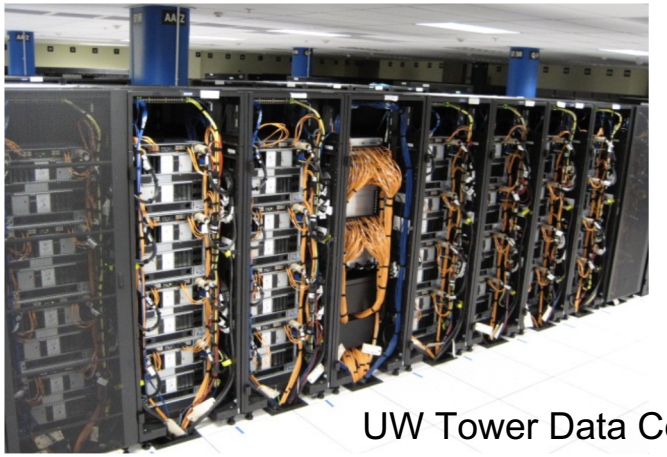


UW Tower Data Center

Research Computing Club Presents

Hyak Training Session

January 24, 2022

Brenden Pelkie (Training Coordinator)

Outline

Where I assume you're at right now:

"I finished the steps to getting access* to Hyak, but don't really know what to do now. How do I get up and running with the computing I want to do?"

I. Hyak overview
- A. Hyak architecture
- B. Logging on to Hyak

II. Navigating Hyak
- A. Basic commands
- B. Important locations
- C. Transferring files

III. Job Scheduling - Slurm
- A. Running a job
- B. Commands
- C. Modules
- D. Interactivity

IV. Other resources
- A. Topics outside of this tutorial
- B. Places to get help

* https://depts.washington.edu/uwrcc/getting-started-2/getting-started/

Things We Won’t Cover:

By the end of this training, you will not:

- Know how to build and install software on Hyak
- Have a complete workflow for your particular project
- Understand how to optimize your code for an HPC system
- (necessarily) know/have the best tools for your problem: There are lots of ways of doing most things I will present here.

What is high performance computing?

Real research problems require large calculations to solve

Compared to your personal computer, HPC has:

- More processor cores (40 vs 4-8)
- More memory
- Multiple nodes (ie multiple computers)
- Faster networking for better parallelization

Examples: Summit, Fugaku, Perlmutter, etc.

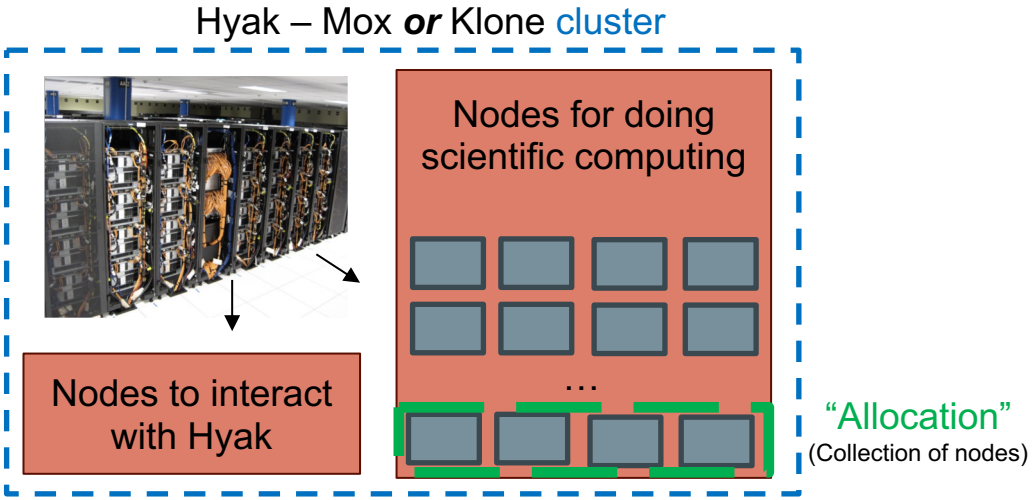
https://phys.org/news/2018-06-ornl-summit-supercomputer.html

What is Hyak?

