Overview of CU Research Computing Resources

Nick Featherstone

CU Applied Math

Outline

- Overview
- Storage systems
- Compute resources
- Logging in
- Software stack / Compiling programs
- Submitting batch jobs
- Globus file transfer

What is Research Computing?

- University-wide resource freely available to students/faculty

- High Performance Computing (HPC) Resources for CU
  - Large pool of networked compute nodes
  - (node = motherboard with multiple processors/cores onboard)
  - Parallel mass file storage systems
  - HPC software stack:
    - Commonly used applications: e.g., Matlab, Tensorflow, OpenFOAM
    - Popular Compilers (Intel, GNU, PGI)
    - Optimized, compiled libraries (BLAS, MPI, etc.)

- User support
  - Basic support (how do I log in / use Linux)?
  - Advanced support (code optimization, parallelization)
  - Software installs upon request
  - Instructional seminars (https://www.colorado.edu/rc/userservices/training)

Getting Help

- Research Computing Web Page:
  - https://www.colorado.edu/rc
- Read the documentation (first!):
  - https://curc.readthedocs.io/en/latest/
- Email:
  - rc-help@colorado.edu
- Note:
  - Remember this is a small team of about **10 people** responsible for serving the **entire university's** HPC needs**.** Sometimes the response time is long. Sometimes you have to iterate before your problem is truly understood.
  - Be cool!

Compute Systems Overview

- Summit
  - 10,848 Intel Haswell Cores (4.8 GB/core, 24-cores/node)
  - 480 Intel Skylake Cores (7.8 GB/core, 24-core/node)
  - And more: https://www.colorado.edu/rc/articles/rmaccsummit
  - Free-to-use by entire university
  - General, low-priority compute 'allocation' provided to every user
  - High-priority allocations available to PI's (speak to your advisor)

- Blanca
  - Dedicated, grant- or department-funded compute system.
  - Heterogeneous node types (Westmere through Skylake)
  - Speak to your advisor
  - Applied Math has 2, 32-core Intel Skylake nodes
    - Speak to/email Dominique Ingoglia for access: am_itsup@colorado.edu

