Intro to HPC

A crash course on the High Performance Computing system at the University of Arizona

by Ethan Jahn

Outline

1. General HPC Background
2. The UA HPC System
3. Accessing the System
4. Job Scheduling
5. Bash + Linux
6. Batch Job Example

My Background

- BS Physics 2016 IUP w CS+Math Minors
- MS Physics & Astronomy 2017 UCR
- PhD Physics & Astronomy 2021 UCR
  - 3 papers published in MNRAS on **computational galaxy formation** w Laura Sales
  - Worked with *FIRE* (P Hopkins, CalTech) and *Illustris/Smuggle* simulations
  - **Coding on HPC since 2017**
    - analysis, simulations
- 2 years community college prof (1st - 3rd semester physics)

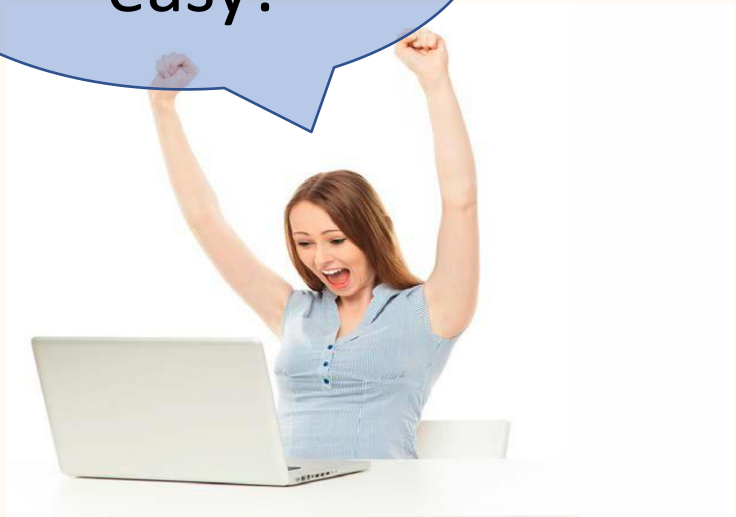
HPC consultant here at UA as of August 2023

General HPC Background

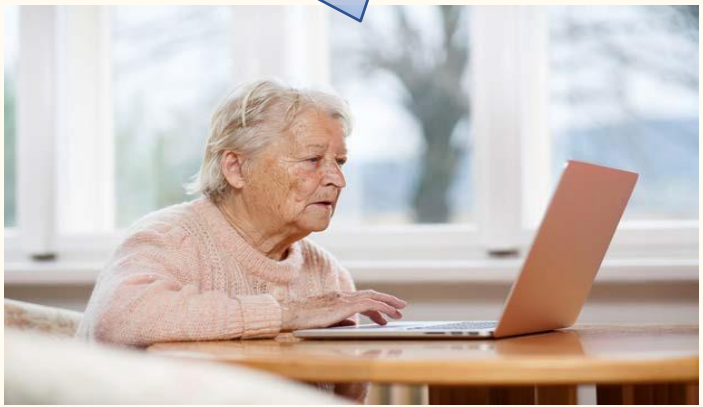

and many more!
DISCIPLINARY
POSITIVIST
POSITIVISM
SOCIAL
SCIENCES
JURISPRUDENCE
ADMINISTRATION
LANGUAGE
MONOGRAPH
PUBLIC
QUANTITATIVE
ACADEMIC
BRANCH

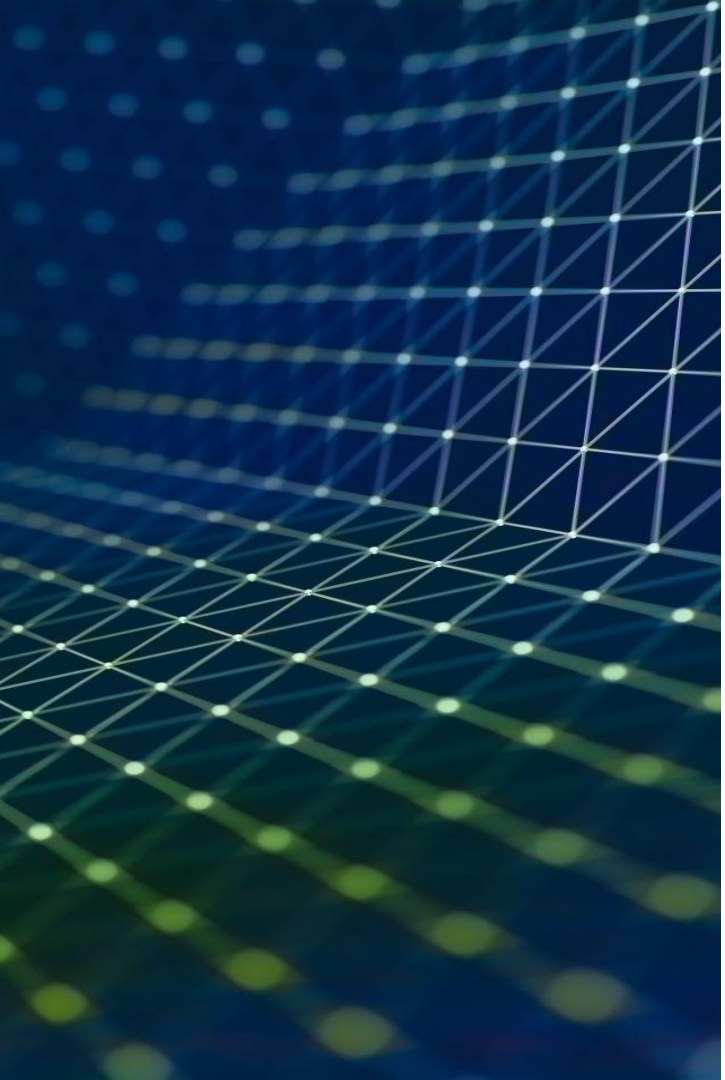
Why HPC?

Research is easy!



It’s still running…





Why HPC?

Problems

- Computation takes too long
- Computation is too big
- Too many computations

Why HPC?

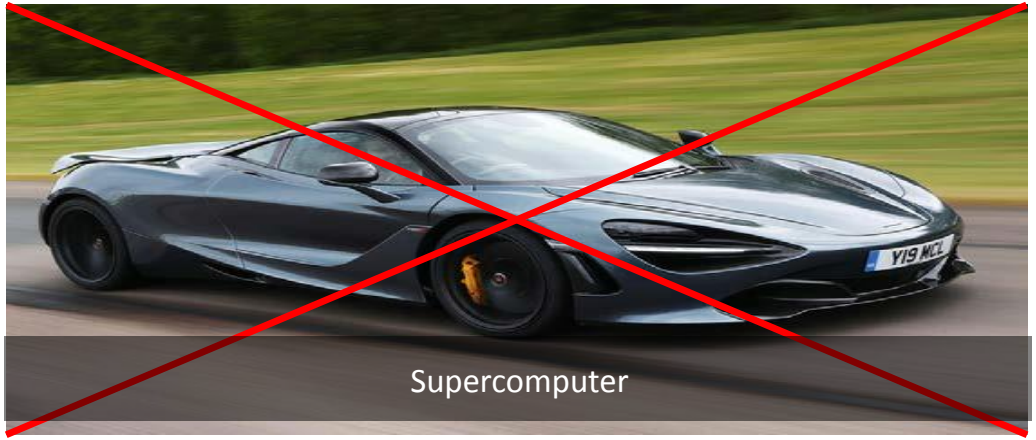
Problems

- Computation takes too long
  -> Get a more powerful computer
- Computation is too big
  -> Link multiple computers
- Too many computations
  -> Use a separate one for each job

Why HPC?

Modern instrument for High-Performance Computing is a **cluster**, consisting of lots of connected individual computers (nodes).

Supercomputer is a commonly used nickname.



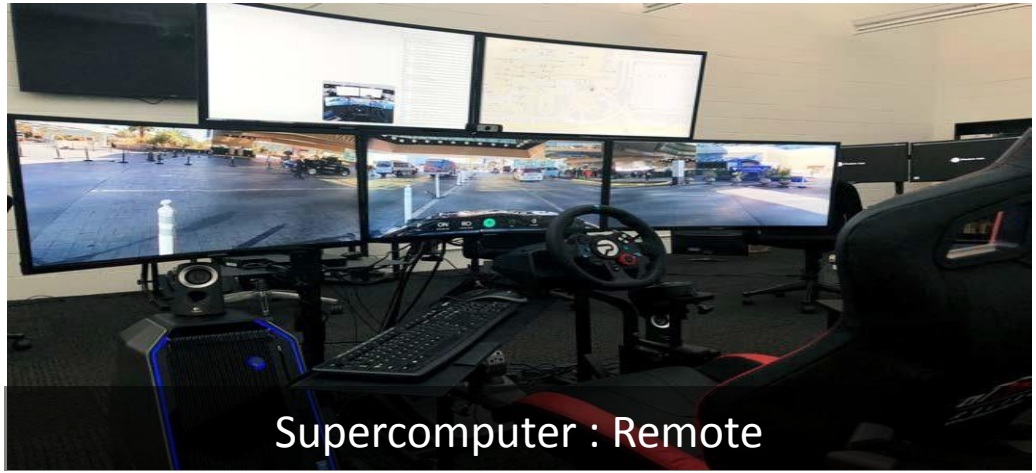
Why HPC?

Why HPC?

Laptop : Personal

Supercomputer : Shared

Why HPC?



Why HPC?

