Alpine Noob! - A Brief Introduction to HPC and Alpine

Written By:
Tonya Brunetti

IMMU 6110/Alpine HPC Active Users

Reminder! Alpine cannot be used on sensitive data!

- No HIPAA or Protected Health Information (PHI)
- No General Data Protection Regulation (GDPR)
- No CMMC
- No FERPA
- No data that requires special security and compliance

PHI
PROTECTED HEALTH INFORMATION

DARPA

biobank uk
Enabling scientific discoveries that improve human health

CMMC
CYBERSECURITY MATURITY MODEL CERTIFICATION

DEPARTMENT OF DEFENSE
UNITED STATES OF AMERICA

DEPARTMENT OF EDUCATION
UNITED STATES OF AMERICA

FERPA
Family Educational Rights & Privacy Act

All data on Alpine must be 100% de-identified and make sure it is compliant with your IRB and/or grant funding agency

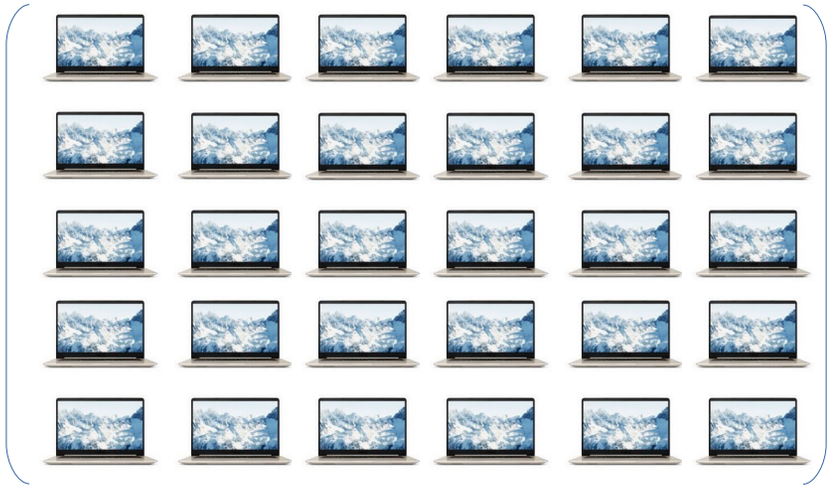
Benefits of HPC – Resource availability

Standard Laptop
~8GB RAM,
500GB hard drive,
Dual-core processor

Standard CPU Node on Alpine (amilan nodes)
64 cores, ~3.74GB/core
=240GB RAM

Alpine High Performance Compute Cluster
382 compute nodes for a total of 22,180 cores*

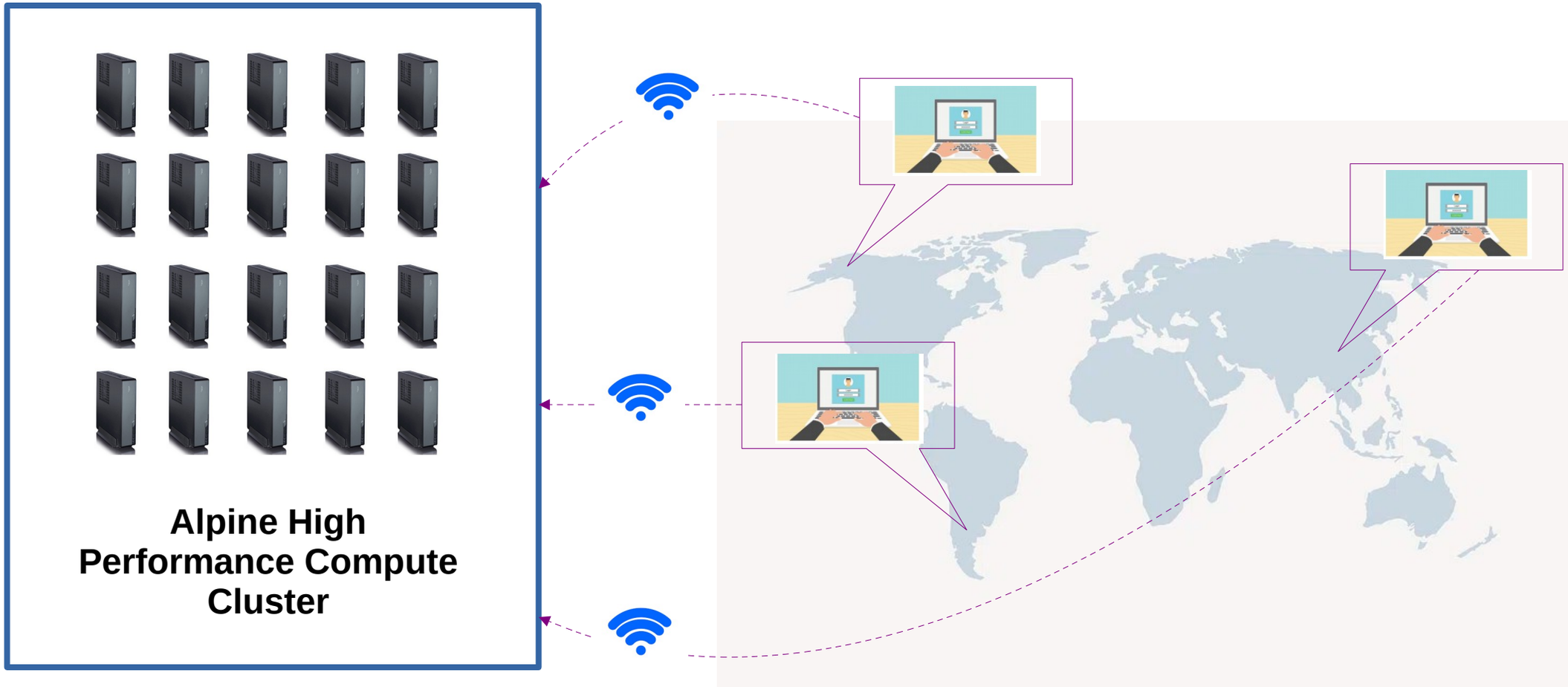
30 standard laptops = 1 CPU node on Alpine!



Alpine also has special nodes such as GPUs and high memory nodes which contain 1TB of RAM

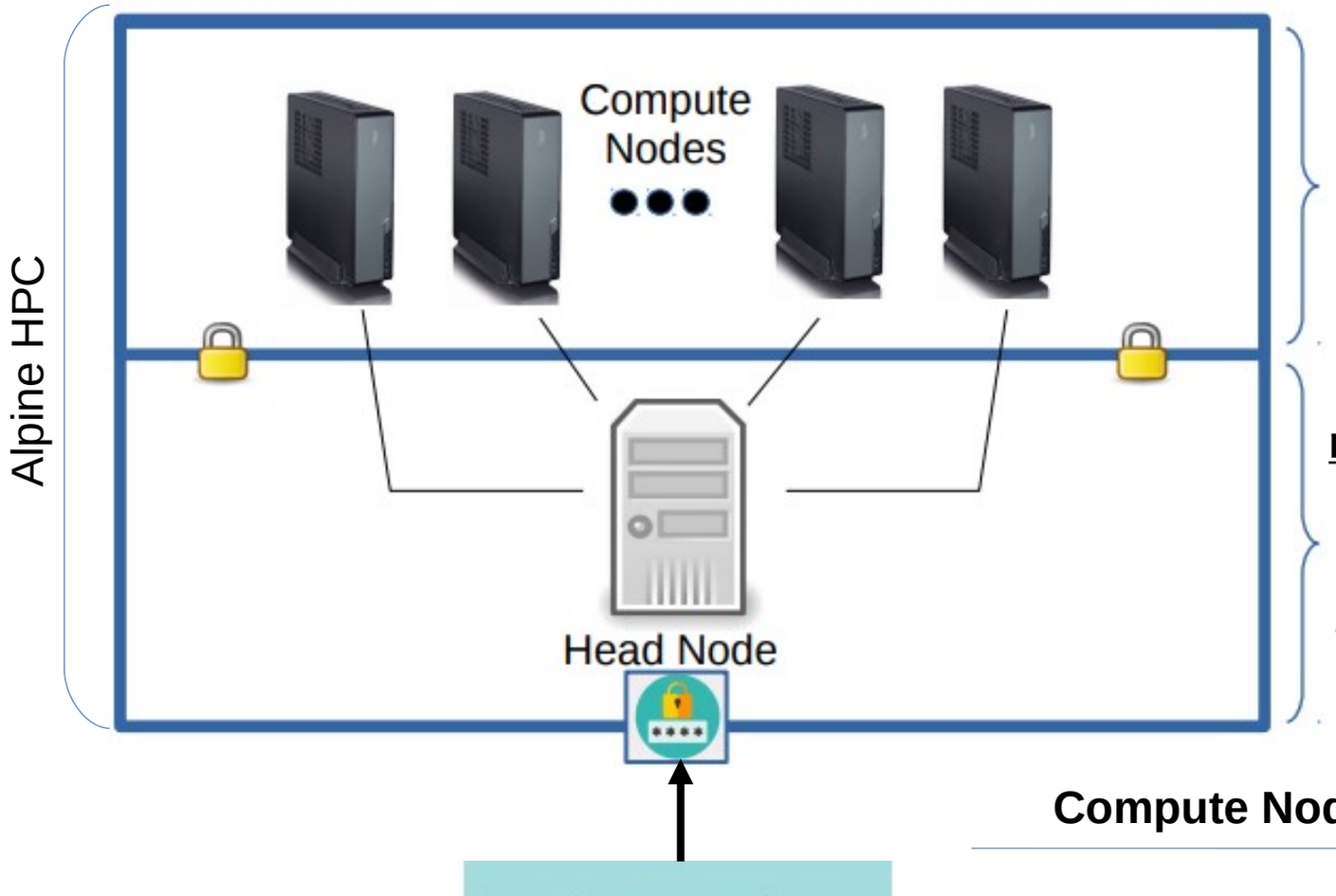
* includes GPU & high mem

Benefits of HPC – Remote computation



Once you submit your job to Alpine or an HPC, you can turn off your computer. At this point, all the data, commands, scripts, etc… are copied over to the Boulder side where the power to the server is always on, therefore **turning off your computer after you submit your job is not at all tied to your local computer power**.

Anatomy of HPC



The compute nodes are where all the power of the HPC is harnessed. **This is where all the computationally heavy work occurs**.

Sometimes referred to as the **login node**. Its main purpose is to take your job requests and find compute resources for you to use.

This node is the master or brain of the HPC.

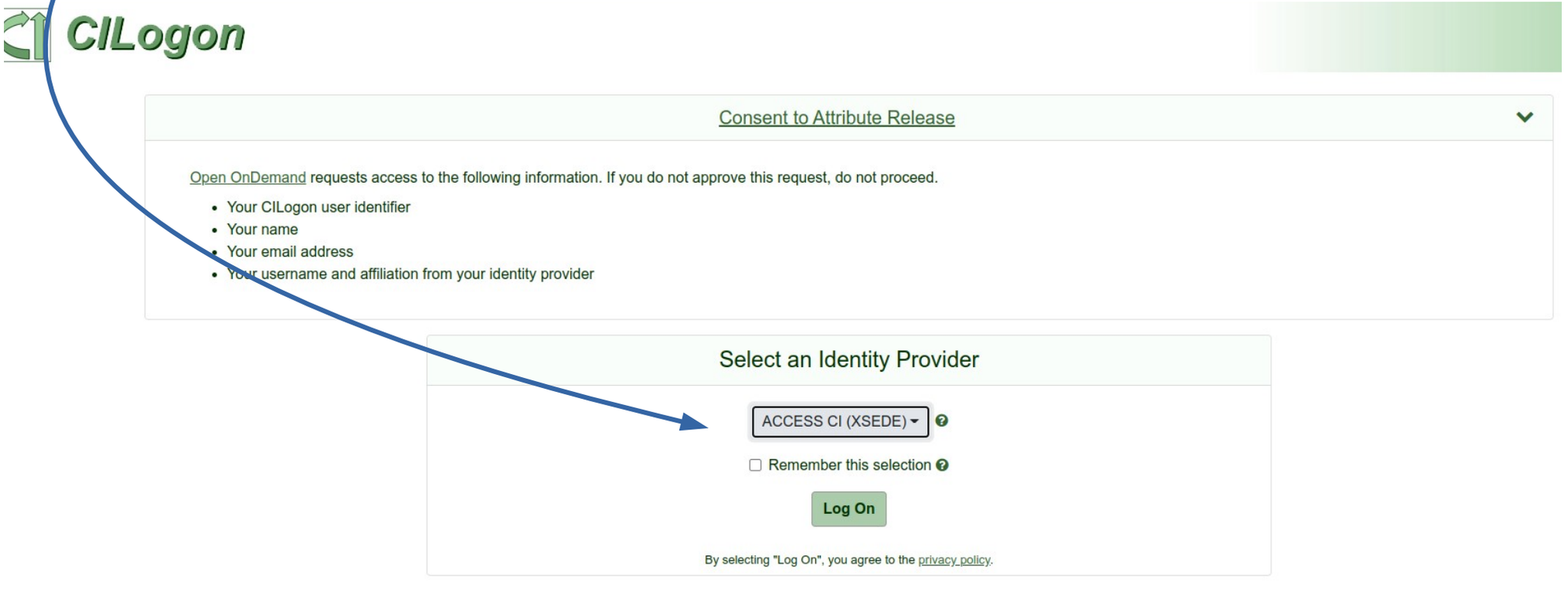
You, on your local computer →

Compute Node	Head/Login Node
Any really long (> few minutes) and/or large (>1GB of memory) programs or calls needs to br run on compute nodes. The compute nodes wait for the head node to assign them a job or task.	Anytime a user makes calls through the command line on Alpine (except in acompile/sinteractive sessions), it is being performed on the head node. The head node is very small and limited in ability. Its main function is to work with the scheduler and delegate tasks and jobs to the compute nodes that users want to utilize

First let's login to Alpine!

1 Go to https://ondemand-rmacc.rc.colorado.edu

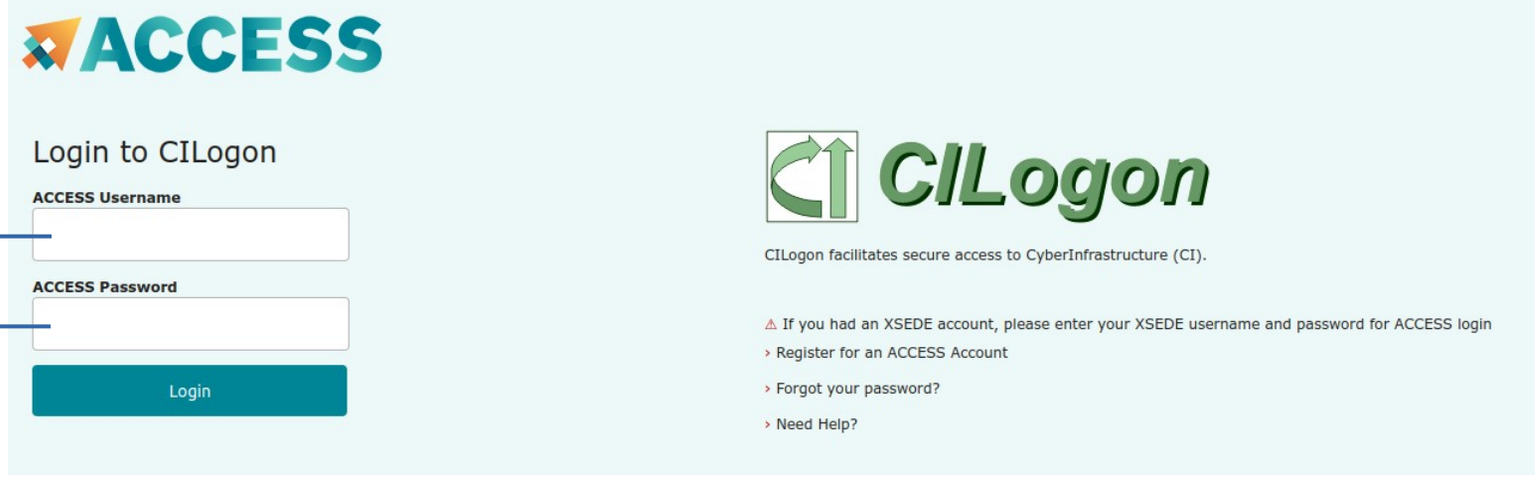
2 It should redirect you to CILogon which is how you authenticate your Alpine session. Make sure you select "ACCESS CI (XSEDE)" as your identity provide and then press "Log On"



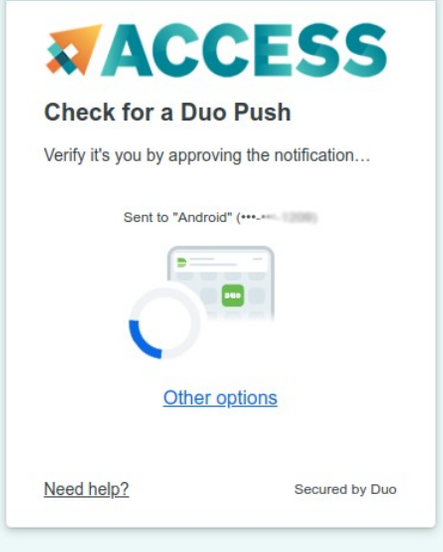
First let's login to Alpine!

3 Next it will take you to this page where you will put in your ACCESS ID and ACCESS password and press Login. This is NOT your CU Anschutz ID!!

ACCESS ID and ACCESS password **not** CU Anschutz credentials!!

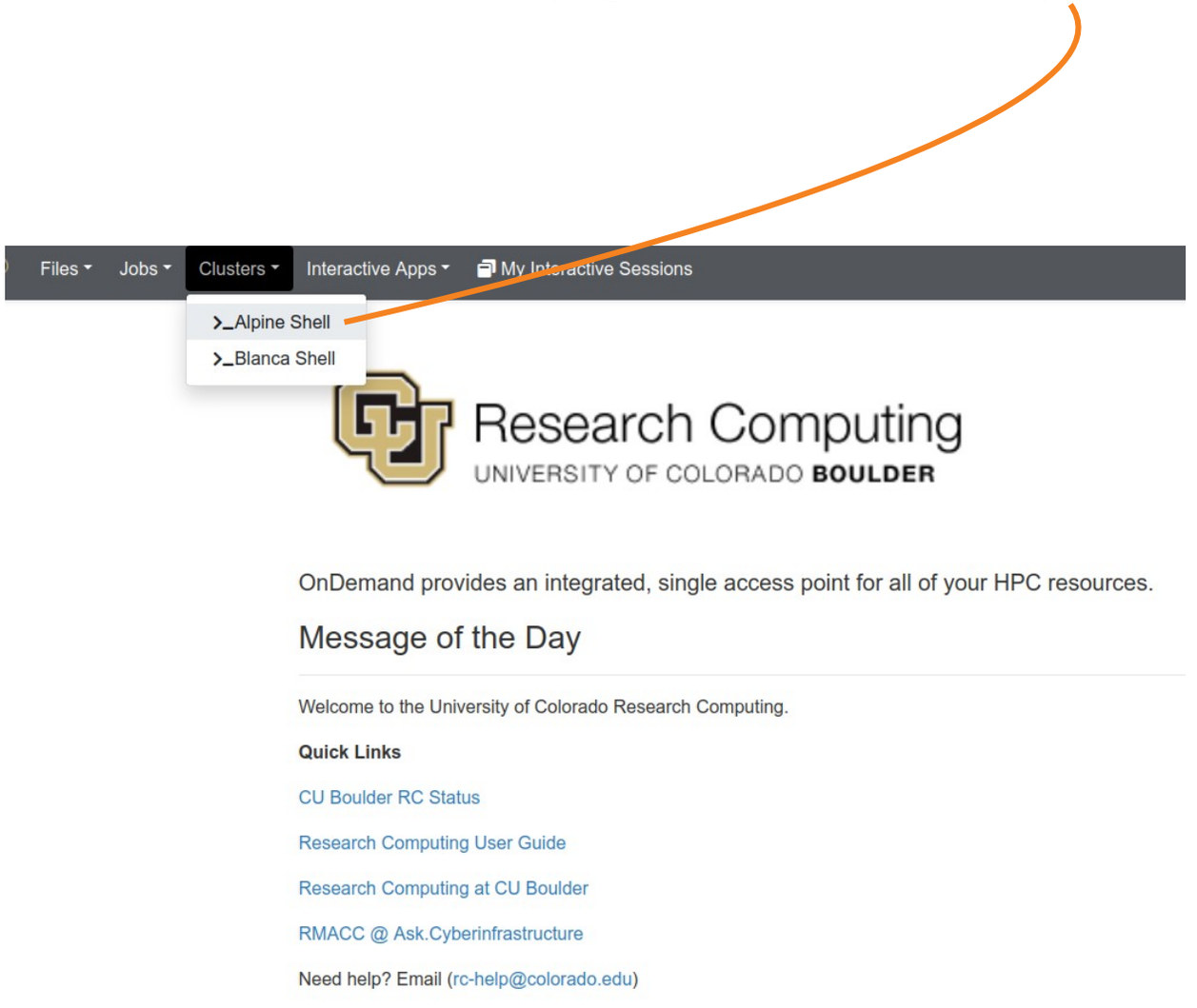


4 This will prompt a DUO MFA push to your phone or whichever way you have DUO set up on your phone to authenticate. Accept the push sent to your device.



First let's login to Alpine!

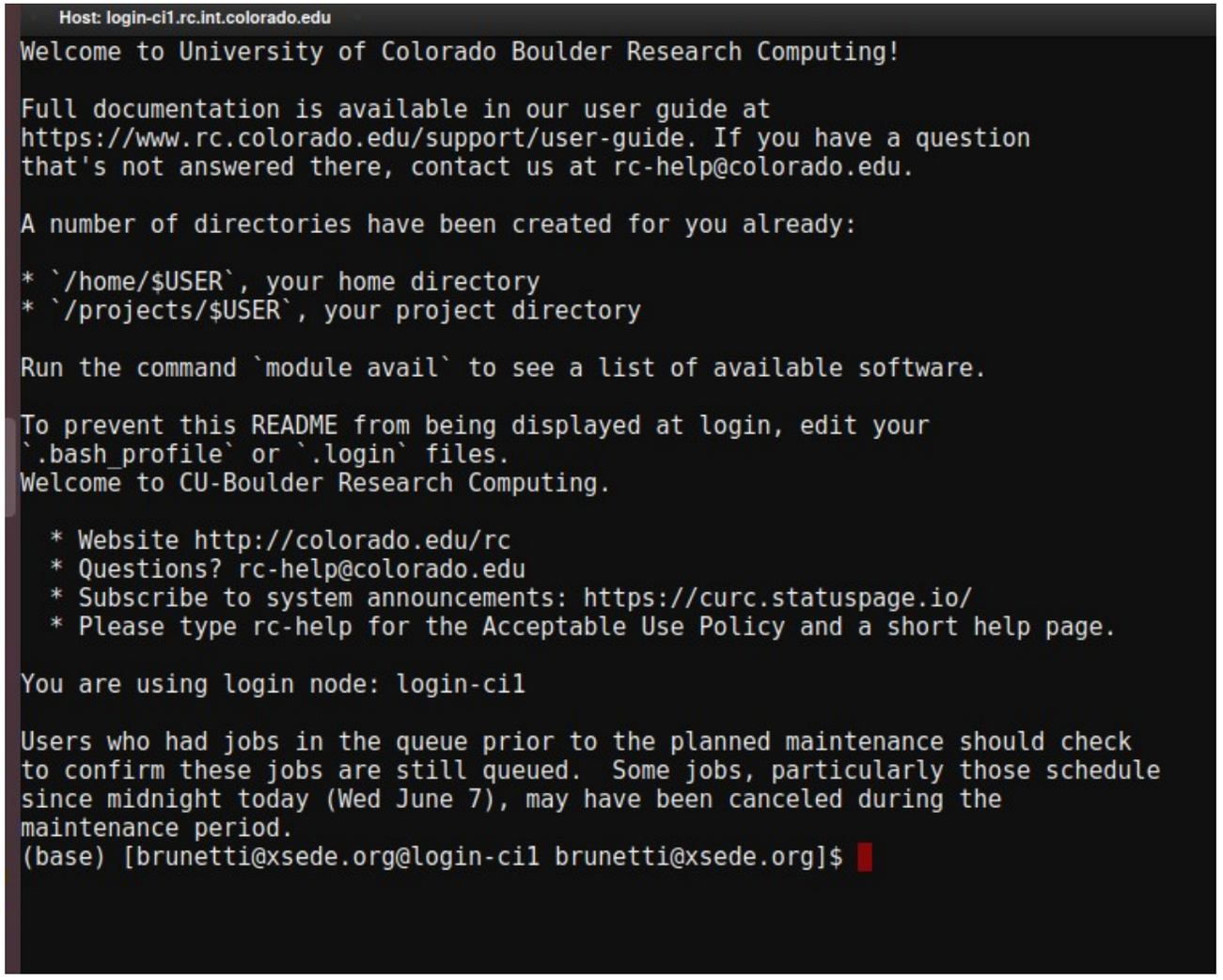
5 This will take to the official CURC page. Let's select the Alpine terminal



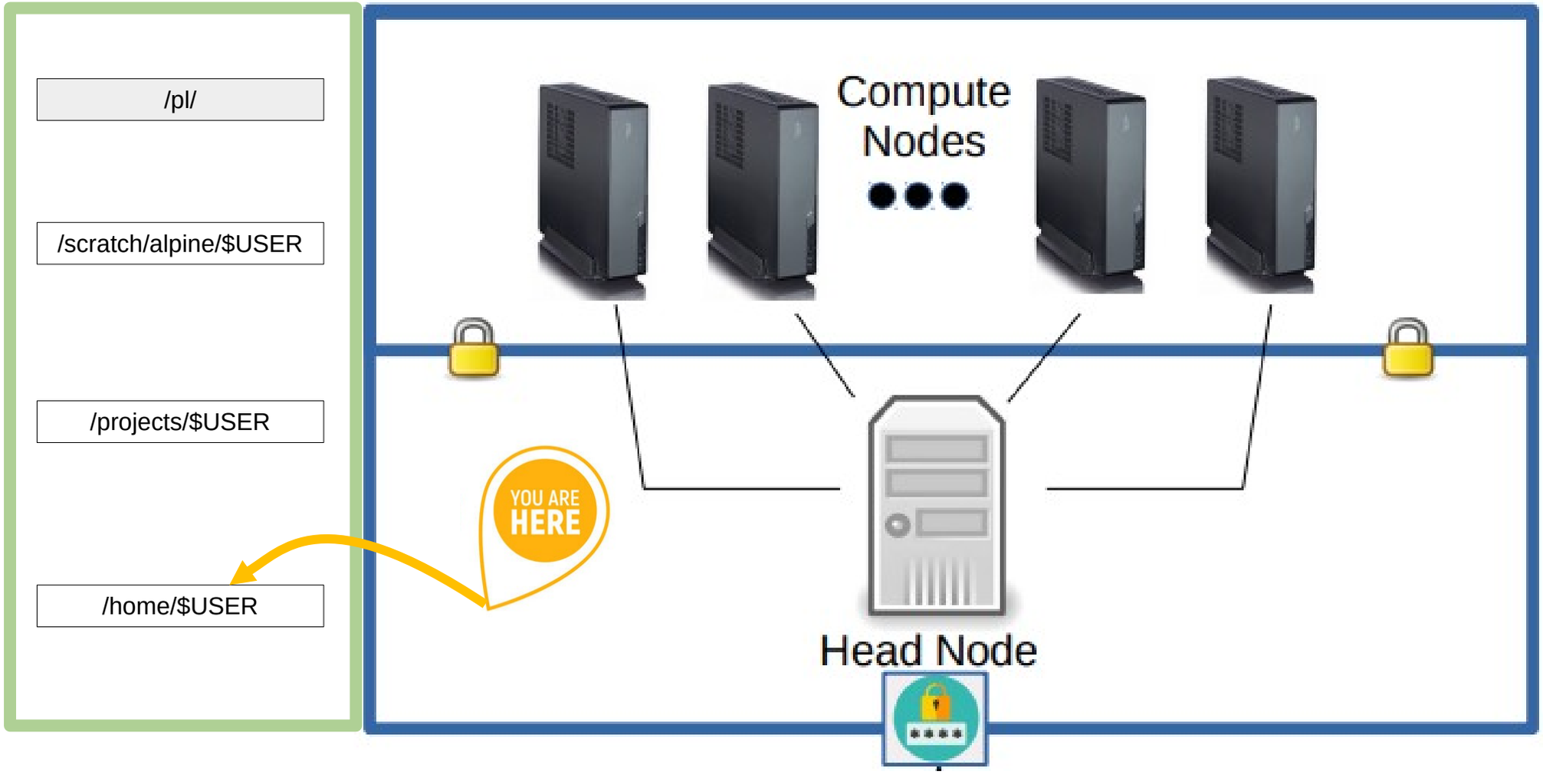
First let’s login to Alpine!

