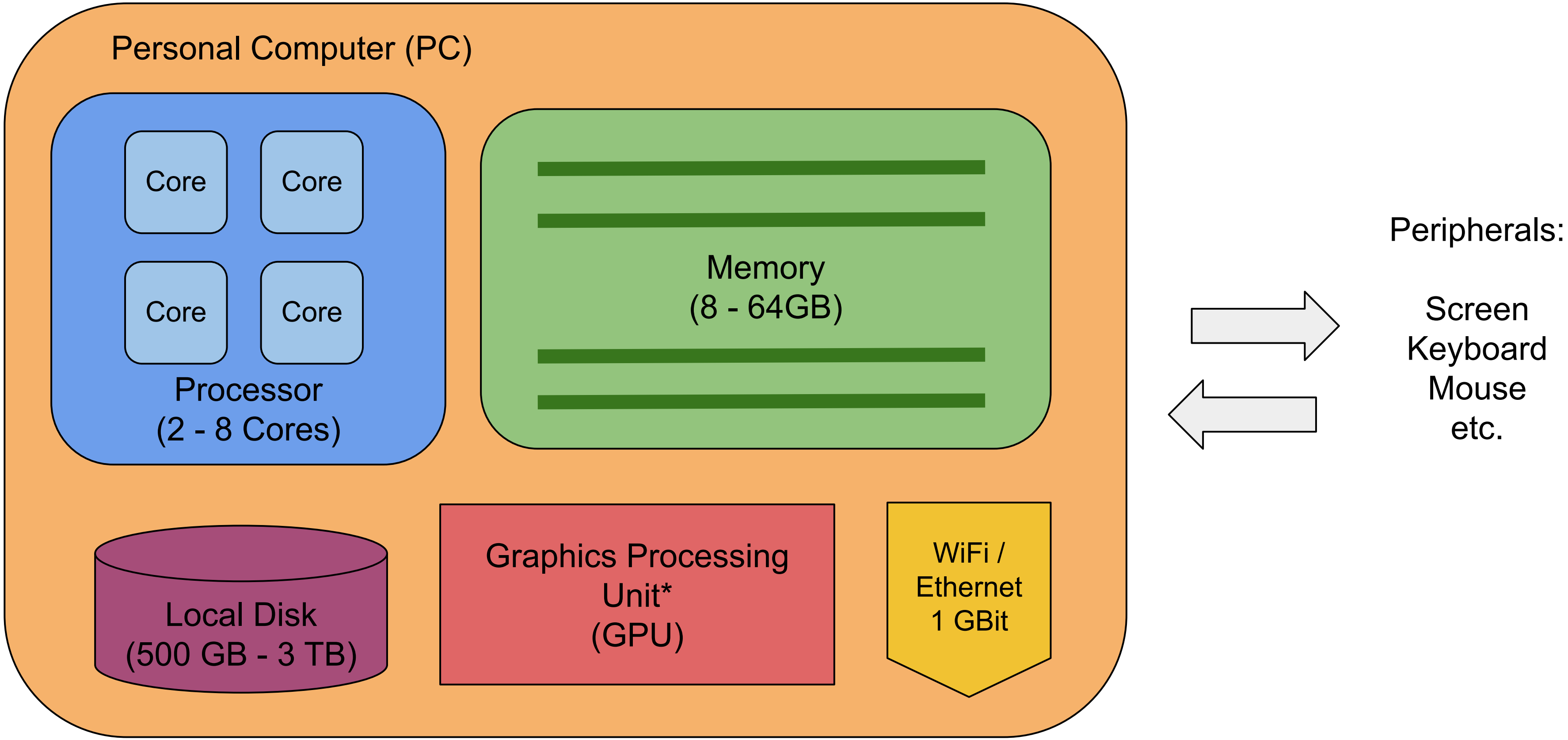
Working on a personal computer (PC)

- Everyone is familiar with this.
    - You’re using one now.
- These machines are designed to facilitate the most common needs of the most people.
- However, the limits of these machines can be easily reached when performing modern neuroscience analysis.

Working on an HPC

- These are not “personal” systems, they are multi-user.
- You do not interact with them directly, but over a network connection.
- The interaction with your analysis entails:
  - Transferring data / results.
  - Managing your analysis “jobs”.
- There is minimal “interactive” computing performed on a HPC.
  - Typically little to no visualization is performed on HPC systems.
- They almost exclusively run Linux.

The Schematic of a Computer



Personal Computer (PC)

Core

Core

Core

Core

Processor
(2 - 8 Cores)

Memory
(8 - 64GB)

Local Disk
(500 GB - 3 TB)