File Systems Overview

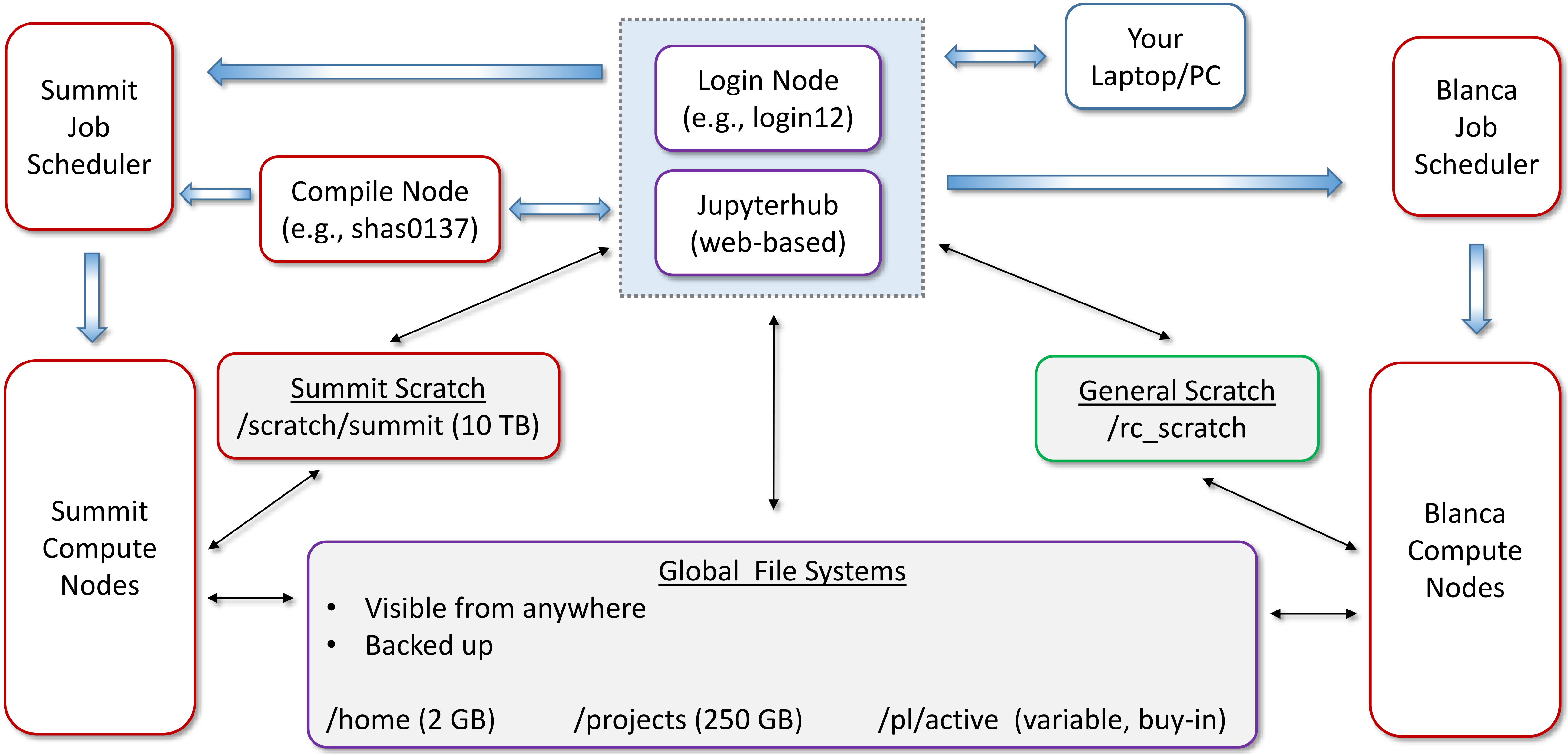
Backed-up Filesystems

- Use these as you see fit, but space is limited
- Non-parallel
- DO NOT have your programs write to these F/S.
- /home/username
  - 2 GB (recommend store source code)
  - Snapshots every 2 hours
- /projects/username
  - 250 GB (permanent input files, etc.)
  - Snapshots every 6 hours
- /pl/active/project_name
  - Ask your advisor (grant or dept. funded)
  - Snapshot frequency varies

Purged Filesystems

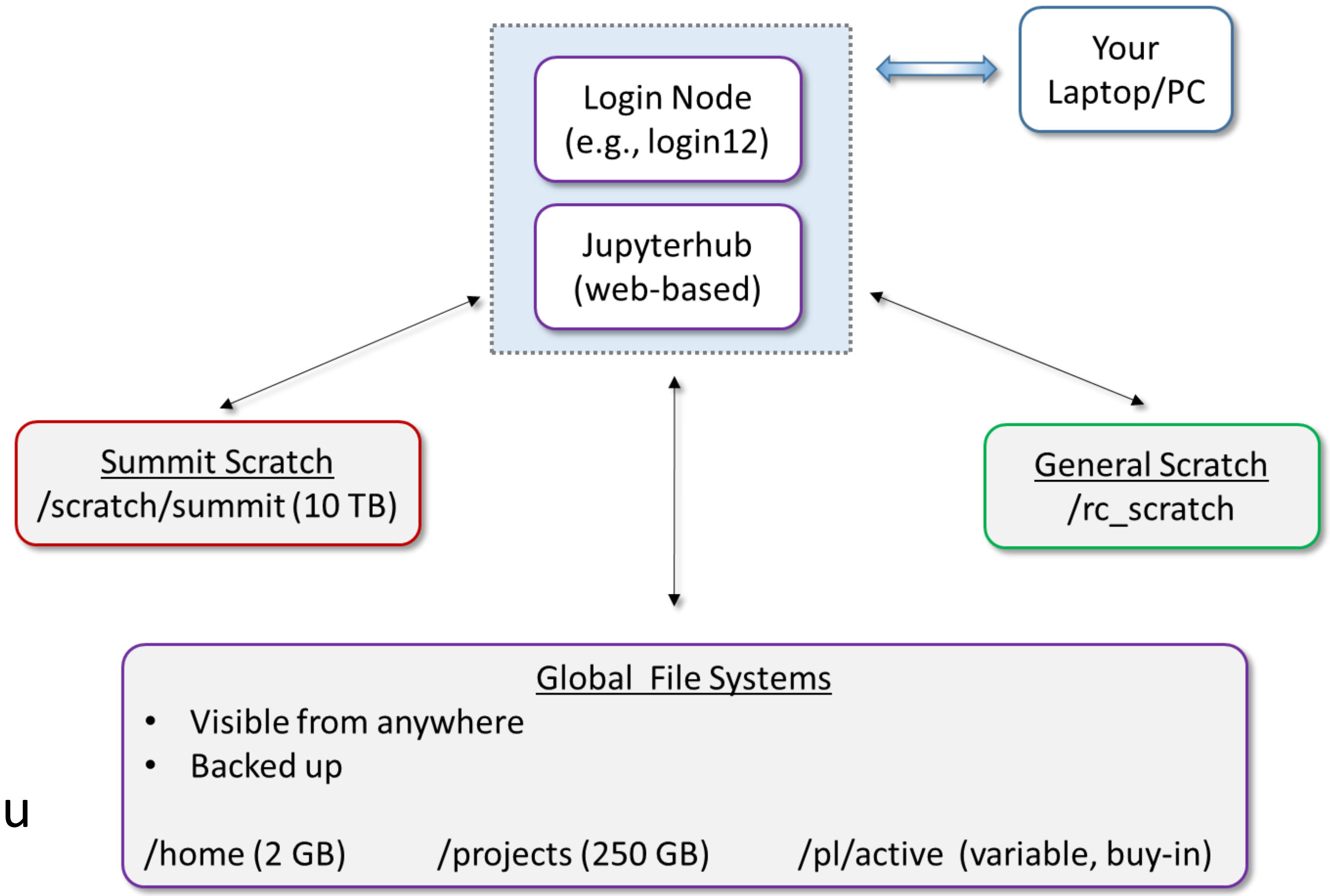
- *Temporary* storage
- Use for program output
- Parallel
- Purged
  - every 90 days
  - per file, based on creation date
- Purged files cannot be retrieved!
- Summit scratch
  - /scratch/summit/username
  - 10 TB
- Blanca scratch
  - /rc_scratch/username
  - Variable storage