6

This will log you into the head node of Alpine. It will always default you to your home directory.



All 4 storage directories you have for storage are accessible by both the head/login node and the compute nodes



Every time you login to Alpine, **<u>you will always be located in your /home/$USER directory on the head/login node</u>**

Alpine HPC RoadMap

/home/$USER

/projects/$USER

/scratch/alpine/$USER

/pl/

Alpine HPC RoadMap

/home/$USER

/projects/$USER

/scratch/alpine/$USER

/pl/

Every time you login to Alpine, **you will always be located in your /home/$USER directory**

Alpine HPC RoadMap

/home/$USER

/projects/$USER

/scratch/alpine/$USER

/pl/

Every time you login to Alpine, **you will always be located in your /home/$USER directory**

It doesn't matter if you were in your scratch space when you logged out or where you were in any prior session; Alpine does not remember where you were last time, so it always puts you in your /home/$USER directory space

Alpine HPC RoadMap

/home/$USER

/home/$USER is very, very small!
2GB total

For most users, you never want to do anything to or in home; this is where most your configuration files are located, such as your .bashrc, .bash_profile, .condarc, etc…

The only time you will do anything in your home space is if you want to modify these hidden configuration files

Do not place any data transfer here or store any data here!

Alpine HPC RoadMap

/home/$USER

/projects/$USER

/scratch/alpine/$USER

/pl/

Let’s change directories into our projects directory...

```
cd /projects/$USER
```

Alpine HPC RoadMap

/projects/$USER

This directory is slightly larger, at 250GB of space.

This should be used for more permanent storage of smaller files.

I recommend any local software installs (i.e. software you install yourself because it is not installed on Alpine) get installed here.

Additionally, all of your job scripts and code should be stored here

If small enough, reference data should be installed here; reference data meaning data that is used over and over again. In bioinformatics, this usually means genome index files, reference genome files (.fasta, .gtf), etc…

Again, not to be used for data storage of large files!

Alpine HPC RoadMap

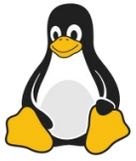
/home/$USER

/projects/$USER

/scratch/alpine/$USER

/pl/

Let’s change directories into our scratch directory...



```
cd /scratch/alpine/$USER
```

Alpine HPC RoadMap

/scratch/alpine/$USER

This is your largest directory, at 10TB!

This is the location where most of your data should be transferred into when you are ready for analysis

Any computation where the software or program is expected to write out output files, should be redirected to this directory space

This has the best read/write performance and compute nodes will be able to handle large volumes of data from this locations.

Alpine HPC RoadMap

/scratch/alpine/$USER

This is your largest directory, at 10TB!

This is the location where most of your data should be transferred into when you are ready for analysis

Any computation where the software or program is expected to write out output files, should be redirected to this directory space

This has the best read/write performance and compute nodes will be able to handle large volumes of data from this locations.

However, there are a few caveats…

- Storage is temporary – files are automatically deleted 90 days from the date of creation/transfer
- Do NOT have your only copy of data here or you risk losing it after the allotted time

Alpine HPC RoadMap

/home/$USER	/projects/$USER	/scratch/alpine/$USER	/pl/

For those of you that have a PetaLibrary allocation, you can access your allocation by using changing directories to the path that was given to you. Your allocation is mounted to your Alpine space automatically.

```
cd /pl/active/nameOfAllocation
```

Alpine HPC RoadMap

/pl/

The benefits of having a petalibrary allocation is if you want permanent storage of data files so that you don’t have to worry about constantly moving data in and out of your scratch space.

Consider a petalibrary allocation if:

- You have several TBs of data that you want permanently stored somewhere that Alpine has access to
- You don’t want to manage moving files back and forth from scratch to your local computer

There is a cost associated with PL, which is ~$45/TB/year, or $65/TB/year if you want a second duplicate copy (recommended if you don’t have a second copy save elsewhere) saved in case the hardware crashes

https://www.colorado.edu/rc/resources/petalibrary

Alpine HPC RoadMap

/home/$USER

2GB

- Do not use this for anything regarding data transfer, storage, software installs, etc…
- One access pre-existing configuration files here if you need to modify them (.bashrc, .bash_profile, .condrc, etc...)

/projects/$USER

250GB

- Store all code, scripts, and sbatch files here
- Local software that you need to install yourself should go here
- If small enough, reference files that will be re-used, i.e. genome index, genome fasta or gtf annotation files, etc...

/scratch/alpine/$USER

10TB

- Location for all data transfers and large data to be used to computation and analysis
- Output files generated by software should be redirected here

There is a 90-day automated data purge from the date of file creation/transfer; not to be used for permanent storage or any data that is single copy

/pl/

1TB-??

- OPTIONAL – only available to those that are paying for an allocation
- Permanent data storage mounted to your Alpine user space
- There is a cost associated with it

Now you are on the head/login node. What are your options?

Option A

Make a request to the the head node to login to a compute node so you can interactively run your code on a compute node, test code in real-time, or look at what software is installed on Alpine.

Option B

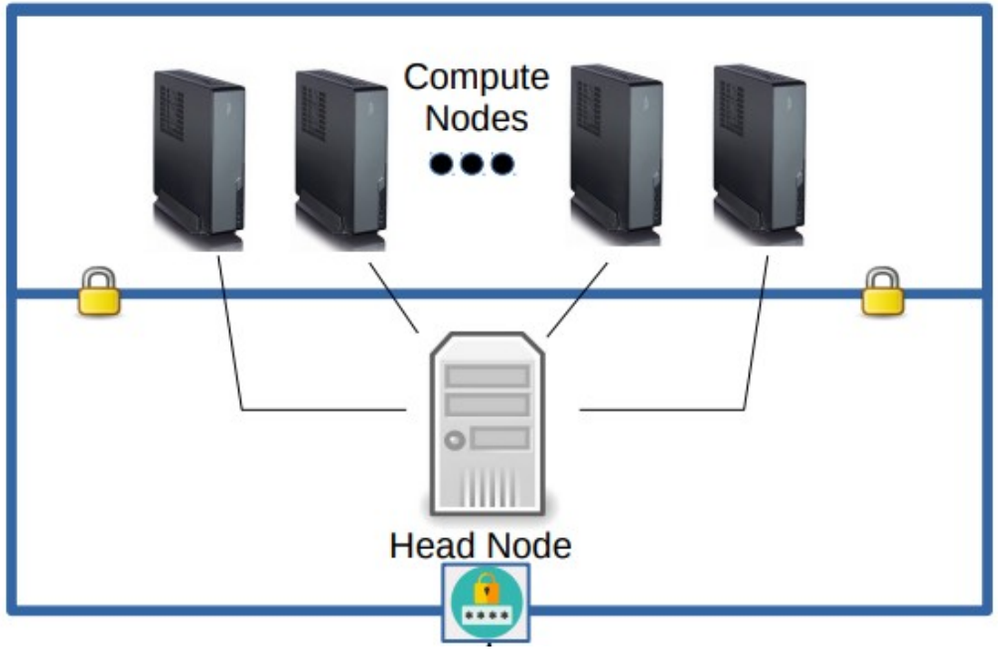
Stay on the head node because you want to submit a job remotely to a compute node.

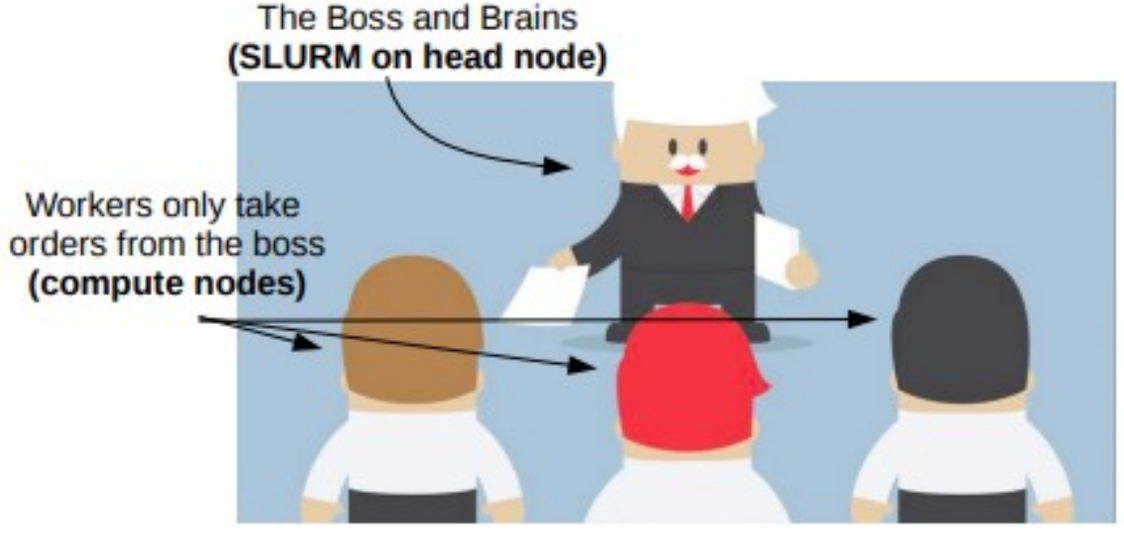
Regardless of which option you choose, you still need to communicate that to the head/login node, so how do we do that?

- You can communicate via **S**imple **L**inux **U**tility for **R**esource **M**anagement (SLURM) job scheduling language

- The head/login node and compute nodes all speak SLURM, so your task is to translate what you want into SLURM format.

- SLURM is a workload manager and is just one of the many flavor of job scheduling softwares available for HPCs. You may have heard of LSF (bsub, bqueues, etc…) or PBS (qsub, qstat, etc…) which are other flavors that are commonly used in Academia and all accomplish the same task – scheduling jobs by orchestrating communication between head/login nodes and compute nodes.





HPCs use job schedulers

- What is a job scheduler?
  - Responsible for executing multiple job requests (i.e. your scripts and programs) whether they come from the same or different user(s) and determining what resources are available, and whose jobs are next in line to be run
- Why do we need a scheduler?
  - “Fair-Share”
    - Sophisticated algorithm to schedule jobs fairly on a shared resource
    - The scheduler knows when each user made submitted requests so it can plan jobs effectively and know how long a job has been waiting in the queue or line
    - In the even all resources are being utilized, the “fair-share” policy is implemented meaning those users who have used the least amount of compute hours/resources within a window of time, get pushed to the front of the queue.
      - Prevents one individual from constantly using all the resources as the become available
  - Efficiency
    - Jobs that are shorter or require less resources typically are in the queue shorter since resources become available more quickly and are more likely to fit into untapped resources
    - The scheduler know how many resources every user has requested and the time and memory constraints of those resources

Now you are on the head/login node. What are your options?

Option A

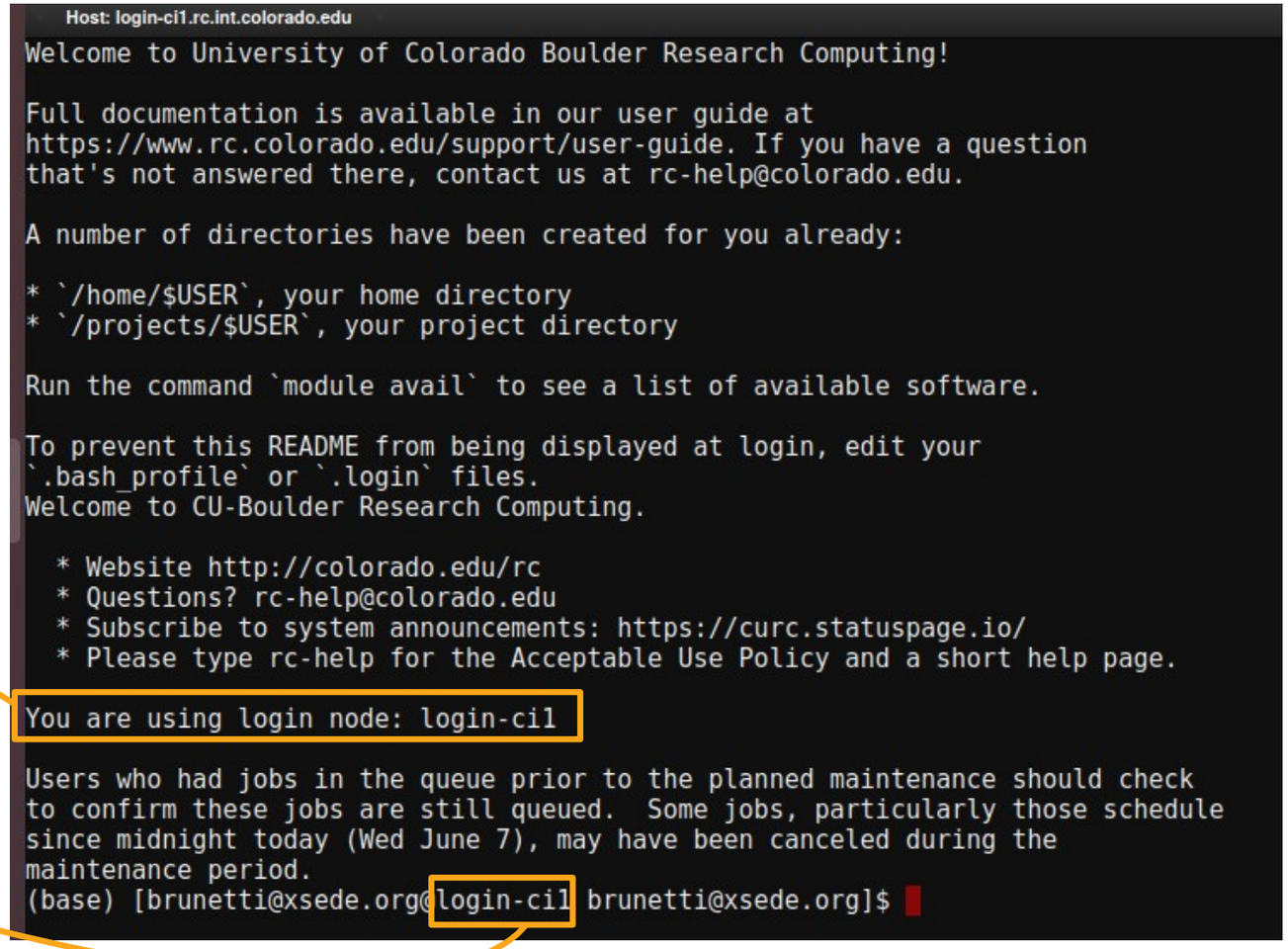
Make a request to the the head node to login to a compute node so you can interactively run your code on a compute node, test code in real-time, or look at what software is installed on Alpine.

Option B

Stay on the head node because you want to submit a job remotely to a compute node.

Option A: Request an interactive compute node

In case you get confused or are unsure if you are on the head/login node or a compute node you can check here! Anything that say `login-ci` followed by a number is the login/head node. I am on `login-ci1`, but you might have a different ci number



Request an short interactive compute node

```
[brunetti@xsede.org@login-cli brunetti@xsede.org]$ acompile
```

Now, we are inside of a compute node!

```
Welcome to University of Colorado Boulder Research Computing!

Full documentation is available in our user guide at
https://www.rc.colorado.edu/support/user-guide. If you have a question
that's not answered there, contact us at rc-help@colorado.edu.

A number of directories have been created for you already:

* `/home/$USER`, your home directory
* `/projects/$USER`, your project directory

Run the command `module avail` to see a list of available software.

To prevent this README from being displayed at login, edit your
`.bash_profile` or `.login` files.
Welcome to CU-Boulder Research Computing.

  * Website http://colorado.edu/rc
  * Questions? rc-help@colorado.edu
  * Subscribe to system announcements: https://curc.statuspage.io/
  * Please type rc-help for the Acceptable Use Policy and a short help page.

You are using login node: login-ci1

Users who had jobs in the queue prior to the planned maintenance should check
to confirm these jobs are still queued.  Some jobs, particularly those schedule
since midnight today (Wed June 7), may have been canceled during the
maintenance period.
(base) [brunetti@xsede.org@login-ci1 brunetti@xsede.org]$ acompile
acompile: submitting job... salloc --nodes=1 --partition=acompile --ntasks=1  --time=01:00:00  --qos=compile --job-name=acompile --bell --oversubscribe srun --pty /bin/bash
salloc: Granted job allocation 2650311
salloc: Nodes c3cpu-c15-u7-2 are ready for job
(base) [brunetti@xsede.org@c3cpu-c15-u7-2 brunetti@xsede.org]$
```

You can now see that I am on a compute node. Some of the numbers and characters may be a bit different as there are 382 different compute nodes on Alpine as of September 2023, but the big thing is you see it says `cpu` somewhere in there.

Now, we are inside of a compute node!

```
Welcome to University of Colorado Boulder Research Computing!

Full documentation is available in our user guide at
https://www.rc.colorado.edu/support/user-guide. If you have a question
that's not answered there, contact us at rc-help@colorado.edu.

A number of directories have been created for you already:

* `/home/$USER`, your home directory
* `/projects/$USER`, your project directory

Run the command `module avail` to see a list of available software.

To prevent this README from being displayed at login, edit your
`.bash_profile` or `.login` files.
Welcome to CU-Boulder Research Computing.

  * Website http://colorado.edu/rc
  * Questions? rc-help@colorado.edu
  * Subscribe to system announcements: https://curc.statuspage.io/
  * Please type rc-help for the Acceptable Use Policy and a short help page.

You are using login node: login-ci1

Users who had jobs in the queue prior to the planned maintenance should check
to confirm these jobs are still queued.  Some jobs, particularly those schedule
since midnight today (Wed June 7), may have been canceled during the
maintenance period.
(base) [brunetti@xsede.org@login-ci1 brunetti@xsede.org]$ acompile
acompile: submitting job... salloc --nodes=1 --partition=acompile --ntasks=1  --time=01:00:00  --qos=compile --job-name=acompile --bell --oversubscribe srun --pty /bin/bash
salloc: Granted job allocation 2650311
salloc: Nodes c3cpu-c15-u7-2 are ready for job
(base) [brunetti@xsede.org@c3cpu-c15-u7-2 brunetti@xsede.org]$
```

Notice that if you call `acompile` without any specifications it shows these parameters. This means you are requesting a CPU with 1 core (`--ntasks=1`), for 1 hour (`--time=01:00:00`), which is the default

This means that your session will automatically close/get killed if you use if longer than 1 hour.

The command below from the head node would not kill your job until 12 hours, which is also the maximum time you can request and interactive job

```
[brunetti@xsede.org@login-cli brunetti@xsede.org]$ acompile --time=12:00:00
```

Option A: Things we can do once we are interactively inside of a compute node...

- Check what software is installed on Alpine

- Install our own software

- Run software and code interactively

Option A: Things we can do once we are interactively inside of a compute node...

- Check what software is installed on Alpine

- Install our own software

- Run software and code interactively

Option A: Things we can do once we are interactively inside of a compute node...

- **Check what software is installed on Alpine**


```
module avail
```

Option A: Things we can do once we are interactively inside of a compute node...

- **Check what software is installed on Alpine**


```
module avail
```

```
base) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$ module avail

---------------------------------------------------------------------------------- /curc/sw/modules/slurm ----------------------------------------------------------------------------------
  StdEnv    curc-quota/latest (D)    slurm/alpine (D)    slurm/blanca    slurm/core    slurmtools (D)

------------------------------------------------------------------------------ /usr/share/lmod/lmod/modulefiles/Core ------------------------------------------------------------------------------
  lmod    settarg

------------------------------------------------------------------------------------ Compilers -------------------------------------------------------------------------------------
  aocc/3.1.0 (D)    aocc/3.2.0    gcc/10.3.0    gcc/11.2.0 (D)    intel/2022.1.2 (m)    nvhpc_sdk/2022.229    nvhpc_sdk/2023.233 (D)

---------------------------------------------------------------------------------- Independent Applications ----------------------------------------------------------------------------------
  R/3.6.3              cuda/11.3          (g)    ghostscript/9.56.0          julia/1.6.6              paraview/5.0.1          rclone/1.58.0             tdom/0.9.2          (D)
  R/4.2.2        (D)   cuda/11.4          (g)    git-lfs/3.1.2               julia/1.8.1        (D)   paraview/5.6.0          rhel7for8/1.0             texlive/2021
  allinea/6.0.4  (m)   cuda/11.8          (g)    git/2.31.0                  lftp/4.8.4               paraview/5.9.0          rocm/5.2.3          (g)   totalview/2016.
6.21
  anaconda/2020.11     cuda/12.1.1        (g,D)  gmsh/2.16.0                 loadbalance/0.2          paraview/5.10.0   (D)   rocm/5.3.0          (g)   ucx/1.10.1
  anaconda/2022.10 (D) cudnn/8.1          (g)    gmsh/4.11.1           (D)   mambaforge/23.1.0-1      pdtoolkit/3.22          rocm/5.5.0          (g,D) ucx/1.12.1          (D)
  arm-forge/19.1.3 (m) cudnn/8.2          (g)    gnu_parallel/20160622       mathematica/9.0          pdtoolkit/3.25.1  (D)   ruby/2.3.1                udunits/2.2.20
  autotools/2.69       cudnn/8.6          (g,D)  gnu_parallel/20210322 (D)   mathematica/11.1.0 (D)   perl/5.16.3             ruby/3.0.0          (D)   udunits/2.2.24
  autotools/2.71 (D)   curc-quota/latest         gnuplot/5.4.3               matlab/R2018b            perl/5.24.0       (D)   singularity/3.6.4 (D)     udunits/2.2.25
  chimerax/1.2.5       dmtcp/2.6.0                idl/8.7                     matlab/R2019b            perl/5.28.1             singularity/3.7.4         udunits/2.2.28      (D)
  cmake/3.5.2          eigen/3.4.0                imagemagick/6.9.12          matlab/R2020b            perl/5.36.0             slurmtools/0.0.0          valgrind/3.11.0
  cmake/3.9.2          emacs/25.3                 jdk/1.7.0                   matlab/R2021b      (D)   pigz/2.7                slurmtools/0.0.1          valgrind/3.17.0     (D)
  cmake/3.14.1         emacs/27.2         (D)     jdk/1.8.0_91                matlab/R2022b            python/2.7.18           subversion/1.8.16         vapor/3.3.0
  cmake/3.20.2         expat/2.1.1                jdk/1.8.0_281               maven/3.8.1              python/3.10.2     (D)   subversion/1.10.2         vapor/3.4.0         (D)
  cmake/3.25.0   (D)   expat/2.3.0        (D)     jdk/1.8.0                   ncl/6.3.0                qchem/4010              subversion/1.14.1 (D)     vtf3/1.43
  cube/3.4.3           ffmpeg/4.4                 jdk/18.0.1.1          (D)   ncl/6.6.2          (D)   qt/5.6.0                tcltk/8.6.5               zip/rhel7
  cube/4.3.4     (D)   gdb/8.1                    julia/0.6.2                 papi/5.4.3               qt/5.9.1                tcltk/8.6.11      (D)
  cuda/11.2      (g)   gdb/10.1           (D)     julia/1.6.0                 papi/5.5.1         (D)   qt/5.15           (D)   tdom/0.8.3

------------------------------------------------------------------------------------ Bioinformatics --------------------------------------------------------------------------------------
  alphafold/2.2.0       bbtools/39.01     bowtie2/2.5.0     cutadapt/4.2     homer/4.11     nextflow/22.10.6       plink2/2.00a2.3    sra-toolkit/3.0.0
  alphafold/2.3.1 (D)   bcftools/1.16     bwa/0.7.17        fastqc/0.11.9    htslib/1.16    nextflow/23.04   (D)   qiime2/2023.5      star/2.7.10b
  bamtools/2.5.2        bedtools/2.29.1   cellranger/7.1.0  gatk/4.3.0.0     multiqc/1.14   picard/2.27.5          samtools/1.16.1    trimmomatic/0.39

---------------------------------------------------------------------------------- Lmod Internal Modules ----------------------------------------------------------------------------------
  StdEnv    lmod    settarg

 Where:
  g:  built for GPU
  m:  built for host and native MIC
  D:  Default Module

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

Option A: Things we can do once we are interactively inside of a compute node...

- **Check what software is installed on Alpine**

`module avail`

Just a reminder! **I have to be inside of a compute node** to see this, not the head/login node!!