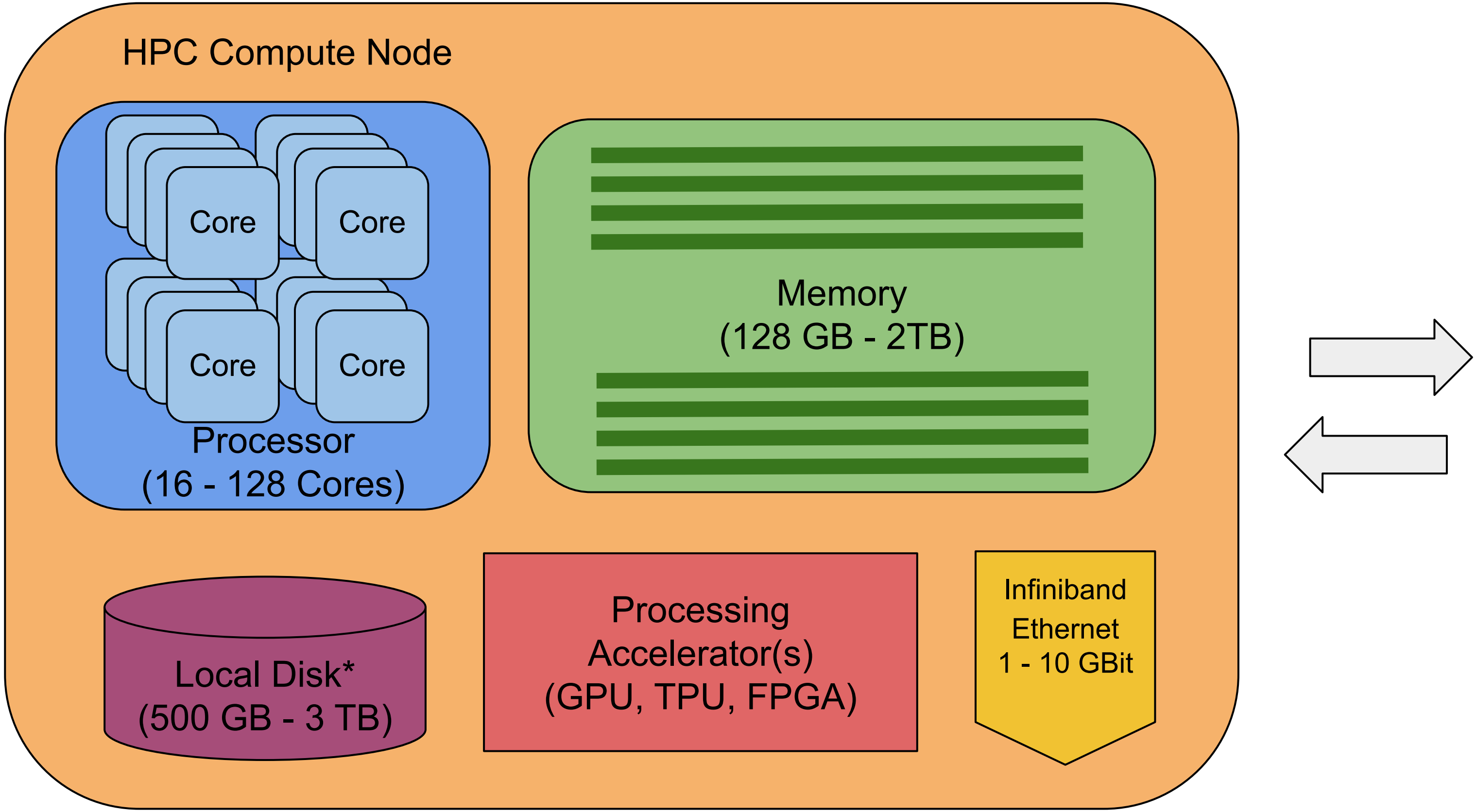
Graphics Processing
Unit*
(GPU)

WiFi /
Ethernet
1 GBit

Peripherals:

Screen
Keyboard
Mouse
etc.

* not necessarily present in all systems (laptops)



HPC Compute Node

Core

Core

Core

Core

Processor
(16 - 128 Cores)

Memory
(128 GB - 2TB)

Local Disk*
(500 GB - 3 TB)

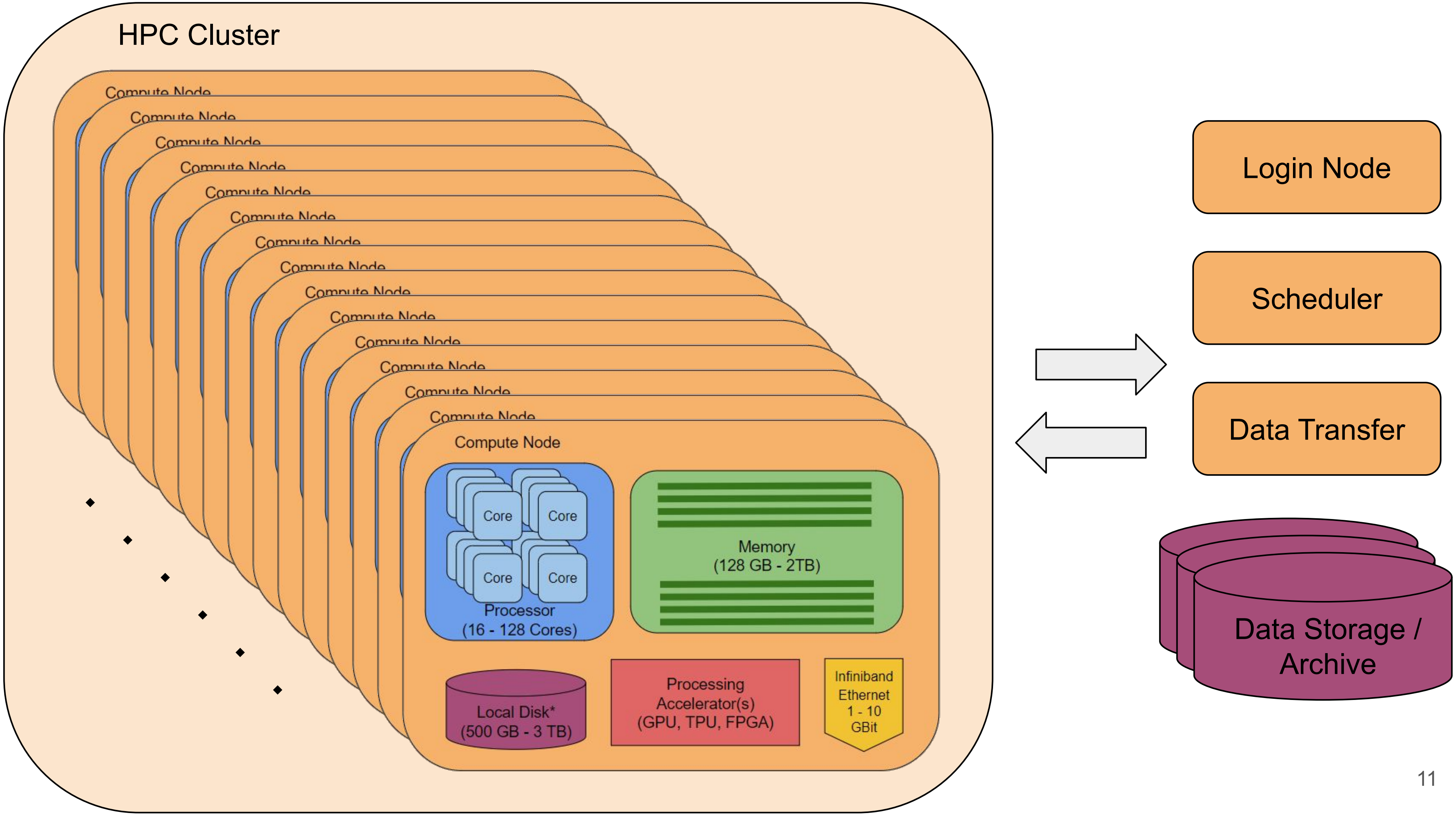
Processing
Accelerator(s)
(GPU, TPU, FPGA)