Laptop : Interactive

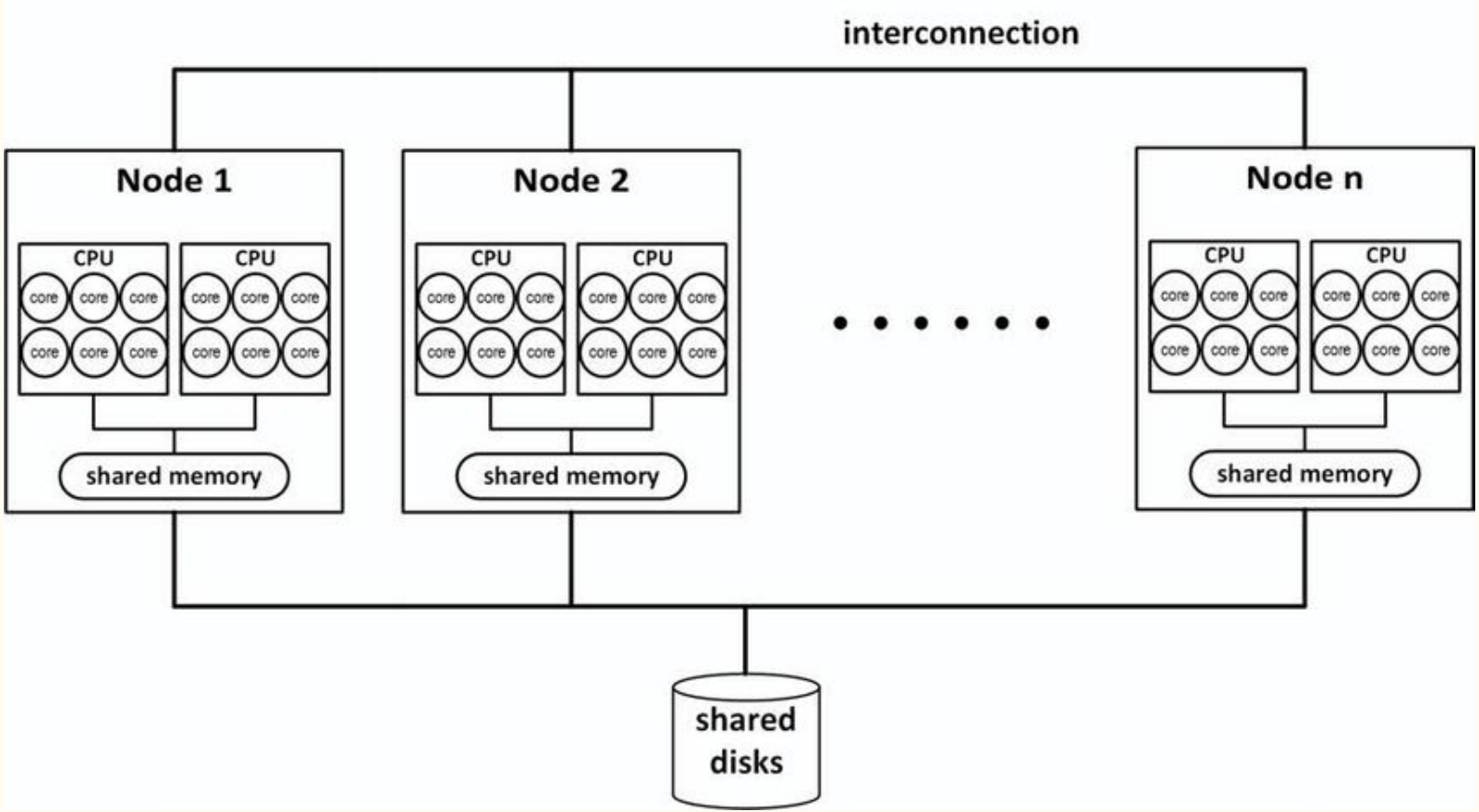
Supercomputer : Batch

Why HPC?

Why HPC?

- Modern instrument for High-Performance Computing is a **cluster**, consisting of lots of connected individual computers (nodes).
- Supercomputer is a commonly used nickname.

Architecture of a Supercomputer




IBM
Blue Gene/P

Summit Overview

OpenPOWER™

Compute Node

2 x POWER9
6 x NVIDIA GV100
NVMe-compatible PCIe 1600 GB SSD

25 GB/s EDR IB- (2 ports)
512 GB DRAM- (DDR4)
96 GB HBM- (3D Stacked)
Coherent Shared Memory

Compute Rack

18 Compute Servers
Warm water (70°F direct-cooled components)
RDHX for air-cooled components

39.7 TB Memory/rack
55 KW max power/rack

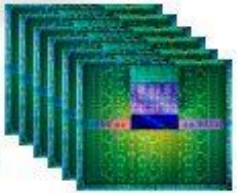
Compute System

10.2 PB Total Memory
256 compute racks
4,608 compute nodes
Mellanox EDR IB fabric
200 PFLOPS
~13 MW

Components

IBM POWER9
- 22 Cores
- 4 Threads/core
- NVLink

NVIDIA GV100
- 7 TF
- 16 GB @ 0.9 TB/s
- NVLink



GPFS File System

250 PB storage
2.5 TB/s read, 2.5 TB/s write

The UA HPC System

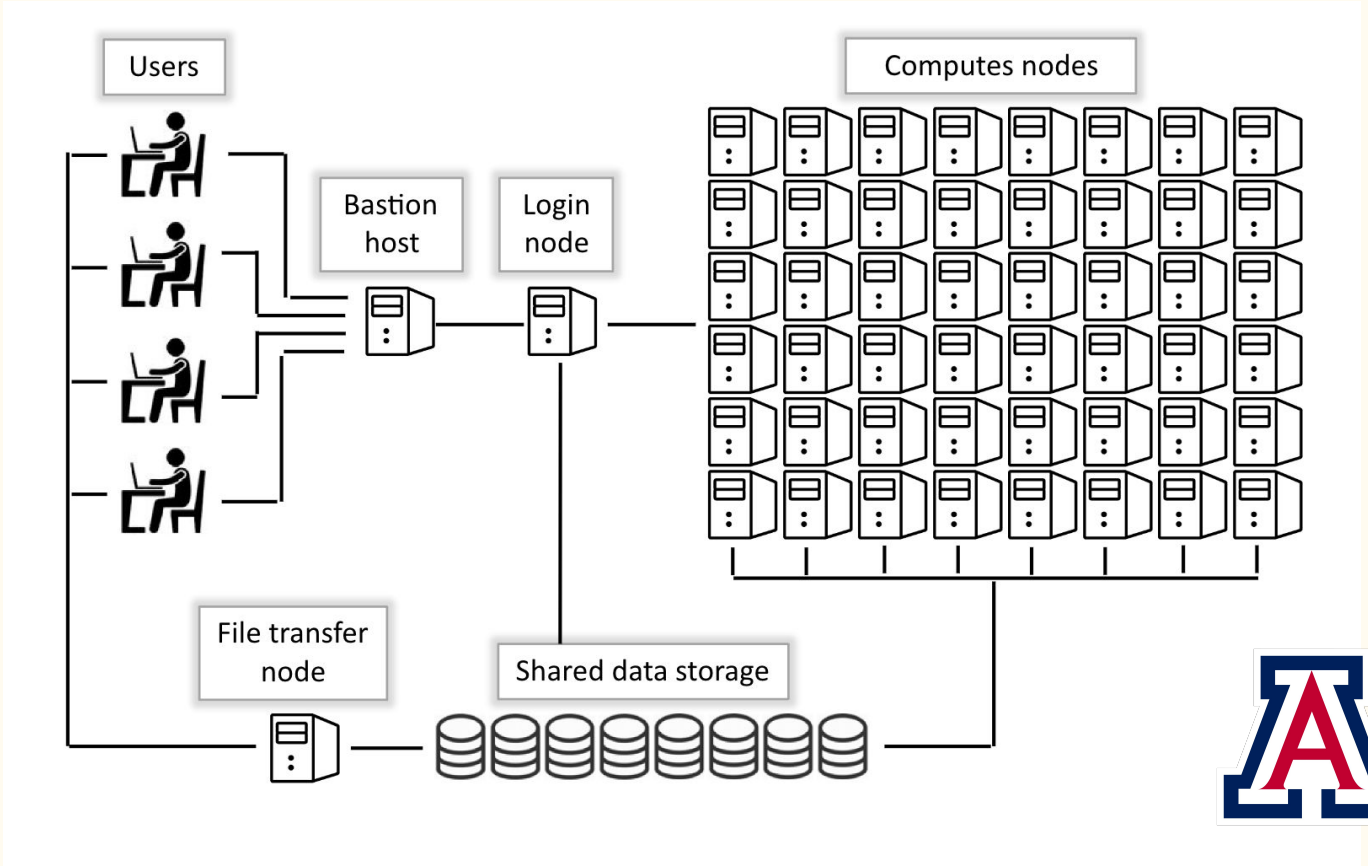
UA HPC Reso

not expected for user g HPC!

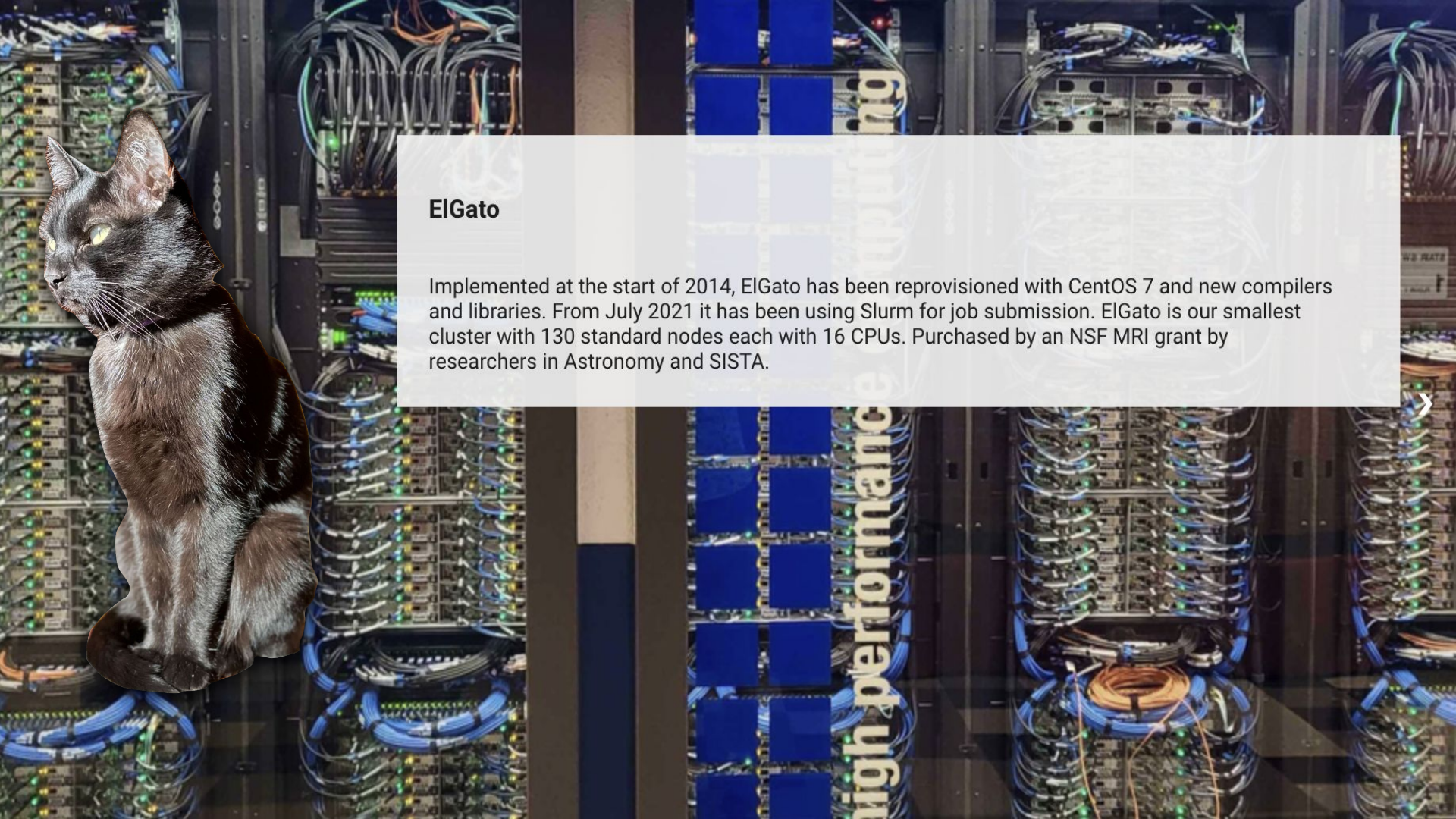
The HPC Consult Tea onalized help

- **Documentation:**
- **GitHub**: ua-rese
- **ServiceNow**: HP
- **Email**: hpc-cons
- **Office Hours**: ev n