```
base) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$ module avail

------------------------------------------------------------------------------ /curc/sw/modules/slurm -------------------------------------------------------------------------------
   StdEnv    curc-quota/latest (D)    slurm/alpine (D)    slurm/blanca    slurm/core    slurmtools (D)

------------------------------------------------------------------------- /usr/share/lmod/lmod/modulefiles/Core --------------------------------------------------------------------------
   lmod    settarg

------------------------------------------------------------------------------ Compilers -------------------------------------------------------------------------------
   aocc/3.1.0 (D)    aocc/3.2.0    gcc/10.3.0    gcc/11.2.0 (D)    intel/2022.1.2 (m)    nvhpc_sdk/2022.229    nvhpc_sdk/2023.233 (D)

---------------------------------------------------------------------------- Independent Applications ----------------------------------------------------------------------------
   R/3.6.3              cuda/11.3          (g)    ghostscript/9.56.0          julia/1.6.6              paraview/5.0.1          rclone/1.58.0          tdom/0.9.2          (D)
   R/4.2.2        (D)   cuda/11.4          (g)    git-lfs/3.1.2               julia/1.8.1        (D)   paraview/5.6.0          rhel7for8/1.0          texlive/2021
   allinea/6.0.4  (m)   cuda/11.8          (g)    git/2.31.0                  lftp/4.8.4               paraview/5.9.0          rocm/5.2.3         (g)  totalview/2016.
6.21
   anaconda/2020.11     cuda/12.1.1        (g,D)  gmsh/2.16.0                 loadbalance/0.2          paraview/5.10.0    (D)  rocm/5.3.0         (g)  ucx/1.10.1
   anaconda/2022.10 (D) cudnn/8.1          (g)    gmsh/4.11.1           (D)   mambaforge/23.1.0-1      pdtoolkit/3.22          rocm/5.5.0         (g,D)ucx/1.12.1          (D)
   arm-forge/19.1.3 (m) cudnn/8.2          (g)    gnu_parallel/20160622       mathematica/9.0          pdtoolkit/3.25.1   (D)  ruby/2.3.1                  udunits/2.2.20
   autotools/2.69       cudnn/8.6          (g,D)  gnu_parallel/20210322 (D)   mathematica/11.1.0 (D)   perl/5.16.3             ruby/3.0.0         (D)  udunits/2.2.24
   autotools/2.71 (D)   curc-quota/latest       gnuplot/5.4.3               matlab/R2018b            perl/5.24.0        (D)  singularity/3.6.4  (D)  udunits/2.2.25
   chimerax/1.2.5       dmtcp/2.6.0            idl/8.7                     matlab/R2019b            perl/5.28.1             singularity/3.7.4          udunits/2.2.28      (D)
   cmake/3.5.2          eigen/3.4.0            imagemagick/6.9.12          matlab/R2020b            perl/5.36.0             slurmtools/0.0.0           valgrind/3.11.0
   cmake/3.9.2          emacs/25.3             jdk/1.7.0                   matlab/R2021b      (D)   pigz/2.7                slurmtools/0.0.1           valgrind/3.17.0     (D)
   cmake/3.14.1         emacs/27.2       (D)   jdk/1.8.0_91                matlab/R2022b            python/2.7.18           subversion/1.8.16          vapor/3.3.0
   cmake/3.20.2         expat/2.1.1            jdk/1.8.0_281               maven/3.8.1              python/3.10.2      (D)  subversion/1.10.2          vapor/3.4.0         (D)
   cmake/3.25.0   (D)   expat/2.3.0      (D)   jdk/1.8.0                   ncl/6.3.0                qchem/4010              subversion/1.14.1  (D)  vtf3/1.43
   cube/3.4.3           ffmpeg/4.4             jdk/18.0.1.1          (D)   ncl/6.6.2          (D)   qt/5.6.0                tcltk/8.6.5                zip/rhel7
   cube/4.3.4     (D)   gdb/8.1                julia/0.6.2                 papi/5.4.3               qt/5.9.1                tcltk/8.6.11       (D)
   cuda/11.2      (g)   gdb/10.1           (D) julia/1.6.0                 papi/5.5.1         (D)   qt/5.15            (D)  tdom/0.8.3

------------------------------------------------------------------------------ Bioinformatics --------------------------------------------------------------------------------
   alphafold/2.2.0       bbtools/39.01      bowtie2/2.5.0      cutadapt/4.2     homer/4.11      nextflow/22.10.6       plink2/2.00a2.3    sra-toolkit/3.0.0
   alphafold/2.3.1 (D)   bcftools/1.16      bwa/0.7.17         fastqc/0.11.9    htslib/1.16     nextflow/23.04   (D)   qiime2/2023.5      star/2.7.10b
   bamtools/2.5.2        bedtools/2.29.1    cellranger/7.1.0   gatk/4.3.0.0     multiqc/1.14    picard/2.27.5          samtools/1.16.1    trimmomatic/0.39

---------------------------------------------------------------------------- Lmod Internal Modules ----------------------------------------------------------------------------
   StdEnv    lmod    settarg

  Where:
   g:  built for GPU
   m:  built for host and native MIC
   D:  Default Module

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

Option A: Things we can do once we are interactively inside of a compute node...

- Check what software is installed on Alpine

- Install our own software

- Run software and code interactively

Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software**

You will notice that not all software is installed on Alpine but you can install Alpine yourself locally. There are typically two recommended ways of doing this:

- Through a mamba (preferred) or conda environment

- Following the website instructions to install from source locally

Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – using mamba and/or conda**

Recall, that your home directory is really, really small (only 2GB), so we always want to make sure software is installed in our projects directory (`/projects/$USER/`). By default conda likes to install in home, so before do any conda work, we must change our `.condarc` file to include the following lines:

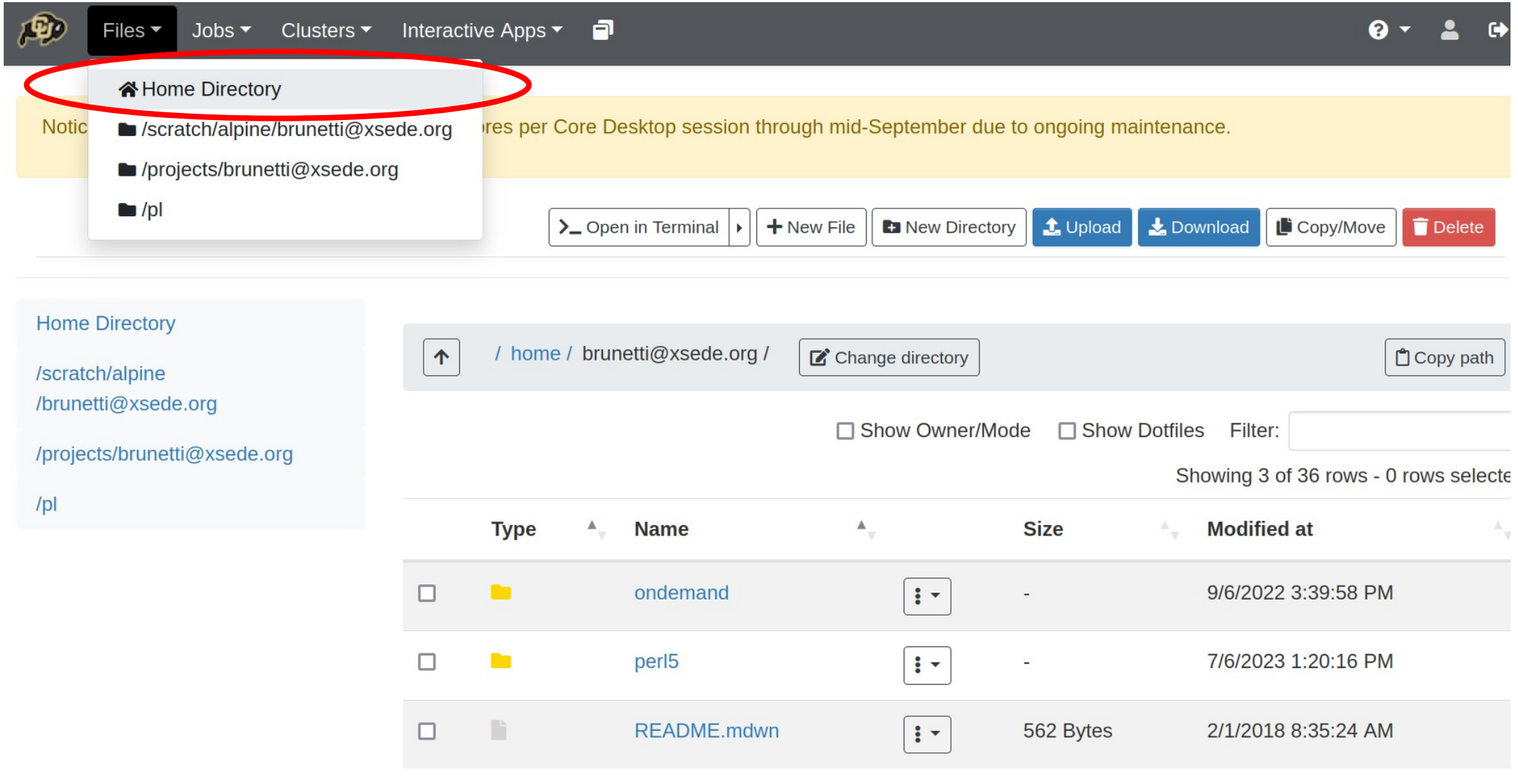
```
pkgs_dirs:
  - /projects/$USER/.conda_pkgs
envs_dirs:
  - /projects/$USER/software/anaconda/envs
```

This only every needs to be done 1 time for the lifetime of your Alpine account!

Option A: Things we can do once we are interactively inside of a compute node...

- **Through a mamba (preferred) or conda environment**

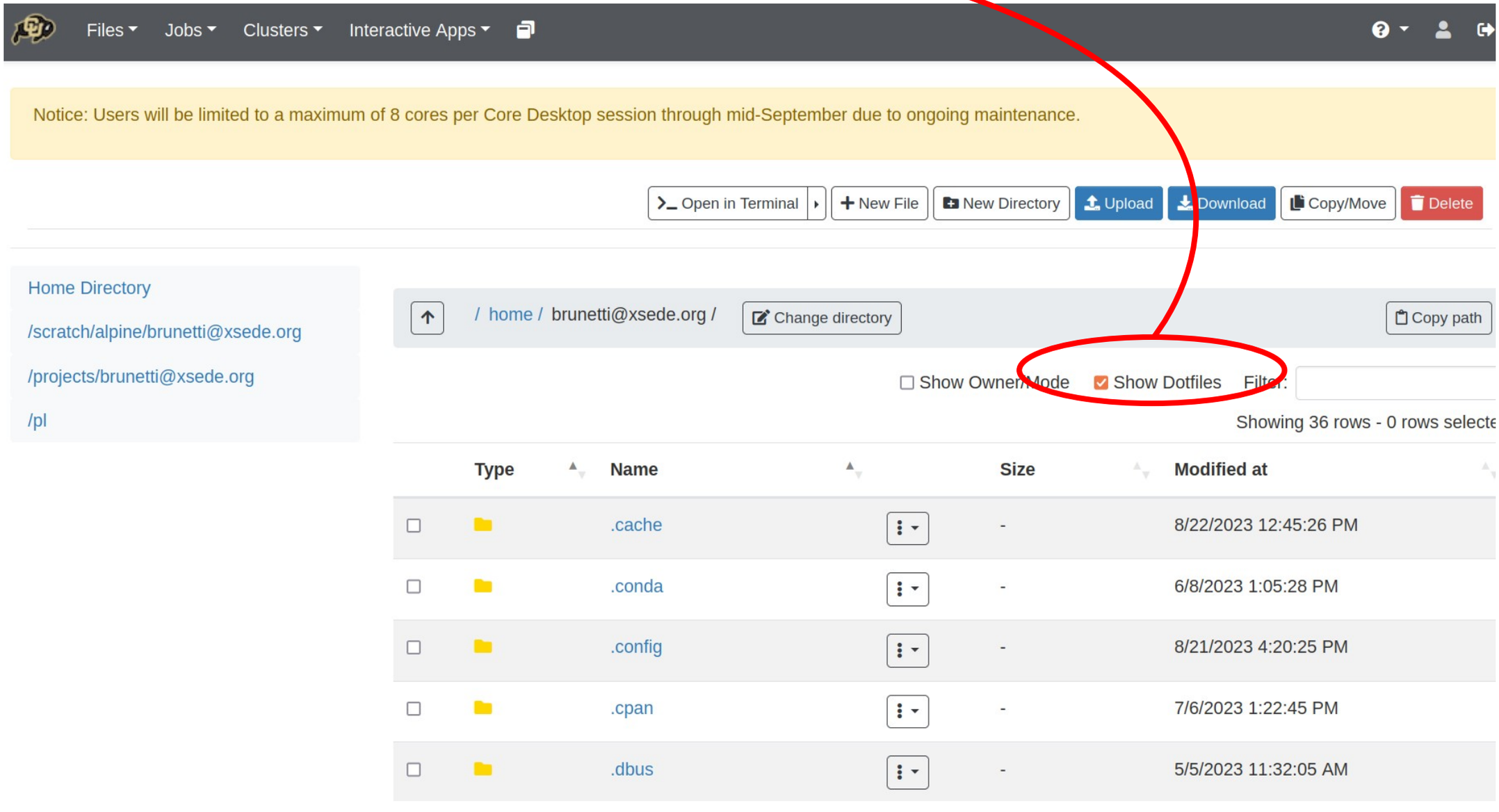
1 Navigate to your Home Directory within your OnDemand session:



Option A: Things we can do once we are interactively inside of a compute node...

- **Through a mamba (preferred) or conda environment**

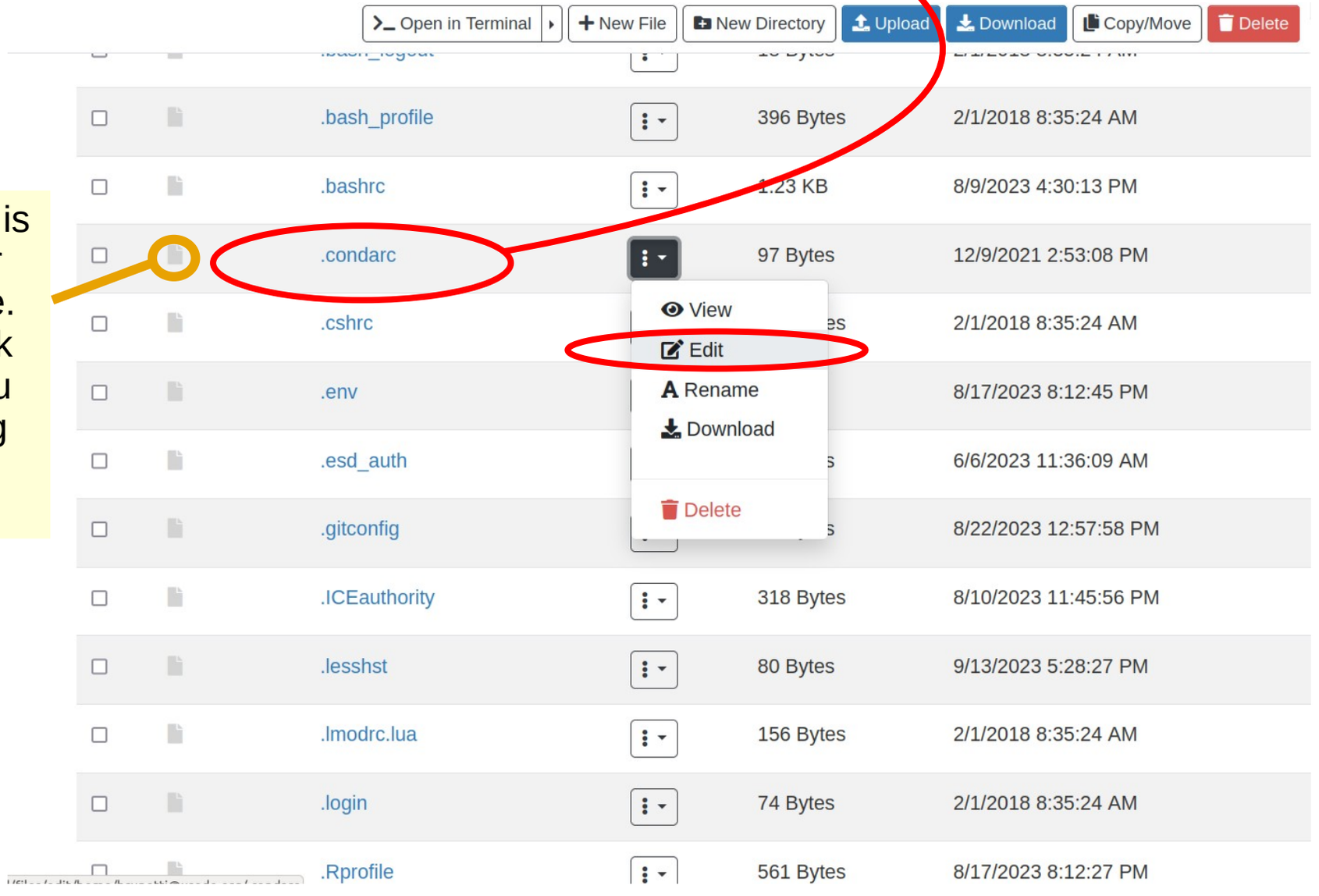
2 Make sure to click on the box that say “Show Dotfiles”



Option A: Things we can do once we are interactively inside of a compute node...

- **Through a mamba (preferred) or conda environment**

3 Scrolls down your list and find the file called .condarc and click the three dots and select Edit from the drop down menu

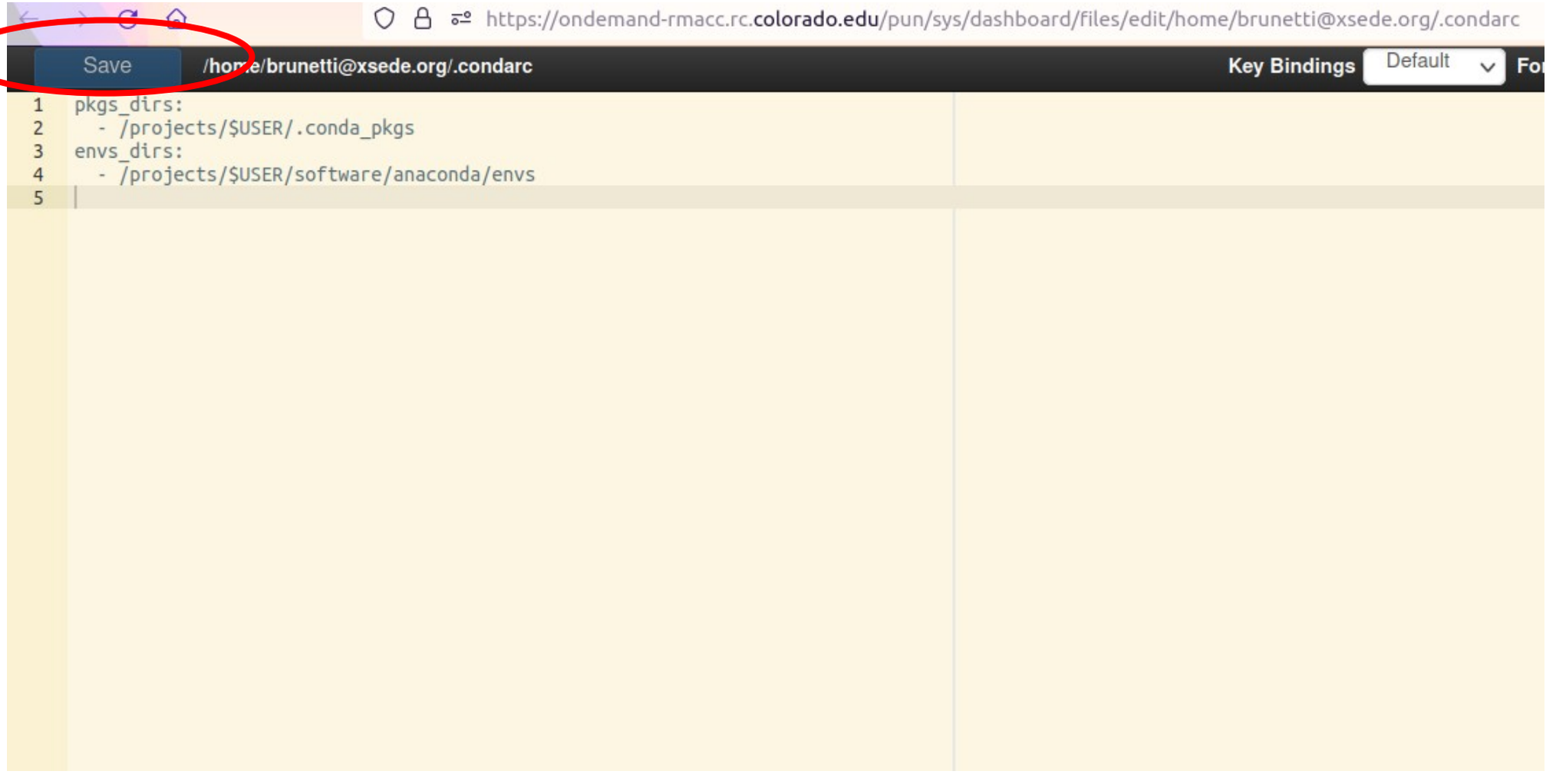


Be careful! There is a .condarc folder and a .condarc file. Make sure to click the file, which you can tell by seeing the little paper image to the left

Option A: Things we can do once we are interactively inside of a compute node...

- **Through a mamba (preferred) or conda environment**

4 This takes you to a text editor where you can copy and paste the following lines. Then be sure to press Save in the upper left hand corner



Option A: Things we can do once we are interactively inside of a compute node...

- **Through a mamba (preferred) or conda environment**

5 After you press save, you can exit the browser tab and you are done configuring conda!

Option A: Things we can do once we are interactively inside of a compute node...

- **Through a mamba (preferred) or conda environment**

5 After you press save, you can exit the browser tab and you are done configuring conda!

Now that conda is configured, let's go through a tutorial of how to install software that has a conda installation option.

Let's try to install the commonly used bioinformatics package, RSEM, which is a way to count/quantify reads mapping to a genome.

Option A: Things we can do once we are interactively inside of a compute node...

- **Through a mamba (preferred) or conda environment**

5 After you press save, you can exit the browser tab and you are done configuring conda!

Now that conda is configured, let's go through a tutorial of how to install software that has a conda installation option.

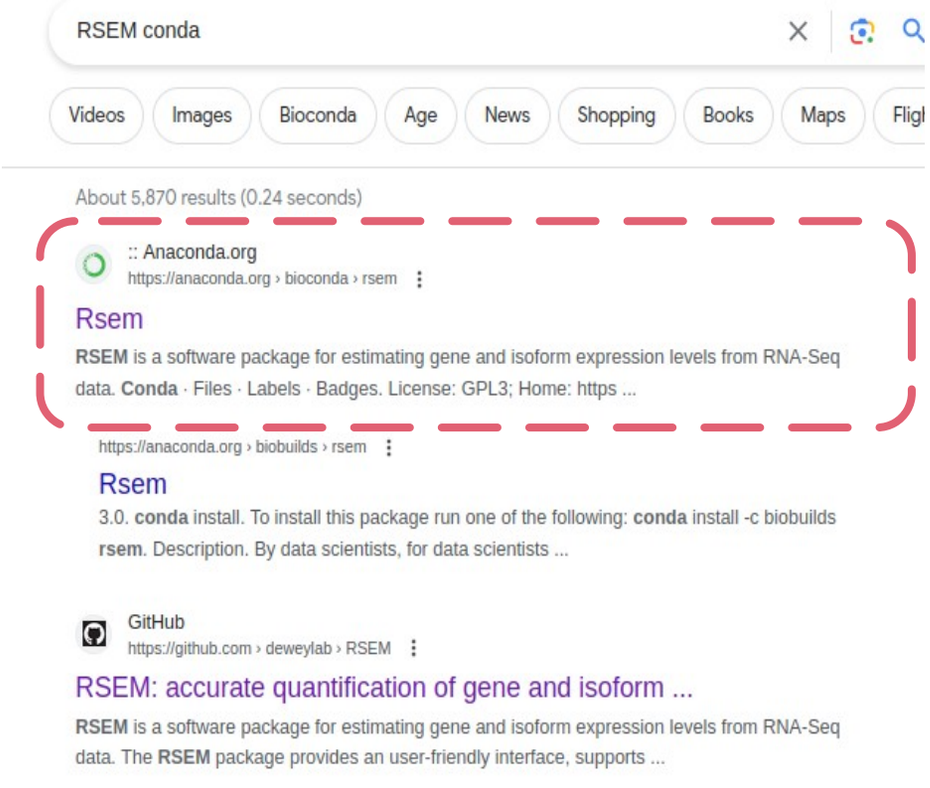
Let's try to install the commonly used bioinformatics package, RSEM, which is a way to count/quantify reads mapping to a genome.

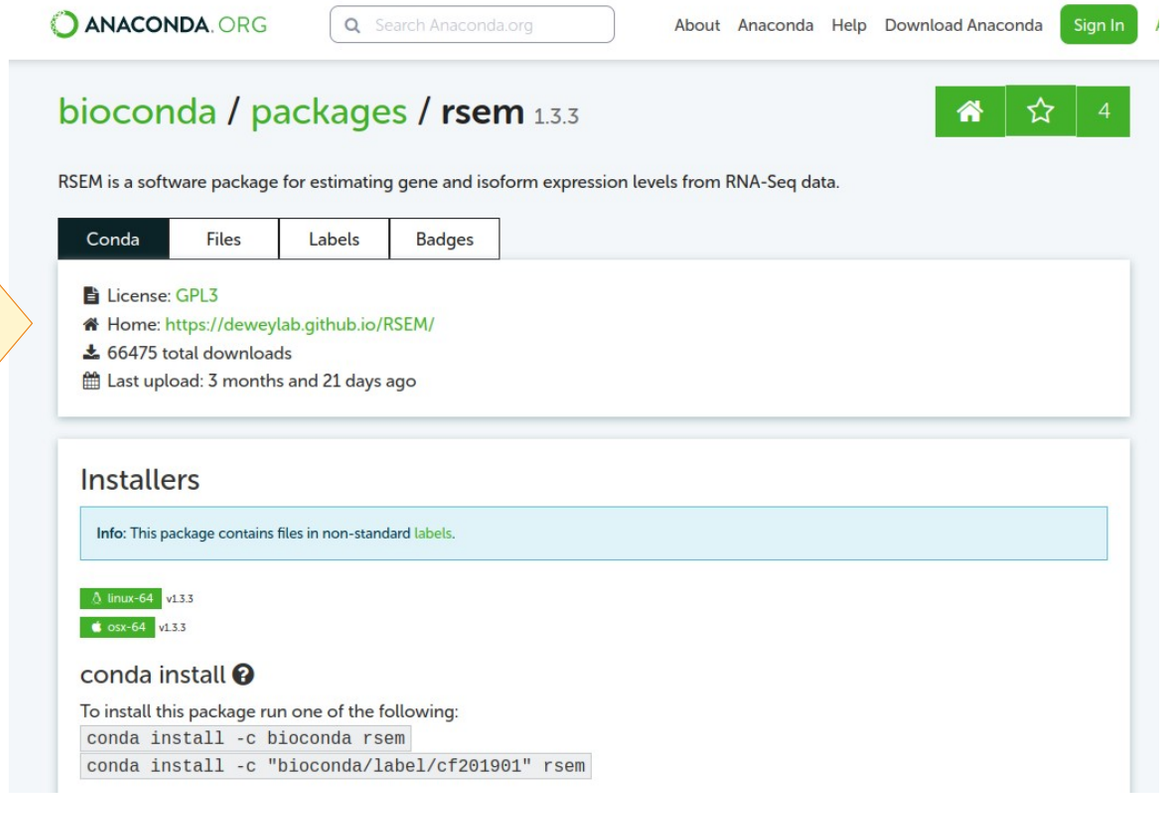
If you want to use mamba, you still need to go through the same .condarc configuration we just went through. Kevin has made a nice guide on how to proceed from there on our github page located at:
https://github.com/kf-cuanschutz/CU-Anschutz-HPC-documentation/blob/main/mamba_tutorial.md

Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – using conda**

1 Let's confirm that RSEM has a conda installation version – you can do this by going to the anaconda site and searching, looking at a software's install instructions, or by googling:





Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – using conda**

2 Let's go back to our terminal session in OnDemand. Remember, **we are still in a compute node at this point**! Now, in order to use conda to install software, we must first load in anaconda to our space. Recall, when we ran `module avail` we can see anaconda as a preinstalled software on Alpine.


```
module load anaconda
```

`module load` can be used on any software listed in the `module avail` section to load the software into your terminal shell. Note, that some software has multiple versions, so copy and paste the software version you want to use

Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – using conda**

3 Now, we need to create a conda environment for the software. I would suggest you name your environment something that relates to the software; no spaces or special characters. Keep in mind you can create as many conda environments as you want, so keep software in separate environments to prevent compatibility issues.

```
conda create --name rsem_install
```

Replace `rsem_install` with whatever you want to name your install of RSEM. Again, no spaces or special characters! Replace spaces with underscores

Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – using conda**

4 When you run the last command, you will see something similar to the following:

```
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 22.9.0
  latest version: 23.7.4

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: /projects/brunetti@xsede.org/software/anaconda/envs/rsem_install


Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate rsem_install
#
# To deactivate an active environment, use
#
#     $ conda deactivate

Retrieving notices: ...working... done
(base) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$
```

Make sure when it asks if you want to proceed that you type in `y` and then hit enter.