Infiniband
Ethernet
1 - 10 GBit

Peripherals:
None

Connects to
the rest of the
compute
cluster.

* it may be less and / or have higher access speed


HPC Cluster
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
Core
Core
Core
Core
Processor
(16 - 128 Cores)
Memory
(128 GB - 2TB)
Local Disk*
(500 GB - 3 TB)
Processing
Accelerator(s)
(GPU, TPU, FPGA)
Infiniband
Ethernet
1 - 10
GBit
Login Node
Scheduler
Data Transfer
Data Storage /
Archive

Other important things to keep in mind

- For a single, small process, an HPC may be slower than your laptop
    - The clock speed of processors is often slower when more cores are present.
    - There is also the overhead of moving / staging data, waiting for walltime, etc.
- Some HPCs have a different system architecture
    - Most PCs are 64-bit (x86) system architectures.
        - ARM is also common for some laptops (Apple M1)
    - Some HPCs do not have an x86 architecture
        - Big Red II (Cray), Longhorn (PPCx64)
    - This may cause challenges for deploying your code.

The Anatomy of a Computer

PCs

- 1 CPU
  - 2 - 8 cores
  - 3.5 - 4.5 GHz
- 8 - 64 GB RAM
- 500 GB - 3 TB Storage
- 1 GPU
  - This may be built into the CPU
- WiFi / 1 GBit Ethernet
- x86 (maybe ARM) architecture
- Single-User System
- Good for most general usage
  - This is likely where you will develop your analysis.

HPCs

- Multiple CPUs
  - 16 - 128 cores per CPU
  - 2.4 - 3.5 GHz
- 128 GB - 2 TB RAM
- 500 GB - 3 TB Storage
  - Not for long term storage
- 0 - 4 Processor Accelerators
  - GPUs w/ double precision CUDA
  - TPUs, FPGAs, Coprocessors, etc.
- 1 - 10 GBit Ethernet, Infiniband
- Different architectures (x86, PPC, Cray)
- Multi-User System
- Good for high throughput / large models.

Working on the Cloud - An expensive middle ground

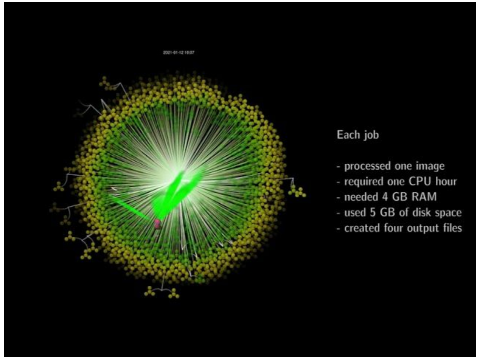
- This is most often an option through dedicated research proposals.
- The cloud system can be tailored to fit your specific analysis and data requirements.
  - A large virtual analysis computer
  - Your own dedicated cluster
- The data may be distributed on the cloud already.
  - Many open datasets are already on AWS.
- This will be *very* expensive for long term use.

The advantages of a HPC over a PC

- The ability to handle large datasets in a more time efficient manner.
- The option to “**Scale Out**” or “**Scale Up**” an analysis.
  - **Scale Out**: the ability to analyze independent parts of a dataset simultaneously
    - Independent permutations / cross-validations / simulations.
    - Different subjects through the same preprocessing preparations.
      - “Embarrassingly Parallel”
      - High Throughput Computing
  - **Scale Up**: create a larger single instance of computing resources to run a larger model, estimation, or analysis.
- We’ll be introducing you to “**Scaling Out**” your analysis in the example today.