UA HPC Architecture

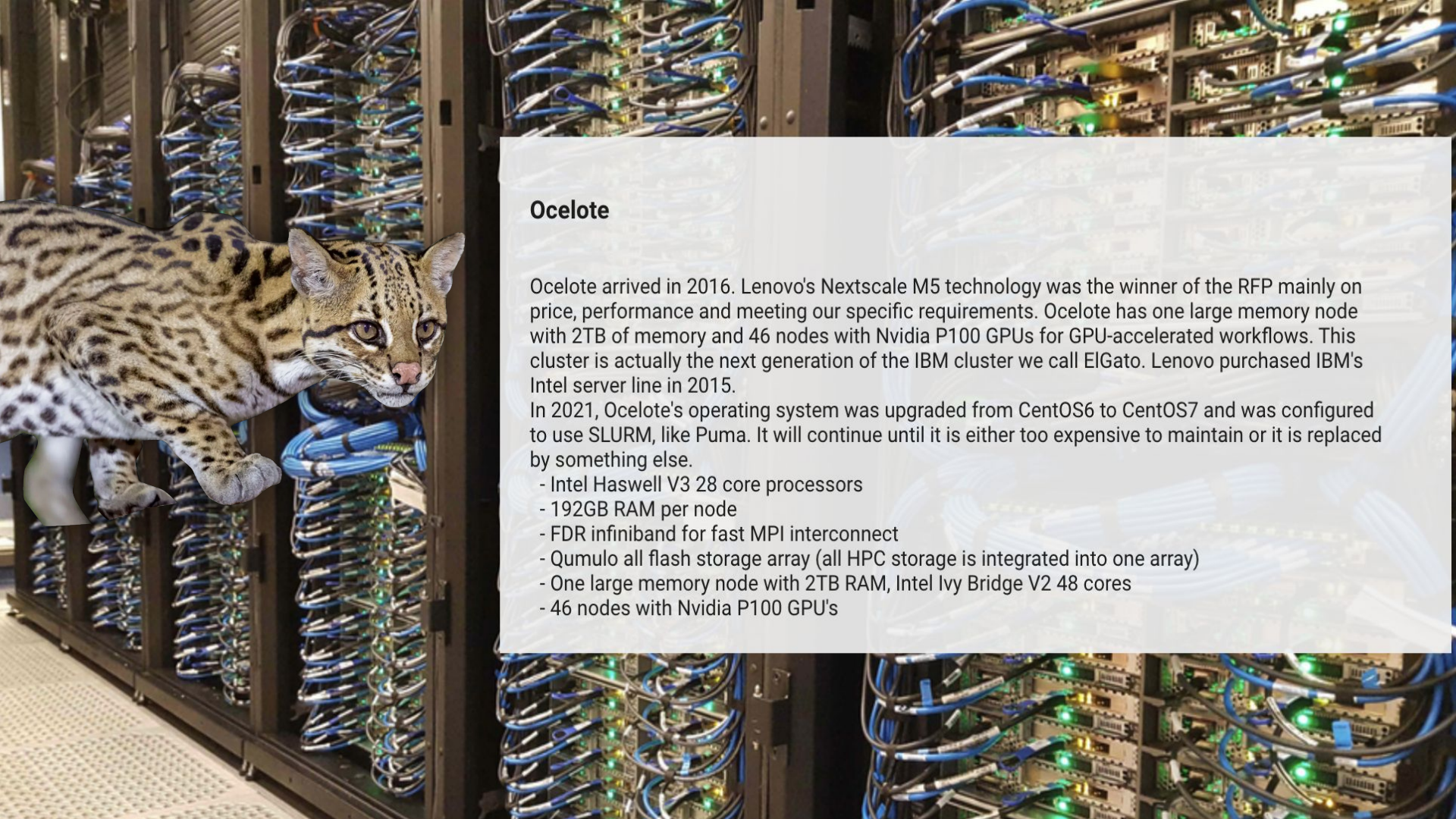


Our Clusters!



ElGato

Implemented at the start of 2014, ElGato has been reprovisioned with CentOS 7 and new compilers and libraries. From July 2021 it has been using Slurm for job submission. ElGato is our smallest cluster with 130 standard nodes each with 16 CPUs. Purchased by an NSF MRI grant by researchers in Astronomy and SISTA.



Ocelote

Ocelote arrived in 2016. Lenovo's Nextscale M5 technology was the winner of the RFP mainly on price, performance and meeting our specific requirements. Ocelote has one large memory node with 2TB of memory and 46 nodes with Nvidia P100 GPUs for GPU-accelerated workflows. This cluster is actually the next generation of the IBM cluster we call ElGato. Lenovo purchased IBM's Intel server line in 2015.

In 2021, Ocelote's operating system was upgraded from CentOS6 to CentOS7 and was configured to use SLURM, like Puma. It will continue until it is either too expensive to maintain or it is replaced by something else.

- Intel Haswell V3 28 core processors
- 192GB RAM per node
- FDR infiniband for fast MPI interconnect
- Qumulo all flash storage array (all HPC storage is integrated into one array)
- One large memory node with 2TB RAM, Intel Ivy Bridge V2 48 cores
- 46 nodes with Nvidia P100 GPU's

Puma

Implemented in the middle of 2020, Puma is the biggest cat yet. Similar to Ocelote, it has standard CPU nodes (with 94 cores and 512 GB of memory per node), GPU nodes (with Nvidia V100) and two high-memory nodes (3 TB). Local scratch storage increased to ~1.4 TB. Puma runs on CentOS 7.

As is the case for our other supercomputers, we use the RFP process to get the best value for our financial resources, that meet our technical requirements. This time Penguin Computing one with AMD processors. This is tremendously valuable as each node comes with:

- Two AMD Zen2 48 core processors
- 512GB RAM
- 25Gb path to storage
- 25Gb path to other nodes for MPI
- 2TB internal NVME disk (largely available as /tmp)
- Qumulo all flash storage array for shared filesystems
- Two large memory nodes with 3TB memory and the same processors and memory as the other nodes
- Six nodes with four Nvidia V100S GPU's each

UA HPC Compute Resources

Cluster	Node Type	N Nodes	N CPU/ Node	RAM/CPU	CPU RAM/ Node	N GPU/ Node	RAM/GPU	GPU RAM/ Node	Total N GPUs
Puma	Standard	236	94	5 gb	470 gb	-	-	-	-
	High Mem	3 standard	94	32 gb	3008 gb	-	-	-	-
		2 buy-in							
	GPU	8 standard	94	5 gb	470 gb	4	32 gb	128 gb	32
		7 buy-in							28
Ocelote	Standard	400	28	6 gb	168 gb	-	-	-	-
	High Mem	1	48	41 gb	1968 gb	-	-	-	-
	GPU	46	28	8 gb	224 gb	1	16 gb	16 gb	46
El Gato	Standard	130	16	4 gb	62 gb	-	-	-	-

CPU TIME ALLOCATION

Every PI gets an allocation of free [illegible]me, and

Allocation increased as of 3/1/24!!

- **150k** CPU hours on Puma
- **100k** CPU hours on Ocelote

- **Windfall**
  - Unlimited
  - Preemptable

UA HPC Storage Resources

High Performance Storage

- mounted directly on HPC filesystem → *speed!*
- each user gets **`/home`** 50GB
- each PI gets **`/groups`** 500GB
- PIs can request **`/xdisk`** up to 20TB

Rental Storage

- PIs can purchase storage in **`/rental`**
- mounted to file transfer node (must be copied to HPC FS to use)

R-DAS

- PIs can request up to 5TB
- shares can be remotely mounted to personal computers (not on HPC)

Tier 2/AWS

- Long term backup options for archival data (not on HPC)

UA HPC Accounts

PI/Sponsor Accounts

- mostly faculty
- can create and own groups
- receive storage and CPU time allocations

Research Groups

- access both storage and full CPU

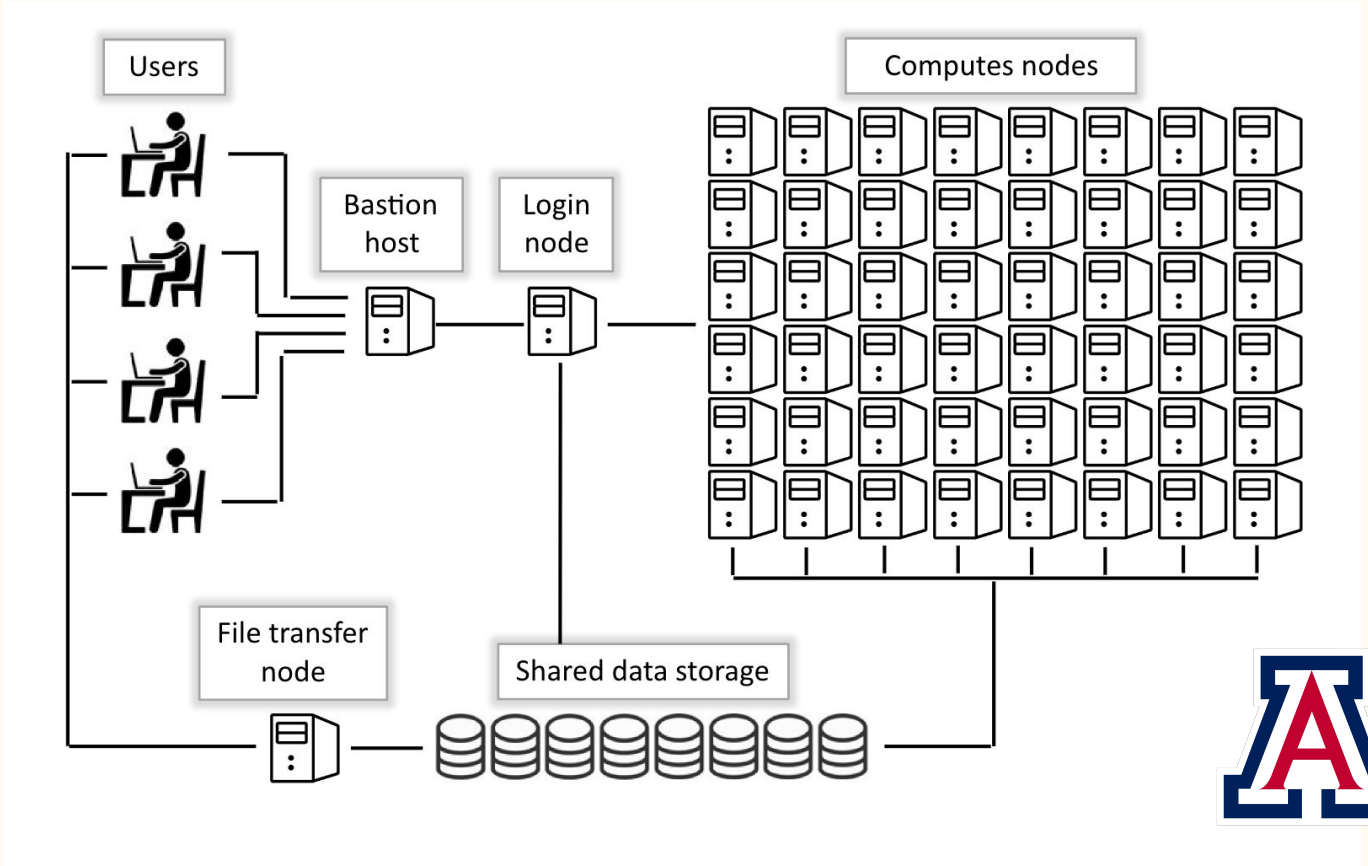
Class Groups

- access only storage and limited CPU

Normal/Sponsored Accounts

- students, post-docs, etc
- must request sponsorship from a PI
- can be added to groups
- given access to storage and compute resources according to group membership

UA HPC Architecture



system access

okay... HPC sounds cool...

but how do I actually access it?