You will also notice, once the environment is created, it tells you how activate it and deactivate it

Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – using conda**

5 To confirm you environment is made, you can run the following and you should see all the conda environments you have made on Alpine

```
conda env list
```

```
(base) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$ conda env list
# conda environments:
#
base                  *  /curc/sw/anaconda3/2022.10
ATOC_NWP                 /curc/sw/anaconda3/2022.10/envs/ATOC_NWP
bash_spr23               /curc/sw/anaconda3/2022.10/envs/bash_spr23
globus                   /curc/sw/anaconda3/2022.10/envs/globus
ood_jupyter_base         /curc/sw/anaconda3/2022.10/envs/ood_jupyter_base
pyomp_2022               /curc/sw/anaconda3/2022.10/envs/pyomp_2022
pyscenic                 /projects/brunetti@xsede.org/software/anaconda/envs/pyscenic
rhapsody_v2.0_py3.11.4   /projects/brunetti@xsede.org/software/anaconda/envs/rhapsody_v2.0_py3.11.4
rsem_install             /projects/brunetti@xsede.org/software/anaconda/envs/rsem_install
signet_env               /projects/brunetti@xsede.org/software/anaconda/envs/signet_env
t1k                      /projects/brunetti@xsede.org/software/anaconda/envs/t1k

(base) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$
```

Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – using conda**

6 Now, let's activate our environment:

```
conda activate rsem_install
```

You will notice when your environment is activated, you will see the name of your environment in parentheses to the left of your shell prompt:

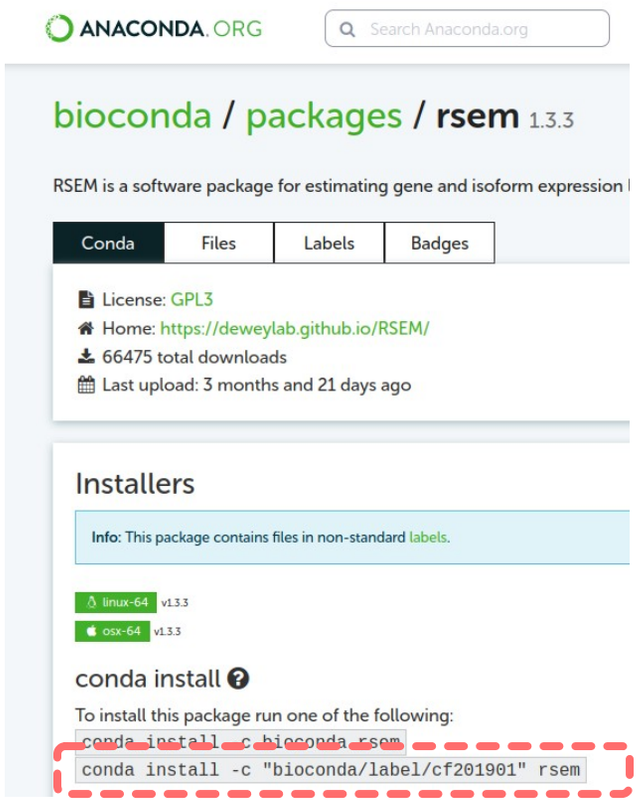
```
(base) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$
(base) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$ conda env list
# conda environments:
#
base                  *  /curc/sw/anaconda3/2022.10
ATOC_NWP                 /curc/sw/anaconda3/2022.10/envs/ATOC_NWP
bash_spr23               /curc/sw/anaconda3/2022.10/envs/bash_spr23
globus                   /curc/sw/anaconda3/2022.10/envs/globus
ood_jupyter_base         /curc/sw/anaconda3/2022.10/envs/ood_jupyter_base
pyomp_2022               /curc/sw/anaconda3/2022.10/envs/pyomp_2022
pyscenic                 /projects/brunetti@xsede.org/software/anaconda/envs/pyscenic
rhapsody_v2.0_py3.11.4     /projects/brunetti@xsede.org/software/anaconda/envs/rhapsody_v2.0_py3.11.4
rsem_install             /projects/brunetti@xsede.org/software/anaconda/envs/rsem_install
signet_env               /projects/brunetti@xsede.org/software/anaconda/envs/signet_env
t1k                      /projects/brunetti@xsede.org/software/anaconda/envs/t1k

(base) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$ conda activate rsem_install
(rsem_install) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$
```

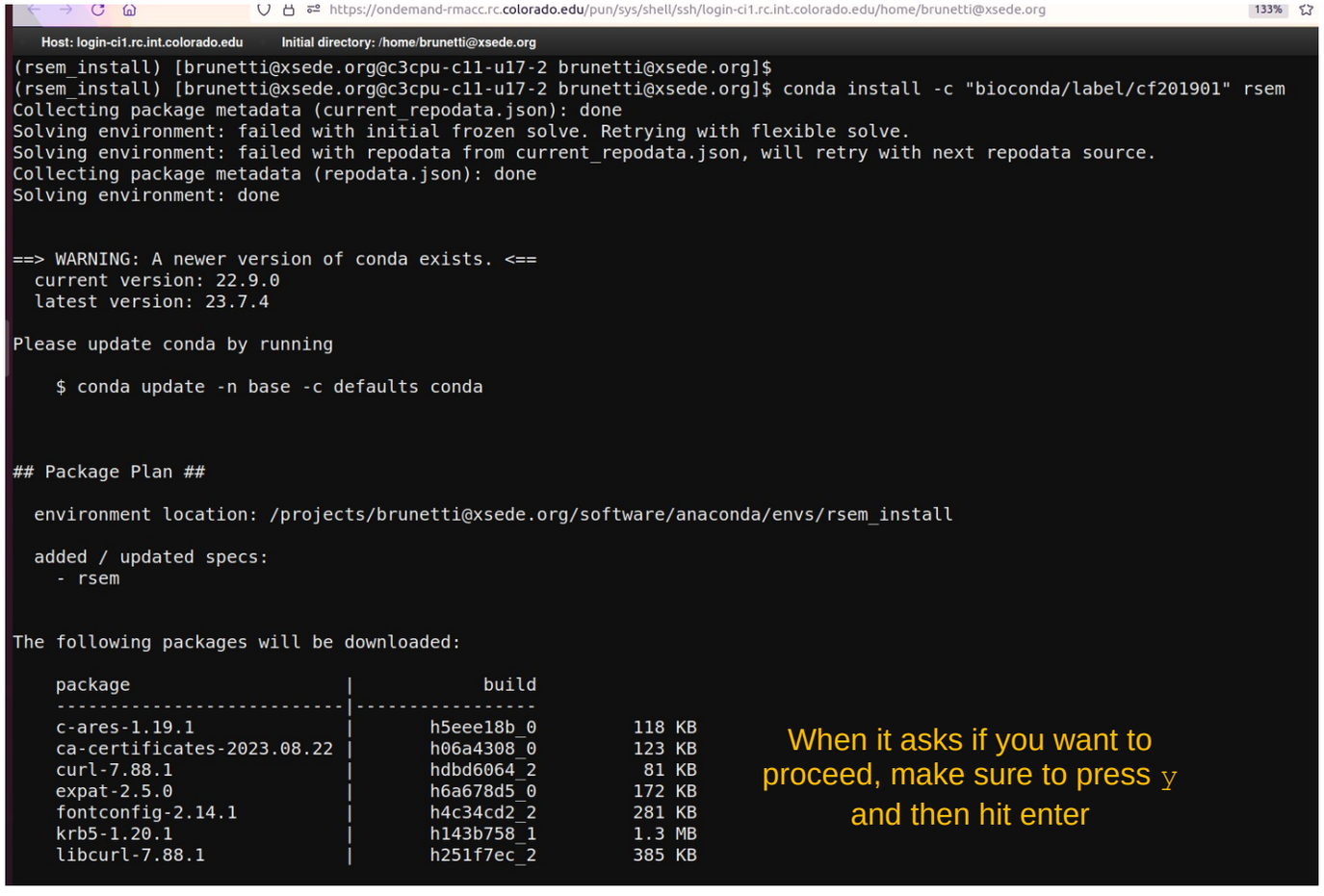
Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – using conda**

7 After our environment is activated, you can go back to the google page and copy and paste one of the conda commands of your choosing to install RSEM:



```
conda install -c "bioconda/label/cf201901" rsem
```



Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – using conda**

8 If you get your an empty prompt back with no errors, it means it has successfully installed!

```
r-codetools-0.2_15    | 47 KB     | ########################################################################################## | 100%
r-class-7.3_14        | 89 KB     | ########################################################################################## | 100%
r-mgcv-1.8_22         | 2.4 MB    | ########################################################################################## | 100%
ca-certificates-2023  | 123 KB    | ########################################################################################## | 100%
openssl-3.0.10        | 5.2 MB    | ########################################################################################## | 100%
r-boot-1.3_20         | 606 KB    | ########################################################################################## | 100%
libev-4.33            | 111 KB    | ########################################################################################## | 100%
libnghttp2-1.52.0     | 672 KB    | ########################################################################################## | 100%
r-spatial-7.3_11      | 135 KB    | ########################################################################################## | 100%
libwebp-base-1.3.2    | 387 KB    | ########################################################################################## | 100%
krb5-1.20.1           | 1.3 MB    | ########################################################################################## | 100%
r-cluster-2.0.6       | 499 KB    | ########################################################################################## | 100%
expat-2.5.0           | 172 KB    | ########################################################################################## | 100%
c-ares-1.19.1         | 118 KB    | ########################################################################################## | 100%
libssh2-1.10.0        | 292 KB    | ########################################################################################## | 100%
r-lattice-0.20_35     | 686 KB    | ########################################################################################## | 100%
libtiff-4.2.0         | 466 KB    | ########################################################################################## | 100%
preparing transaction: done
verifying transaction: done
executing transaction: done
retrieving notices: ...working... done
(rsem_install) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$
```

Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – using conda**

9 To test your install, you can look up the software documentation on how to use it and the software should come up. In the case of RSEM, I know that their instructions say I should be able to call: `rsem-prepare-reference`

```
Host: login-ci1.rc.int.colorado.edu   Initial directory: /home/brunetti@xsede.org   Themes: Default
(rsem_install) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$ rsem-prepare-reference
Can't locate Env.pm in @INC (you may need to install the Env module) (@INC contains: /projects/brunetti@xsede.org/software/anaconda/envs/rsem_install/lib
/rsem /home/brunetti@xsede.org/perl5/lib/perl5 /usr/local/lib64/perl5 /usr/local/share/perl5 /usr/lib64/perl5/vendor_perl /usr/share/perl5/vendor_perl /u
sr/lib64/perl5 /usr/share/perl5) at /projects/brunetti@xsede.org/software/anaconda/envs/rsem_install/bin/../lib/rsem/rsem-prepare-reference line 10.
BEGIN failed--compilation aborted at /projects/brunetti@xsede.org/software/anaconda/envs/rsem_install/bin/../lib/rsem/rsem-prepare-reference line 10.
(rsem_install) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$
```

Sometimes you will notice some software has other dependencies that don't get installed. Once common one is Perl. If you see the following error where the suggestion is @INC or the file in question end in .pm, that is a Perl dependency issue. It can be easily resolved by loading Perl into your space which is a language already installed on Alpine

To fix issues that are Perl dependent, run the following:

```
module load perl
```

Then try again, and you should see the `rsem-prepare-reference` manual/help page pop up – successfully installed!

```
(rsem_install) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$
(rsem_install) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$ module load perl
(rsem_install) [brunetti@xsede.org@c3cpu-c11-u17-2 brunetti@xsede.org]$ rsem-prepare-reference
Invalid number of arguments!
NAME
    rsem-prepare-reference

SYNOPSIS
     rsem-prepare-reference [options] reference_fasta_file(s) reference_name

ARGUMENTS
    reference_fasta_file(s)
        Either a comma-separated list of Multi-FASTA formatted files OR a
        directory name. If a directory name is specified, RSEM will read all
        files with suffix ".fa" or ".fasta" in this directory. The files
        should contain either the sequences of transcripts or an entire
        genome, depending on whether the '--gtf' option is used.

    reference name
        The name of the reference used. RSEM will generate several
        reference-related files that are prefixed by this name. This name
        can contain path information (e.g. '/ref/mm9').
```

Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – using conda**

In summary, once you install your software, in this case, RSEM, you can now add it to any script or compute environment and use the software by adding the following lines to your code:

```
module load anaconda
conda activate rsem_install
module load perl
```

If you ever want to deactivate your environment, usually for the sake of using a different version of the same software you can run the following:

```
conda deactivate rsem_install
```

Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – using conda**

In summary, once you install your software, in this case, RSEM, you can now add it to any script or compute environment and use the software by adding the following lines to your code:

```
module load anaconda
conda activate rsem_install
module load perl
```

Tells SLURM you want to use a conda environment

Tells conda that you want to use the `rsem_install` environment

Tells SLURM that you also need to make sure to use Perl (remember this is because we found out that RSEM had a perl dependency, so this is specific to RSEM)

If you ever want to deactivate your environment, usually for the sake of using a different version of the same software you can run the following:

```
conda deactivate rsem_install
```

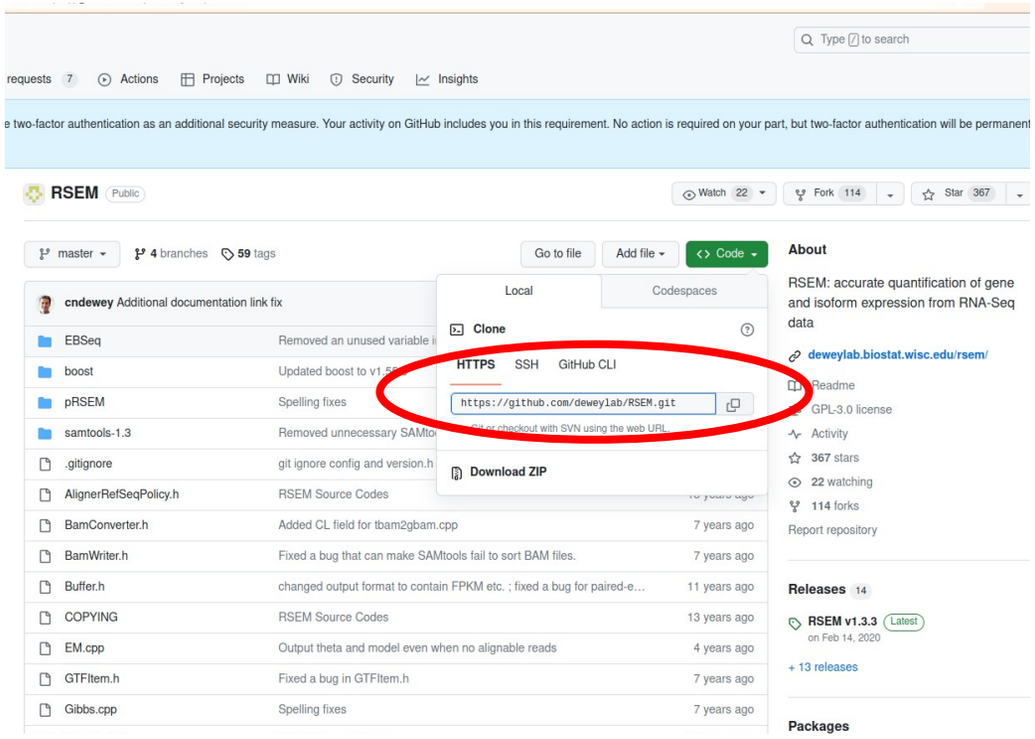
Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – from source**

You can always install your own software from source as well. Just be sure to follow the instructions on their install and note that you do not have any `sudo` privileges so any commands that require `sudo` will not work. 99% of software can be installed from source without `sudo` privileges.

For example, here is RSEM's github page and an example of how to install RSEM from source:

1 Click the green code button and copy the HTTPS github link



2 On Alpine, make sure you are inside of a compute node and not on the head/login node

3 Navigate to your `/projects/$USER` directory

4 Download the source code using `git`

```
git clone https://github.com/deweylab/RSEM.git
```

5 Change into the directory of the dowloaded code (hint: if you don't know the name, you can type `ls` to see what is available in your directory)

```
cd RSEM/
```

Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – from source**

You can always install your own software from source as well. Just be sure to follow the instructions on their install and note that you do not have any `sudo` privileges so any commands that require `sudo` will not work. 99% of software can be installed from source without `sudo` privileges.

For example, here is RSEM's github page and an example of how to install RSEM from source:

6

Now scroll down on their github site and find the prerequisites, compilation and install directions

Compilation & Installation

To compile RSEM, simply run

```
make
```

For Cygwin users, run

```
make cygwin=true
```

To compile EBSeq, which is included in the RSEM package, run

```
make ebseq
```

To install RSEM, simply put the RSEM directory in your environment's PATH variable. Alternatively, run

```
make install
```

By default, RSEM executables are installed to `/usr/local/bin`. You can change the installation location by setting `DESTDIR` and/or `prefix` variables. The RSEM executables will be installed to `${DESTDIR}${prefix}/bin`. The default values of `DESTDIR` and `prefix` are `DESTDIR=` and `prefix=/usr/local`. For example,

```
make install DESTDIR=/home/my_name prefix=/software
```

will install RSEM executables to `/home/my_name/software/bin`.

Note that `make install` does not install `EBSeq` related scripts, such as `rsem-generate-ngvector`, `rsem-run-ebseq`, and `rsem-control-fdr`. But `rsem-generate-data-matrix`, which generates count matrix for differential expression analysis, is installed.

Prerequisites

C++, Perl and R are required to be installed.

To use the `--gff3` option of `rsem-prepare-reference`, Python is also required to be installed.

7

You will notice in the prerequisites, it tells us we must have C++, Perl, and R in order to be installed. Well, lucky for us, these are already installed on Alpine! In order to activate them you need to use `module load` (you can find these in the `module avail` command). There is one exception, in that C++ is already loaded by default into everyone's space, so no need to load that.

```
module load perl
module load R
```

Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – from source**

You can always install your own software from source as well. Just be sure to follow the instructions on their install and note that you do not have any `sudo` privileges so any commands that require `sudo` will not work. 99% of software can be installed from source without `sudo` privileges.

For example, here is RSEM's github page and an example of how to install RSEM from source:

8

When you follow the instructions of the github, it says to run make inside of the RSEM directory:

Compilation & Installation

To compile RSEM, simply run

```
make
```

For Cygwin users, run

```
make cygwin=true
```

To compile EBSeq, which is included in the RSEM package, run

```
make ebseq
```

To install RSEM, simply put the RSEM directory in your environment's PATH variable. Alternatively, run

```
make install
```

By default, RSEM executables are installed to `/usr/local/bin`. You can change the installation location by setting `DESTDIR` and/or `prefix` variables. The RSEM executables will be installed to `${DESTDIR}${prefix}/bin`. The default values of `DESTDIR` and `prefix` are `DESTDIR=` and `prefix=/usr/local`. For example,

```
make install DESTDIR=/home/my_name prefix=/software
```

will install RSEM executables to `/home/my_name/software/bin`.

Note that `make install` does not install `EBSeq` related scripts, such as `rsem-generate-ngvector`, `rsem-run-ebseq`, and `rsem-control-fdr`. But `rsem-generate-data-matrix`, which generates count matrix for differential expression analysis, is installed.

Prerequisites

C++, Perl and R are required to be installed.

To use the `--gff3` option of `rsem-prepare-reference`, Python is also required to be installed.

```
(base) [brunetti@xsede.org@c3cpu-c11-u17-2 RSEM]$ module load R
(base) [brunetti@xsede.org@c3cpu-c11-u17-2 RSEM]$ make
g++ -std=gnu++98 -Wall -I. -I. -Isamtools-1.3/htslib-1.3  -O3 -c -o extractRef.o extractRef.cpp
g++  -o rsem-extract-reference-transcripts extractRef.o
g++ -std=gnu++98 -Wall -I. -I. -Isamtools-1.3/htslib-1.3  -O3 -c -o synthesisRef.o synthesisRef.cpp
g++  -o rsem-synthesis-reference-transcripts synthesisRef.o
g++ -std=gnu++98 -Wall -I. -I. -Isamtools-1.3/htslib-1.3  -O3 -c -o preRef.o preRef.cpp
g++  -o rsem-preref preRef.o
g++ -std=gnu++98 -Wall -I. -I. -Isamtools-1.3/htslib-1.3  -O3 -c -o buildReadIndex.o buildReadIndex.cpp
g++  -o rsem-build-read-index buildReadIndex.o
g++ -std=gnu++98 -Wall -I. -I. -Isamtools-1.3/htslib-1.3  -O3 -ffast-math -c -o simulation.o simulation.cpp
In file included from ./boost/mpl/aux_/na_assert.hpp:23,
                 from ./boost/mpl/arg.hpp:25,
                 from ./boost/mpl/placeholders.hpp:24,
                 from ./boost/mpl/apply.hpp:24,
                 from ./boost/mpl/aux_/iter_apply.hpp:17,
                 from ./boost/mpl/aux_/find_if_pred.hpp:14,
                 from ./boost/mpl/find_if.hpp:17,
                 from ./boost/mpl/find.hpp:17,
                 from ./boost/mpl/aux_/contains_impl.hpp:20,
                 from ./boost/mpl/contains.hpp:20,
                 from ./boost/math/policies/policy.hpp:10,
                 from ./boost/math/tools/precision.hpp:19,
                 from ./boost/math/tools/fraction.hpp:17,
                 from ./boost/math/special_functions/gamma.hpp:18,
                 from ./boost/math/special_functions/detail/bessel_jy.hpp:14,
                 from ./boost/math/special_functions/bessel.hpp:18,
                 from ./boost/math/special_functions/airy.hpp:10,
                 from ./boost/math/special_functions.hpp:15,
                 from ./boost/random/generate_canonical.hpp:22,
                 from boost/random.hpp:52,
                 from simul.h:6,
                 from Orientation.h:8,
                 from SingleModel.h:16,
                 from simulation.cpp:22:
./boost/mpl/assert.hpp:187:21: warning: unnecessary parentheses in declaration of 'assert_arg' [-Wparentheses]
 failed ************ (Pred::************
                     ^
./boost/mpl/assert.hpp:192:21: warning: unnecessary parentheses in declaration of 'assert_not_arg' [-Wparentheses]
 failed ************ (boost::mpl::not_<Pred>::************
                     ^
In file included from ./boost/format/alt_sstream.hpp:20,
                 from ./boost/format/internals.hpp:23,
                 from ./boost/format.hpp:38,
```

All of this stuff is expected! It is going to take a couple minutes and lots of stuff is going to print to your screen

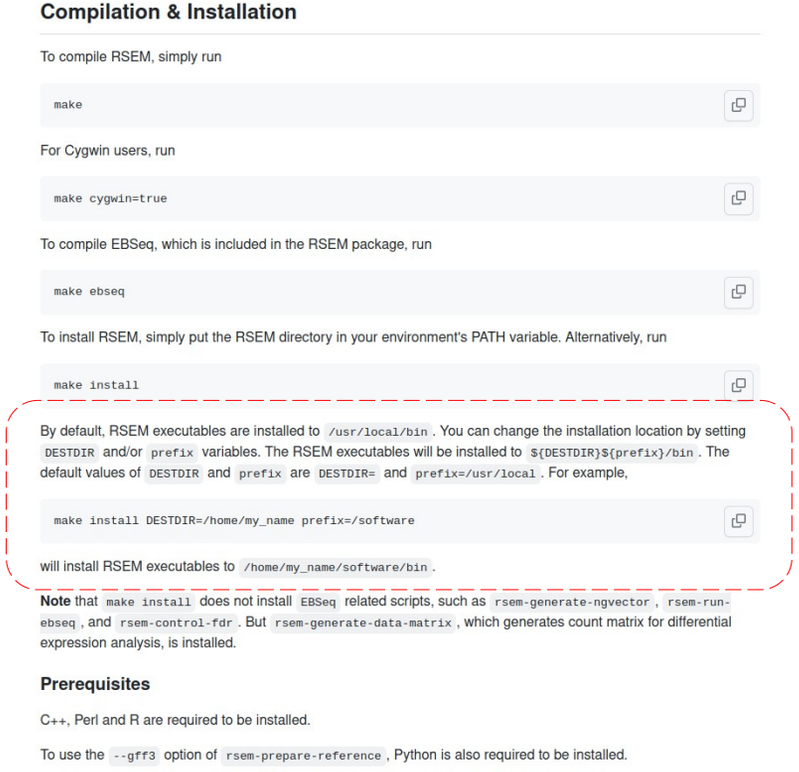
Option A: Things we can do once we are interactively inside of a compute node...

- **Install our own software – from source**

You can always install your own software from source as well. Just be sure to follow the instructions on their install and note that you do not have any `sudo` privileges so any commands that require `sudo` will not work. 99% of software can be installed from source without `sudo` privileges.

For example, here is RSEM's github page and an example of how to install RSEM from source:

9 When you follow the instructions of the github, it says to run make inside of the RSEM directory:



Compilation & Installation

To compile RSEM, simply run

```
make
```

For Cygwin users, run

```
make cygwin=true
```

To compile EBSeq, which is included in the RSEM package, run

```
make ebseq
```

To install RSEM, simply put the RSEM directory in your environment's PATH variable. Alternatively, run

```
make install
```

By default, RSEM executables are installed to `/usr/local/bin`. You can change the installation location by setting `DESTDIR` and/or `prefix` variables. The RSEM executables will be installed to `${DESTDIR}${prefix}/bin`. The default values of `DESTDIR` and `prefix` are `DESTDIR=` and `prefix=/usr/local`. For example,

```
make install DESTDIR=/home/my_name prefix=/software
```

will install RSEM executables to `/home/my_name/software/bin`.

Note that `make install` does not install `EBSeq` related scripts, such as `rsem-generate-ngvector`, `rsem-run-ebseq`, and `rsem-control-fdr`. But `rsem-generate-data-matrix`, which generates count matrix for differential expression analysis, is installed.

Prerequisites

C++, Perl and R are required to be installed.

To use the `--gff3` option of `rsem-prepare-reference`, Python is also required to be installed.

Option A: Things we can do once we are interactively inside of a compute node...

- Check what software is installed on Alpine

- Install our own software

- Run software and code interactively

Option A: Things we can do once we are interactively inside of a compute node...