Scaling Out

biobank uk



2021-01-12 18:07

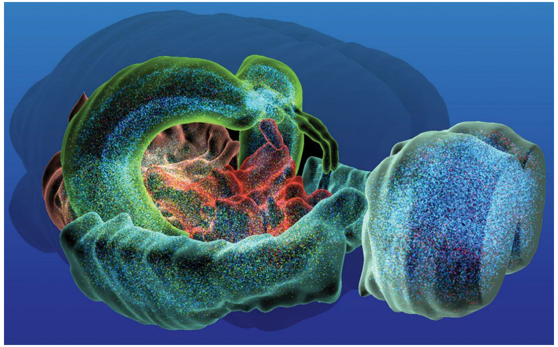
Each job

- processed one image
- required one CPU hour
- needed 4 GB RAM
- used 5 GB of disk space
- created four output files

[UKBB high-throughput replication of over 40K subjects]

Scaling Up

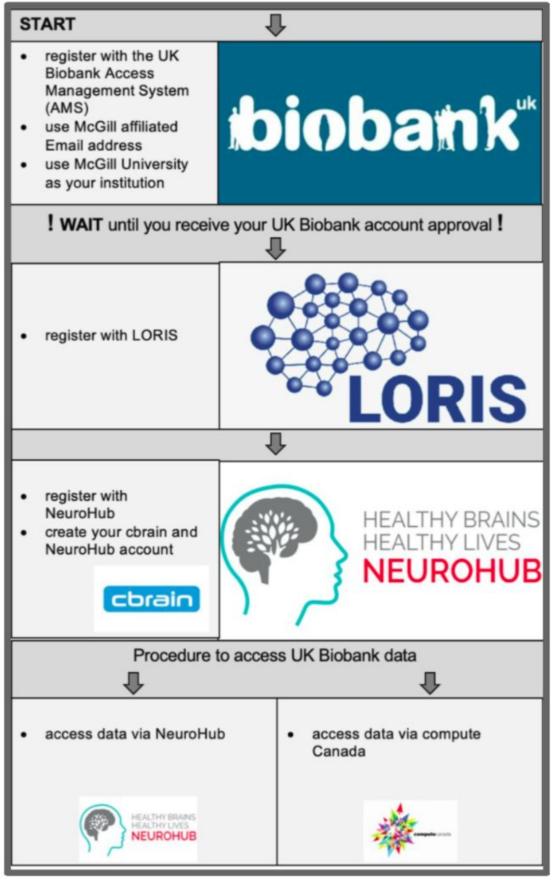
Blue Brain Project



UKBB Data Access

Multi-step process!

1. Register with UKBB AMS
2. Register with LORIS
   a. Project and data management platform
3. Get a NeuroHub account
   a. Data and compute management platform
4. Get a Compute Canada account
   a. High performance compute cluster



START

- register with the UK Biobank Access Management System (AMS)
- use McGill affiliated Email address
- use McGill University as your institution

! WAIT until you receive your UK Biobank account approval !

- register with LORIS

- register with NeuroHub
- create your cbrain and NeuroHub account

Procedure to access UK Biobank data

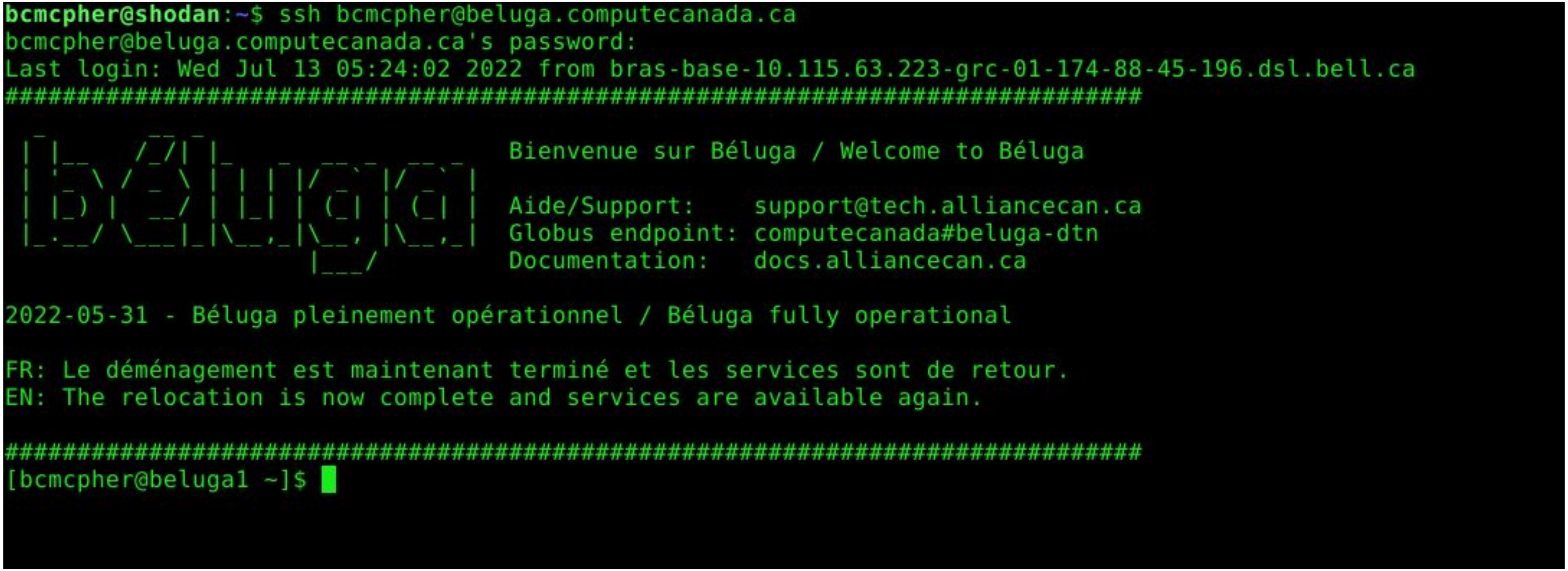
- access data via NeuroHub
- access data via compute Canada

Thanks Nikhil!