Hyak is UW's research computing environment

Comprised of two supercomputing "clusters":

- Mox - in phase-out process
- Klone – current cluster



Nodes on Mox and on Klone are only accessible if signed onto that cluster

On Hyak, jobs are run on *nodes:*

- physically an individual computer/server
- Can use part of a node or multiple nodes for a job

Mox nodes:

- 28 cores
- 128 GB RAM

Klone nodes:

- 40 cores (can be split between users)
- 192 GB RAM (or 768 GB)

Resource Allocations on Mox and Klone

Access to nodes is through a specific allocation or account

`stf`: All tuition-paying students have access to the student technology fee (‘stf’) allocation

- Ideally for use on personal or class work – **not for research projects**
- Mox: 54 compute nodes, 9 GPU nodes
- Klone: 15 compute nodes, 1 GPU node, 3 hugemem nodes

`Research group allocation`: Purchased by advisor. Use this for research.

`checkpoint:` allows you to use idle resources from other allocations. Job may be terminated at any time – plan accordingly

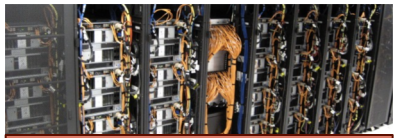
Use `hyakalloc` to view available resources

Accessing Nodes - Hyak “Architecture”

- All nodes on same cluster share the same filesystem

Hyak / Mox or Hyak / Klone
on your screen, but a
separate computer

Your Computer



Login Nodes
(Shared space)

via Slurm

Computing Nodes

- High performance, lots of nodes
- Takes batch submission
- Interactive optional
- Actual work is done here

for navigation, not calculation

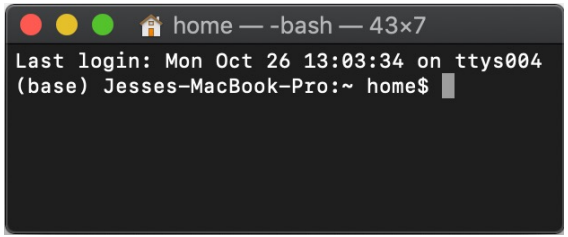
- Transfer files to/from Mox or Klone
- Submit work *to other nodes*
- Do **NOT** run intensive calculations here

login via ssh command
(next slide)

Logging on to Hyak

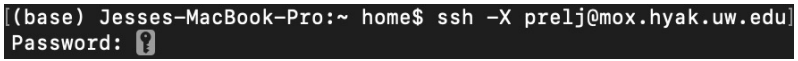
1. Launch a new terminal

Terminal – “command line”
Mac/Linux: you have this
Windows: Hang tight



2. Log into Hyak with SSH

“secure shell” command
Enables connection to remote machine

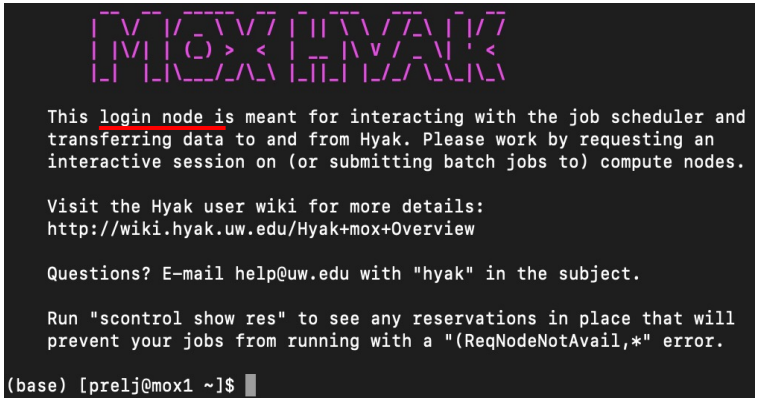



```
ssh <uwnetid>@mox.hyak.uw.edu
ssh <uwnetid>@klone.hyak.uw.edu
```