- **Run software and code interactively**

Once you are in a compute node you can also run code interactively, similar to how you have been doing on our command line. You may have noticed that Alpine also has several programming languages already installed that can be run on the command line and any other software (STAR, cutadapt, fastqc, Python, R, Ruby, Perl, C/C++, Java, Matlab, Mathematica, Julia, etc...)

slurm workload manager

```
module avail
```

```
------------------------------------------------------------------------------ /curc/sw/modules/slurm ------------------------------------------------------------------------------
   StdEnv    curc-quota/latest (D)    slurm/alpine (D)    slurm/blanca    slurm/core    slurmtools (D)

------------------------------------------------------------------------------ /usr/share/lmod/lmod/modulefiles/Core ------------------------------------------------------------------------------
   lmod    settarg

------------------------------------------------------------------------------ Compilers ------------------------------------------------------------------------------
   aocc/3.1.0 (D)    aocc/3.2.0    gcc/10.3.0    gcc/11.2.0 (D)    intel/2022.1.2 (m)    nvhpc_sdk/2022.229    nvhpc_sdk/2023.233 (D)

------------------------------------------------------------------------------ Independent Applications ------------------------------------------------------------------------------
   R/3.6.3                cuda/11.3          (g)    ghostscript/9.56.0          julia/1.6.6                paraview/5.0.1          rclone/1.58.0              tdom/0.9.2            (D)
   R/4.2.2          (D)   cuda/11.4          (g)    git-lfs/3.1.2               julia/1.8.1          (D)   paraview/5.6.0          rhel7for8/1.0              texlive/2021
   allinea/6.0.4    (m)   cuda/11.8          (g)    git/2.31.0                  lftp/4.8.4                 paraview/5.9.0          rocm/5.2.3           (g)   totalview/2016.
06.21
   anaconda/2020.11       cuda/12.1.1        (g,D)  gmsh/2.16.0                 loadbalance/0.2            paraview/5.10.0  (D)    rocm/5.3.0           (g)   ucx/1.10.1
   anaconda/2022.10 (D)   cudnn/8.1          (g)    gmsh/4.11.1           (D)   mambaforge/23.1.0-1        pdtoolkit/3.22             rocm/5.5.0           (g,D) ucx/1.12.1            (D)
   arm-forge/19.1.3 (m)   cudnn/8.2          (g)    gnu_parallel/20160622       mathematica/9.0                                       ruby/2.3.1                 udunits/2.2.20
   autotools/2.69         cudnn/8.6          (g,D)  gnu_parallel/20210322 (D)   mathematica/11.1.0       (D)   perl/5.16.3                ruby/3.0.0           (D)   udunits/2.2.24
   autotools/2.71   (D)   curc-quota/latest        gnuplot/5.4.3               matlab/R2018b                  perl/5.24.0              (D)   singularity/3.6.4 (D)      udunits/2.2.25
   chimerax/1.2.5         dmtcp/2.6.0              idl/8.7                     matlab/R2019b                  perl/5.28.1                    singularity/3.7.4          udunits/2.2.28      (D)
   cmake/3.5.2            eigen/3.4.0              ImageMagick/6.9.12          matlab/R2020b                  perl/5.36.0                    slurmtools/0.0.0           valgrind/3.11.0
   cmake/3.9.2            emacs/25.3               jdk/1.7.0                   matlab/R2021b            (D)   pigz/2.7                       slurmtools/0.0.1           valgrind/3.17.0     (D)
   cmake/3.14.1           emacs/27.2         (D)   jdk/1.8.0_91                matlab/R2022b                  python/2.7.18                  subversion/1.8.16          vapor/3.3.0
   cmake/3.20.2           expat/2.1.1              jdk/1.8.0_281               maven/3.8.1                    python/3.10.2            (D)   subversion/1.10.2          vapor/3.4.0         (D)
   cmake/3.25.0     (D)   expat/2.3.0        (D)   jdk/1.8.0                   ncl/6.3.0                      qchem/4010                     subversion/1.14.1 (D)      vtf3/1.43
   cube/3.4.3             ffmpeg/4.4               jdk/18.0.1.1          (D)   ncl/6.6.2                (D)   qt/5.6.0                       tcltk/8.6.5                zip/rhel7
   cube/4.3.4       (D)   gdb/8.1                  julia/0.6.2                 papi/5.4.3                     qt/5.9.1                       tcltk/8.6.11         (D)
   cuda/11.2        (g)   gdb/10.1           (D)   julia/1.6.0                 papi/5.5.1               (D)   qt/5.15                  (D)   tdom/0.8.3

------------------------------------------------------------------------------ Bioinformatics ------------------------------------------------------------------------------
   alphafold/2.2.0        bbtools/39.01      bowtie2/2.5.0       cutadapt/4.2      homer/4.11      nextflow/22.10.6       plink2/2.00a2.3    sra-toolkit/3.0.0
   alphafold/2.3.1 (D)    bcftools/1.16      bwa/0.7.17          fastqc/0.11.9     htslib/1.16     nextflow/23.04   (D)   qiime2/2023.5      star/2.7.10b
   bamtools/2.5.2         bedtools/2.29.1    cellranger/7.1.0    gatk/4.3.0.0      multiqc/1.14    picard/2.27.5          samtools/1.16.1    trimmomatic/0.39

------------------------------------------------------------------------------ Lmod Internal Modules ------------------------------------------------------------------------------
   StdEnv    lmod    settarg

Where:
  g:  built for GPU
  m:  built for host and native MIC
  D:  Default Module
```

Option A: Things we can do once we are interactively inside of a compute node...

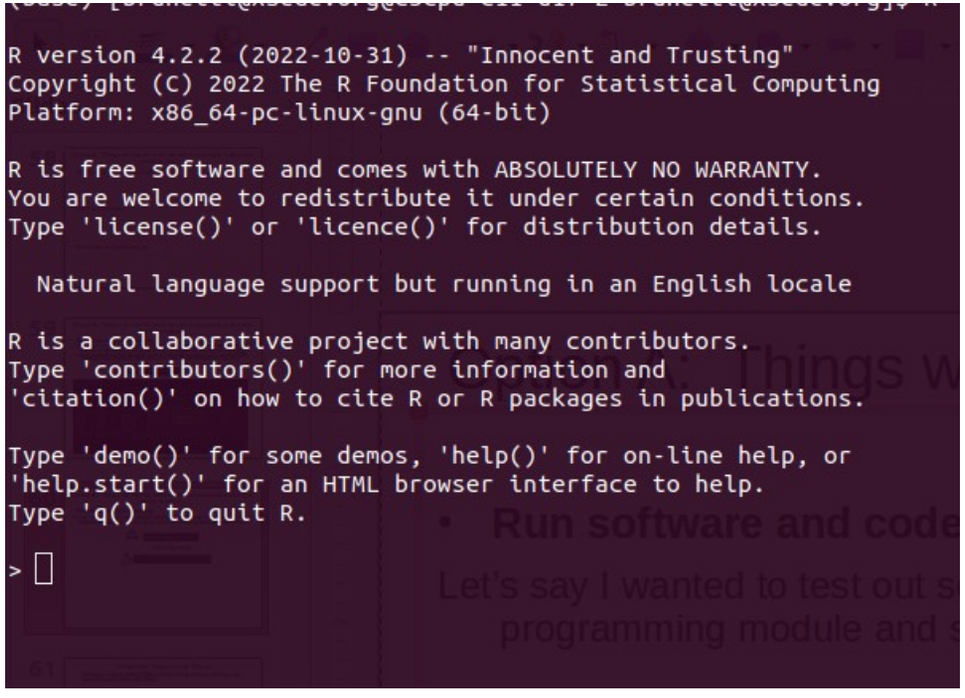
- **Run software and code interactively**

Let's say I wanted to test out some code in R in real-time. I can go inside of a compute node, load the programming module and start it by typing in the name of the programming language/software.

```
module load R/4.2.2
```

Followed by entering:

```
R
```



Now we are in an R session! We can install libraries, load libraries, write code and it will run; I am not going to talk about this much more in that this is really only good for testing since if you close your computer, turn it off etc… it will kill your session.

Now you are on the head/login node. What are your options?

Option A

Make a request to the the head node to login to a compute node so you can interactively run your code on a compute node, test code in real-time, or look at what software is installed on Alpine.

Option B

Stay on the head node because you want to submit a job remotely to a compute node.

Let’s say we wanted to submit a job to Alpine instead of running it interactively. How do we do it?

Let’s say we wanted to submit a job to Alpine instead of running it interactively. How do we do it?

We are going to focus on a simple bioinformatics example, where we want to generate quality reports for next-generation sequencing (NGS) data. This is generally always the first step in any NGS analysis when the output files are of the FASTQ type. The most commonly used software to generate this analysis is called, FastQC.

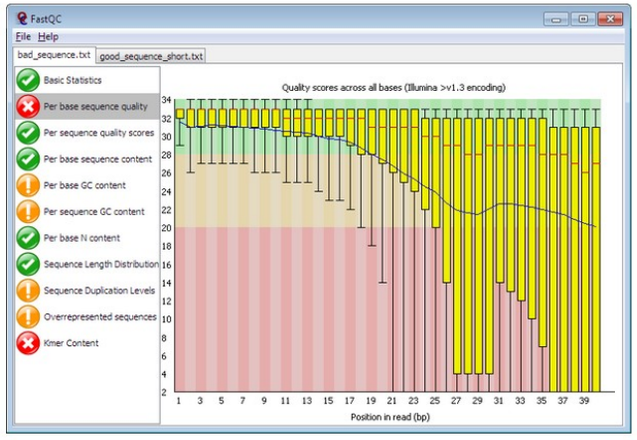
Babraham Bioinformatics

About | People | Services | Projects | Training | Publications

FastQC

Function	A quality control tool for high throughput sequence data.
Language	Java
Requirements	A suitable Java Runtime Environment The Picard BAM/SAM Libraries (included in download)
Code Maturity	Stable. Mature code, but feedback is appreciated.
Code Released	Yes, under GPL v3 or later.
Initial Contact	Simon Andrews

Download Now



Let’s say we wanted to submit a job to Alpine instead of running it interactively. How do we do it?

Task 1: Write a script to run FastQC on all samples

Task 1: Write a script to run FastQC on all samples

Go through our checklist for how we would run FastQC by making a batch script and submitting it to the head node

- [ ] Create a file that ends in .sh and name our script
- [ ] Add a bash/shell directive
- [ ] Add SLURM directives
- [ ] Load modules if needed
- [ ] Set environmental variables
- [ ] Write shell code or command to run
- [ ] Submit script to head node to delegate to compute nodes based on resources requested

Note: the same logic provided in this check list will apply to anything you use on Alpine, so this example can be modified for other scripts you write

Task 1: Write a script to run FastQC on all samples

Go through our checklist for how we would run FastQC by making a batch script and submitting it to the head node

- [ ] Create a file that ends in .sh and name our script
- [ ] Add a bash/shell directive
- [ ] Add SLURM directives
- [ ] Load modules if needed
- [ ] Set environmental variables
- [ ] Write shell code or command to run
- [ ] Submit script to head node to delegate to compute nodes based on resources requested

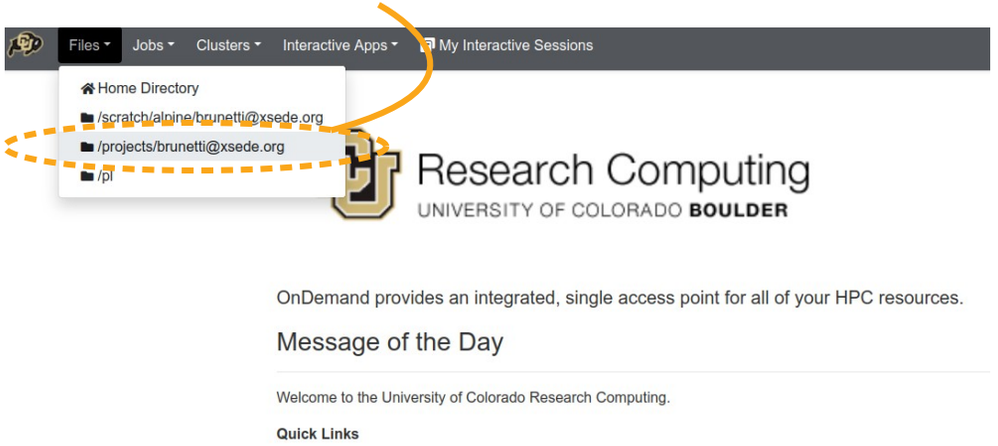
Note: the same logic provided in this check list will apply to anything you use on Alpine, so this example can be modified for other scripts you write

Create a file that ends in .sh and name our script

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

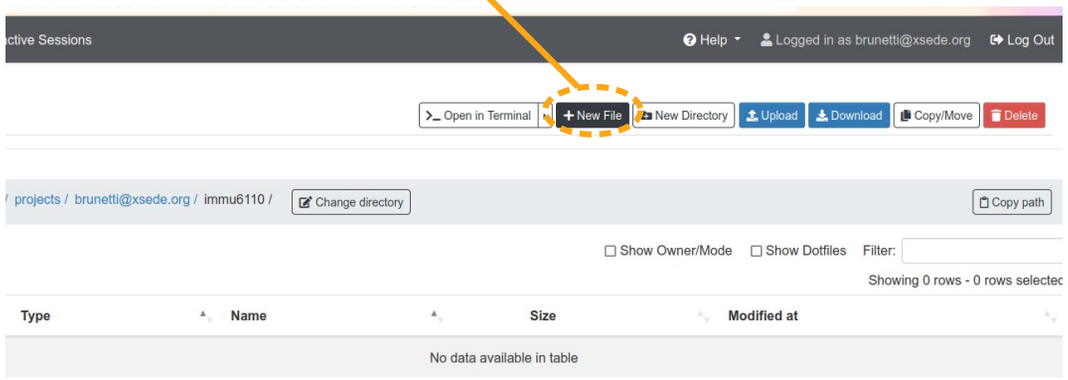
1

Go to your /projects directory (remember this is small, only 250GB of space, so I usually only use this space to save scripts and install software)



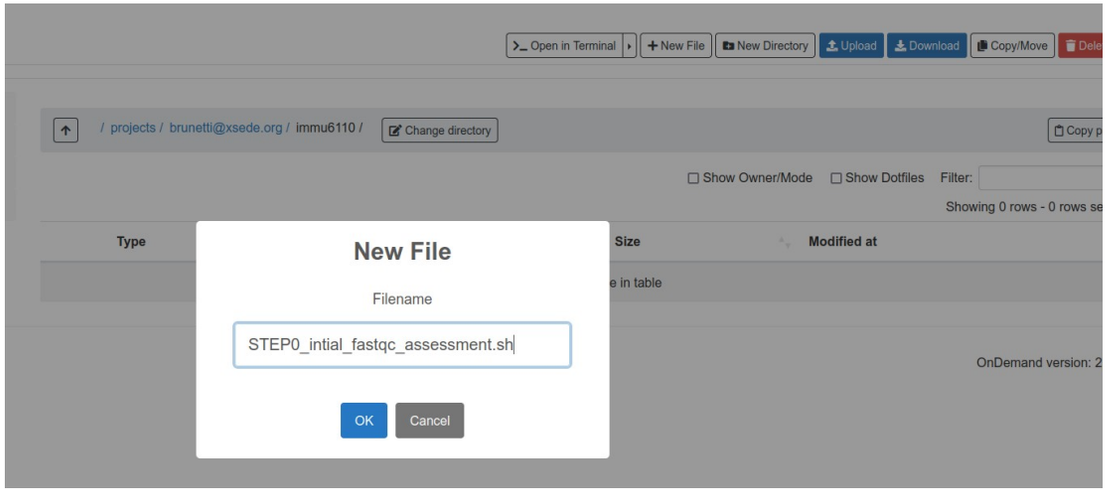
2

Click the “+ New File” button to open an empty text file



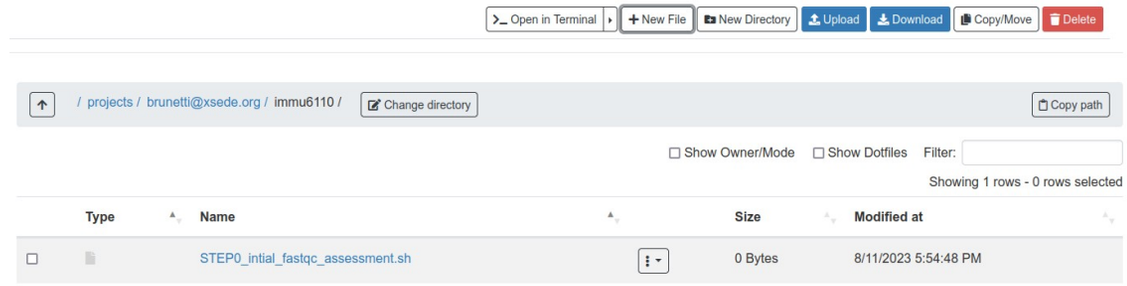
3

You can name your file whatever you want, but I always try to label it with the step of the pipeline, short command description and a .sh ending so I know it is a shell script



4

Now you have created an empty shell file in your /projects/$USER directory on Alpine!



Task 1: Write a script to run FastQC on all samples

Go through our checklist for how we would run FastQC by making a batch script and submitting it to the head node

- [x] Create a file that ends in .sh and name our script
- [ ] Add a bash/shell directive
- [ ] Add SLURM directives
- [ ] Load modules if needed
- [ ] Set environmental variables
- [ ] Write shell code or command to run
- [ ] Submit script to head node to delegate to compute nodes based on resources requested

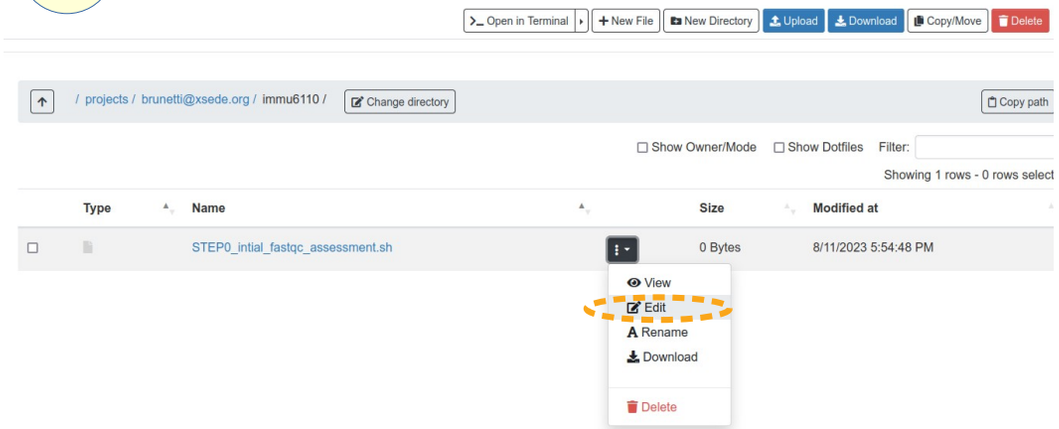
Note: the same logic provided in this check list will apply to anything you use on Alpine, so this example can be modified for other scripts you write

Add a bash/shell directive

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

1

Click on the “Edit” drop down for the empty file we just created

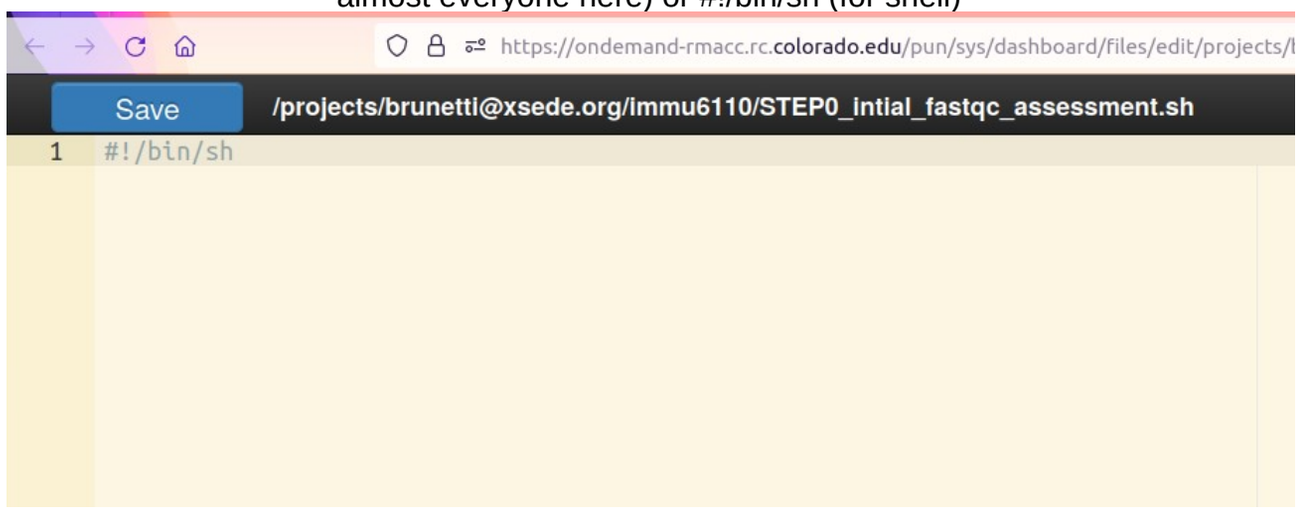


2

This opens up a **text editor**. A text editor is anything that does basic text writing without any formatting (ex: notepad) **NOT Word!!**

3

At the very top of the file add the following:
The #!/bin/sh tells the computer which software language interpreter to use. For Alpine batch scripts, this will almost always be !#/bin/bash (for bash, which is the default for almost everyone here) or #!/bin/sh (for shell)



Task 1: Write a script to run FastQC on all samples

Go through our checklist for how we would run FastQC by making a batch script and submitting it to the head node

- [x] Create a file that ends in .sh and name our script
- [x] Add a bash/shell directive
- [ ] Add SLURM directives
- [ ] Load modules if needed
- [ ] Set environmental variables
- [ ] Write shell code or command to run
- [ ] Submit script to head node to delegate to compute nodes based on resources requested

Note: the same logic provided in this check list will apply to anything you use on Alpine, so this example can be modified for other scripts you write

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

SLURM directives are special commands that allow the head node (which speaks the SLURM language) to take your compute reservation order.

It requires you to know roughly how many resources you need to run your program so it can find different pockets of compute resources that are open/available for use that fit the request you have made.

Inside of a script, these commands are located at the top and start with the string: `#SBATCH` followed by some keyword options.

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=fastqc
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16
```

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Add SLURM directives

If you have experience with text editors on th…lease go ahead and do this in on the command line wi…ow you a

What kind of computer do you need and what size? How long do you you need to use it for?

Save /proje…ssessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=fastqc
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16
```

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=fastqc
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16
```

How many nodes do you want to use?

Always set this to 1 until you understand how to use multiple nodes in the same script – even if you set to more than 1 it won't use it in the way you think without coding it specially to do so using something like openMPI,MPI,etc...

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=fastqc
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16
```

What quality of service do you need?

Most of the time it will be normal, but you can see based on what you need.

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Quality of Service Documentation
https://curc.readthedocs.io/en/latest/running-jobs/job-resources.html#quality-of-service

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8
9
10
11
12
13
14
15
16
```

What quality of service do you need?

Most of the time it will be normal, but you can see based on what you need.

The available QoS's for Alpine are:

QOS name	Description	Max walltime	Max jobs/user	Node limits	Partition lim
normal	Default	1D	tbd	tbd	n/a
long	Longer wall times	7D	tbd	tbd	tbd
mem	High-memory jobs	7D	tbd	12	amem only

- `normal` means you only need the node no longer than 24 hours or 1 day
- `long` means you need to use the node for greater than 24 hours up to 7 days
- `mem` means you need to use a special partition (`amem`) because you need more than 240GB of RAM/memory; many things that are high memory also tend to need longer run times, so you can use this for up to 1 week

Quality of Service Documentation
https://curc.readthedocs.io/en/latest/running-jobs/job-resources.html#quality-of-service

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/in

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
```

If you need a high memory node, you must specify `--partition=amem` and `--qos=mem`

What type of node do you need?

For most bioinformatics tools this will be amilan (CPU nodes) or amem (high memory nodes) but we do have two types of GPU nodes as well

Partition	Description	# of nodes	cores/node	RAM/core (GB)
amilan	AMD Milan (default)	270	64	3.74
ami100	GPU-enabled (3x AMD MI100)	8	64	3.74
aa100	GPU-enabled (3x NVIDIA A100)	12	64	3.74
amem[1]	High-memory	14	48	21.486

Partition (node type) Documentation

https://curc.readthedocs.io/en/latest/running-jobs/job-resources.html#quality-of-service

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=
11 #SBATCH --output=S
12 #SBATCH --error=STEP0_
13 #SBATCH --mail-user=to
14 #SBATCH --mail-type=EN
15
16
```

How long do you need the node?

Format is d-hh:mm:ss

In this case we have it set for 30 minutes. You always want to give yourself a little bit more time than you think you need. I**f you job runs past this max time, the SLURM head node will kill/cancel your job and you will need to run it again.**

If you need to run a job for longer than 24 hours, you will need to set `--qos=long`, warning `--qos=long` will not work unless your time is set to at least 24 hours and 1 minute!

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=f tqc
11 #SBATCH --output=ST
12 #SBATCH --error=STEP0_f
13 #SBATCH --mail-user=ton
14 #SBATCH --mail-type=END
15
16
```

How much memory do you need?

In this case, we have asked for 10G of RAM, however, you can modify and use M(megabytes), G(gigabytes), T(terabytes). Similar to time, you will always want to give yourself a little extra because **as soon as you surpass this amount, SLURM will kill/cancel your job and you will need to resubmit.**

Our amilan nodes have a max RAM of 240GB for the full node; if you need more than that you will need to request a high memory node and set the memory to at least 241G of RAM or it won't work! Our high memory nodes have a max of 1T of RAM

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=fastqc
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16
```

How many CPUs/cores do you want?

If you every see bioinformatics software that has a “threading” or “CPU” option, you will want to set this to match the number or processes you spawn off and/or the number of treads/CPUs you tell the software to use. Our normal amilan nodes have up to 64 cores you can request.

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=fastqc
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16
```

What institute are you a part of?

This is required and it should always be set to `amc-general` for this group as that tells me you have an Anschutz account.

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=fastqc
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16
```

What is the name of your job?

This isn't very useful, as it only appears on the scheduling queue side and only the first 8 letters show up or so. You can put any sting here

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=fastqc
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16
```

Save printed output messages

These two will fields will save all of the printed output on your screen to a file. --output is for printed standard out (basically what 1> does in Linux) and –error will save all of your standard error to a file (basically what 2> does in Linux). You can name these whatever you want but no spaces!

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=fastqc
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16
```

Save printed output messages

These two will fields will save all of the printed output on your screen to a file. --output is for printed standard out (basically what 1> does in Linux) and –error will save all of your standard error to a file (basically what 2> does in Linux). You can name these whatever you want but no spaces!

What is this %J?

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=fastqc
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16
```

Save printed output messages

These two will fields will save all of the printed output on your screen to a file. --output is for printed standard out (basically what 1> does in Linux) and –error will save all of your standard error to a file (basically what 2> does in Linux). You can name these whatever you want but no spaces!

What is this %J?

This %J when listed in a SLURM script #SBATCH directive means it will automatically substitute the unique job ID so you don't risk overwriting these files with the same name. Important for troubleshooting!

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Add SLURM directives

If you have experience with text editors on the command line (nano, vi/vim, emacs), please go ahead and do this in on the command line within Alpine but for users that do not I am going to show you a way to do it in a GUI.

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --qos=normal
5  #SBATCH --partition=amilan
6  #SBATCH --time=00:30:00
7  #SBATCH --mem=10G
8  #SBATCH --ntasks=10
9  #SBATCH --account=amc-general
10 #SBATCH --job-name=fastqc
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16
```

(Optional) Email job status

These lines are completely optional in that you can specify your email address and the SLURM scheduler will email you based on what you select for the –mail-type command. In this case, END means it will email me when my job is finished running.

Some common options include:

- BEGIN
- END
- ALL
- FAIL

SLURM is very sensitive about its directives; there is no space between the # and SBATCH, it is all one word – `#SBATCH`. Also, no space before or after = signs!

Task 1: Write a script to run FastQC on all samples

Go through our checklist for how we would run FastQC by making a batch script and submitting it to the head node

- [x] Create a file that ends in .sh and name our script
- [x] Add a bash/shell directive
- [x] Add SLURM directives
- [ ] Load modules if needed
- [ ] Set environmental variables
- [ ] Write shell code or command to run
- [ ] Submit script to head node to delegate to compute nodes based on resources requested

Note: the same logic provided in this check list will apply to anything you use on Alpine, so this example can be modified for other scripts you write

Refresher: What is a module again?

Refresher: What is a module again?

Remember, a module is just a fancy term to say that a particular piece of software is installed and accessible to the user on a high performance compute cluster.

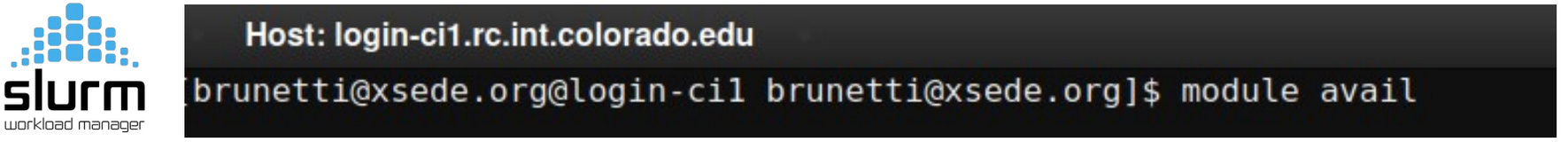
If software you need is listed as a module, it means you don’t need to install it!

How do we know what software is already installed and available for us to use?

Refresher: What is a module again?

Remember, a module is just a fancy term to say that a particular piece of software is installed and accessible to the user on a high performance compute cluster.

If software you need is listed as a module, it means you don’t need to install it!



Refresher: What is a module again?

Remember, a module is just a fancy term to say that a particular piece of software is installed and accessible to the user on a high performance compute cluster.

If software you need is listed as a module, it means you don't need to install it!