Beginning to use a Cluster Computer

Accessing the Cluster

- This is done exclusively through a remote connection.
    - `ssh`, a remote desktop client, etc.
- You will connect to the **Login Node** of the cluster.
    - The login in node manages the coordination of user processes on the cluster
    - It also reports the overall status of the system.
- You will submit your jobs and query their status from the Login Node.
    - DO NOT run analyses or do other work on the Login Node.



```
bcmcpher@shodan:~$ ssh bcmcpher@beluga.computecanada.ca
bcmcpher@beluga.computecanada.ca's password:
Last login: Wed Jul 13 05:24:02 2022 from bras-base-10.115.63.223-grc-01-174-88-45-196.dsl.bell.ca
###############################################################################

  _      __ _
 | |__  /_/| |_   _  __ _  __ _      Bienvenue sur Béluga / Welcome to Béluga
 | '_ \ / _ \ | | | |/ _` |/ _` |
 | |_) |  __/ | |_| | (_| | (_| |    Aide/Support:    support@tech.alliancecan.ca
 |_.__/ \___|_|\__,_|\__, |\__,_|    Globus endpoint: computecanada#beluga-dtn
                     |___/           Documentation:   docs.alliancecan.ca

2022-05-31 - Béluga pleinement opérationnel / Béluga fully operational

FR: Le déménagement est maintenant terminé et les services sont de retour.
EN: The relocation is now complete and services are available again.

###############################################################################
[bcmcpher@beluga1 ~]$ 
```

Getting your data on the cluster

- I assume your data is on a computer not accessible by the cluster.
- You will use a file transfer tool to move your data from where it is to the data server.
  - `scp`, sftp, rsync, FileZilla, mobaXterm, etc.
  - This data server will be accessible to the compute nodes
- This may be done through the Login Node, but there may be a specific Data Transfer Node to move data.
  - Check with the system administrators.

Getting your data on the cluster - Things to Remember

- Be aware of the policies of this server!
  - Files are often automatically removed after a (potentially short) window of inactivity.
  - Your analysis code, jobs, and data are probably not backed up on the HPC.
- It is your responsibility to make sure your data and code are securely backed up somewhere else.
- Once your analysis is complete you need to have a plan for storing your data long term.
  - OSF
  - OpenNeuro

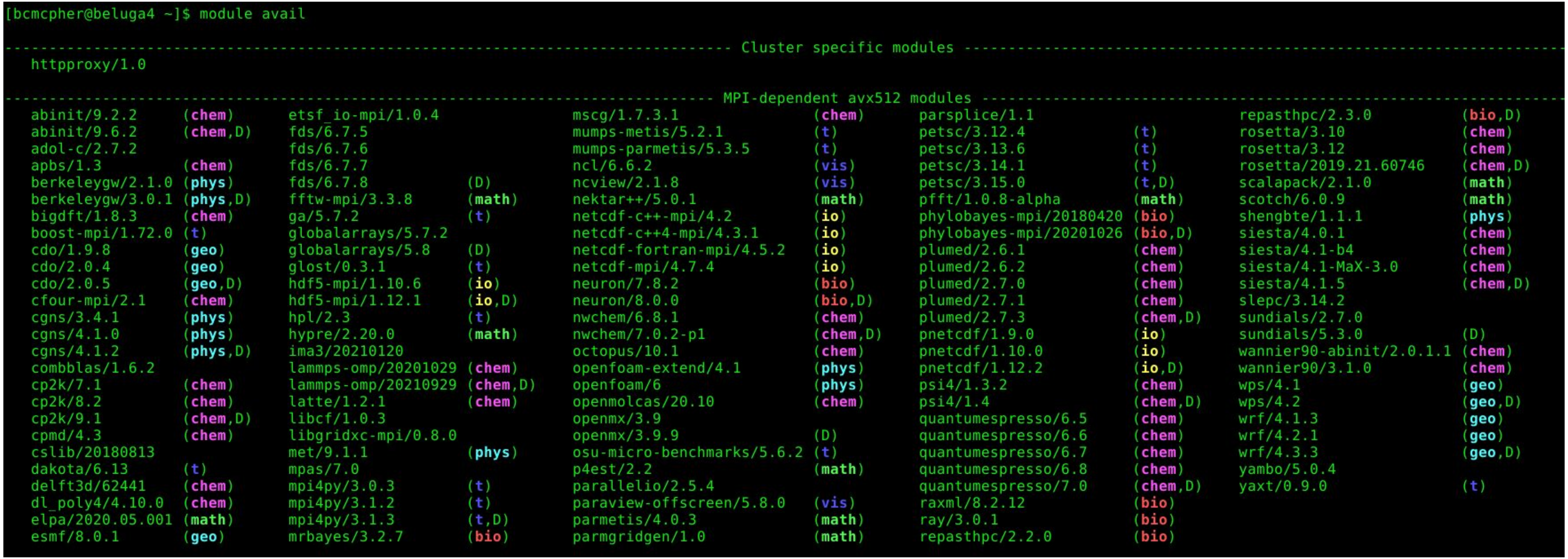
Getting your software on the cluster

- The cluster administrators will likely need to install it for you.
  - If you are confident you know how the tools works, you may be able to set it up yourself.
- Many tools will be readily available on the cluster through **modules**.
  - These are an efficient way to add and remove tools installed on the cluster to your session.
  - They are like an interactive `venv` for any tool.
  - They also allow for easier management of different versions of the same tool.
  - Hopefully, most of the tools you need will already be installed.
- Running your code in **containers** guarantees you have the tools you need to run your analysis.
  - You do have to a have (or be able to build) a working container of the tool.