Create an Account

Everyone

- step 1: log into portal.hpc.arizona.edu

Faculty/PIs

- sponsor your own account: https://portal.hpc.arizona.edu/portal/sendlink.php
- put your own email address in the link, and follow the steps

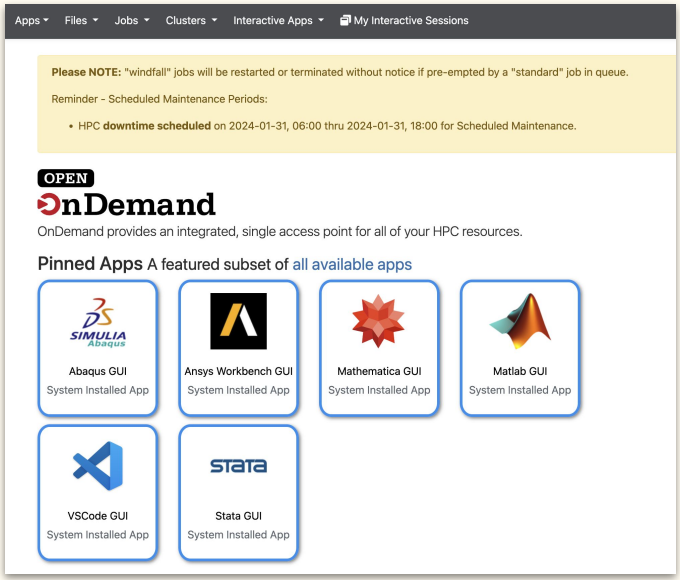
Students, Post-Docs, etc

- request sponsorship from a PI
- put your PI's email address in the above link

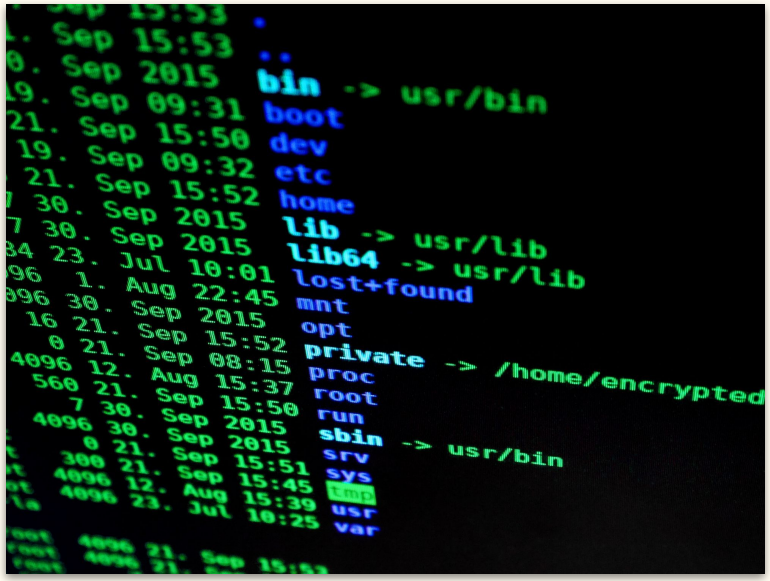
More details in our documentation

Connecting to the HPC

Method 1: Browser



Method 2: Command Line



Open On Demand

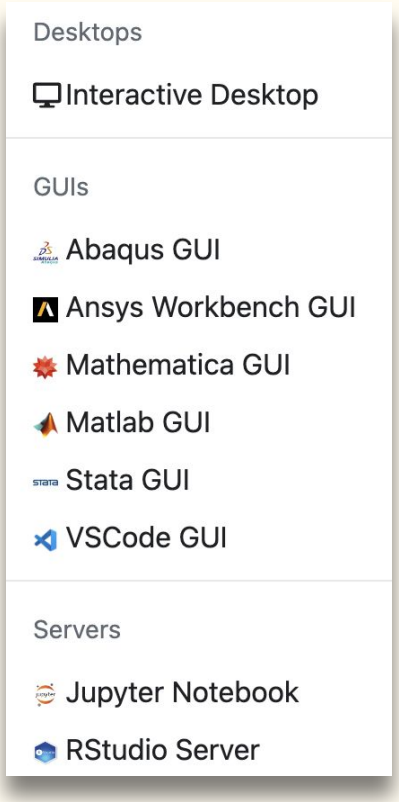
Website: ood.hpc.arizona.edu

Browser-based graphic user portal

- file browser
- interactive desktop
- GUI-based applications
- easy access to terminal

Downside

- can be a little slow and clunky



OPEN OnDemand

OnDemand provides an integrated, single access point for all of your HPC resources.

Pinned Apps A featured subset of all available apps

Abaqus GUI

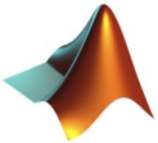
System Installed App

Ansys Workbench GUI

System Installed App

Mathematica GUI

System Installed App



Matlab GUI

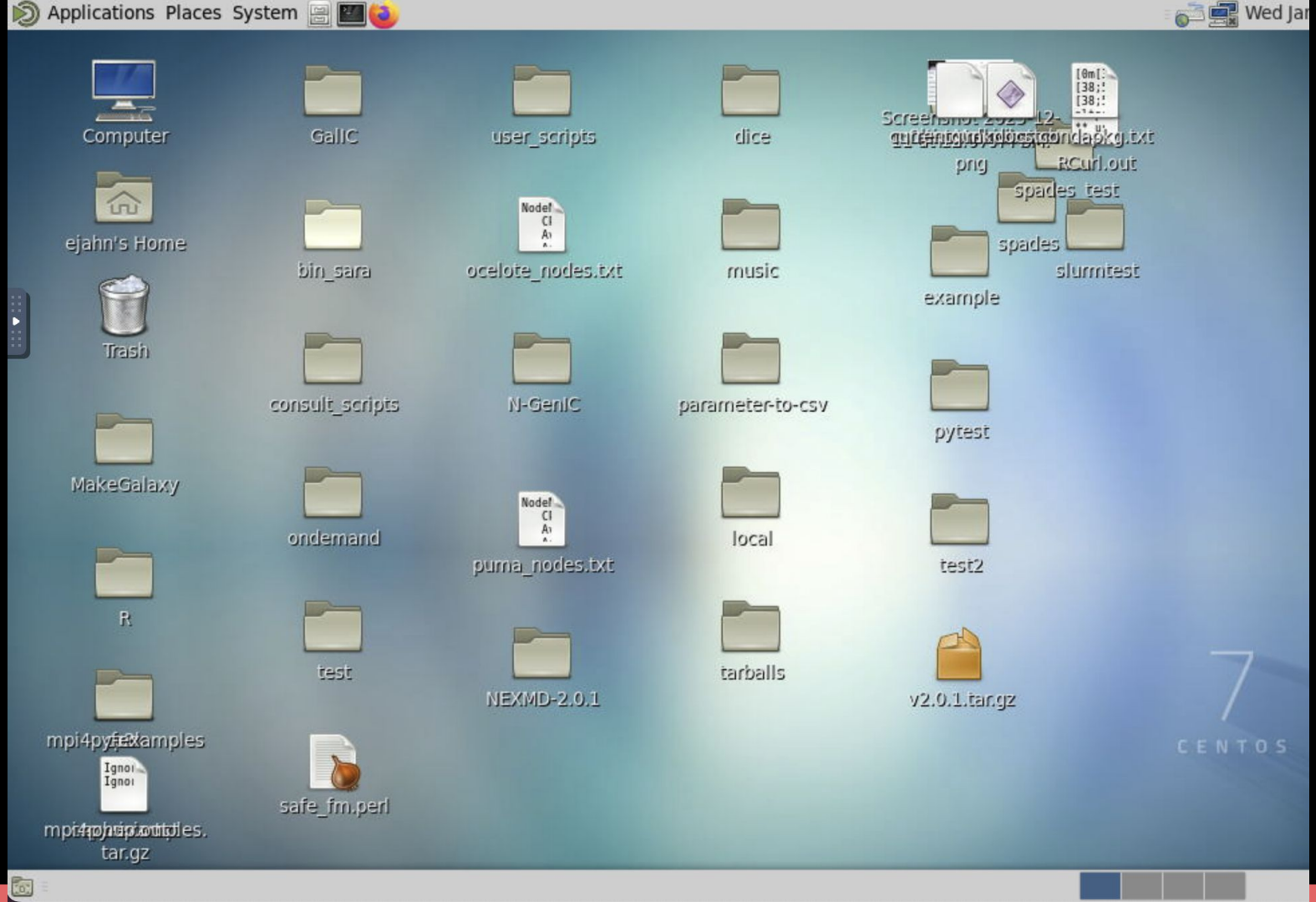
System Installed App

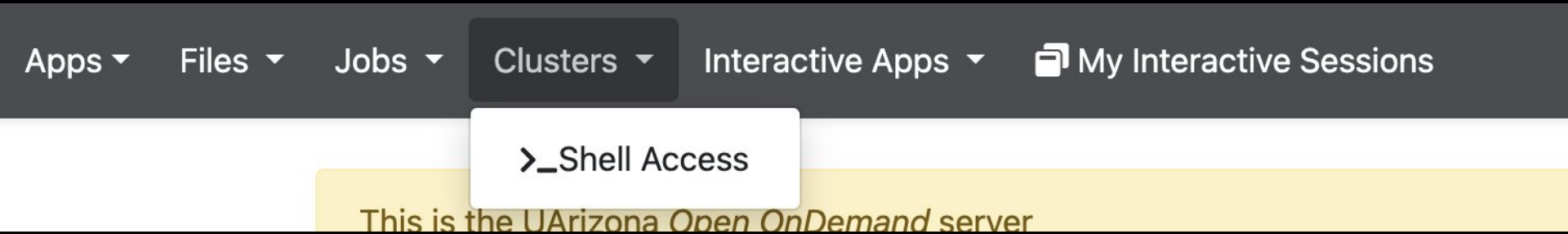
Stata GUI

System Installed App

VSCode GUI

System Installed App


Applications Places System
Wed Jan
Computer
GalIC
user_scripts
dice
Screenshot 2023-12-
png
RCurl.out
ejahn's Home
bin_sara
ocelote_nodes.txt
music
spades_test
spades
slumtest
example
Trash
consult_scripts
N-GenIC
parameter-to-csv
pytest
MakeGalaxy
ondemand
puma_nodes.txt
local
test2
R
test
NEXMD-2.0.1
tarballs
v2.0.1.tar.gz
mpi4py-ex
safe_fm.perl
tar.gz
7
CENTOS




```
***
The default cluster for job submission is Puma
***
Shortcut commands change the target cluster
----------------------------------------
Puma:
$ puma
(puma) $
Ocelote:
$ ocelote
(ocelote) $
ElGato:
$ elgato
(elgato) $
----------------------------------------

(puma) [ejahn@junonia ~]$
```

