

# The Illinois Campus Cluster

A Short Tutorial by Prof Jared Hutchins

# Have you or someone you love experienced these symptoms?

- **Data won't load because your computer's memory is too small?**
- **Repetitive task is taking too long?**
- **Computer labs are already being used by other people?**



# Ask your advisor if *The Illinois Campus Cluster* is right for you!

- **Greater access to memory and CPU than your own computer!**
- **Ability to read in large datasets!**
- **Ability to parallelize code!**





# Our Department's Computing Resources

- Decentralized
- Resources are first come first serve, regardless of needs





# The Campus Cluster

- Central planner
- Resources are distributed according to needs

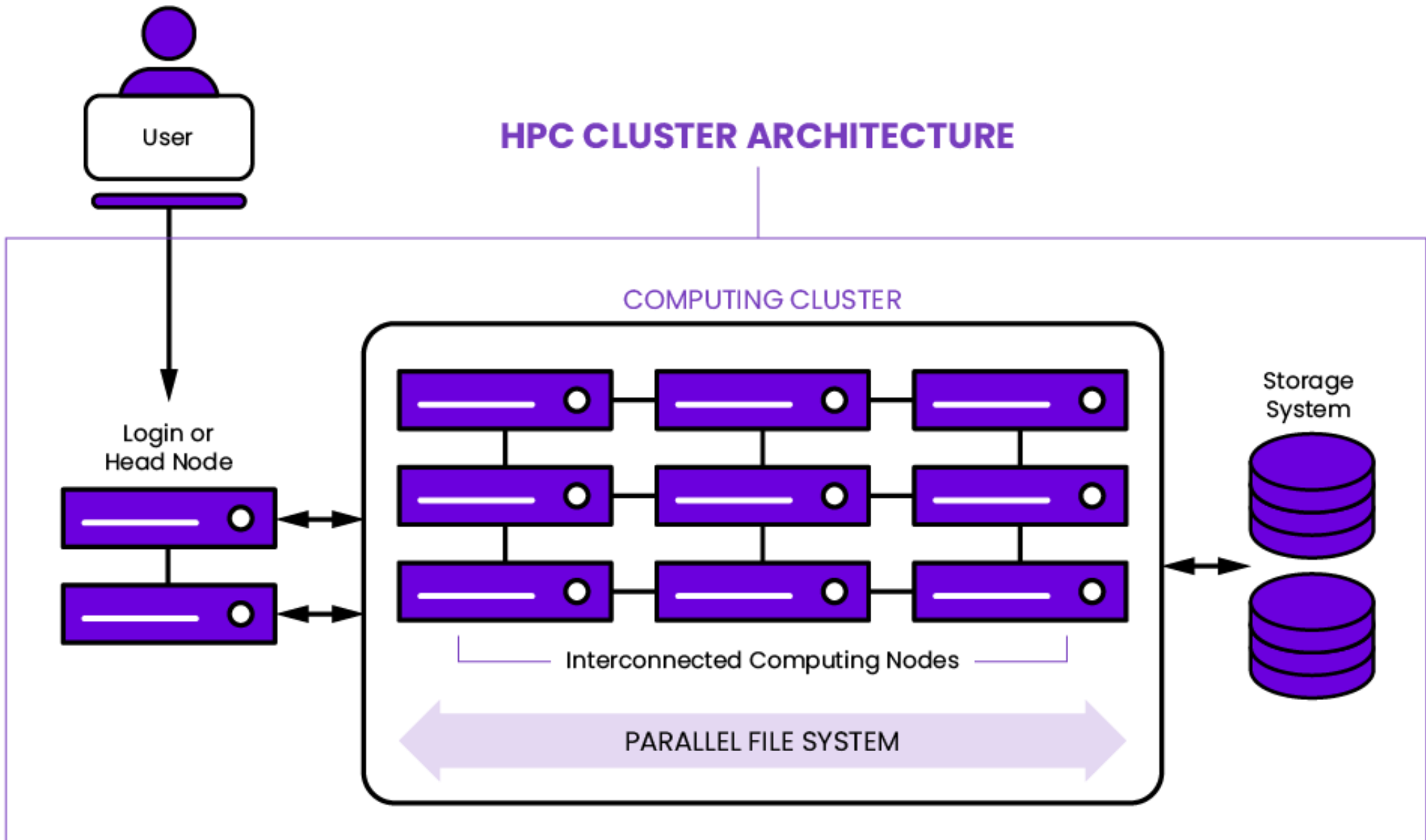




# What is a supercomputing cluster?

- A “high performance computing” (HPC) cluster = connected **nodes** (computers)
- To do big computations, you can request specific nodes that have more **memory or CPU than your own computer.**