Getting your software on the cluster - Modules

- How most software is made available to users on a HPC.
- A convenient way to set up basic development environments and change between versions.

```
> module avail
> module load python
> module load python/3.10.2
> module unload python
> module swap python/3.9.6
```



```
[bcmcpher@beluga4 ~]$ module avail

---------------------------------------------------------------------------------- Cluster specific modules ---------------------------------------------------------------------
   httpproxy/1.0

-------------------------------------------------------------------------------- MPI-dependent avx512 modules -------------------------------------------------------------------
   abinit/9.2.2     (chem)      etsf_io-mpi/1.0.4               mscg/1.7.3.1               (chem)      parsplice/1.1                           repasthpc/2.3.0          (bio,D)
   abinit/9.6.2     (chem,D)    fds/6.7.5                       mumps-metis/5.2.1          (t)         petsc/3.12.4            (t)             rosetta/3.10             (chem)
   adol-c/2.7.2                 fds/6.7.6                       mumps-parmetis/5.3.5       (t)         petsc/3.13.6            (t)             rosetta/3.12             (chem)
   apbs/1.3         (chem)      fds/6.7.7                       ncl/6.6.2                  (vis)       petsc/3.14.1            (t)             rosetta/2019.21.60746    (chem,D)
   berkeleygw/2.1.0 (phys)      fds/6.7.8           (D)         ncview/2.1.8               (vis)       petsc/3.15.0            (t,D)           scalapack/2.1.0          (math)
   berkeleygw/3.0.1 (phys,D)    fftw-mpi/3.3.8      (math)      nektar++/5.0.1             (math)      pfft/1.0.8-alpha        (math)          scotch/6.0.9             (math)
   bigdft/1.8.3     (chem)      ga/5.7.2            (t)         netcdf-c++-mpi/4.2         (io)        phylobayes-mpi/20180420 (bio)           shengbte/1.1.1           (phys)
   boost-mpi/1.72.0 (t)         globalarrays/5.7.2              netcdf-c++4-mpi/4.3.1      (io)        phylobayes-mpi/20201026 (bio,D)         siesta/4.0.1             (chem)
   cdo/1.9.8        (geo)       globalarrays/5.8    (D)         netcdf-fortran-mpi/4.5.2   (io)        plumed/2.6.1            (chem)          siesta/4.1-b4            (chem)
   cdo/2.0.4        (geo)       glost/0.3.1         (t)         netcdf-mpi/4.7.4           (io)        plumed/2.6.2            (chem)          siesta/4.1-MaX-3.0       (chem)
   cdo/2.0.5        (geo,D)     hdf5-mpi/1.10.6     (io)        neuron/7.8.2               (bio)       plumed/2.7.0            (chem)          siesta/4.1.5             (chem,D)
   cfour-mpi/2.1    (chem)      hdf5-mpi/1.12.1     (io,D)      neuron/8.0.0               (bio,D)     plumed/2.7.1            (chem)          slepc/3.14.2
   cgns/3.4.1       (phys)      hpl/2.3             (t)         nwchem/6.8.1               (chem)      plumed/2.7.3            (chem,D)        sundials/2.7.0
   cgns/4.1.0       (phys)      hypre/2.20.0        (math)      nwchem/7.0.2-p1            (chem,D)    pnetcdf/1.9.0           (io)            sundials/5.3.0           (D)
   cgns/4.1.2       (phys,D)    ima3/20210120                   octopus/10.1               (chem)      pnetcdf/1.10.0          (io)            wannier90-abinit/2.0.1.1 (chem)
   combblas/1.6.2               lammps-omp/20201029 (chem)      openfoam-extend/4.1        (phys)      pnetcdf/1.12.2          (io,D)          wannier90/3.1.0          (chem)
   cp2k/7.1         (chem)      lammps-omp/20210929 (chem,D)    openfoam/6                 (phys)      psi4/1.3.2              (chem)          wps/4.1                  (geo)
   cp2k/8.2         (chem)      latte/1.2.1         (chem)      openmolcas/20.10           (chem)      psi4/1.4                (chem,D)        wps/4.2                  (geo,D)
   cp2k/9.1         (chem,D)    libcf/1.0.3                     openmx/3.9                             quantumespresso/6.5     (chem)          wrf/4.1.3                (geo)
   cpmd/4.3         (chem)      libgridxc-mpi/0.8.0             openmx/3.9.9               (D)         quantumespresso/6.6     (chem)          wrf/4.2.1                (geo)
   cslib/20180813               met/9.1.1           (phys)      osu-micro-benchmarks/5.6.2 (t)         quantumespresso/6.7     (chem)          wrf/4.3.3                (geo,D)
   dakota/6.13      (t)         mpas/7.0                        p4est/2.2                  (math)      quantumespresso/6.8     (chem)          yambo/5.0.4
   delft3d/62441    (chem)      mpi4py/3.0.3        (t)         parallelio/2.5.4                       quantumespresso/7.0     (chem,D)        yaxt/0.9.0               (t)
   dl_poly4/4.10.0  (chem)      mpi4py/3.1.2        (t)         paraview-offscreen/5.8.0   (vis)       raxml/8.2.12            (bio)
   elpa/2020.05.001 (math)      mpi4py/3.1.3        (t,D)       parmetis/4.0.3             (math)      ray/3.0.1               (bio)
   esmf/8.0.1       (geo)       mrbayes/3.2.7       (bio)       parmgridgen/1.0            (math)      repasthpc/2.2.0         (bio)
```