3. Enter your UW Credentials

Two-factor authentication required
(Through UW-IT)

4. You’re in!



- Notice you are defaulted to the *login node:* **Jobs are not run here**

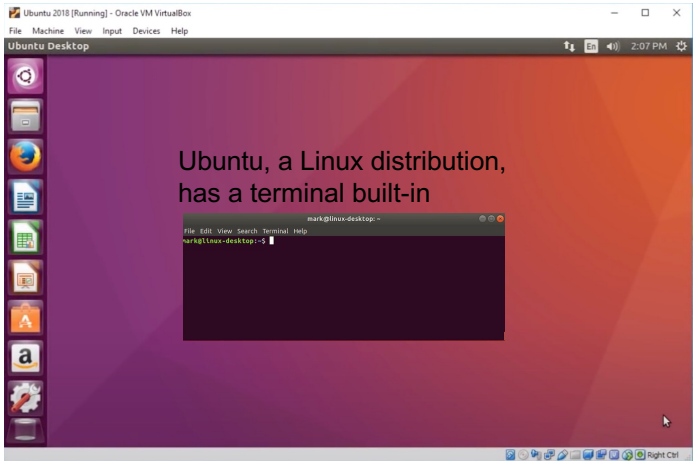
What if I'm on Windows?

You'll need a terminal somehow, and Windows does not provide a good one by default
RCC recommendation: Windows Subsystem for Linux* (I am not a windows user)

- Allows you to run a linux shell in Windows
- Access windows filesystem from linux shell (useful for transferring data to/from hyak)

To install on a recent Windows version:

1. Launch Powershell and enter command "`wsl –install`"
2. After install completes, reboot
3. Open linux distribution from start menu and set up profile
4. Use SSH to access Hyak, as described on last slide



Other options for Windows:

- PuTTY (Terminal emulator)
- Linux virtual machine (Similar concept to WSL)

More help:

https://docs.microsoft.com/en-us/windows/wsl/install
http://wiki.cac.washington.edu/display/hyakusers/Logging+In

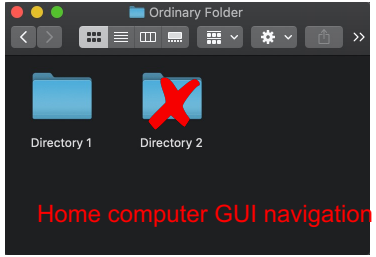
Practice: Log In

1. Open a new terminal window
2. Enter the command “ssh <your netid>@klone.hyak.uw.edu
3. Follow the prompts for your password (same as UW password) and 2FA
    - Note: As you type your password, nothing will appear. This is normal.

I’m in

… Now what?

Exploring Hyak/Command line 101



Hyak works on a command line interface: no mouse and pointer

- Commands are written in *bash*
- Basic commands transferable to all linux/mac systems

```
(base) [prelj@mox1 ~]$
```

Essential bash commands for navigation:

Note: On a computer, a “folder” and a “directory” are the same thing

Where in the computer am I?

`pwd` – “Print working (current) directory”

```
(base) [prelj@mox1 ~]$ pwd
/usr/lusers/prelj
```

My home directory (see next slide)

What folders and files are here?

`ls` – “List” files in current directory (folder)

```
(base) [prelj@mox1 ~]$ ls
d  g_perf  intel  opt
```

Change to a different directory:

`cd <filepath>` – “Change directory”

```
(base) [prelj@mox1 ~]$ cd /gscratch
(base) [prelj@mox1 gscratch]$
```

`cd ..` – “Move up one directory”

What does this file say?

`cat <filename>`

```
(base) bgpelkie@klone1:~$cat .bash_profile
```

More shell (bash) commands: Try them out!

File system manipulation:

- `Mkdir <dir. name>`
  - “Make directory”
- `Mv <filename> <new location>`
  - Move (rename) file or directory
- Cp `<filename> <new location>`
  - “Copy” files and/or directories (-r)
- rm <filename>
  - “Remove” files and/or directories (-r)
  - Add “-r” between rm and filename for directories
  - This deletes the file permanently. Be careful. Have a backup of important stuff.