Command Line Access

Text-only interface

Available on Mac, Windows, and Linux

Uses a *scripting language*

- windows: powershell
- mac/linux: bash (bourne-again shell); zsh
- windows also provides bash shell

Connect to remote server (like HPC) using secure shell protocol: SSH

Can access a terminal through OOD, but is also a program that runs on your personal computer

Command Line Access – *why?*

Faster *(don't waste bandwidth on pesky visuals!)*

More control with use of commands

Write and edit code directly in the terminal using a text editor like vim or nano

Manually access compute nodes and load software

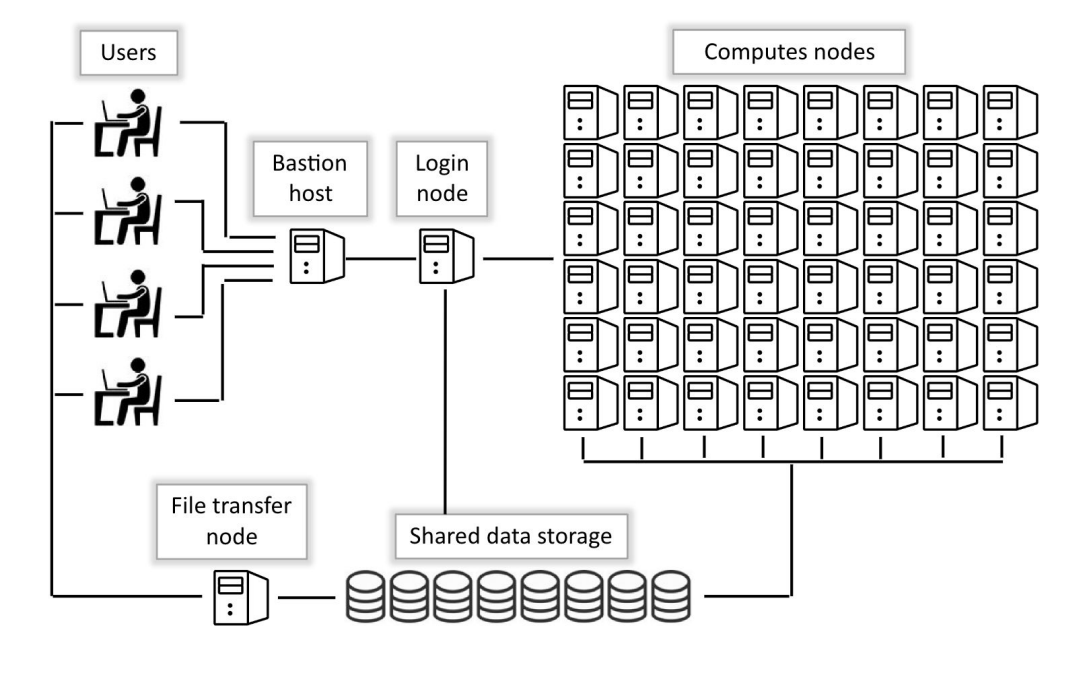
Submit *batch* jobs

- ***no need to babysit your session!***

Downside: slightly steeper learning curve

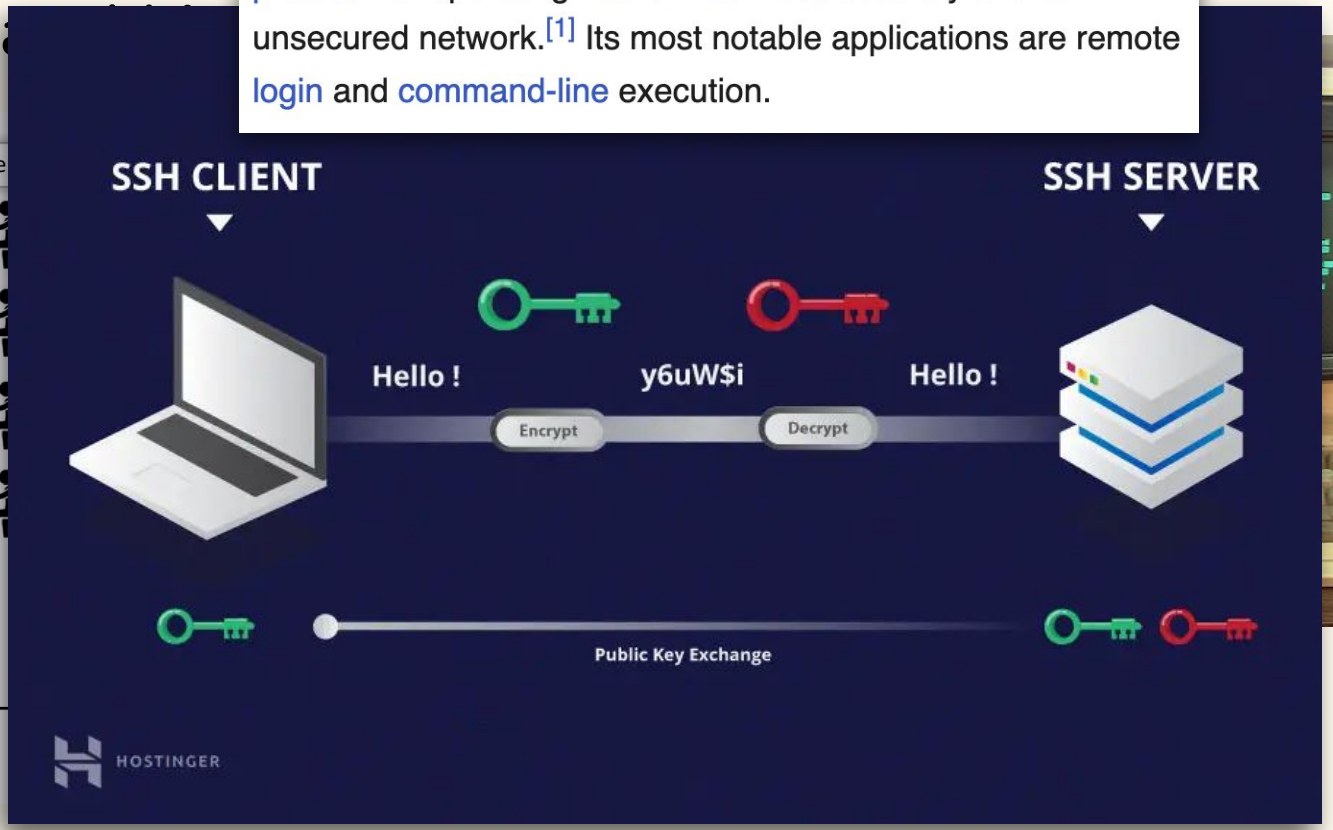
Upside: more powerful user in the long run

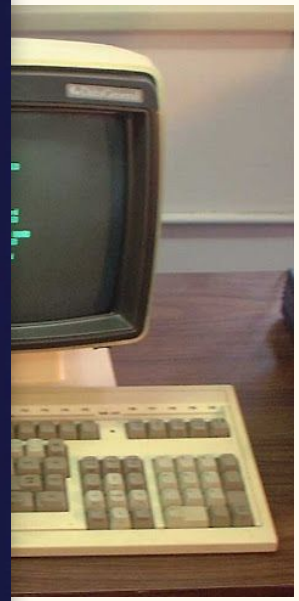
Command Line Access – *how?*



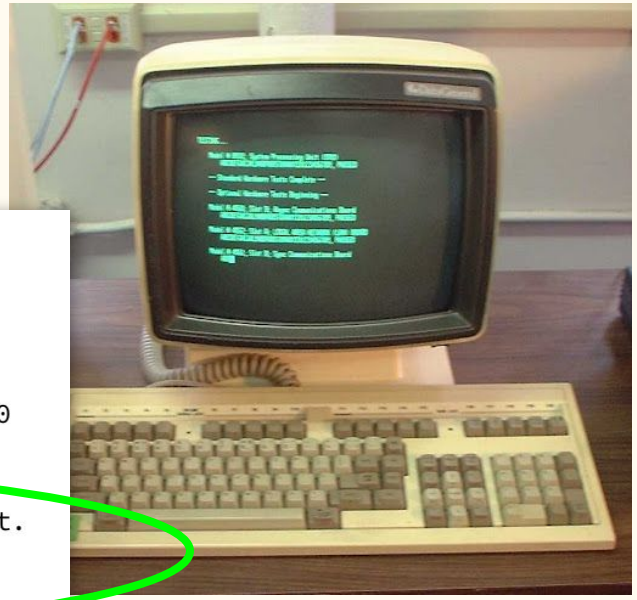
Comma

The **Secure Shell Protocol** (**SSH**) is a cryptographic network protocol for operating network services securely over an unsecured network.[1] Its most notable applications are remote login and command-line execution.