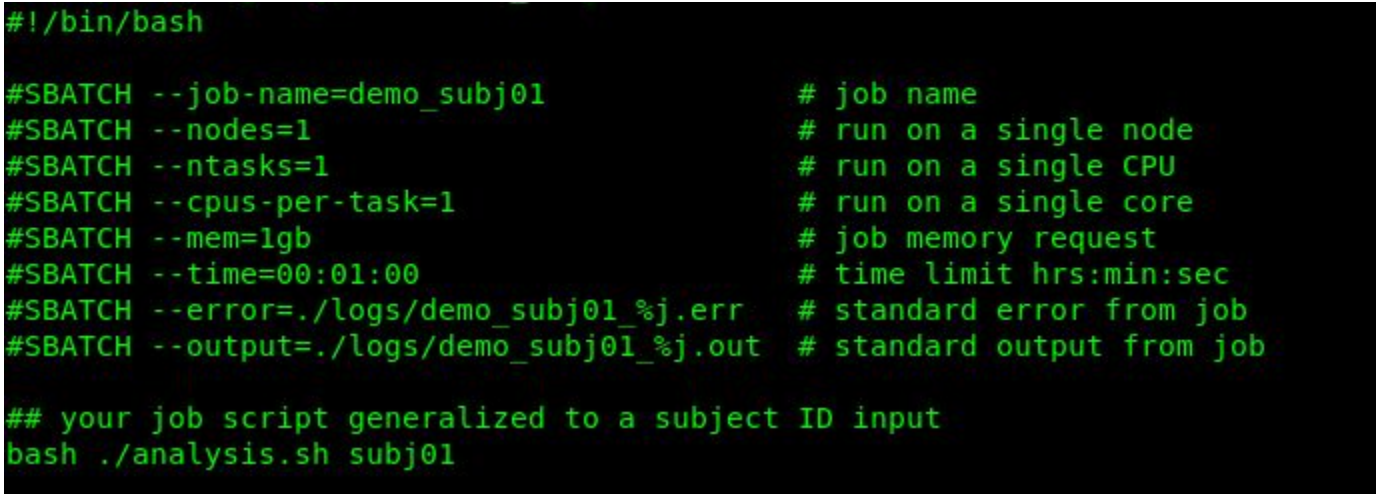
Running your analysis - Working with a Scheduler

- With your data and `env` ready you can run your analysis.
- You will do this by submitting your work as a **Job**.
- **Jobs** are submitted to the Scheduler to be run.
  - This deploys the analysis onto the compute nodes.
- Your entire analysis will need to be scripted.
  - You cannot interact with or manually input information to a job when it’s running.
- Once your analysis script is complete, you create your job script for the analysis.
  - This script describes the resources needed to successfully run your analysis on the cluster.

Running your analysis - Creating Job Scripts

- Your job scripts define the system resources required by your analysis.
- The job script will be submitted to the scheduler from the login node to be added to the queue.
  - The scheduler uses your resource requests and your priority to determine what runs when.
- Your outputs, along with logs, will be created for each job you submit.
- The example today will assume you are **Scaling Out** your analysis.
  - You want to run a common process across many subjects.
  - i.e. fMRIPrep, FreeSurfer, etc.

demo_subj01.slurm



```
#!/bin/bash

#SBATCH --job-name=demo_subj01              # job name
#SBATCH --nodes=1                           # run on a single node
#SBATCH --ntasks=1                          # run on a single CPU
#SBATCH --cpus-per-task=1                   # run on a single core
#SBATCH --mem=1gb                           # job memory request
#SBATCH --time=00:01:00                     # time limit hrs:min:sec
#SBATCH --error=./logs/demo_subj01_%j.err   # standard error from job
#SBATCH --output=./logs/demo_subj01_%j.out  # standard output from job

## your job script generalized to a subject ID input
bash ./analysis.sh subj01
```

Commands for working with Jobs

- `sbatch`
  - Submit a job to the scheduler.
  - `> sbatch job.slurm`
- `squeue`
  - Monitor the status of jobs on the cluster.
  - `> squeue -u USERNAME`
- `scancel`
  - Cancels a job that hasn’t completed.
  - `> scanel JOBID`
- `sacct`
  - Provides a summary of the jobs your have recently submitted.
  - `> sacct`
- `srun`
  - Start an interactive job to test that your job will work correctly.
  - `> srun --nodes=1 --ntasks-per-node=1 --time=01:00:00 --pty bash -i`