There are many more – here’s one resource: https://guide.bash.academy/

File editing:

Edit text files (ie code, scripts) with a command line text editor

Pick one and get comfy with it

- Vim
- Nano
- emacs

Changing Bash Profile

You can change behavior of commands (ie get more information from ls) and add colors by editing **.bash_profile** file in home directory. See files from this training for a starting point

Important locations on Hyak (get there with cd)

- ⭐ `/gscratch/stf`
  - Main work location for stf users
  - **Any files untouched for >30 days will be auto-deleted!!**
- ⭐ `~`
  - Home directory "~", *you are put here by default*
  - `/usr/lusers/<username>` on Mox
  - `/mmfs1/home/<username>` on Klone
  - **Only 10 GB of storage per user**
- `/tmp`
  - Node local storage (separate from shared filesystem)
- `/sw`
  - All software installs
- `/sw/contrib`
  - User installed software – not *yet* operational on Klone
- `/sw/modules-1.775/modulefiles/contrib`
  - User added modulefiles – on mox, still not up for Klone

Where pre-installed programs (and their supporting files) are housed

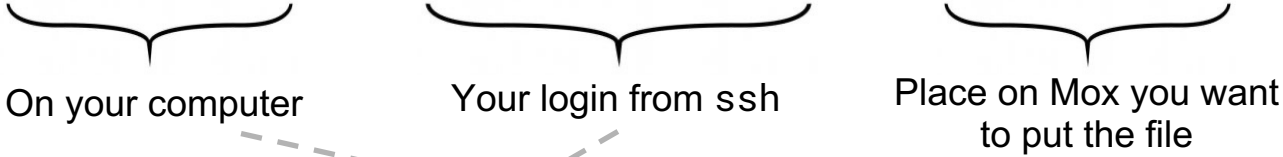
You will seldom need to go here

Transferring files

`scp` – Secure Copy

- Send file To Mox or Klone: …>@klone.hyak…

```
scp <path/to/file> <username>@mox.hyak.uw.edu:<path/to/dest>
```



- Get file From Mox:

```
scp <username>@mox.hyak.uw.edu:<path/to/file> <path/to/dest>
```

…>@klone.hyak…

Flag `-r` needed to copy whole folder at once

An alternative copier for synchronizing directories is `scopy`

Running Your First Job

Real computing jobs are passed to Slurm

- Ensures fair share between users
- Manages multi-node jobs

Interactive vs. Batch jobs:

- Interactive: requires user input during job
- Batch: job runs unattended

https://slurm.schedmd.com/

Some information is needed during job setup:

Partition: whose CPUs/GPUs are we using? (we'll use stf)
Allocation: which bank account pays for these? (also stf)
Locations: where to write/read files to/from
Resources: how many CPUs/GPUs are needed, and for how long?
Modules: which programs should be loaded to use?

Packaged into a "Slurm script"

Batch Jobs use a “slurm script”

In a text file called “submit.slurm” on Klone

```
#!/bin/bash
## The first line has to say this, as a bash script

## Job Name
#SBATCH --job-name=test_python

## Partition and Allocation
#SBATCH -p compute
#SBATCH -A stf

## Resources
#SBATCH --nodes=1
#SBATCH --time=0:01:00
#SBATCH --ntasks=1
#SBATCH --mem=100G

## Specify the working directory for this job
#SBATCH --chdir=.

## Import any modules here
module load contrib/anaconda/anaconda4.4.0

## Scripts to be executed here
python test_run.py

## Clean up
exit 0
```

Any line that begins with ## is a comment and is not read
Each line that begins with #SLURM specifies info for slurm
The #SLURM options can be provided in any order

- Name is optional
- Allocation: Use `stf` funding/budget
- Partition: *Different* on Klone! “`compute`” or “`ckpt`”

- Request (1) node
- Time – Hours : Minutes : Seconds
- **Be mindful of your node capacity**: 28 vs. 40 core, memory

- Directory “.” is *current location* to read/write files (the same place submit.slurm is saved)
- Load in a pre-installed python package

- The real run command

To execute this job,

```
sbatch submit.slurm
```

Then We Wait

- `sbatch <script>`
  - Submits a script for non-interactive use
  - Used this to submit a job on previous slide

Want to check the status?