Command Line Access – *how?*



```
~ ssh ejahn@hpc.arizona.edu
Last login: Wed Jan 17 10:14:00 2014 from ip72-201-152-35.ph.ph.cox.net

***
  Reminder - Scheduled Maintenance Periods:
    * HPC downtime scheduled on 2024-01-31, 06:00 thru 2024-01-31, 18:00
      for Scheduled Maintenance.
***
This is a bastion host used to access the rest of the RT/HPC environment.

Type "shell" to access the job submission hosts for all environments
-------------------------------------------------------------------

[ejahn@gatekeeper ~]$
```

Command Line Access – *how?*

```
[ejahn@gatekeeper ~]$ shell
Last login: Wed Jan 17 10:40:39 2024 from gatekeeper.hpc.arizona.edu
***
The default cluster for job submission is Puma
***
Shortcut commands change the target cluster
-----------------------------------------
Puma:
$ puma
(puma) $
Ocelote:
$ ocelote
(ocelote) $
ElGato:
$ elgato
(elgato) $
-----------------------------------------


(puma) ejahn@wentletrap:~
 ->>>
```

how to change cluster

prompt
- (cluster)
- user
- @
- node

login node names
- junonia
- wentletrap

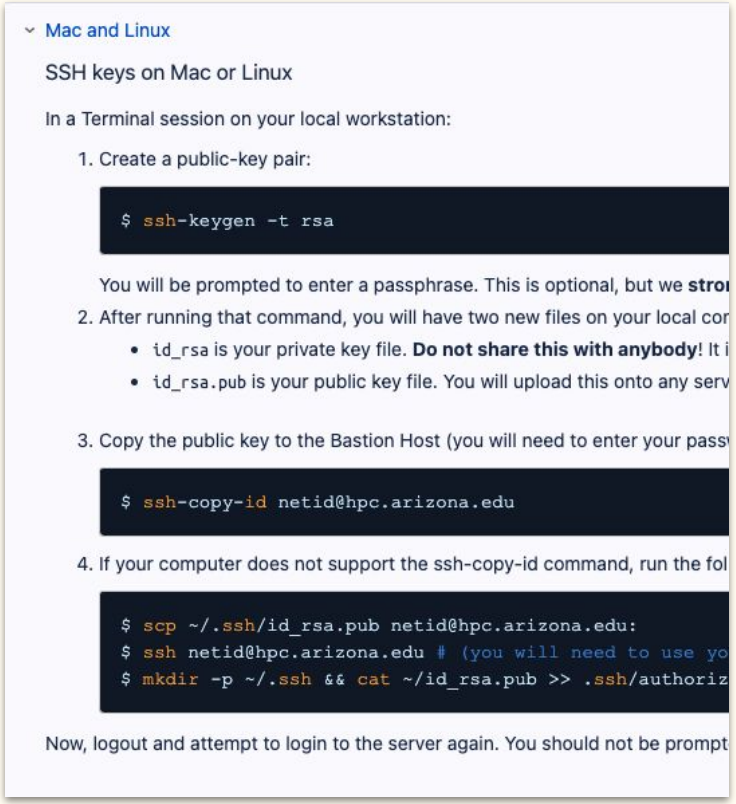
once you see the prompt with a login node name… *you're in!*

now, what to do?

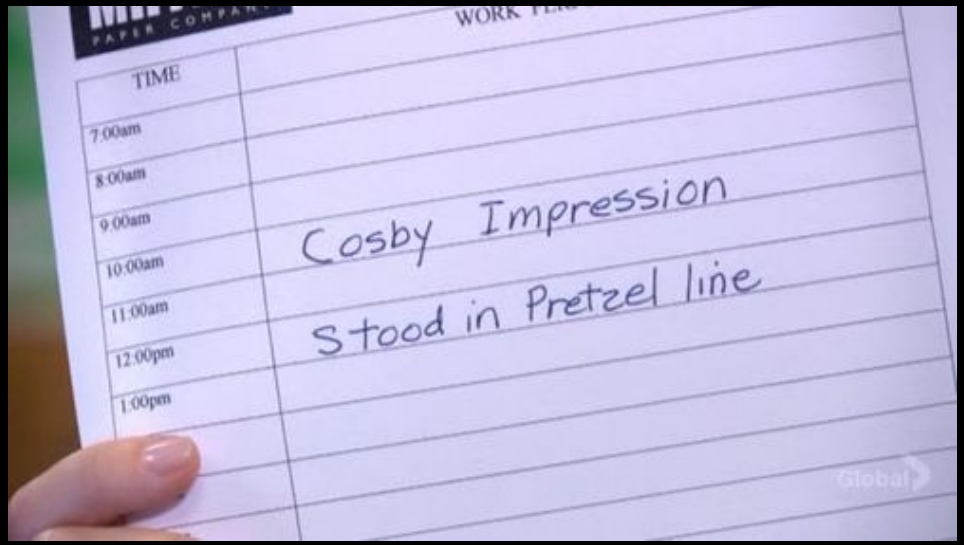
- use **bash commands** to manage files
- **install** code or software needed for your research
  - `note: compile on a compute node!`
- use an **interactive session** for testing
- submit a **batch job**

Easy Access: SSH Keys

- create a **private/public key pair** to place on your local machine and the HPC to reduce the number of times you need to enter your password
- **3 failed password attempts means system lockout 1 hour**
  - ***avoid this with SSH keys!***
- instructions in our docs



job scheduling



Job Scheduling

- Many users
- Many jobs
- Limited resources

→ ***task scheduler*** required to balance the requirements of hundreds of jobs

→ once your job has been requested, it goes to the scheduler, which places it in the ***queue*** according to *when* it was submitted and the *resources* it requested

→ jobs requesting **more resources** will generally take **longer** to start

→ jobs with **high time limits** will generally take **longer** to start

→ sometimes usage is low and jobs start soon

→ sometimes the queue is busy, and it takes a while

Job Scheduling

- Many users
- Many jobs
- Limite

→ ***task schedu*** dreds of jobs

→ once your jo which places it in the ***queue*** according t requested

→ jobs requesting **m** **onger** to start

→ jobs with **high time limits** wi take **longer** to start

→ sometimes usage is low and jobs start soon

→ sometimes the queue is busy, and it takes a while

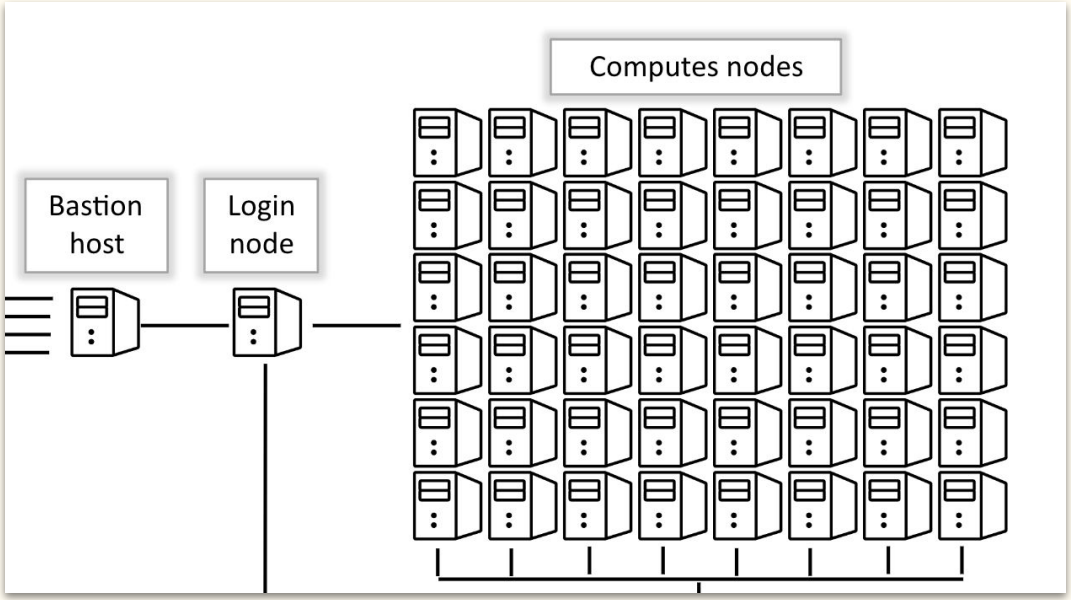
When the system is busy, it may take several hours for jobs on Puma to begin!

Requesting a Job

Login node is used to manage files, etc.

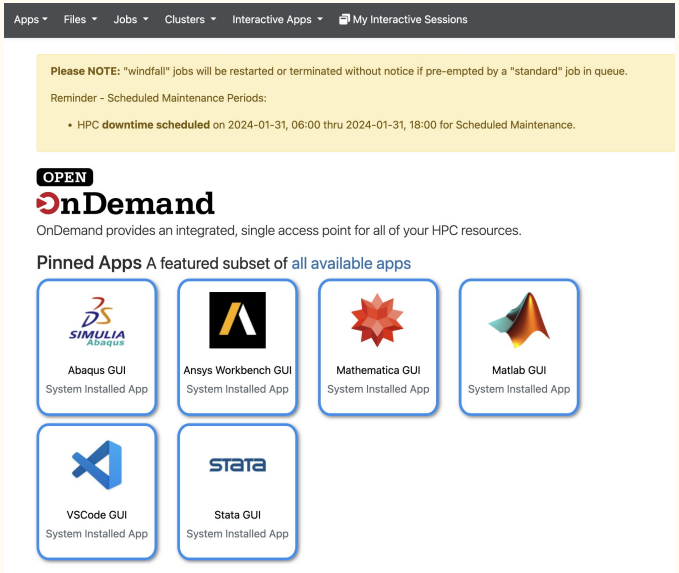
Computations are run using requested **compute resources**.. i.e. a Job

Do not run computations on the login node!