```
Host: login-ci1.rc.int.colorado.edu
[brunetti@xsede.org@login-ci1 brunetti@xsede.org]$ module avail
```

```
Host: login-ci1.rc.int.colorado.edu
brunetti@xsede.org@login-ci1 brunetti@xsede.org]$ module avail

------------------------------------------------------------------------------ /curc/sw/modules/slurm -------------------------------
  StdEnv (L)    curc-quota/latest (L)    slurm/alpine (L,D)    slurm/blanca    slurm/core    slurmtools

------------------------------------------------------------------------------ /usr/share/lmod/lmod/modulefiles/Core ----------------------------
  lmod    settarg

 Where:
  L:  Module is loaded
  D:  Default Module

hese modules apply to the login environment only. To see modules
or the compute environment, run module commands from acompile or
n interactive job.
```

Hmmm…. There are not many softwares installed on Alpine…???

Refresher: What is a module again?

Remember, a module is just a fancy term to say that a particular piece of software is installed and accessible to the user on a high performance compute cluster.

If software you need is listed as a module, it means you don't need to install it!

```
Host: login-ci1.rc.int.colorado.edu
[brunetti@xsede.org@login-ci1 brunetti@xsede.org]$ module avail
```

```
Host: login-ci1.rc.int.colorado.edu
brunetti@xsede.org@login-ci1 brunetti@xsede.org]$ module avail

------------------------------------------------------------------------ /curc/sw/modules/slurm ------------------------------------
  StdEnv (L)    curc-quota/latest (L)    slurm/alpine (L,D)    slurm/blanca    slurm/core    slurmtools

---------------------------------------------------------------------- /usr/share/lmod/lmod/modulefiles/Core ----------------------
  lmod    settarg

 Where:
  L:  Module is loaded
  D:  Default Module

hese modules apply to the login environment only. To see modules
or the compute environment, run module commands from acompile or
n interactive job.
```

Hmmm…. There are not many softwares installed on Alpine…???

Are we on a head node or a compute node?

Bonus: what is the difference again?

Let’s access an interactive session so we can get into a compute node to check modules


```
Host: login-ci1.rc.int.colorado.edu
[brunetti@xsede.org@login-ci1 brunetti@xsede.org]$ acompile
```

Let’s access an interactive session so we can get into a compute node to check modules


```
Host: login-ci1.rc.int.colorado.edu
[brunetti@xsede.org@login-ci1 brunetti@xsede.org]$ acompile
```

```
acompile: submitting job... salloc --nodes=1 --partition=acompile --ntasks=1  --time=01:00:00  --qos=compile --job-name=acompile --bell --oversubscribe srun --pty /bin/bash
salloc: Granted job allocation 2736270
salloc: Nodes c3cpu-c9-u3-2 are ready for job
```

You will notice this long command; this is because **`acompile` is just an alias** for that entire SLURM command!

Are we on the head node or a compute node?

Let’s access an interactive session so we can get into a compute node to check modules


```
Host: login-ci1.rc.int.colorado.edu
[brunetti@xsede.org@login-ci1 brunetti@xsede.org]$ acompile
```

```
acompile: submitting job... salloc --nodes=1 --partition=acompile --ntasks=1  --time=01:00:00  --qos=compile --job-name=acompile --bell --oversubscribe srun --pty /bin/bash
salloc: Granted job allocation 2736270
salloc: Nodes c3cpu-c9-u3-2 are ready for job
```

You will notice this long command; this is because **`acompile` is just an alias** for that entire SLURM command!

Are we on the head node or a compute node?

```
[brunetti@xsede.org@c3cpu-c9-u3-2 brunetti@xsede.org]$
```

Let’s access an interactive session so we can get into a compute node to check modules


```
Host: login-ci1.rc.int.colorado.edu
[brunetti@xsede.org@login-ci1 brunetti@xsede.org]$ acompile
```

```
acompile: submitting job... salloc --nodes=1 --partition=acompile --ntasks=1  --time=01:00:00  --qos=compile --job-name=acompile --bell --oversubscribe srun --pty /bin/bash
salloc: Granted job allocation 2736270
salloc: Nodes c3cpu-c9-u3-2 are ready for job
```

You will notice this long command; this is because **`acompile` is just an alias** for that entire SLURM command!

Are we on the head node or a compute node?

```
[brunetti@xsede.org@c3cpu-c9-u3-2 brunetti@xsede.org]$
```

Let’s access an interactive session so we can get into a compute node to check modules

Now that we are in a compute node, let’s see what modules are available now


```
[brunetti@xsede.org@c3cpu-c9-u3-2 brunetti@xsede.org]$ module avail
```

Let’s access an interactive session so we can get into a compute node to check modules

Now that we are in a compute node, let’s see what modules are available now

```
[brunetti@xsede.org@c3cpu-c9-u3-2 brunetti@xsede.org]$ module avail

---------------------------------------------------------------------------- /curc/sw/modules/slurm -----------------------------------------------------------------------------
StdEnv (L)    curc-quota/latest (L,D)    slurm/alpine (L,D)    slurm/blanca    slurm/core    slurmtools (D)

-------------------------------------------------------------------- /usr/share/lmod/lmod/modulefiles/Core --------------------------------------------------------------------
lmod    settarg

------------------------------------------------------------------------------ Compilers ------------------------------------------------------------------------------
aocc/3.1.0 (D)    aocc/3.2.0    gcc/10.3.0    gcc/11.2.0 (D)    intel/2022.1.2 (m)    nvhpc_sdk/2022.229    nvhpc_sdk/2023.233 (D)

---------------------------------------------------------------------- Independent Applications ----------------------------------------------------------------------
R/3.6.3                cuda/12.1.1          (g,D)    gnu_parallel/20210322 (D)    matlab/R2020b           python/3.10.2     (D)    tcltk/8.6.11          (D)
R/4.2.2          (D)   cudnn/8.1            (g)      gnuplot/5.4.3                matlab/R2021b     (D)   qchem/4010                tdom/0.8.3
allinea/6.0.4    (m)   cudnn/8.2            (g)      idl/8.7                      matlab/R2022b           qt/5.6.0                  tdom/0.9.2            (D)
anaconda/2020.11       cudnn/8.6            (g,D)    imagemagick/6.9.12           maven/3.8.1             qt/5.9.1                  texlive/2021
anaconda/2022.10 (D)   curc-quota/latest             jdk/1.7.0                    ncl/6.3.0               qt/5.15           (D)     totalview/2016.06.21
arm-forge/19.1.3 (m)   dmtcp/2.6.0                   jdk/1.8.0_91                 ncl/6.6.2         (D)   rclone/1.58.0             ucx/1.10.1
autotools/2.69         eigen/3.4.0                   jdk/1.8.0_281                papi/5.4.3              rhel7for8/1.0             ucx/1.12.1            (D)
autotools/2.71   (D)   emacs/25.3                    jdk/1.8.0                    papi/5.5.1        (D)   rocm/5.2.3        (g)     udunits/2.2.20
chimerax/1.2.5         emacs/27.2           (D)      jdk/18.0.1.1          (D)    paraview/5.0.1          rocm/5.3.0        (g)     udunits/2.2.24
cmake/3.5.2            expat/2.1.1                   julia/0.6.2                  paraview/5.6.0          rocm/5.5.0        (g,D)   udunits/2.2.25
cmake/3.9.2            expat/2.3.0          (D)      julia/1.6.0                  paraview/5.9.0          ruby/2.3.1                udunits/2.2.28        (D)
cmake/3.14.1           ffmpeg/4.4                    julia/1.6.6                  paraview/5.10.0   (D)   ruby/3.0.0        (D)     valgrind/3.11.0
cmake/3.20.2           gdb/8.1                       julia/1.8.1           (D)    pdtoolkit/3.22          singularity/3.6.4 (D)     valgrind/3.17.0       (D)
cmake/3.25.0     (D)   gdb/10.1             (D)      lftp/4.8.4                   pdtoolkit/3.25.1  (D)   singularity/3.7.4         vapor/3.3.0
cube/3.4.3             ghostscript/9.56.0            loadbalance/0.2              perl/5.16.3             slurmtools/0.0.0          vapor/3.4.0           (D)
cube/4.3.4       (D)   git-lfs/3.1.2                 mambaforge/23.1.0-1          perl/5.24.0       (D)   slurmtools/0.0.1          vtf3/1.43
cuda/11.2        (g)   git/2.31.0                    mathematica/9.0              perl/5.28.1             subversion/1.8.16         zip/rhel7
cuda/11.3        (g)   gmsh/2.16.0                   mathematica/11.1.0    (D)    perl/5.36.0             subversion/1.10.2
cuda/11.4        (g)   gmsh/4.11.1          (D)      matlab/R2018b                pigz/2.7                subversion/1.14.1 (D)
cuda/11.8        (g)   gnu_parallel/20160622         matlab/R2019b                python/2.7.18           tcltk/8.6.5

---------------------------------------------------------------------- Bioinformatics ----------------------------------------------------------------------
alphafold/2.2.0       bbtools/39.01      bowtie2/2.5.0      cutadapt/4.2     homer/4.11      nextflow/22.10.6       plink2/2.00a2.3    sra-toolkit/3.0.0
alphafold/2.3.1 (D)   bcftools/1.16      bwa/0.7.17         fastqc/0.11.9    htslib/1.16     nextflow/23.04   (D)   qiime2/2023.5      star/2.7.10b
bamtools/2.5.2        bedtools/2.29.1    cellranger/7.1.0   gatk/4.3.0.0     multiqc/1.14    picard/2.27.5          samtools/1.16.1    trimmomatic/0.39
```

We want to see if `fastqc` is a software that is already installed on Alpine. Do you see it?

```
------------------------------------------------------------------------------ /curc/sw/modules/slurm ------------------------------------------------------------------------------
StdEnv (L)    curc-quota/latest (L,D)    slurm/alpine (L,D)    slurm/blanca    slurm/core    slurmtools (D)

---------------------------------------------------------------------- /usr/share/lmod/lmod/modulefiles/Core ----------------------------------------------------------------------
lmod    settarg

------------------------------------------------------------------------------ Compilers -------------------------------------------------------------------------------
aocc/3.1.0 (D)    aocc/3.2.0    gcc/10.3.0    gcc/11.2.0 (D)    intel/2022.1.2 (m)    nvhpc_sdk/2022.229    nvhpc_sdk/2023.233 (D)

---------------------------------------------------------------------- Independent Applications -----------------------------------------------------------------------
R/3.6.3                cuda/12.1.1          (g,D)    gnu_parallel/20210322 (D)    matlab/R2020b           python/3.10.2     (D)    tcltk/8.6.11         (D)
R/4.2.2          (D)   cudnn/8.1            (g)      gnuplot/5.4.3                matlab/R2021b     (D)   qchem/4010                tdom/0.8.3
allinea/6.0.4    (m)   cudnn/8.2            (g)      idl/8.7                      matlab/R2022b           qt/5.6.0                  tdom/0.9.2           (D)
anaconda/2020.11       cudnn/8.6            (g,D)    imagemagick/6.9.12           maven/3.8.1             qt/5.9.1                  texlive/2021
anaconda/2022.10 (D)   curc-quota/latest              jdk/1.7.0                    ncl/6.3.0               qt/5.15           (D)     totalview/2016.06.21
arm-forge/19.1.3 (m)   dmtcp/2.6.0                    jdk/1.8.0_91                 ncl/6.6.2         (D)   rclone/1.58.0             ucx/1.10.1
autotools/2.69         eigen/3.4.0                    jdk/1.8.0_281                papi/5.4.3              rhel7for8/1.0             ucx/1.12.1           (D)
autotools/2.71   (D)   emacs/25.3                     jdk/1.8.0                    papi/5.5.1        (D)   rocm/5.2.3        (g)     udunits/2.2.20
chimerax/1.2.5         emacs/27.2           (D)      jdk/18.0.1.1          (D)    paraview/5.0.1          rocm/5.3.0        (g)     udunits/2.2.24
cmake/3.5.2            expat/2.1.1                    julia/0.6.2                  paraview/5.6.0          rocm/5.5.0        (g,D)   udunits/2.2.25
cmake/3.9.2            expat/2.3.0          (D)      julia/1.6.0                  paraview/5.9.0          ruby/2.3.1                udunits/2.2.28       (D)
cmake/3.14.1           ffmpeg/4.4                     julia/1.6.6                  paraview/5.10.0   (D)   ruby/3.0.0        (D)     valgrind/3.11.0
cmake/3.20.2           gdb/8.1                        julia/1.8.1           (D)    pdtoolkit/3.22          singularity/3.6.4 (D)     valgrind/3.17.0      (D)
cmake/3.25.0     (D)   gdb/10.1             (D)      lftp/4.8.4                   pdtoolkit/3.25.1  (D)   singularity/3.7.4         vapor/3.3.0
cube/3.4.3             ghostscript/9.56.0             loadbalance/0.2              perl/5.16.3             slurmtools/0.0.0          vapor/3.4.0          (D)
cube/4.3.4       (D)   git-lfs/3.1.2                  mambaforge/23.1.0-1          perl/5.24.0       (D)   slurmtools/0.0.1          vtf3/1.43
cuda/11.2        (g)   git/2.31.0                     mathematica/9.0              perl/5.28.1             subversion/1.8.16         zip/rhel7
cuda/11.3        (g)   gmsh/2.16.0                    mathematica/11.1.0    (D)    perl/5.36.0             subversion/1.10.2
cuda/11.4        (g)   gmsh/4.11.1          (D)      matlab/R2018b                pigz/2.7                subversion/1.14.1 (D)
cuda/11.8        (g)   gnu_parallel/20160622          matlab/R2019b                python/2.7.18           tcltk/8.6.5

------------------------------------------------------------------------------ Bioinformatics --------------------------------------------------------------------------------
alphafold/2.2.0       bbtools/39.01      bowtie2/2.5.0      cutadapt/4.2      homer/4.11      nextflow/22.10.6       plink2/2.00a2.3    sra-toolkit/3.0.0
alphafold/2.3.1 (D)   bcftools/1.16      bwa/0.7.17         fastqc/0.11.9     htslib/1.16     nextflow/23.04   (D)   qiime2/2023.5      star/2.7.10b
bamtools/2.5.2        bedtools/2.29.1    cellranger/7.1.0   gatk/4.3.0.0      multiqc/1.14    picard/2.27.5          samtools/1.16.1    trimmomatic/0.39
```

You can see that fastqc is listed as a module and the version that is install is version 0.11.9

Let’s add a line of code in our sbatch script to load the `fastqc` software into our environment

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh Key Bindin

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --time=00:30:00
5  #SBATCH --account=amc-general
6  #SBATCH --partition=amilan
7  #SBATCH --qos=normal
8  #SBATCH --ntasks=10
9  #SBATCH --job-name=fastqc
10 #SBATCH --mem=10G
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16 module load fastqc/0.11.9
17
```

Task 1: Write a script to run FastQC on all samples

Go through our checklist for how we would run FastQC by making a batch script and submitting it to the head node

- [x] Create a file that ends in .sh and name our script
- [x] Add a bash/shell directive
- [x] Add SLURM directives
- [x] Load modules if needed
- [ ] Set environmental variables
- [ ] Write shell code or command to run
- [ ] Submit script to head node to delegate to compute nodes based on resources requested

Note: the same logic provided in this check list will apply to anything you use on Alpine, so this example can be modified for other scripts you write

Environmental variables are reserved variable names (usually in all caps) that specify a temporary change in location/path to a directory or software

Here are some common ones you might eventually encounter during your bioinformatics career:

- PATH
- PERL5LIB
- TMPDIR
- PYTHONHOME
- LD_LIBRARY_PATH

You can always see the value these variables by opening a terminal session and printing out their value. In shell/bash, if something is a variable, whether you make it or it is an environmental variable, a `$` must precede the variable name when you want to expand it. Expand means replace the variable name with the value.

```
[brunetti@xsede.org@c3cpu-c9-u3-2 brunetti@xsede.org]$ echo $PATH
/home/brunetti@xsede.org/perl5/bin:/opt/TurboVNC/bin:/usr/lpp/mmfs/bin:/home/brunetti@xsede.org/perl5/bin:/curc/slurm/alpine/scripts:/home/brunetti@xsede.org/perl5/bin:/curc/sw/anaconda3
/2022.10/condabin:/curc/sw/curc-quota/latest/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
[brunetti@xsede.org@c3cpu-c9-u3-2 brunetti@xsede.org]$
```

If you wanted to add a location to your `PATH` when you submit you sbatch script, you can use the export command in Linux and add your path using a `:`

Let's say I want to append my `/projects/$USER` directory to my path, I could add the following line in my script:

```
export PATH=$PATH:/projects/$USER
```

Temporarily overwrite the PATH variable with the following for the life of the current shell session

Extract everything that was in the `$PATH` variable originally

Append the `:/projects/$USER` to the end of the original $PATH; : is the symbol used to separate a list of paths

If you wanted to add a location to your `PATH` when you submit you sbatch script, you can use the export command in Linux and add your path using a `:`

Let's say I want to append my `/projects/$USER` directory to my path, I could add the following line in my script:

```
export PATH=$PATH:/projects/$USER
```

Temporarily overwrite the PATH variable with the following for the life of the current shell session

Extract everything that was in the `$PATH` variable originally

Append the `:/projects/$USER` to the end of the original $PATH; : is the symbol used to separate a list of paths

If you print the contents of your `PATH`, you should see that `/projects/$USER` is appended to the end

```
echo $PATH
```

If you open a new terminal or start a new shell/bash session, any variable that has used the `export` command, will revert back what it was originally before you changed it

Although it is not an environmental variable if you plan on needing to use any conda environments, now would be a good time to add them to your sbatch script. For more information on how to use conda on Alpine, please visit:

https://curc.readthedocs.io/en/latest/software/python.html?highlight=conda

We do not need to set any environmental variables or create any new variables for `fastqc`, therefore, we don’t need to add anything to our script

Task 1: Write a script to run FastQC on all samples

Go through our checklist for how we would run FastQC by making a batch script and submitting it to the head node

- [x] Create a file that ends in .sh and name our script
- [x] Add a bash/shell directive
- [x] Add SLURM directives
- [x] Load modules if needed
- [x] Set environmental variables
- [ ] Write shell code or command to run
- [ ] Submit script to head node to delegate to compute nodes based on resources requested

Note: the same logic provided in this check list will apply to anything you use on Alpine, so this example can be modified for other scripts you write

Now we can write the code to run `fastqc` on one of our fastq files

I know that the command to run `fastqc` is the following when you are using Linux because I use fastqc all the time. You will change this command to reflect the software you want to run


```
/path/to/fastqc --outdir /path/to/myOutputDir/ --threads 1 /path/to/fastqFile1.gz /path/to/fastqFile2.gz
```

But I don't know the `/path/to/fastqc` part?

Now we can write the code to run `fastqc` on one of our fastq files

I know that the command to run `fastqc` is the following when you are using Linux because I use fastqc all the time. You will change this command to reflect the software you want to run


```
/path/to/fastqc --outdir /path/to/myOutputDir/ --threads 1 /path/to/fastqFile1.gz /path/to/fastqFile2.gz
```

But I don't know the `/path/to/fastqc` part?

Yes you do! Anytime you use `module load` all it is doing is taking the long location and adding that location to your `$PATH` variable. Anything in your `$PATH` means if the software name/executable is in one of those locations you can just call it by the software executable name...no full path required! COOL!

Not convinced? Print out your `$PATH` before running `module load fastqc` and then print your `$PATH` after running `module load fastqc`. Can you tell me where the full location of `fastqc` is located on Alpine?

Let’s add it to our script

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh Key Bindings Default Font Size 12px Mode SH Theme

```
#!/bin/sh

#SBATCH --nodes=1
#SBATCH --time=00:30:00
#SBATCH --account=amc-general
#SBATCH --partition=amilan
#SBATCH --qos=normal
#SBATCH --ntasks=10
#SBATCH --job-name=fastqc
#SBATCH --mem=10G
#SBATCH --output=STEP0_fastqc_08112023_%J.log
#SBATCH --error=STEP0_fastqc_08112023_%J.err
#SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
#SBATCH --mail-type=END

module load fastqc/0.11.9

fastqc --outdir /scratch/alpine/$USER/ --threads 10 /pl/active/courses/2023_fall/IMMU6110/data/bulk_RNA_seq/fastq_files/0_min_unstimulated_rep1_1.fastq.gz

```

Let's add it to our script

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh Key Bindings Default Font Size 12px Mode SH Theme

```
#!/bin/sh

#SBATCH --nodes=1
#SBATCH --time=00:30:00
#SBATCH --account=amc-general
#SBATCH --partition=amilan
#SBATCH --qos=normal
#SBATCH --ntasks=10
#SBATCH --job-name=fastqc
#SBATCH --mem=10G
#SBATCH --output=STEP0_fastqc_08112023_%J.log
#SBATCH --error=STEP0_fastqc_08112023_%J.err
#SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
#SBATCH --mail-type=END

module load fastqc/0.11.9

fastqc --outdir /scratch/alpine/$USER/ -threads 10 /pl/active/courses/2023_fall/IMMU6110/data/bulk_RNA_seq/fastq_files/0_min_unstimulated_rep1_1.fastq.gz
```

Remember on Alpine, everything should be output to `/scratch/alpine/$USER` since you have 10TB of free space. You can make directories and subdirectories in there and redirect your output to a directory or subdirectory of your `/scratch/alpine/$USER/` space.

Careful! All data is `/scratch/alpine/$USER` is purged/deleted 90 days after its creation, so be sure to download your results and never have only a single copy of your raw data stored there!

Let’s add it to our script

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh Key Bindings Default Font Size 12px Mode SH Theme

```
#!/bin/sh

#SBATCH --nodes=1
#SBATCH --time=00:30:00
#SBATCH --account=amc-general
#SBATCH --partition=amilan
#SBATCH --qos=normal
#SBATCH --ntasks=10
#SBATCH --job-name=fastqc
#SBATCH --mem=10G
#SBATCH --output=STEP0_fastqc_08112023_%J.log
#SBATCH --error=STEP0_fastqc_08112023_%J.err
#SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
#SBATCH --mail-type=END

module load fastqc/0.11.9

fastqc --outdir /scratch/alpine/$USER/ --threads 10 /pl/active/courses/2023_fall/IMMU6110/data/bulk_RNA_seq/fastq_files/0_min_unstimulated_rep1_1.fastq.gz

```

How did I pick 10 for the `--threads` parameter?

Let’s add it to our script

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh Key Bindings Default Font Size 12px Mode SH Theme

```
#!/bin/sh

#SBATCH --nodes=1
#SBATCH --time=00:30:00
#SBATCH --account=amc-general
#SBATCH --partition=amilan
#SBATCH --qos=normal
#SBATCH --ntasks=10
#SBATCH --job-name=fastqc
#SBATCH --mem=10G
#SBATCH --output=STEP0_fastqc_08112023_%J.log
#SBATCH --error=STEP0_fastqc_08112023_%J.err
#SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
#SBATCH --mail-type=END

module load fastqc/0.11.9

fastqc --outdir /scratch/alpine/$USER/ --threads 10 /pl/active/courses/2023_fall/IMMU6110/data/bulk_RNA_seq/fastq_files/0_min_unstimulated_rep1_1.fastq.gz
```

How did I pick 10 for the `--threads` parameter?

Several softwares offer the ability to speed up computational processes by providing the option to use more cores/CPUs. Although not 100% synonymous most tools use the word “threads” to indicate the number of core processors or CPUs you want the software to use (a bit of a misnomer, but that is for a different day). In this case, recall I have the `#SBATCH` directive `--ntasks=10`, which requests 10 cores of a node, so I might as well have the software use all 10 cores as threads.

Just FYI – most software also hit a wall with CPU, in that it can only be parallelized so much so just because you give it a ton, does not necessarily mean it can use all of it efficiently

Let’s add it to our script

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh Key Bindings Default Font Size 12px Mode SH Theme

```
#!/bin/sh

#SBATCH --nodes=1
#SBATCH --time=00:30:00
#SBATCH --account=amc-general
#SBATCH --partition=amilan
#SBATCH --qos=normal
#SBATCH --ntasks=10
#SBATCH --job-name=fastqc
#SBATCH --mem=10G
#SBATCH --output=STEP0_fastqc_08112023_%J.log
#SBATCH --error=STEP0_fastqc_08112023_%J.err
#SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
#SBATCH --mail-type=END

module load fastqc/0.11.9

fastqc --outdir /scratch/alpine/$USER/ --threads 10 /pl/active/courses/2023_fall/IMMU6110/data/bulk_RNA_seq/fastq_files/0_min_unstimulated_rep1_1.fastq.gz
```

Now, you can ahead and save your script and don’t forget the location where your script is saved!

BUT WAIT!!!

Let's add it to our script

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh Key Bindings Default Font Size 12px Mode SH Theme

```
#!/bin/sh

#SBATCH --nodes=1
#SBATCH --time=00:30:00
#SBATCH --account=amc-general
#SBATCH --partition=amilan
#SBATCH --qos=normal
#SBATCH --ntasks=10
#SBATCH --job-name=fastqc
#SBATCH --mem=10G
#SBATCH --output=STEP0_fastqc_08112023_%J.log
#SBATCH --error=STEP0_fastqc_08112023_%J.err
#SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
#SBATCH --mail-type=END

module load fastqc/0.11.9

fastqc --outdir /scratch/alpine/$USER/ --threads 10 /pl/active/courses/2023_fall/IMMU6110/data/bulk_RNA_seq/fastq_files/0_min_unstimulated_rep1_1.fastq.gz
```

Now, you can ahead and save your script and don't forget the location where your script is saved!

BUT WAIT!!!

Even though this script is perfectly fine, we can do better!
Wouldn't it be nice if we could run multiple samples and steps in the same submission so we don't need to do this over and over?

What did I change here?

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh Key Bindings Default Font Size 12p

```
#!/bin/sh

#SBATCH --nodes=1
#SBATCH --time=00:30:00
#SBATCH --account=amc-general
#SBATCH --partition=amilan
#SBATCH --qos=normal
#SBATCH --ntasks=10
#SBATCH --job-name=fastqc
#SBATCH --mem=10G
#SBATCH --output=STEP0_fastqc_08112023_%J.log
#SBATCH --error=STEP0_fastqc_08112023_%J.err
#SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
#SBATCH --mail-type=END

module load fastqc/0.11.9
module load multiqc/1.14

fastqc --outdir /scratch/alpine/$USER/ --threads 10 /pl/active/courses/2023_fall/IMMU6110/data/bulk_RNA_seq/fastq_files/*.fastq.gz

multiqc /scratch/alpine/$USER/ --filename STEP0_initial_fastqc.html --outdir /scratch/alpine/$USER/

```

What did I change here?