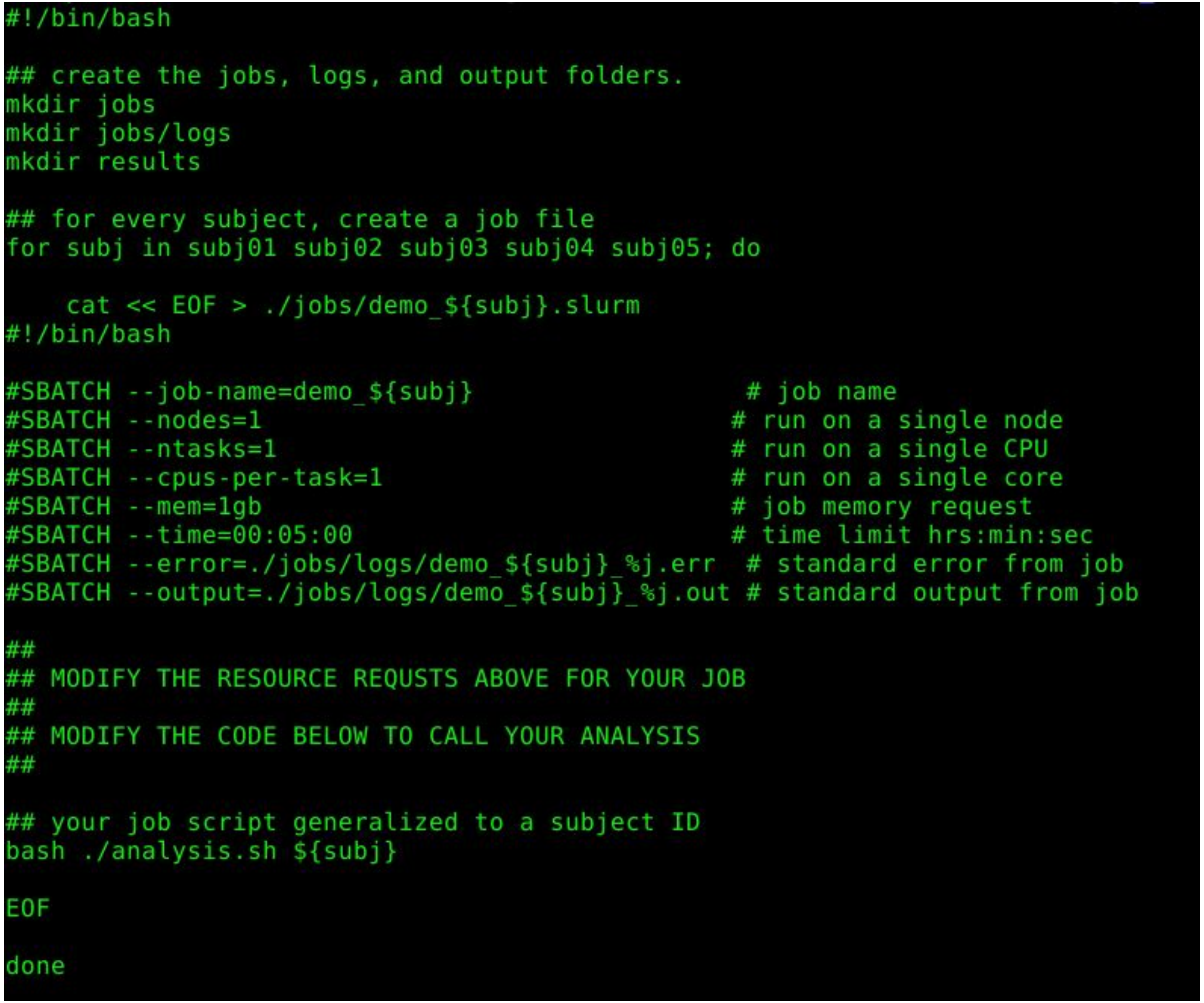
There are a lot of resources available for learning about HPCs online and in your lab. Find a process that works for you and ASK how others have solved similar problems!

Running your analyses - be efficient

- For every job you run, you will have:
    - The input data - *the same files for each subject*
    - The analysis script - *the same analysis for each subject*
    - The job script and logs - *a separate job for every subject with logs for each attempt*
    - The results - *whatever output your analysis produces that you intend to keep*
- Just like your analyses, you should script as much of your cluster work as possible.
    - You should never be individually making or modifying job files for subjects.
    - Having scripts for moving and backing up data is a good idea, too.

The Example

- The example today reports back a short log about the system status of the node before waiting.
    - This will let you see the different nodes your jobs ran on.
- The analysis input wants a subject ID as an input
    - This is is likely how you would set up a symmetric analysis across your subjects.
- The goal is to provide a basic working template for your job scripts that you can modify.



```
#!/bin/bash

## create the jobs, logs, and output folders.
mkdir jobs
mkdir jobs/logs
mkdir results

## for every subject, create a job file
for subj in subj01 subj02 subj03 subj04 subj05; do

    cat << EOF > ./jobs/demo_${subj}.slurm
#!/bin/bash

#SBATCH --job-name=demo_${subj}                  # job name
#SBATCH --nodes=1                               # run on a single node
#SBATCH --ntasks=1                              # run on a single CPU
#SBATCH --cpus-per-task=1                       # run on a single core
#SBATCH --mem=1gb                               # job memory request
#SBATCH --time=00:05:00                         # time limit hrs:min:sec
#SBATCH --error=./jobs/logs/demo_${subj}_%j.err  # standard error from job
#SBATCH --output=./jobs/logs/demo_${subj}_%j.out # standard output from job

##
## MODIFY THE RESOURCE REQUSTS ABOVE FOR YOUR JOB
##
## MODIFY THE CODE BELOW TO CALL YOUR ANALYSIS
##

## your job script generalized to a subject ID
bash ./analysis.sh ${subj}

EOF

done
```

1. Connect to the HPC.
2. Move the example scripts to the HPC.
3. Make and submit your jobs.
4. Monitor the job status.
5. Cancel a job.
6. Run an interactive job.
7. Modify your resource requests / Explore documentation.
8. Run the script in a container.

Commands

```
> ssh USERNAME@beluga.computecanada.ca

> scp /path/to/Lecture11/demo/*.sh USERNAME@beluga.computecanada.ca:~/demo

> cd ~/demo; ./mk_jobs

> find jobs -type f -name "*.slurm" | xargs -n 1 sbatch

> squeue -u USERNAME; watch squeue -u USERNAME

> sacct

> scancel JOBID
```

Thanks