- `sacct`
- `squeue`
  - Flag –u `<uwnetid>` for only your jobs
  - Specify –p or –A for whole queue

What it looks like to check queue:

```
(base) [prelj@mox1 move]$ squeue -p stf
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            417044       stf test_pyt    prelj CG       0:02      1 n2282
            416180       stf [redacted] PD       0:00      1 (Dependency)
            416186       stf [redacted] PD       0:00      1 (Dependency)
            413137       stf [redacted]  R 1-04:27:53     10 n[2143,2151,2166,2168,2171-2172,
            415317       stf [redacted]  R   15:51:35      6 n[2140,2165,2275,2277,2292,2298]
            414068       stf [redacted]  R   22:51:51      6 n[2266,2269-2270,2285,2290,2308]
            413181       stf [redacted]  R 1-04:04:16      6 n[2139,2144,2146,2279,2284,2312]
```

Other peoples' job names and IDs (censored)

Something wrong? / Taking too long?

- `scancel <jobid>`
  - Cancels an unfinished job
  - Can only cancel your own

Always check your output!

JOBID is used to refer to specific jobs
NODELIST has ID #s for individual nodes being used
ST column is status
R – Running (currently active)
PD – Pending (in queue)
CG – Completing (in process of termination)

Practice: Submit a job

While logged into klone:

1. Change directories to gscratch: “`cd /gscratch/stf`”
2. Make a directory for yourself in gscratch: “`mkdir <your netid>`” and switch into that directory
3. From a new terminal on your computer, copy the handout files to your new gscratch directory: “`scp <path to handout files>/* <your netid>@klone.uw.edu:/gscratch/stf/<your netid>`”
4. Submit the FizzBuzz job: “`sbatch slurmscript.sh`”
5. View the output file: “`cat <output file name>`”

Check Memory Resources: Hyakalloc

On Mox:

```
[(base) [prelj@mox1 chains_reg_i]$ hyakalloc
Allocation Name    Nodes     Cores     Total Memory     Max Mem Per Node
---------------    -----     -----     ------------     ----------------
alys                  25        40             500G                500G
anantram                        40             374G                   G
ark                             40             373G                   G
ark-gpu                1        40             373G                373G
astro                 12       336            4992G                500G
baker                 25       700            6136G                248G
barnardlab            24        40             185G                248G
bdata                           40             373G                   G
bdata-gpu              1        40             373G                373G
bkrs                  65      1820           16120G                248G
brunton               25        40             752G                752G
build                  2        56             496G                248G
build-gpu              1        40             373G                370G
build-long                                       G                   G
bumblereem             7       196             980G                248G
cardss                 4       128             740G                185G
```

… (many!)

On Klone:

```
[[prelj@klone-login01 sw]$ hyakalloc

Account-Partition - Resource          Total          Used          Free
----------------------------          -----          ----          ----
chem
   - CPU cores                          480             0           480
   - Memory (GB)                       2220             0          2220

Account-Partition - Resource          Total          Used          Free
----------------------------          -----          ----          ----
pfaendtner
   - CPU cores                          400           288           112
   - Memory (GB)                       1850           260          1590

Account-Partition - Resource          Total          Used          Free
----------------------------          -----          ----          ----
stf
   - CPU cores                          680            59           621
   - Memory (GB)                       3145           900          2245

Account-Partition - Resource          Total          Used          Free
----------------------------          -----          ----          ----
stf-hugemem
   - CPU cores                          120            80            40
   - Memory (GB)                       2256          1500           756
```