# Reasons to learn to use the cluster:

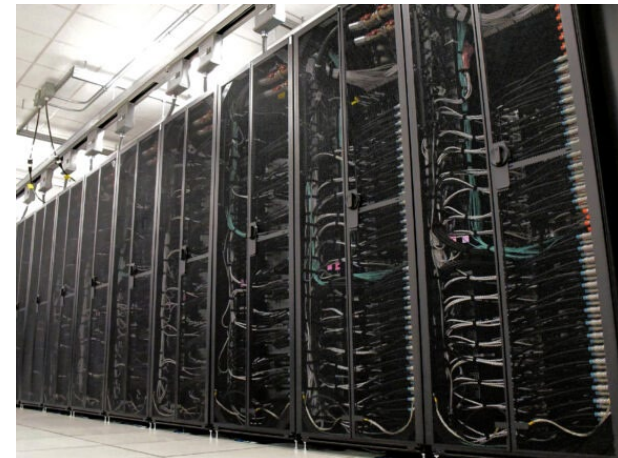
- Our datasets are getting bigger, not smaller.
- The college pays big money to have this resource (and may stop paying for it if we don't use it!)
- A much more efficient way to allocate resources to people!

**Illinois Campus Cluster Program**

AT THE UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

**I ILLINOIS**

NCSA | National Center for  
Supercomputing Applications

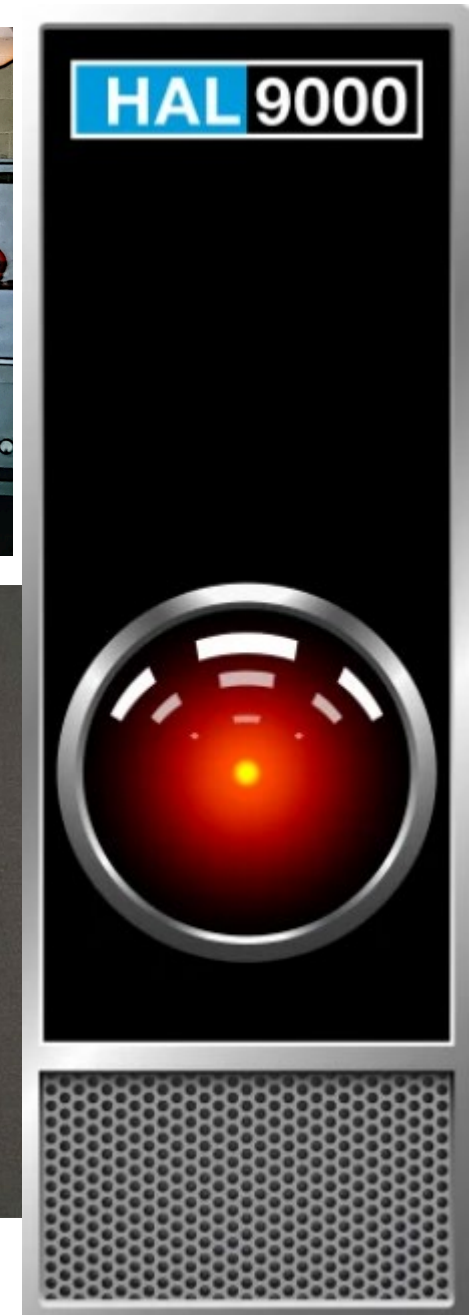




# A quick history

UIUC has been pioneering research in computers since the 1950s:

- 1952: **ILLIAC I was created**, first supercomputer built by a US educational institution.
- 1986: **National Center for Supercomputer Applications (NCSA)** started.
- 1992: year that HAL 9000 was made in Urbana, Illinois in 2001: *A Space Odyssey*