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh Key Bindings Default Font Size 12p

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --time=00:30:00
5  #SBATCH --account=amc-general
6  #SBATCH --partition=amilan
7  #SBATCH --qos=normal
8  #SBATCH --ntasks=10
9  #SBATCH --job-name=fastqc
10 #SBATCH --mem=10G
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16 module load fastqc/0.11.9
17 module load multiqc/1.14
18
19 fastqc --outdir /scratch/alpine/$USER/ --threads 10 /pl/active/courses/2023_fall/IMMU6110/data/bulk_RNA_seq/fastq_files/*.fastq.gz
20
21 multiqc /scratch/alpine/$USER/ --filename STEP0_initial_fastqc.html --outdir /scratch/alpine/$USER/
22
```

1

I am telling Alpine I also want to load in the `multiqc` software to my space. Who remembers how I know how to fill out the name and version here?

Follow up: Which node do I need to be on to look at software installs? i.e. head or compute?

What did I change here?

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh Key Bindings Default Font Size 12p

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --time=00:30:00
5  #SBATCH --account=amc-general
6  #SBATCH --partition=amilan
7  #SBATCH --qos=normal
8  #SBATCH --ntasks=10
9  #SBATCH --job-name=fastqc
10 #SBATCH --mem=10G
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16 module load fastqc/0.11.9
17 module load multiqc/1.14
18
19 fastqc --outdir /scratch/alpine/$USER/ --threads 10 /pl/active/courses/2023_fall/IMMU6110/data/bulk_RNA_seq/fastq_files/*.fastq.gz
20
21 multiqc /scratch/alpine/$USER/ --filename STEP0_initial_fastqc.html --outdir /scratch/alpine/$USER/
22
```

1

I am telling Alpine I also want to load in the `multiqc` software to my space. Who remembers how I know how to fill out the name and version here? *Hint: start at slide 52*

Follow up: Which node do I need to be on to look at software installs? i.e. head or compute?

2

`fastqc` is a special software where you can use a regex to grab all all the files and process them in parallel.

This is not the usual case for most software. For example, you cannot do this with `cutadapt`, or alignment software (`STAR`, `Rsubread`), etc… It is software dependent so look up the software if you are not sure.

What did I change here?

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh Key Bindings Default Font Size 12p

```
1  #!/bin/sh
2
3  #SBATCH --nodes=1
4  #SBATCH --time=00:30:00
5  #SBATCH --account=amc-general
6  #SBATCH --partition=amilan
7  #SBATCH --qos=normal
8  #SBATCH --ntasks=10
9  #SBATCH --job-name=fastqc
10 #SBATCH --mem=10G
11 #SBATCH --output=STEP0_fastqc_08112023_%J.log
12 #SBATCH --error=STEP0_fastqc_08112023_%J.err
13 #SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
14 #SBATCH --mail-type=END
15
16 module load fastqc/0.11.9
17 module load multiqc/1.14
18
19 fastqc --outdir /scratch/alpine/$USER/ --threads 10 /pl/active/courses/2023_fall/IMMU6110/data/bulk_RNA_seq/fastq_files/*.fastq.gz
20
21 multiqc /scratch/alpine/$USER/ --filename STEP0_initial_fastqc.html --outdir /scratch/alpine/$USER/
22
```

1

I am telling Alpine I also want to load in the `multiqc` software to my space. Who remembers how I know how to fill out the name and version here? *Hint: start at slide 52*

Follow up: Which node do I need to be on to look at software installs? i.e. head or compute?

2

`fastqc` is a special software where you can use a regex to grab all all the files and process them in parallel.

3

I added a `multiqc` command. Notice, on Alpine, none of the commands themselves change; you can still use the software in the same way you have always used it.

What Alpine will do here, is execute your `fastqc` command first and then wait until it is finished before it runs the `multiqc` command.

This is not the usual case for most software. For example, you cannot do this with `cutadapt`, or alignment software (`STAR`, `Rsubread`), etc… It is software dependent so look up the software if you are not sure.

What did I change here?

Save /projects/brunetti@xsede.org/immu6110/STEP0_intial_fastqc_assessment.sh Key Bindings Default Font Size 12p

```
#!/bin/sh

#SBATCH --nodes=1
#SBATCH --time=00:30:00
#SBATCH --account=amc-general
#SBATCH --partition=amilan
#SBATCH --qos=normal
#SBATCH --ntasks=10
#SBATCH --job-name=fastqc
#SBATCH --mem=10G
#SBATCH --output=STEP0_fastqc_08112023_%J.log
#SBATCH --error=STEP0_fastqc_08112023_%J.err
#SBATCH --mail-user=tonya.brunetti@cuanschutz.edu
#SBATCH --mail-type=END

module load fastqc/0.11.9
module load multiqc/1.14

fastqc --outdir /scratch/alpine/$USER/ --threads 10 /pl/active/courses/2023_fall/IMMU6110/data/bulk_RNA_seq/fastq_files/*.fastq.gz

multiqc /scratch/alpine/$USER/ --filename STEP0_initial_fastqc.html --outdir /scratch/alpine/$USER/

```

Be warned! Every command you run needs to finish within the resources you requested. If your memory is exceeded, or it take longer than the time you requested, Alpine will kill your job and you will need to re-run the parts that didn't finish.

Task 1: Write a script to run FastQC on all samples

Go through our checklist for how we would run FastQC by making a batch script and submitting it to the head node

- [x] Create a file that ends in .sh and name our script
- [x] Add a bash/shell directive
- [x] Add SLURM directives
- [x] Load modules if needed
- [x] Set environmental variables
- [x] Write shell code or command to run
- [ ] Submit script to head node to delegate to compute nodes based on resources requested

Note: the same logic provided in this check list will apply to anything you use on Alpine, so this example can be modified for other scripts you write

We can submit our script using `sbatch`

Go back to your terminal and navigate to the directory of where your saved sbatch script is located

This location is where your `.err` and `.log` files will be generated unless you change the path in the #SBATCH directive

Now, you can submit your script using `sbatch`

```
sbatch STEP0_initial_fastq_assessment.sh
```

Now you can turn off your computer, log off, do whatever you want until your job finishes; your computer power and any Alpine session at this point is not tied to this job as it has been sent to the remote compute nodes in Boulder

We just submitted our first job!

Go through our checklist for how we would run FastQC by making a batch script and submitting it to the head node

- ✓ Create a file that ends in .sh and name our script
- ✓ Add a bash/shell directive
- ✓ Add SLURM directives
- ✓ Load modules if needed
- ✓ Set environmental variables
- ✓ Write shell code or command to run
- ✓ Submit script to head node to delegate to compute nodes based on resources requested

Note: the same logic provided in this check list will apply to anything you use on Alpine, so this example can be modified for other scripts you write

How can we check the status of resources our job is using while it is running?

When you submitted your job, the SLURM scheduler gave you a confirmation message that also contained your job id:


```
sbatch STEP0_initial_fastq_assessment.sh
```

```
Submitted batch job 2789588
```

How can we check the status of resources our job is using while it is running?

When you submitted your job, the SLURM scheduler gave you a confirmation message that also contained your job id:


```
sbatch STEP0_initial_fastq_assessment.sh
```

```
Submitted batch job 2789588
```

This is your job ID

How can we check the status of resources our job is using while it is running?

When you submitted your job, the SLURM scheduler gave you a confirmation message that also contained your job id:

```
sbatch STEP0_initial_fastq_assessment.sh
Submitted batch job 2789588
```

This is your job ID

```
sstat -j 2789588.batch -o JobID,AveRSS,MaxRSS,MinCPU
```

How can we check the status of resources our job is using while it is running?

When you submitted your job, the SLURM scheduler gave you a confirmation message that also contained your job id:

```
sbatch STEP0_initial_fastq_assessment.sh
Submitted batch job 2789588
```

This is your job ID

```
sstat -j 2789588.batch -o JobID,AveRSS,MaxRSS,MinCPU
```

```
JobID                AveRSS      MaxRSS      MinCPU
------------ ---------- ---------- ----------
2789588.bat+        528K       528K   00:00:00
```

How can we check the status of resources our job is using while it is running?

When you submitted your job, the SLURM scheduler gave you a confirmation message that also contained your job id:

```
sbatch STEP0_initial_fastq_assessment.sh
Submitted batch job 2789588
```

This is your job ID

```
sstat -j 2789588.batch -o JobID,AveRSS,MaxRSS,MinCPU
```

```
JobID          AveRSS      MaxRSS      MinCPU
------------ ---------- ---------- ----------
2789588.bat+        528K        528K 00:00:00
```

Average memory (RAM) job has used so far since the start of the job until its most recent sampling (sampled every few seconds)

Maximum memory (RAM) job has used so far since the start of the job until its most recent sampling (sampled every few seconds)

How can we check the status of resources our job is using while it is running?

When you submitted your job, the SLURM scheduler gave you a confirmation message that also contained your job id:

```
sbatch STEP0_initial_fastq_assessment.sh
Submitted batch job 2789588
```

This is your job ID

```
sstat -j 2789588.batch -o JobID,AveRSS,MaxRSS,MinCPU
```

```
JobID              AveRSS      MaxRSS      MinCPU
------------ ---------- ---------- ----------
2789588.bat+        528K       528K   00:00:00
```

The total amount of CPU time your job has used since the start of the job until its most recent sampling

How can we check the amount of resources our job used after it has finished?

```
sacct -j 2789588
```

```
JobID           JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
2789588            fastqc     amilan amc-gener+         15  COMPLETED      0:0
2789588.bat+        batch            amc-gener+         15  COMPLETED      0:0
2789588.ext+       extern            amc-gener+         15  COMPLETED      0:0
```

How can we check the amount of resources our job used after it has finished?


```
sacct -j 2789588.batch -o JobID,Elapsed,AveRSS,MaxRSS,MinCPU
```

How can we check the amount of resources our job used after it has finished?


```
sacct -j 2789588.batch -o JobID,Elapsed,AveRSS,MaxRSS,MinCPU
```

Notice how we now append the string, `.batch` to the job ID in order to get the batch job information

Notice, how this is a comma-separated list, with NO SPACES between them!

How can we check the amount of resources our job used after it has finished?

```
sacct -j 2789588.batch -o JobID,Elapsed,AveRSS,MaxRSS,MinCPU
```

Notice how we now append the string, `.batch` to the job ID in order to get the batch job information

Notice, how this is a comma-separated list, with NO SPACES between them!

```
JobID           Elapsed    AveRSS     MaxRSS      MinCPU
------------ ---------- ---------- ---------- ----------
2789588.bat+   00:03:24    2857636K   2857636K   00:21:59
```

The total time from the time your job started to time your job ended to complete. This is called the walltime and is essentially if you took a stop watch and started it when the job started and stopped it when the job completed. In this case, it to 3 minutes and 24 seconds

What does this mean?

This is the amount of CPU time used based on the start of your job through when it completed. In this case, I used 21 minutes and 59 seconds of CPU time.

How can we check the amount of resources our job used after it has finished?

Alternatively, after you that the job has completed, you can use the following:

```
seff 2789588
```

```
Job ID: 2789588
Cluster: alpine
User/Group: brunetti@xsede.org/brunettipgrp@xsede.org
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 10
CPU Utilized: 00:34:27
CPU Efficiency: 43.15% of 01:19:50 core-walltime
Job Wall-clock time: 00:07:59
Memory Utilized: 1.17 GB
Memory Efficiency: 11.70% of 10.00 GB
```

How can we check the amount of resources our job used after it has finished?

Alternatively, after you that the job has completed, you can use the following:

```
seff 2789588
```

This refers to how much CPU time it took. Recall, we provided multiple threads to help parallelize the call. Although not 100% accurate, you can think of the core-wall-time the time it would take if you didn't parallelize your job

```
Job ID: 2789588
Cluster: alpine
User/Group: brunetti@xsede.org/brunettipgrp@xsede.org
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 10
CPU Utilized: 00:34:27
CPU Efficiency: 43.15% of 01:19:50 core-walltime
Job Wall-clock time: 00:07:59
Memory Utilized: 1.17 GB
Memory Efficiency: 11.70% of 10.00 GB
```

Refers to the real-time (or often referred to as Wall-clock time/Wall time) in the sense if you started a stop watch as soon as Alpine started your job and timed it until it completed, this is how long it took.

ALWAYS check your `.err` and `.log` files to make sure the job completed! You can't always rely on the SLURM job status

Go back to either Lecture2_introduction_to_linux_bash_shell.pdf or your Linux cheat sheet to figure out how to look at these files on the command line; all of those Linux commands will work on the Alpine command line.

Hints for looking through `.log` and `.err` files; especially ones that are very long!

- `grep` is your absolute best friend!
  - You will learn over time that certain software and programming languages write out string patterns when errors are encountered and your program terminates unexpectedly
- `tail` can give you a good indicator of what may be the most recent messages written to your `.err` and `.log` files
- `less` can let you scroll and find patterns within your `.log` and `.err` files

If your jobs fails, these two logs will provide you with hints and error messages as to how to fix your script or code

`grep` hints

Replace `fileName` with nameOfYourLogFile.log or nameOfYourErrorFile.err

```
grep –i “error” fileName
```

```
grep –i “err” fileName
```

```
grep –i “traceback” fileName
```

```
grep –i “except” fileName
```

```
grep –i “fail” fileName
```

```
grep –i “warn” fileName
```

Consider also adding the `–A` flag with grep to print x number of lines following a match. Ex: `grep –i –A 3 “error” fileName`, also, use `less` and/or `more` to your advantage as well!

Submit the cutadapt job

Go to the Alpine terminal and navigate to the directory where your cutadapt script was saved; for me it is the following (keep in mind it may be different for you!):

```
cd /projects/${USER}/immu6110/
```

Now you can submit your script using `sbatch` followed by the name of your script which may be different from mine! (all of your log and error files will go to the location you submitted your job from)


```
sbatch STEP1_cutadapt.sh
```

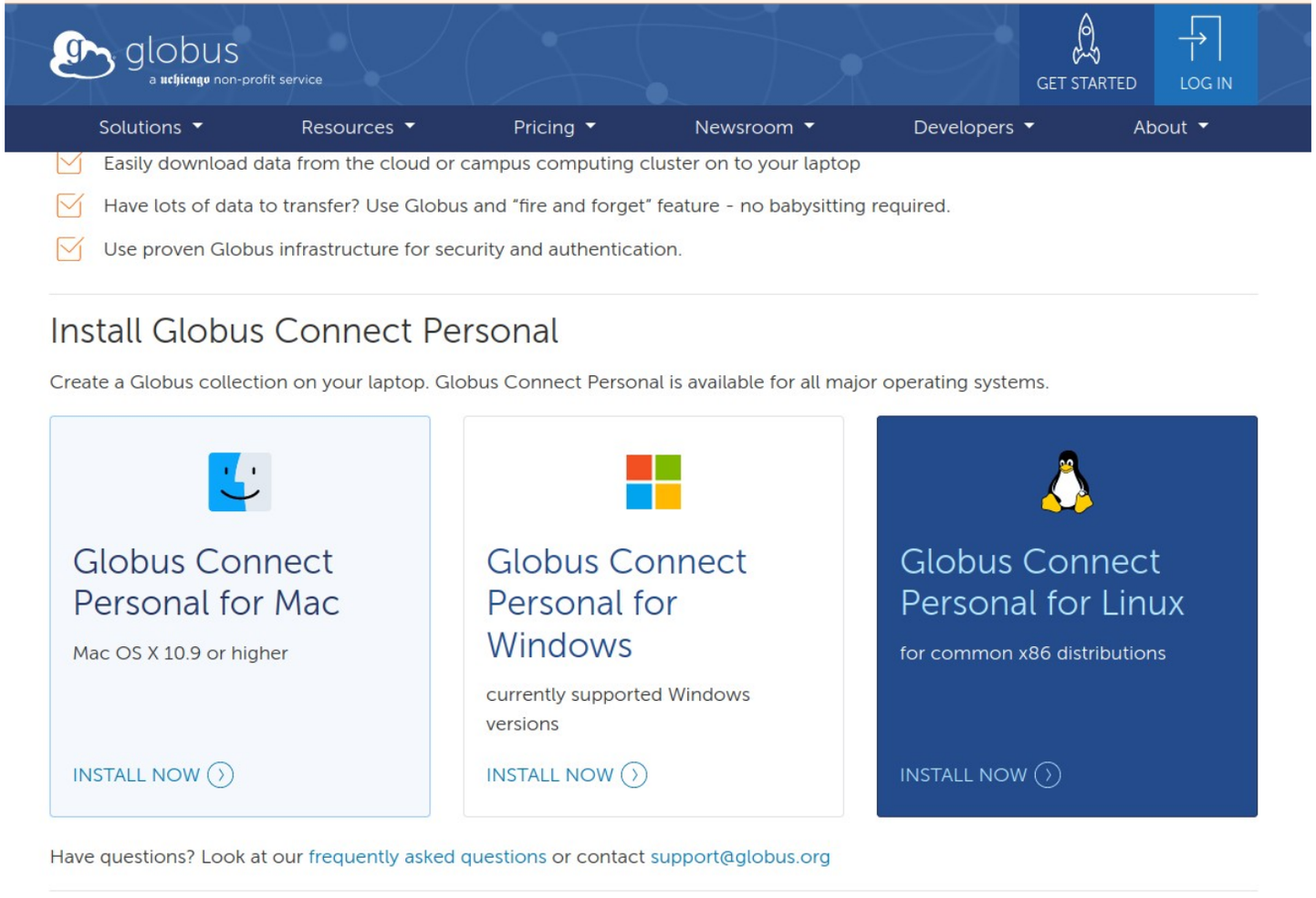
Now you can turn off your computer, log off, do whatever you want until your job finishes; your computer power and any Alpine session at this point is not tied to this job as it has been sent to the remote compute nodes in Boulder

Large Data Transfers with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus

Install globus connect personal first and then connect to it from your local computer

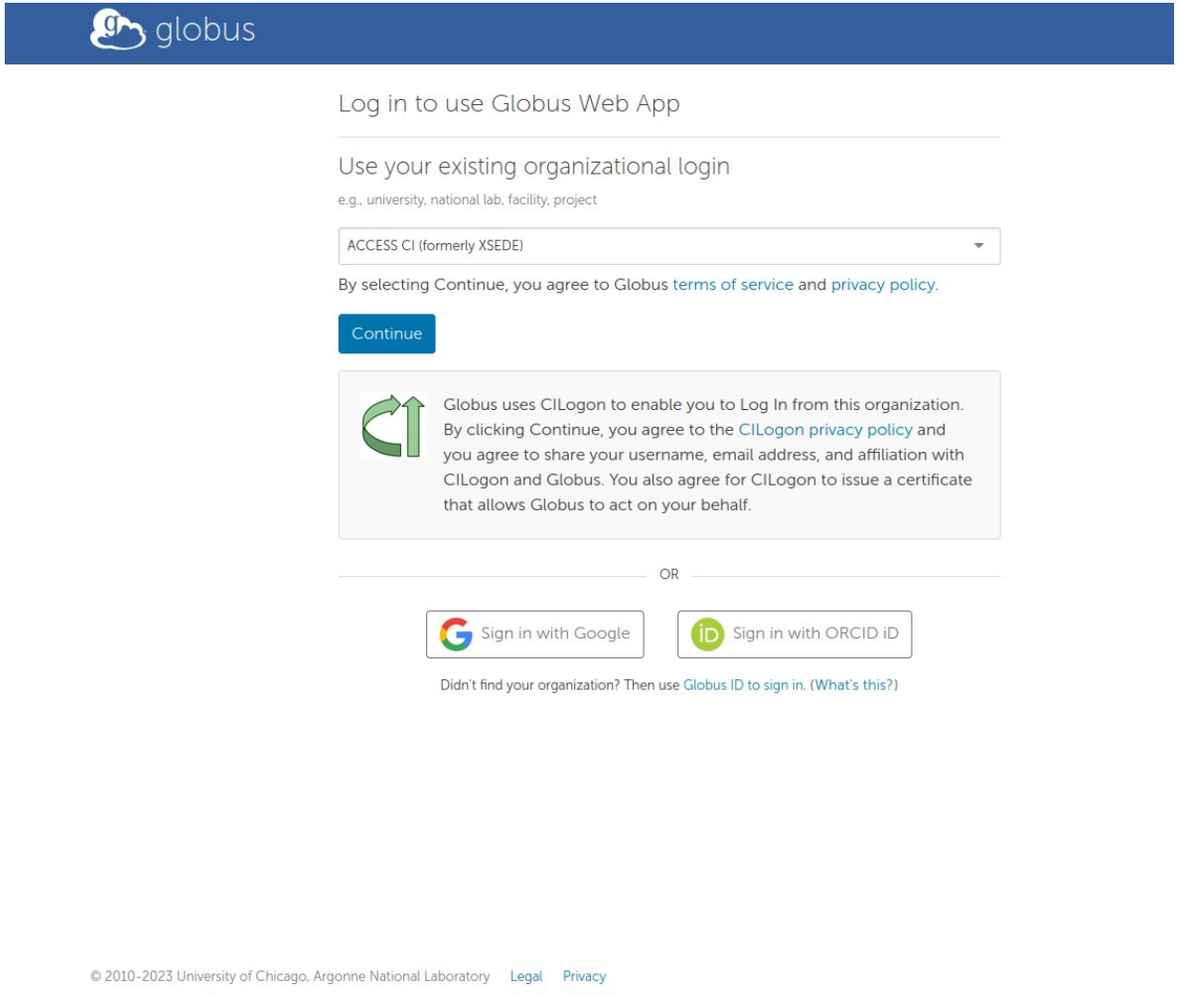
https://www.globus.org/globus-connect-personal



Data Transfer with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus

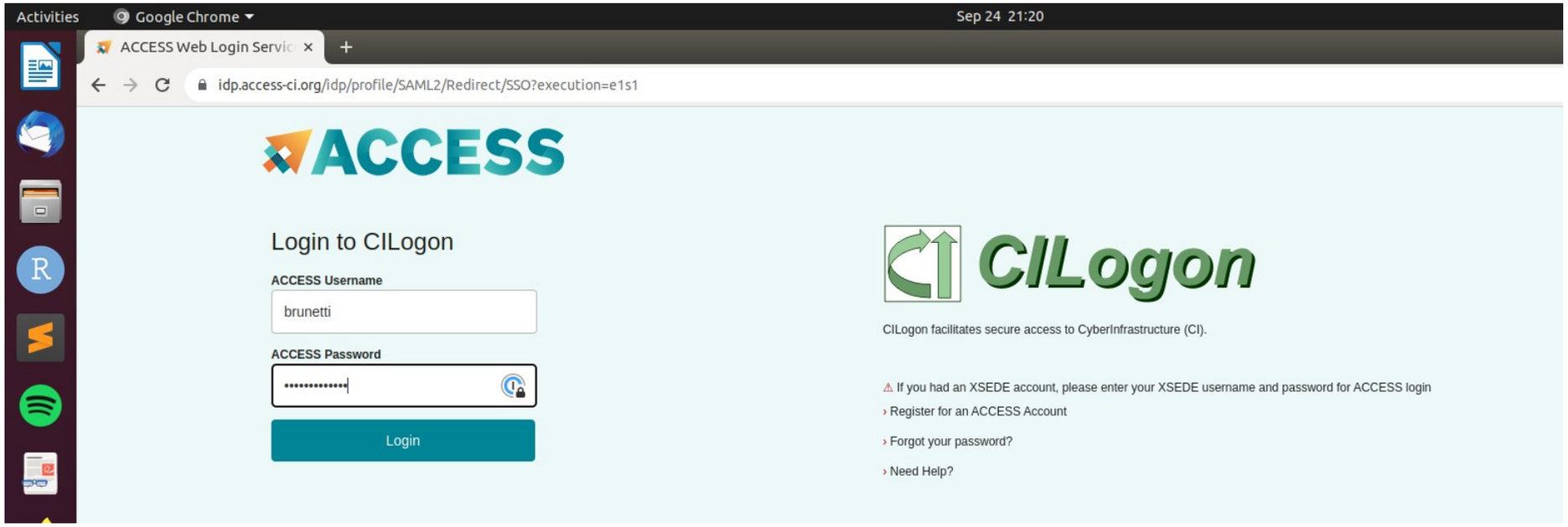
Go to: https://auth.globus.org



Data Transfer with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus

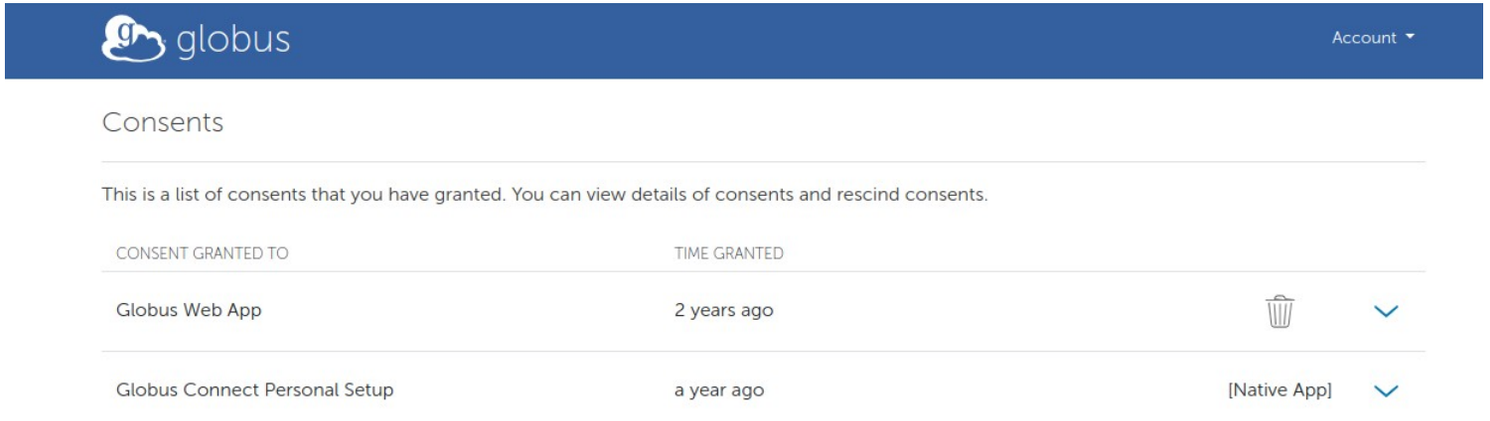
You should receive a DUO push once you login and be sure to accept it



Data Transfer with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus

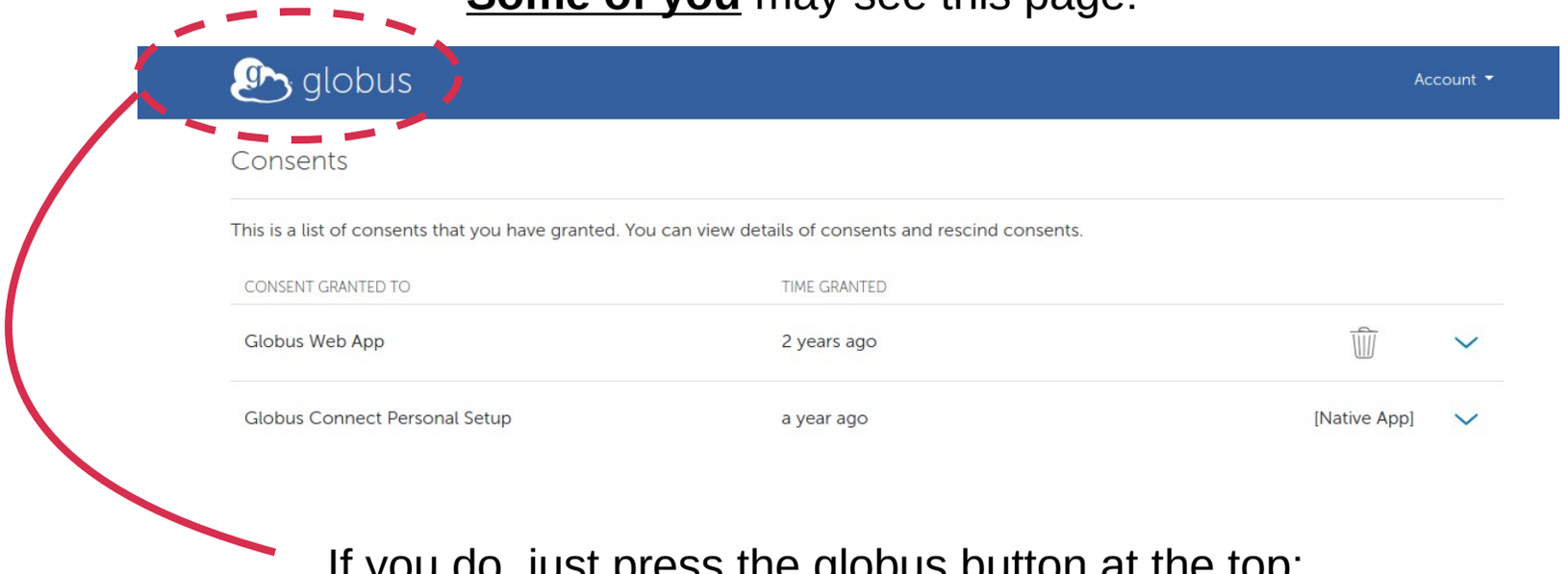
<u>Some of you</u> may see this page:



Data Transfer with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus

<u>Some of you</u> may see this page:

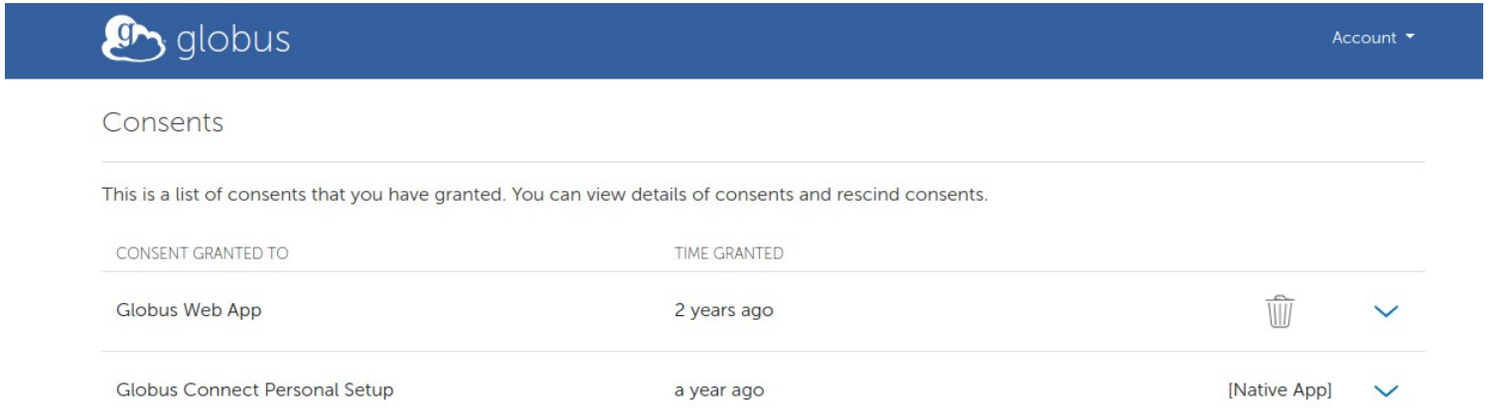


If you do, just press the globus button at the top:

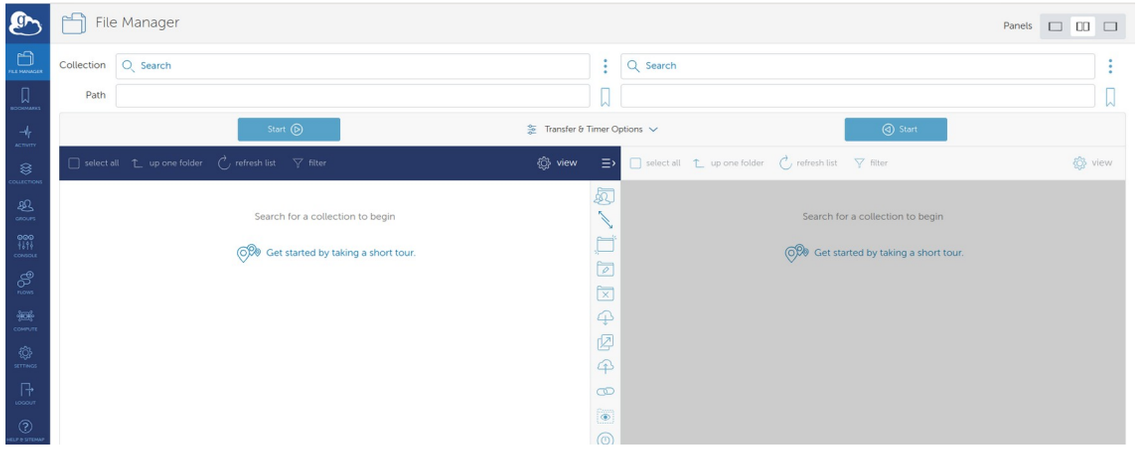
Data Transfer with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus

<u>Some of you</u> may see this page:

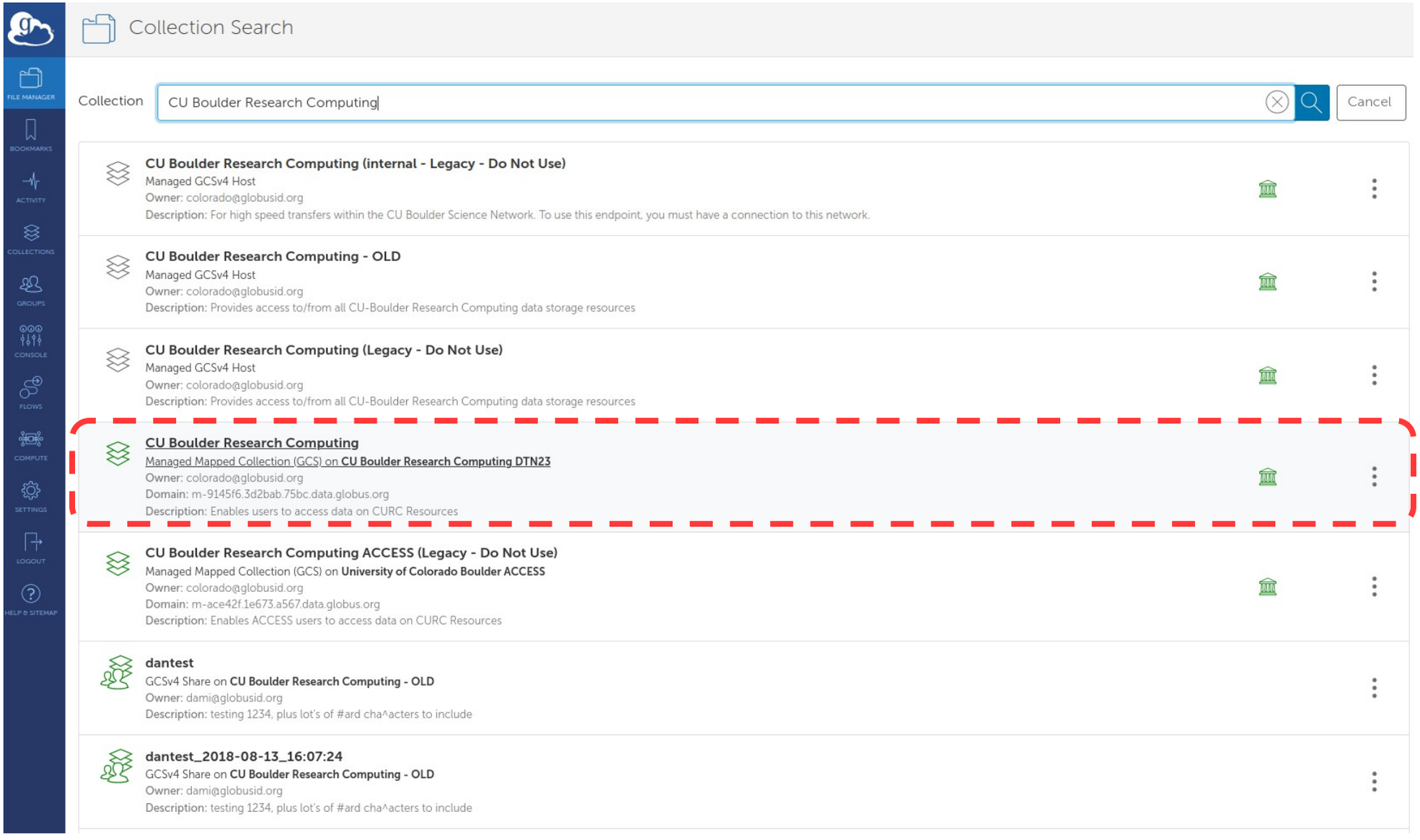


This will take you to this page and then press login:



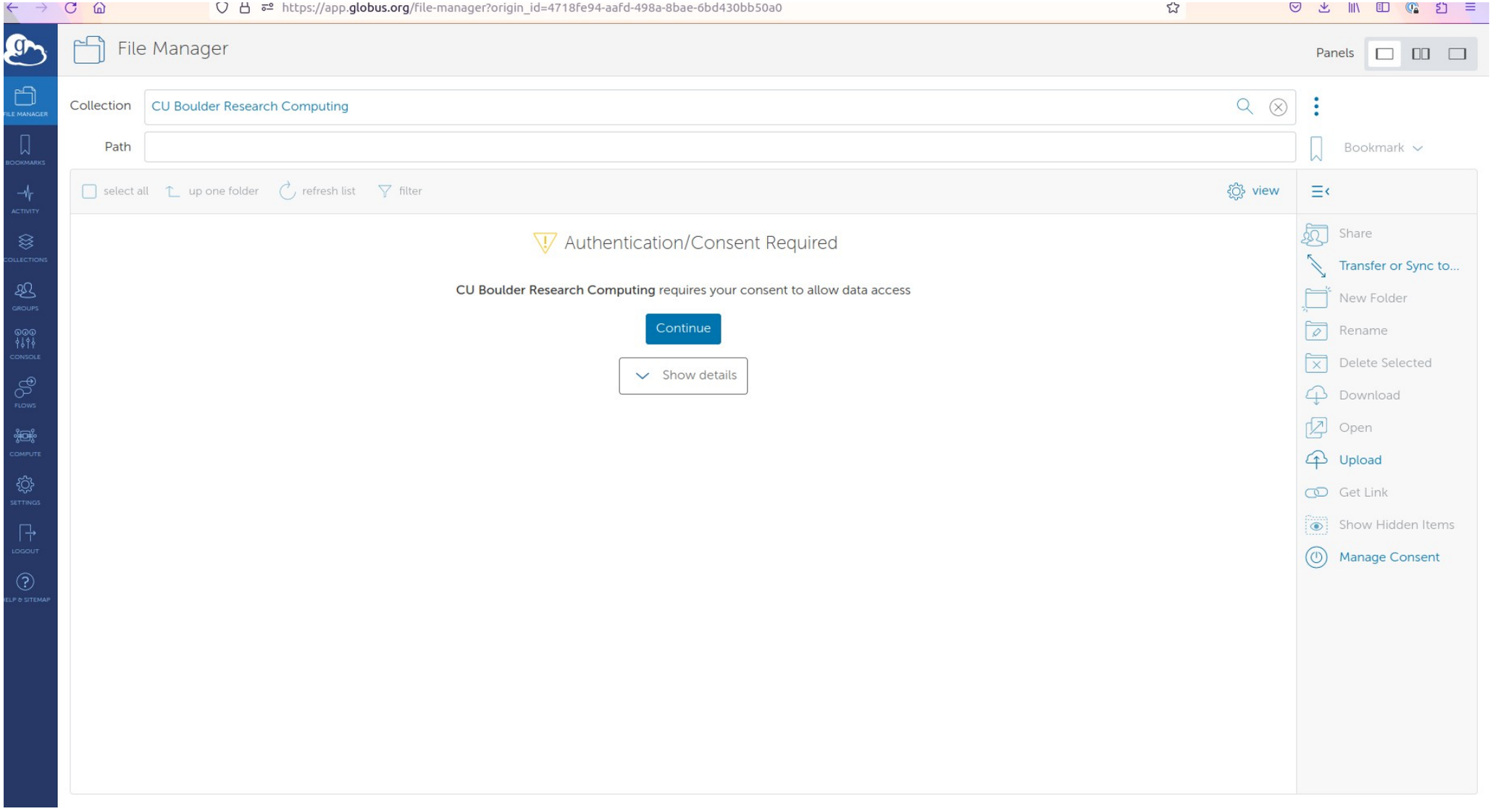
Data Transfer with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus



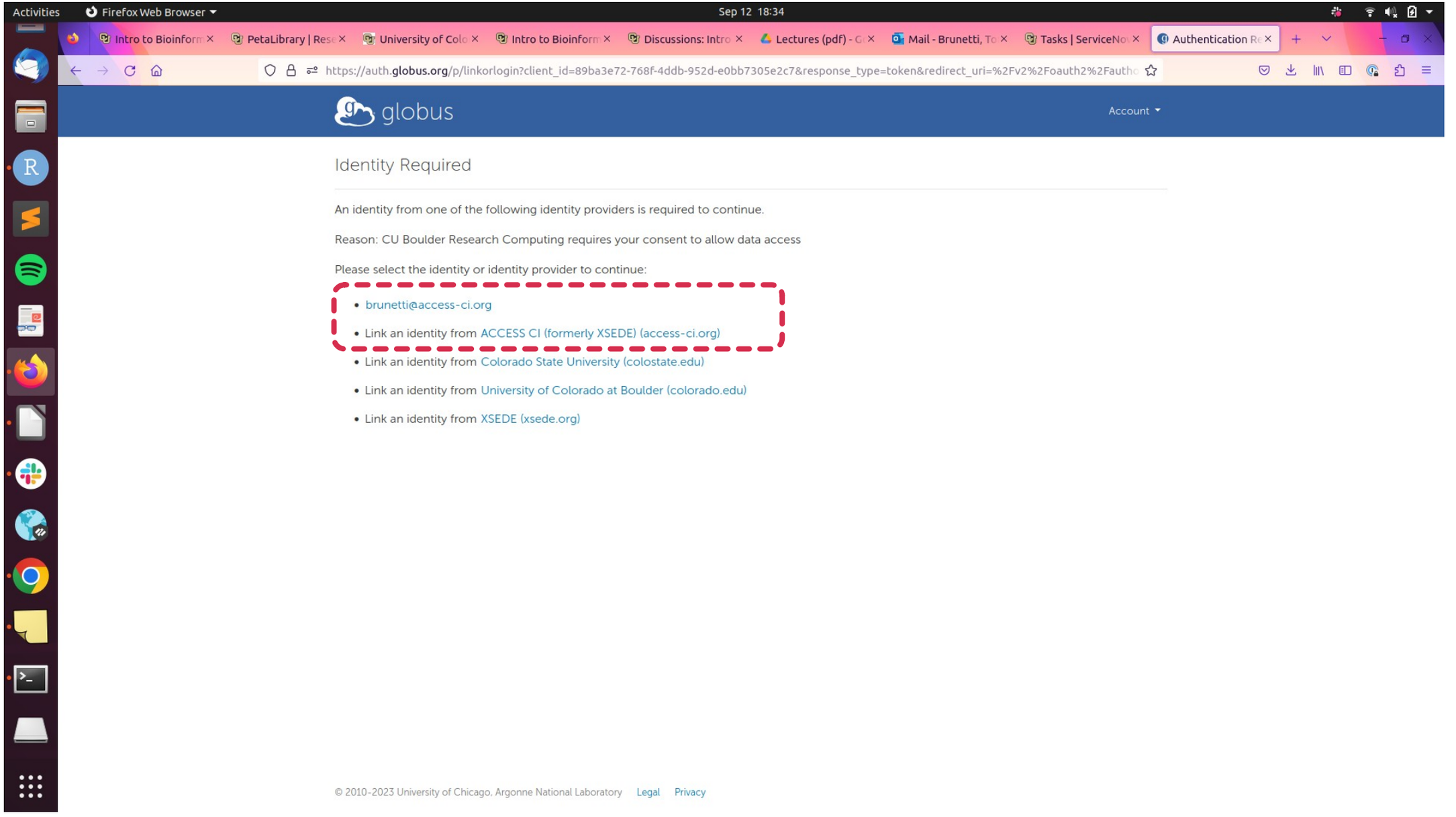
Data Transfer with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus



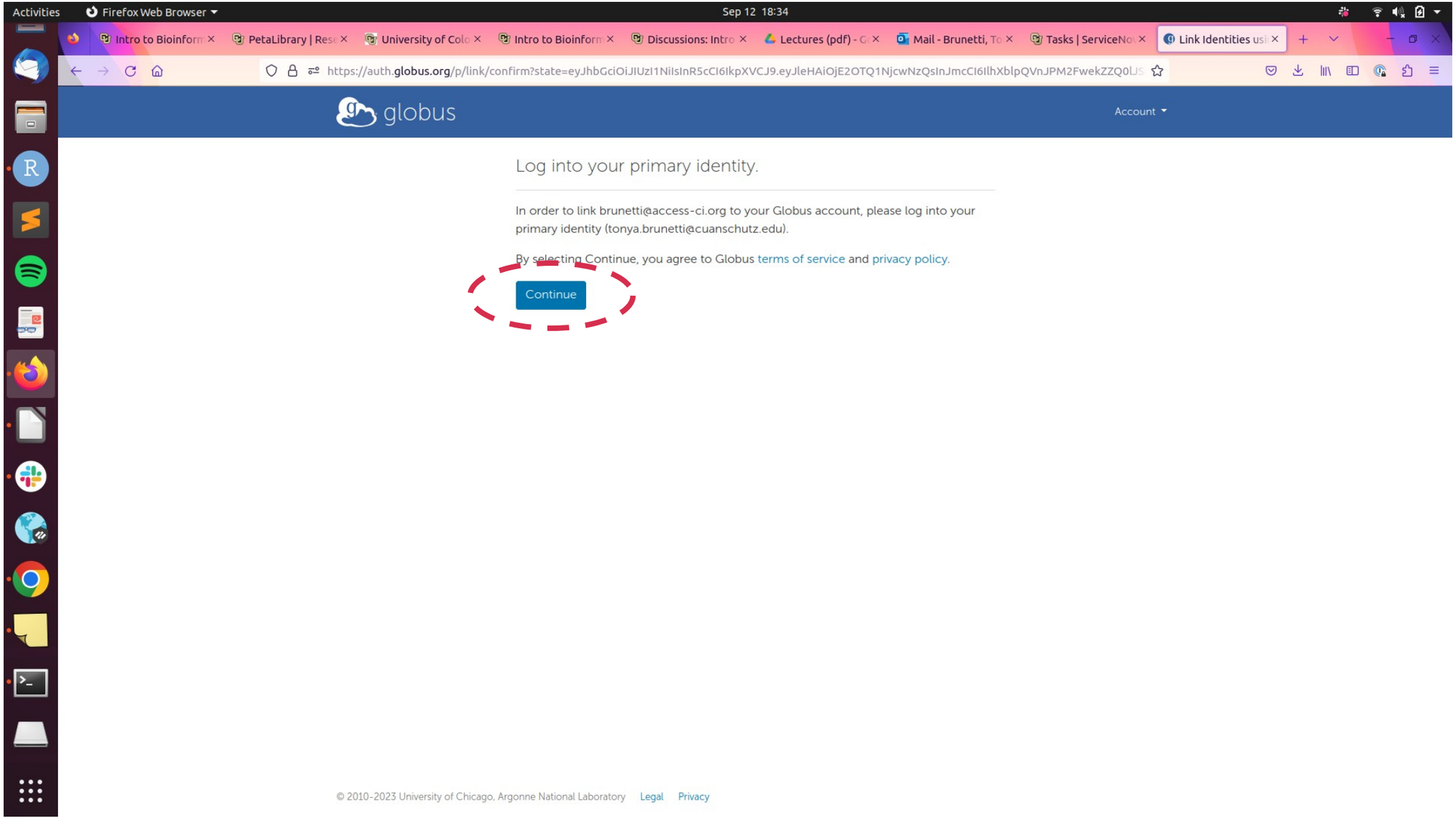
Data Transfer with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus



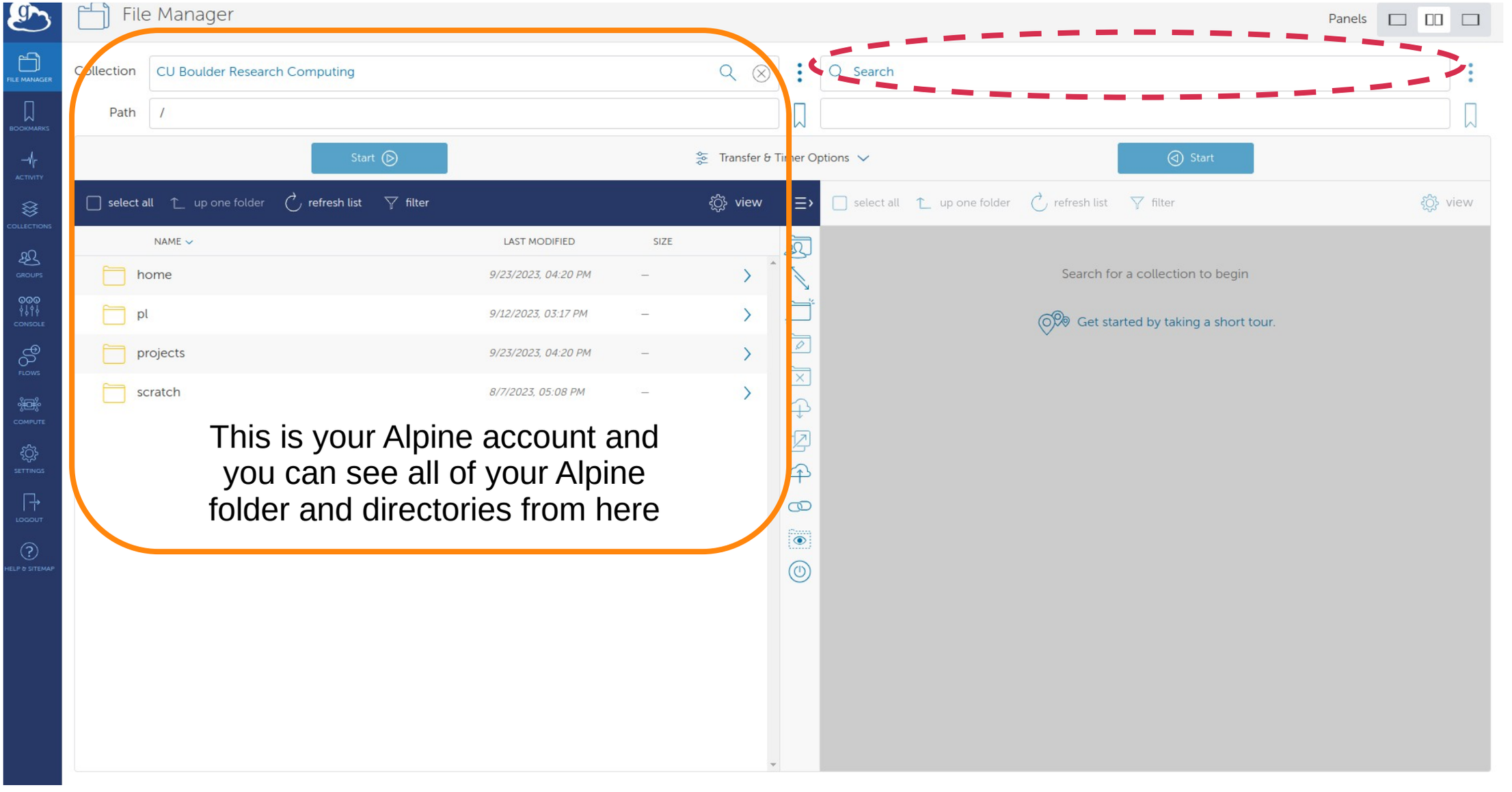
Data Transfer with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus



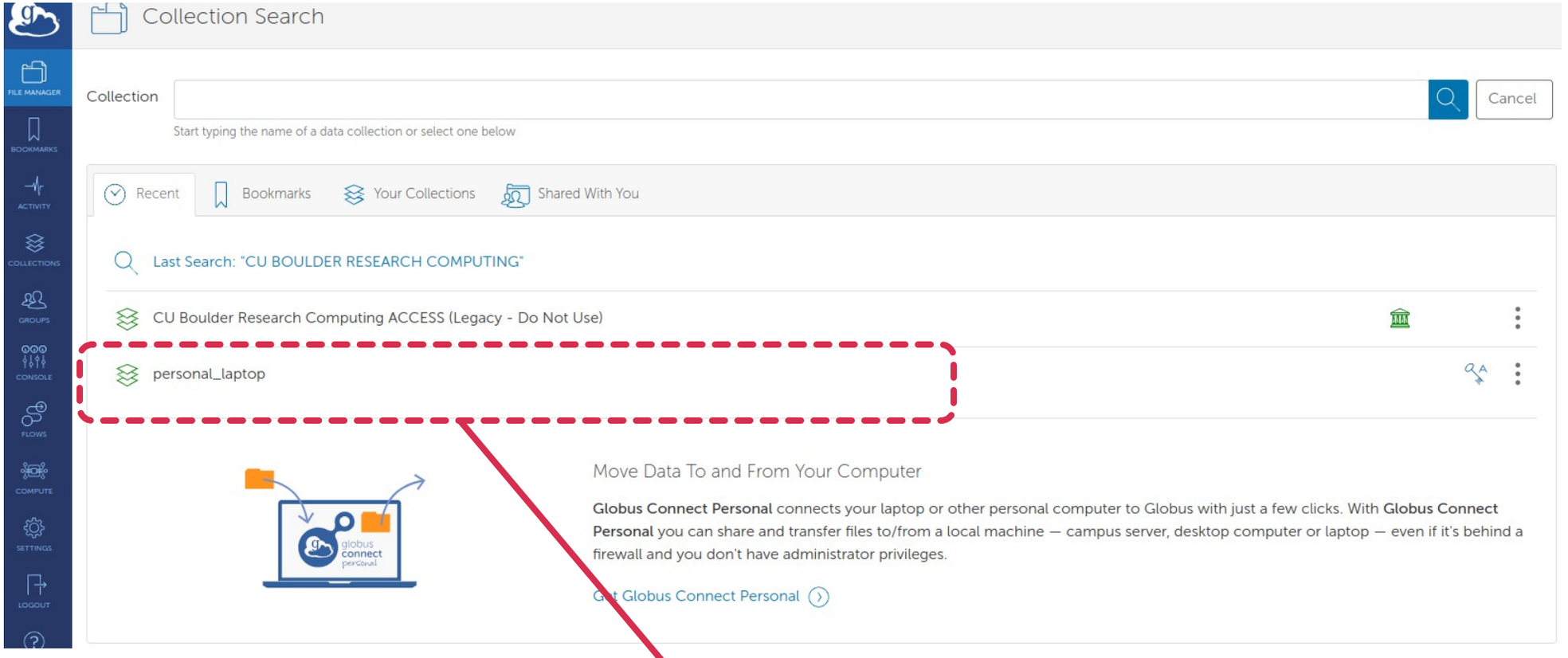
Data Transfer with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus



Data Transfer with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus



This is your globusconnect personal that you downloaded and connected to. It may not be called personal_computer, but rather something you picked when you installed the software

Data Transfer with Globus

- Unless you have ssh enabled for Alpine (which nearly all of you won't) you only option for transfer data is with Globus