Nodes can be automatically shared between users on Klone

Only *cores* are reported

Running Interactive Jobs from Command Line

```
srun -p compute -A stf --ntasks=8 --mem=10G --time=0:10:00 --pty bash -l
```

Time limit in place for interactive nodes: 8 hrs

- Partition
- Account
- Number of processes (*)
- Amount of RAM
- Time
- Command

- Much of the same specifications as submit.slurm
- `--pty bash -l` signifies *interactive*
- Now you have command line control *while* accessing a compute node

Before running interactive job command

```
[(base) [prelj@mox1 jesse]$
```

After running interactive job command

```
[(base) [prelj@n2232 jesse]$
```

- Interactive nodes are computing nodes. You ***can*** submit jobs to other nodes from interactive nodes.

https://slurm.schedmd.com/srun.html

`man srun`

Loading software: the `modules` system

- `module avail`
  - Show all available modules
- `module load <module>`
- `module unload <module>`
  - (Un)load a given module
  - Provide full module name
- `module list`
  - List loaded modules
- `module purge`
  - Unload all modules
- `module help`
  - Print help with commands

Running `module avail`

```
(base) [prelj@mox1 jesse]$ module avail
------------------------------------------------ /sw/modules-1.775/modulefiles -----------------------------
anaconda2_4.3.1                                  contrib/multiz/20090121
anaconda2_5.3                                    contrib/multiz/20190527
anaconda3_4.3.1                                  contrib/mummer/3.23
anaconda3_5.3                                    contrib/mummer/4.0.0b2
ansys_18                                         contrib/muscle/3.8.31
cmake/3.11.2                                     contrib/mysql/8.0.11
cmake_3.8                                        contrib/NAMD2019/namd2
contrib/3ddna/170123                             contrib/nauty26r11/nauty26r11
contrib/9.0                                      contrib/ncbi-blast/2.7.1
contrib/abaqus/2020                              contrib/ncbi-vdb/2.9.0
contrib/abyss/2.0.3                              contrib/nektar
contrib/adapter-removal/2.1.7                    contrib/newton-x
contrib/admixtools/1.0.1                         contrib/newtonX
contrib/admixture/1.3.0                          contrib/ngs/2.9.0
contrib/anaconda-2019.03                         contrib/ngsRelate/1.0
contrib/anaconda/2-5.0.1                         contrib/ngsRelate/2.0
contrib/anaconda/3-4.4.0                         contrib/nvidia_toolkit/9.0
contrib/anaconda/3-5.3.0                         contrib/nvidia_toolkit/10.0
contrib/anaconda/anaconda4.4.0                   contrib/NWChem/NWChem
```

… and more

- 400 modules available on Hyak Mox right now!
- Far fewer on Klone at the moment – hang tight

Useful programs you'd use on your own computer, i.e.

- Python/Anaconda, Gromacs, NAMD, R, Mathematica, Matlab, Gaussian, compilers and more

Parallel computing tools

- CUDA, IMPI, etc.

Advanced user's note:
The modules system works by keeping track of and modifying environment variables (e.g. PATH, LD_LIBRARY_PATH, CPATH, etc.)
https://modules.readthedocs.io/en/latest/

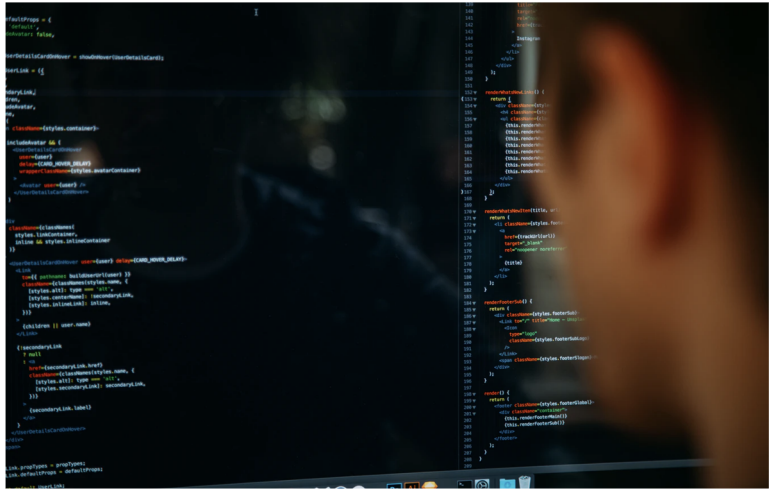
Checkpoint queue – special case

The checkpoint queue allows any user to run on other groups' unused nodes!