# So what's the catch?

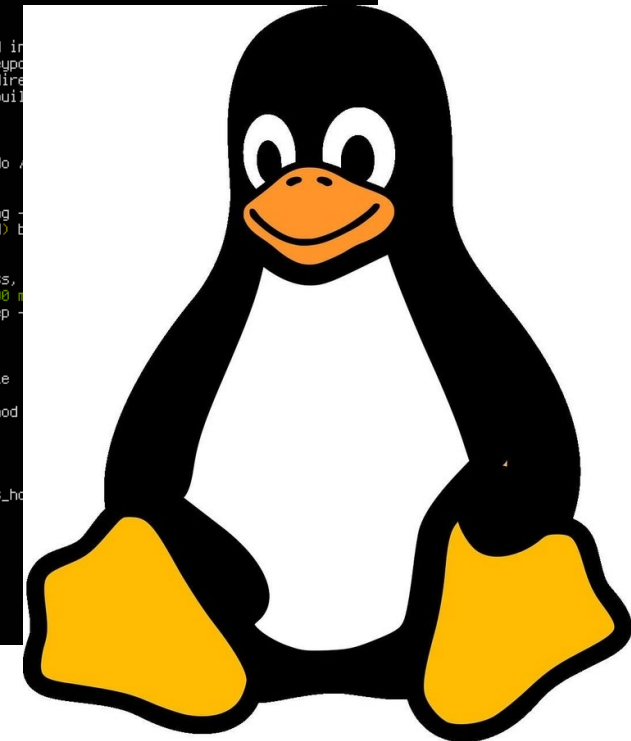
1. You have to know how to ***remote*** into the computer cluster (***no pointing and clicking!***)
2. You have to know how to write a ***batch script*** (limited ability to “interactively program”)
3. You have to set up your computation on a Linux computer (sorry Windows).



# Bash Scripting

- Most compute clusters use **Linux** as their operating system
- At minimum, you need to know how to navigate using **bash**.
- Usually just need a handful of commands.

```
mar@mar$ pwd
/home/mar
mar@mar$ cd /usr/portage/app-shells/bash
mar@mar$ ls -al
total 130
drwxr-xr-x 3 portage portage 1024 Jul 25 10:06 .
drwxr-xr-x 33 portage portage 1024 Aug 7 22:39 ..
-rw-r--r-- 1 root root 35808 Jul 25 10:06 ChangeLog
-rw-r--r-- 1 root root 27002 Jul 25 10:06 Manifest
-rw-r--r-- 1 portage portage 4645 Mar 23 21:37 bash-3.1_p17.ebuild
-rw-r--r-- 1 portage portage 5977 Mar 23 21:37 bash-3.2_p39.ebuild
-rw-r--r-- 1 portage portage 6151 Apr 5 14:37 bash-3.2_p48-r1.ebuild
-rw-r--r-- 1 portage portage 5988 Mar 23 21:37 bash-3.2_p48.ebuild
-rw-r--r-- 1 portage portage 5643 Apr 5 14:37 bash-4.0_p10-r1.ebuild
-rw-r--r-- 1 portage portage 6230 Apr 5 14:37 bash-4.0_p10.ebuild
-rw-r--r-- 1 portage portage 5648 Apr 14 05:52 bash-4.0_p17-r1.ebuild
-rw-r--r-- 1 portage portage 5532 Apr 8 10:21 bash-4.0_p17.ebuild
-rw-r--r-- 1 portage portage 5660 May 30 03:35 bash-4.0_p24.ebuild
-rw-r--r-- 1 root root 5660 Jul 25 09:43 bash-4.0_p28.ebuild
drwxr-xr-x 2 portage portage 2048 May 30 03:35 files
-rw-r--r-- 1 portage portage 468 Feb 9 04:35 metadata.xml
mar@mar$ cat metadata.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pkgmetadata SYSTEM "http://www.gentoo.org/dtd/metadata.dtd">
<pkgmetadata>
  <herd>base-system</herd>
  <use>
    <flag name='bashlogger'>Log ALL commands typed in
      used in restricted environments such as honeypots
    <flag name='net'>Enable /dev/tcp/host/port redirection
    <flag name='plugins'>Add support for loading built-in
      'enable'</flag>
  </use>
</pkgmetadata>
mar@mar$ sudo
Password:
* status: started
mar@mar$ ping -c 1 rr.esams.wikimedia.org
PING rr.esams.wikimedia.org (91.198.174.2) 56(84) bytes of data:
64 bytes from rr.esams.wikimedia.org: icmp_seq=1 ttl=64 time=0.000 ms
--- rr.esams.wikimedia.org ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0.000 ms
rtt min/avg/max/mdev = 49.820/49.820/49.820/0.000 ms
mar@mar$ grep -r /dev/sd[a-c] /boot
/dev/sda1: /boot
/dev/sda2: none
/dev/sda3: /
mar@mar$ date
Sat Aug 8 02:42:24 MSD 2009
mar@mar$ lsmod
Module                Size  Used by
rdis_wlan              23424  0
rdis_host              8696   1 rdis_wlan
cdc_ether               5672   1 rdis_host
usbnet                 18688   3 rdis_wlan,rdis_host
parport_pc             38424   0
fglrx                 2388128  20
parport                39648   1 parport_pc
iTCO_wdt               12272   0
i2c_i801                9380   0
mar@mar$
```



# Software on the cluster

Cluster supports:

- R
- Python
- Matlab
- Mathematica

[Full List](#)

**No STATA** (sorry STATA people)





# What you should check before using the cluster

- Can my code be made more memory efficient?
- Can I fix a speed bottleneck to make the code run faster?
- Can I vectorize any operations?
- Can I run parallel on my own computer?