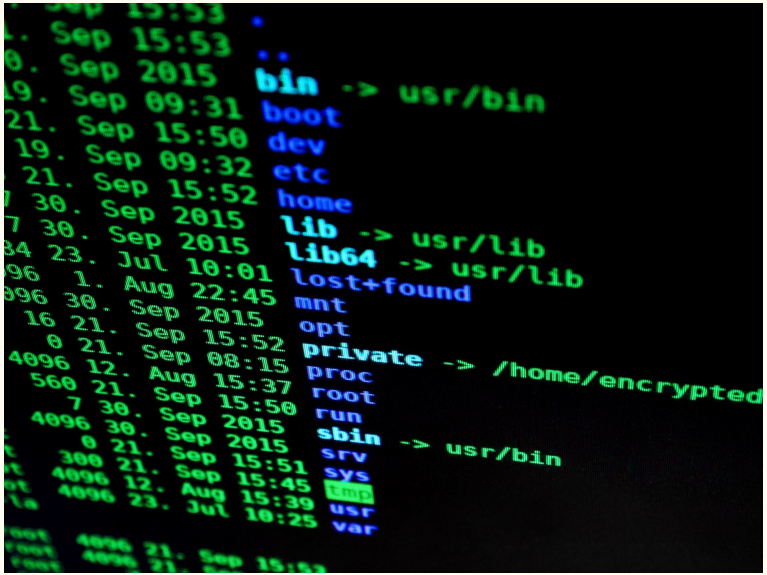


Connecting to the HPC

Method 1: Browser



Method 2: Command Line

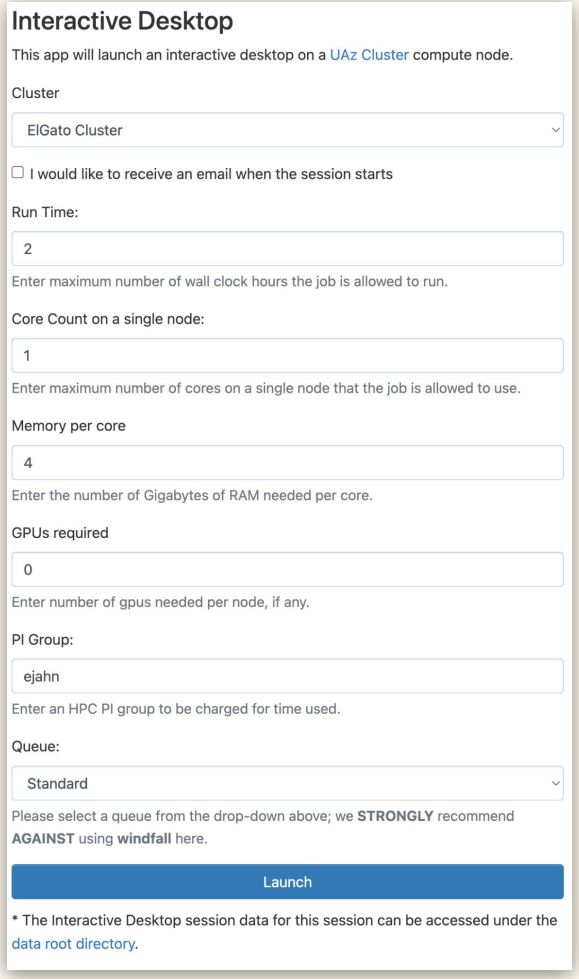


OOD Job Composer

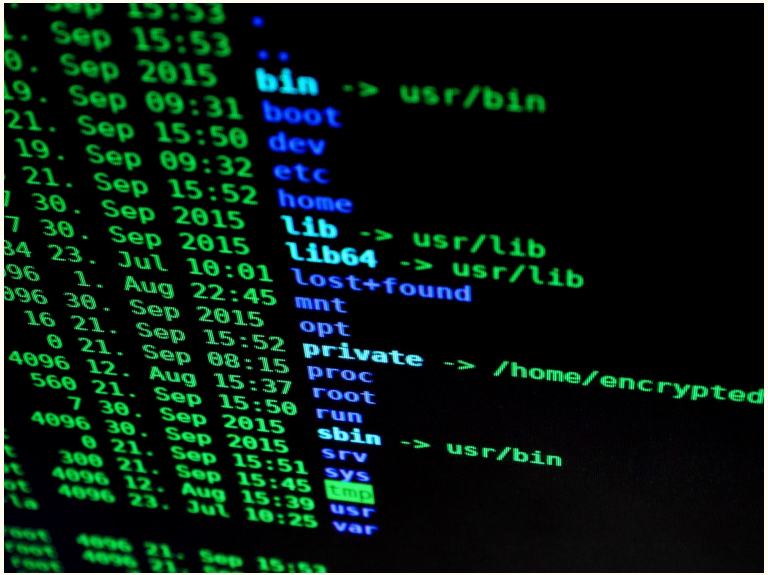
This form will show up when requesting an OOD graphical application

To request a job, you need to tell the system…

- which **cluster** you want to use
- the **CPUs + Memory** you want
- how much **time** you want
- which **account** to charge



Method 2: Command Line



Login node

- Login node is a computer intended for users to prepare and manage computations:
  - submit jobs
  - edit files
  - manage files
  - compile codes - NO
  - small-scale testing - NO

DO NOT run any calculations on the login node

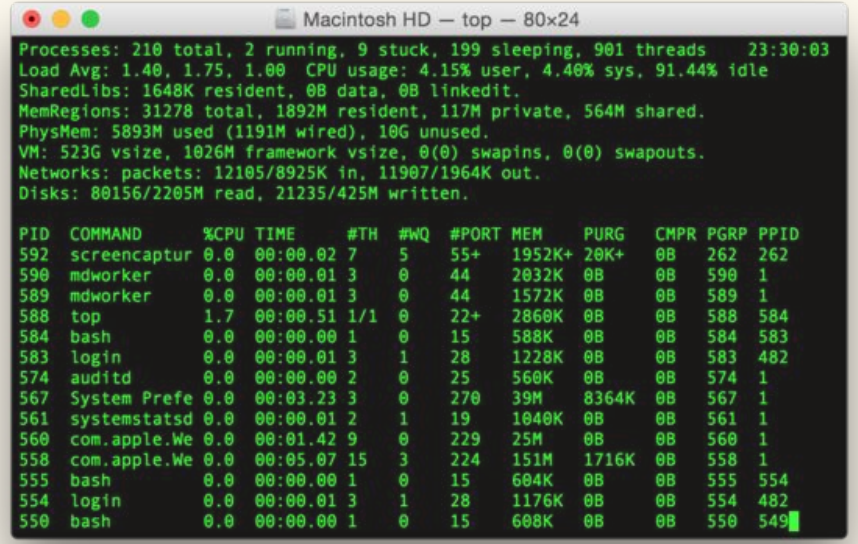
Interactive Session

Terminal-based access

Gain access to a compute node from the login node

command:

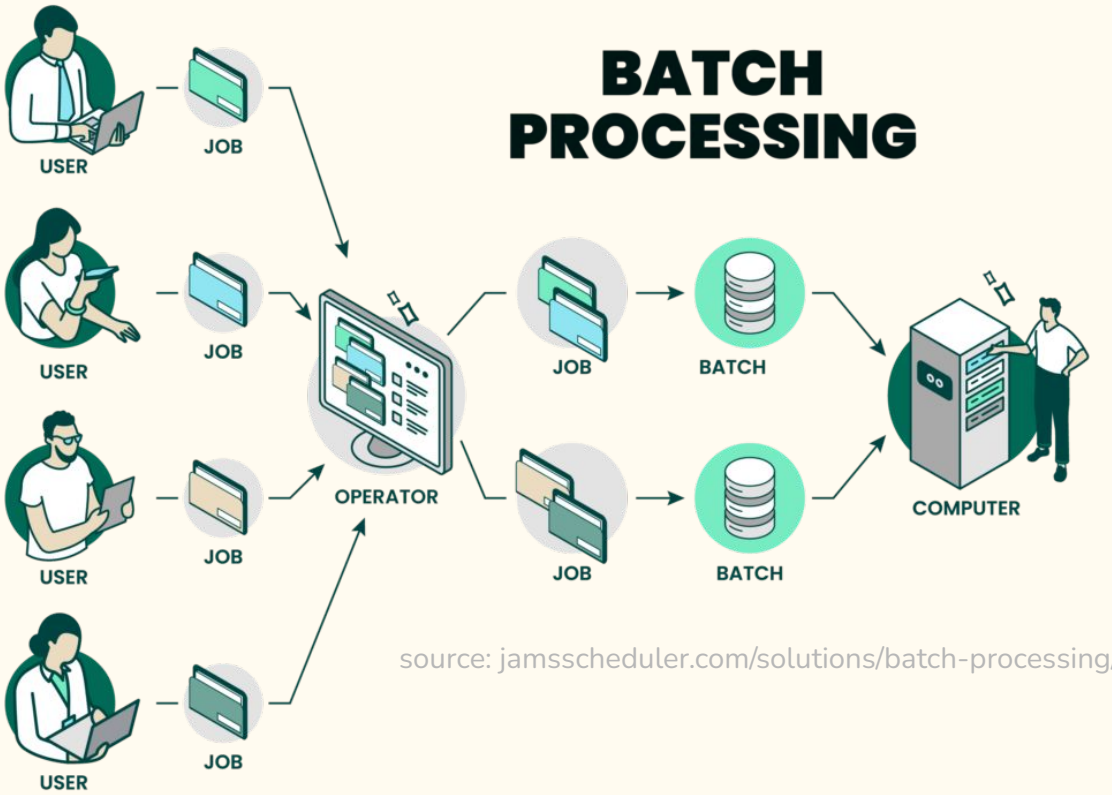
```
interactive

interactive -a <account>
```



Batch Jobs

set-and-forget computing!

1. write a script with resource request and job commands
2. send to scheduler
3. log off and wait!



source: jamsscheduler.com/solutions/batch-processing/

Before requesting a job...

We should know the following things before submitting a request:

- which **cluster** do I want to run on?
- how much **memory** do I want to use?
- how many **CPUs** do I want to use?
- roughly how much **time** do I think the job will take?

Choosing a Cluster

El Gato

- low wait times
- fewer resources
- good for **smaller jobs** or **testing**
- smallest CPU time allocation (7,000hr)

Ocelote

- more resources than El Gato
- **has many nodes with 1 GPU each**
- wait times *vary* depending on usage
- great tool for **many jobs**
- 100,000 hr CPU time

Puma

- most resources in terms of GPU and CPU
- **newest** hardware
- in high demand (*long wait times*)
- 150,000 hr CPU time
- best for **large** production **jobs**

Note on CPUs and Memory

Each compute node has a certain number of processors *physically mounted* on it

Each of those processors has a memory chip *physically connected* to it

That means

Number of CPUs determines Memory*

***think RAM, not disk space!**

Total Mem = Mem/CPU x N(CPU)

Memory is not a continuous scale or arbitrary number!!

Each cluster has different RAM availability

Cluster	Node Type	N Nodes	N CPU/ Node	RAM/CPU	CPU RAM/ Node	N GPU/ Node	RAM/GPU	GPU RAM/ Node	Total N GPUs
Puma	Standard	236	94	5 gb	470 gb	-	-	-	-
	High Mem	3 standard	94	32 gb	3008 gb	-	-	-	-
		2 buy-in							
	GPU	8 standard	94	5 gb	470 gb	4	32 gb	128 gb	32
		7 buy-in							28
Ocelote	Standard	400	28	6 gb	168 gb	-	-	-	-
	High Mem	1	48	41 gb	1968 gb	-	-	-	-
	GPU	46	28	8 gb	224 gb	1	16 gb	16 gb	46
El Gato	Standard	130	16	4 gb	62 gb	-	-	-	-

Interactive vs. Batch Jobs

Interactive

- requested with `'interactive'` or `'salloc'` command
- resources described in **command**
- input commands and receive output in **real time**
- can *change* what it's doing after it starts
- job **cancelled** if connection lost

Both

- access compute nodes by submitting a request to **slurm**
- can run intense calculations
- subject to **time limit**
- access to standard and windfall partitions
- subject to **queue** *(wait times)*

Batch

- requested with `'sbatch'`
- requires a **batch script**
  - `describe resource request`
  - `load software`
  - `initiate the calculation`
- saves **output to files**
- *cannot change* anything once submitted
- will run **without supervision** or active connection to HPC

Use cases

Interactive

- install and compile software
- test new code
- smaller scale computations
- iterative analyses
- debugging

Batch

- jobs that will take a long time to complete
- jobs that require significant resources
- jobs that may end up waiting in the queue for a while
- code that is known to work and requires minimal debugging

Both

- require use **BASH** in the terminal

so... what is bash?