- Partition: `ckpt`
- Account: `<group>-ckpt` (`stf-ckpt`)
- Jobs can be interrupted at any time
- Jobs *will* be interrupted after 4 contiguous hours
- Jobs will be resubmitted after interruption (if under total requested time limit)
- **Your code must be checkpointed to take advantage of this**
  - Save a binary file containing state of some objects, restart from logs, etc.
  - Your script should also account for any checkpointing that is done

Topics not covered

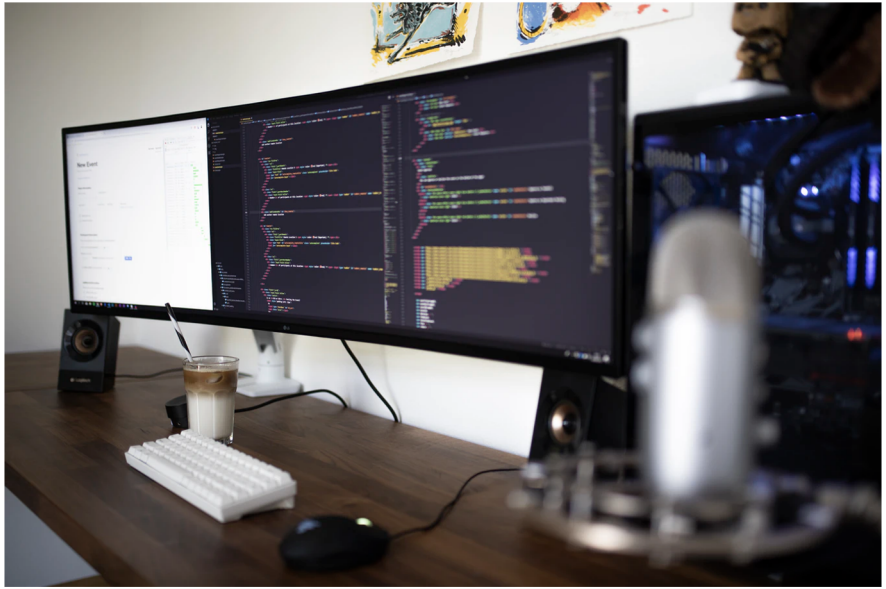
- How to install new software
  - Installation and build systems
  - Writing your own modulefiles
- How to parallelize your code
- Anything with the cloud
- General architecture of HPC systems
  - Interconnectivity and node locality
  - Physical infrastructure
- Archiving on Lolo



Where to get help

- Documentation (`man` or webpages)
- Hyak wiki: https://wiki.cac.washington.edu/display/hyakusers/WIKI+for+Hyak+users
- Slack channel: https://uw-rcc.slack.com/
- Website: https://depts.washington.edu/uwrcc/
- Emails: hpcc@uw.edu or uwrcc@uw.edu
- Office hours: Mondays 1-2 PM this quarter, location varies

Happy computing!



Slurm: Commands

- `sbatch <script>`
  - Submits a script for non-interactive use
- `squeue`
  - Get status of jobs in batch queue
- `scancel <jobid>`
  - Cancels an unfinished job

I use these most often

- `srun`
  - Submits job for interactive use or initiate job steps inside batch script
  - (More on interactive jobs in a minute)
- `sinfo`
  - Get state of partitions and nodes (is the system operational)
- `sacct`
  - Gets accounting information about active and completed jobs

Slurm docs and `man <slurm-command>` are very useful!

(“man” for manual)