RC Systems Conceptual Schematic



Logging in

- Login via ssh with DUO two-factor
  - ssh username@login.rc.Colorado.edu
  - Check your phone for Duo!
- Let's poke around for a bit…
  - cd filesystem/username
  - e.g. cd /scratch/summit/feathern
- Logout (type logout or exit)
- Login in as tutorial user:
  - ssh username@tlogin1.rc.colorado.edu
  - no DUO two-factor



Clone the Seminar Respository

- All of todays materials, including these slides are available online at:
  - https://github.com/feathern/rc_appm_2019

- Let’s work in /scratch/summit:
  - ssh scompile (puts you on a compile node)
  - cd /scratch/summit/username
  - git clone https://github.com/feathern/rc_appm_2019.git
  - cd rc_appm_2019

The Software Stack

- Software controlled through a hierarchical module system
- https://curc.readthedocs.io/en/latest/compute/modules.html
- Must load a software module to use the module
  - PATH, LD_LIBRARY_PATH etc. modified (for those in the know)

- Many versions of same software may exist, but compiled with
  - Different compiler flavors and versions
    (e.g., Intel 17.02 or Intel 16.1 or GNU 7.1)
  - Different MPI flavors and versions
    (e.g., Intel MPI or OpenMPI)

- For this reason, modules must be loaded in order:
  1. Compiler
  2. MPI
  3. Desired software

- Let's have a look…

Loading Software Modules

- Primary commands (try these out)
  - module avail
    - See which modules are now available
    - Based on compiler and MPI loaded
    - As you load a compiler and MPI, new modules appear as available
  - module load [or unload] {module name} (e.g., module load intel)
    - Load or unload a desired software module
  - module list
    - See all modules you currently have loaded
  - module purge
    - Unload all currently loaded modules
  - module spider {module name}
    - Search for software (shows which modules need to be loaded)

Compiling Code

- Programs can be compiled within a job script (more on that soon)