BASH: Bourne Again SHell

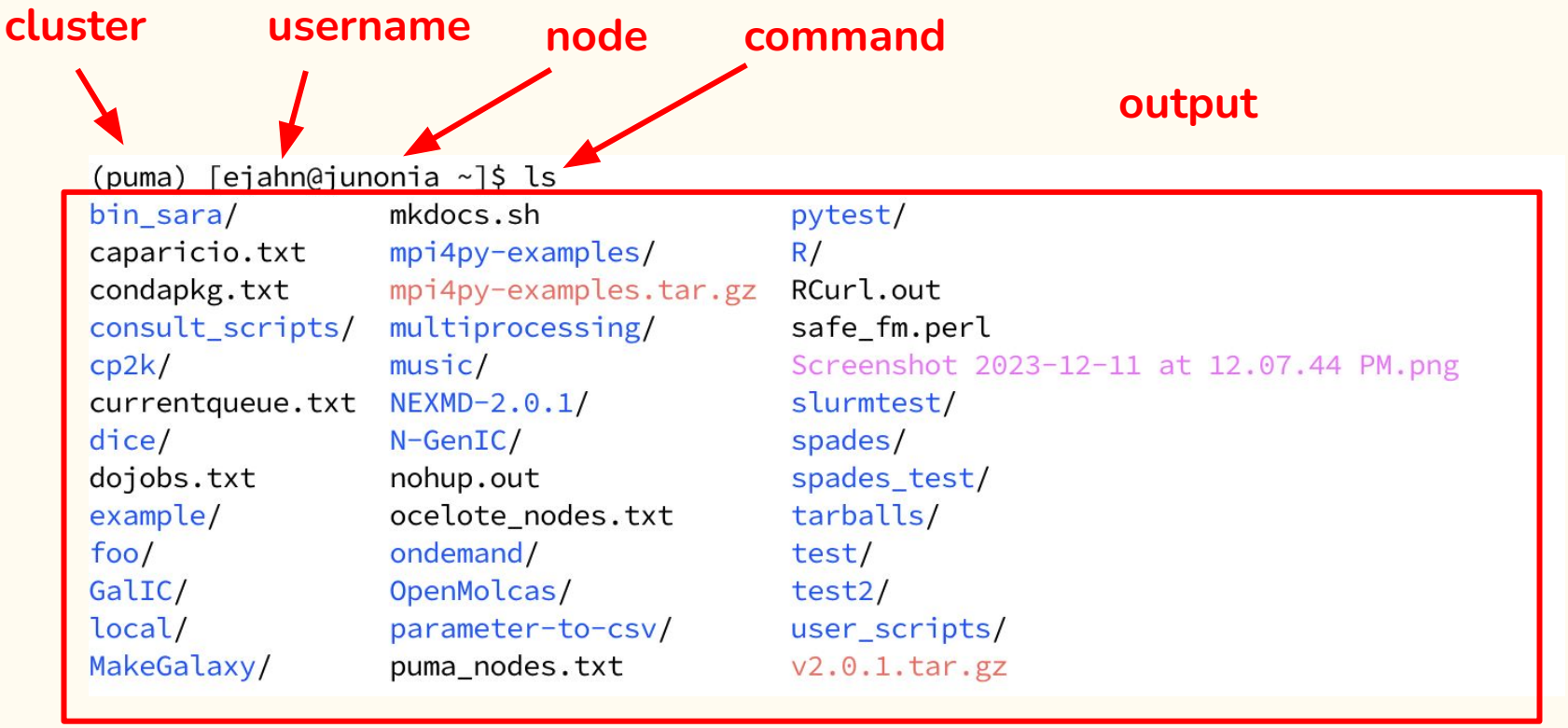
Scripting language for **Unix** shells

Code version of a file browser and application launcher

> Just like a file browser, you always “reside” in a certain folder and can see its contents
>
> Can change your “location”, and move folders/files

```
(puma) [ejahn@junonia ~]$ pwd
/home/u16/ejahn
```

print
working
directory


cluster
username
node
command
output
(puma) [ejahn@junonia ~]$ ls
bin_sara/ mkdocs.sh pytest/
caparicio.txt mpi4py-examples/ R/
condapkg.txt mpi4py-examples.tar.gz RCurl.out
consult_scripts/ multiprocessing/ safe_fm.perl
cp2k/ music/ Screenshot 2023-12-11 at 12.07.44 PM.png
currentqueue.txt NEXMD-2.0.1/ slurmtest/
dice/ N-GenIC/ spades/
dojobs.txt nohup.out spades_test/
example/ ocelote_nodes.txt tarballs/
foo/ ondemand/ test/
GalIC/ OpenMolcas/ test2/
local/ parameter-to-csv/ user_scripts/
MakeGalaxy/ puma_nodes.txt v2.0.1.tar.gz

current lowest-level folder

```
(puma) [ejahn@junonia ~]$ cd MakeGalaxy/

(puma) [ejahn@junonia MakeGalaxy]$ ls
brunscript   diskset.o                    forcetree.h  init.c        read_param.c
bulge.c      distfunc.c                   forcetree.o  init.o        read_param.o
bulge.o      distfunc.o                   gas.c        main.c        save.c
bulgeset.c   effmodel.c                   gas.o        main.o        save.c~
bulgeset.o   effmodel.o                   globvars.h   misc.c        save.o
cooling.c    escape.c                     globvars.o   misc.o        sbc.txt
cooling.h    escape.o                     halo.c       nrsrc/        structure.c
cooling.o    example_input.txt            halo.o       prototypes.h  structure.c~
deldisp.dat  example_input.txt~           haloset.c    README.txt    structure.o
disk_b.c     example_input_wonkytabs.txt  haloset.o    README.txt~   tests/
disk.c       force.c                      Makefile     sbc.c         toomre.c
disk.o       force.o                      Makefile~    sbc.o         toomre.o
diskset.c    forcetree.c                  Makefile.old sbc.txt~      toomre_orig.c

(puma) [ejahn@junonia MakeGalaxy]$ pwd
/home/u16/ejahn/MakeGalaxy
```

Bash Cheat Sheet

Navigating the File System

cd [directory]	Change directory
pwd	Print working directory
ls [options] [directory]	List directory contents
mkdir [directory]	Create a new directory
rmdir [directory]	Remove a directory
cp [source] [destination]	Copy files or directories
mv [source] [destination]	Move or rename files or directories
rm [options] [file]	Remove files or directories
touch [file]	Create an empty file

Archiving and Compression

tar [options] [files/directories]	Create or extract tar archives
gzip [file]	Compress a file
gunzip [file.gz]	Decompress a gzipped file
zip [archive.zip] [files/directories]	Create a zip archive
unzip [archive.zip]	Extract files from a zip archive

Get more cheat sheets and other Linux content at LinuxStans.com

File Manipulation

cat [file]	Output the contents of a file
head [options] [file]	Output the first lines of a file
tail [options] [file]	Output the last lines of a file
less [file]	View the contents of a file interactively
grep [pattern] [file]	Search for a pattern in a file
wc [options] [file]	Count the number of lines, words, or characters in a file

Permissions

chmod [permissions] [file]	Change the permissions of a file or directory
chown [user:group] [file]	Change the owner and group of a file or directory
chgrp [group] [file]	Change the group of a file or directory
umask [mask]	Set the default file permissions for newly created files

Process Management

ps [options]	Display information about active processes
kill [process_ID]	Terminate a process
top	Display and manage the top processes
bg [job_ID]	Move a job to the background
fg [job_ID]	Bring a background job to the foreground

Command Structure

<command> <options> <object>

example: ‘ls -lha ..’

- **command** = ‘ls’
  - ls = list contents
- **options** = ‘-lha’
  - options start with ‘-’ and are often called ‘flags’
  - l = long; h = human readable file sizes; a = all files (including hidden ones)
- **object** = ‘..’
  - shorthand for ‘the directory that contains the current directory’
  - single dot = current directory
  - can be left blank to imply current directory for this command

Other Useful Commands

puma/ocelote/elgato : switch cluster

interactive -a <account> : request interactive session on current cluster

sbatch <file> : submit a batch job to the current cluster

uquota : get a summary of your storage usage

va : view allocation – get a summary of the CPU time used on current cluster

past-jobs -d <N> : view jobs from previous N days

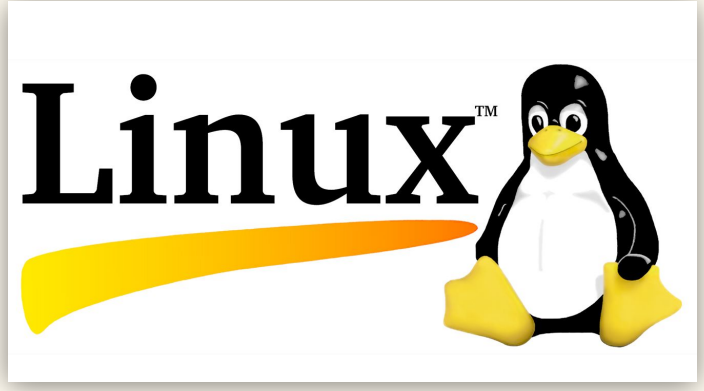
job-history <job ID> : detailed report about given job

...and many more

Intro to Linux

BASH is a shell language for Unix systems

- Linux is a type of Unix system
- So is Mac
- Windows is not



Strictly speaking, Linux is a *kernel* that many different operating systems are built on

Intro to Linux

Most servers in the world run some type of Linux

Linux is also becoming more popular as a daily use OS (for example Debian, Linux Mint, Arch, Fedora, and many many more...)

The UA HPC currently runs CentOS 7

Intro to Linux: What you need to know

- it’s not Mac
- it’s not Windows
- you are one user on a very large shared computer
  - there are things you do not have permission to do (e.g. sudo)
  - there are things that are possible to do, but highly discouraged
    - mainly: don’t run too many things on the login node
  - you can install software for yourself, but not for the entire system (we can do that for you)
- if you want to learn how to do something: ask a friend
  - Fellow researchers
  - HPC Consulting
  - Google, Chat GPT, etc

Activity: Let’s run a batch job!

Let’s write a basic python script, then submit it to the queue

Process:

1. write a python script
2. write a batch script
3. submit the batch script

Python Script

You can write anything you want! It can be as simple as print(‘hello world’), or you can make it more interesting.. up to you ;)

Steps:

1. log onto the HPC
2. create a blank python file: ‘**touch myscript.py**’
3. edit the script: ‘**nano myscript.py**’
   a. feel free to use vim or emacs or any other text editor if you are more comfortable with one of those
   b. add something simple like

```
print(“hello world!”)
```

4. be sure to save it!

Batch Script

add the following to sections to a new file called myscript.slurm

1. Directives: tell the scheduler what resources you want

```
#!/bin/bash
#SBATCH --job-name=test
#SBATCH --output=test_%A.out
#SBATCH --error=test_%A.err
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --partition=standard
#SBATCH --account=<your_account>
#SBATCH --time=00:05:00
```

2. Load your modules

```
module load python
```

3. Call your command

```
python myscript.py
```

Submit!

send to scheduler

```
sbatch myscript.slurm
```

check the queue

```
squeue --user=<my netid>
```

view the output files

```
cat <name>.out
```

```
cat <name>.err
```

check the job-history report

```
job-history <job id>
```

Congrats! You just ran your first batch job!!!

UA HPC Resources and Help

The HPC Consult Team is here for you!

- **Documentation**: docs.hpc.arizona.edu
- **GitHub**: ua-researchcomputing-hpc.github.io
- **ServiceNow**: HPC Support and Consulting Request
- **Email**: hpc-consult@list.arizona.edu
- **Office Hours**: every Wednesday 2-4pm on GatherTown