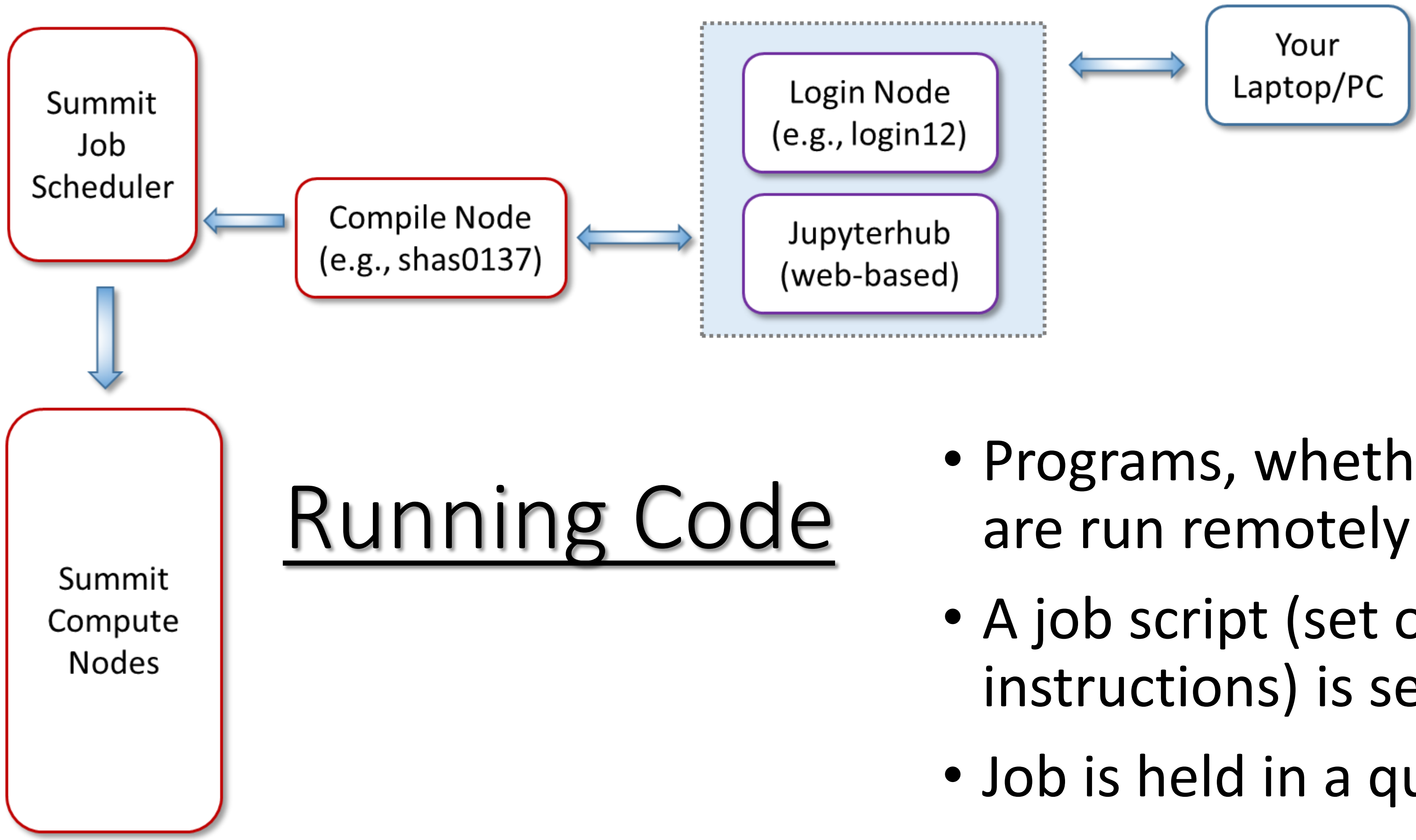
- Programs may also be compiled on the scompile nodes
  - You are already on an scompile node
  - Reached from login node via “ssh scompile” (shas prompt shows)

- When compiling a program written in C or Fortran:
  - You must first load the desired compiler/MPI modules
  - These same modules must also be loaded in your *job script*

- Let’s try a quick example

Compiling Code

- Let’s compile a sample program.
- cd sample_code
- Load a compiler and MPI:
  - module purge
  - module load intel
  - module load impi
- Compile the program:
  - mpicc main.c –o hello.out
- To run the program, we need to interface with the job scheduler…



Running Code

- Programs, whether parallel or serial, are run remotely
- A job script (set of step-by-step instructions) is sent to the scheduler
- Job is held in a queue
- When enough cores available, it runs…
- Priority basis, sometimes you wait…
- Just like all national level HPC systems…

Job Scripts

- Set of step-by-step instructions necessary to run a job
  - Just like you would type at the command line yourself
- cd ../sample_scripts
- more tutorial_2core.sh
- Equivalent Summit and Blanca scripts are provided

Job Scripts: Important Notes

- Total number of cores is controlled by *ntasks* flag
- Be sure to specify enough nodes via the *nodes* flag (24 core/node).
- Never request more cores/nodes than you need (node can be shared).
- ALWAYS write to /scratch/summit or /rc_scratch (parallel filesystems)
- NEVER write to /home or /projects from a remote job
  - These systems are not parallel; can negatively impact other users; account locked
- Instructions are executed from within directory where script is submitted
  - So be sure to run appropriate “cd” commands
- Don’t forget to load necessary modules!

Running a Remote Job

- Submit the two sample jobs:
  - sbatch tutorial_2core.sh
  - sbatch tutorial_8core.sh
- Job monitoring:
  - squeue –u username (check status of all of your jobs)
- If you goof:
  - scancel jobID#   (cancel a job)
- Output normally written to the screen will appear in output file.
- Data output location depends on application

JupyterHub

- Web-based interface to RC Resources
  - Primarily intended for Jupyter notebooks
  - Terminal sessions also available
  - Jupyter sessions spawned on either Summit or Blanca
  - Useful for small quick file transfers (no need to bring ssh into equation)
- Documentation: https://curc.readthedocs.io/en/latest/gateways/jupyterhub.html
- Let's check it out:
  - Go here: https://jupyter.rc.colorado.edu
  - Login with YOUR credentials; Click "Start my Server;" Wait.

Globus

- We don’t quite have time to discuss this
- If you have questions, feel free to ask me or email rc-help@colorado.edu
- Very useful tool for transferring data between RC and:
  - Your laptop/desktop
  - Another remote HPC system (initiate on your laptop, then walk away)
- https://curc.readthedocs.io/en/latest/compute/data-transfer.html