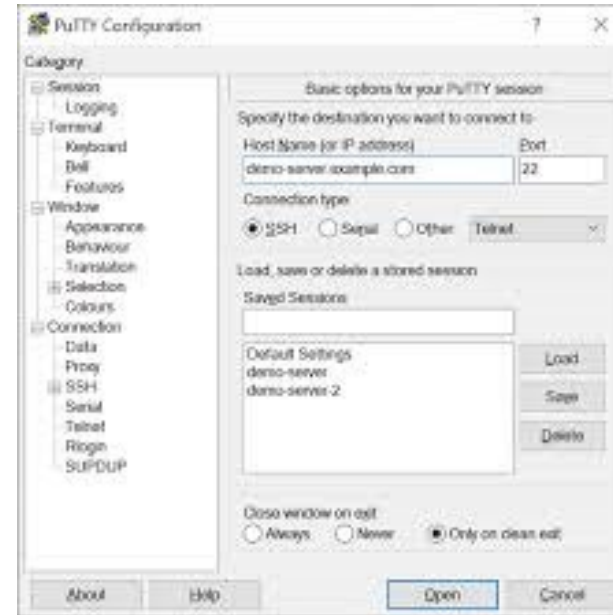


Steps we will go over:



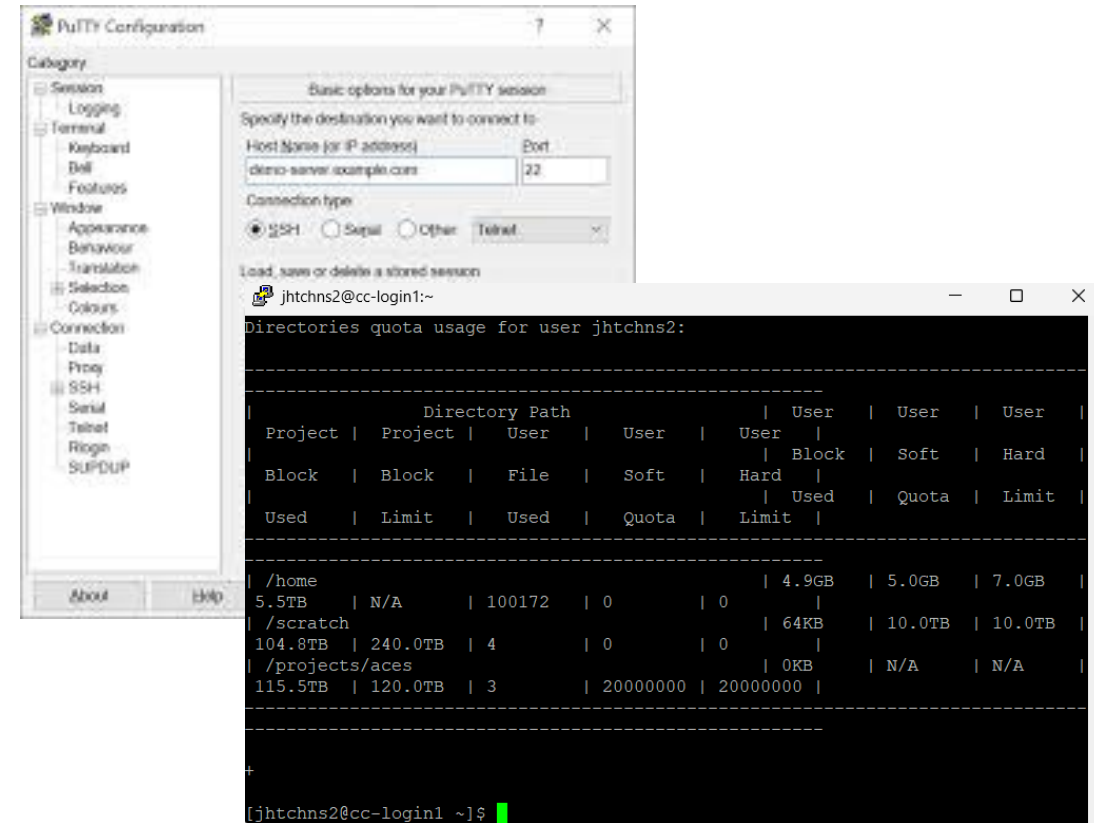
# Steps we will go over:

1. Login to the head node
  - I will demonstrate using the terminal and Windows SSH client “PuTTY”



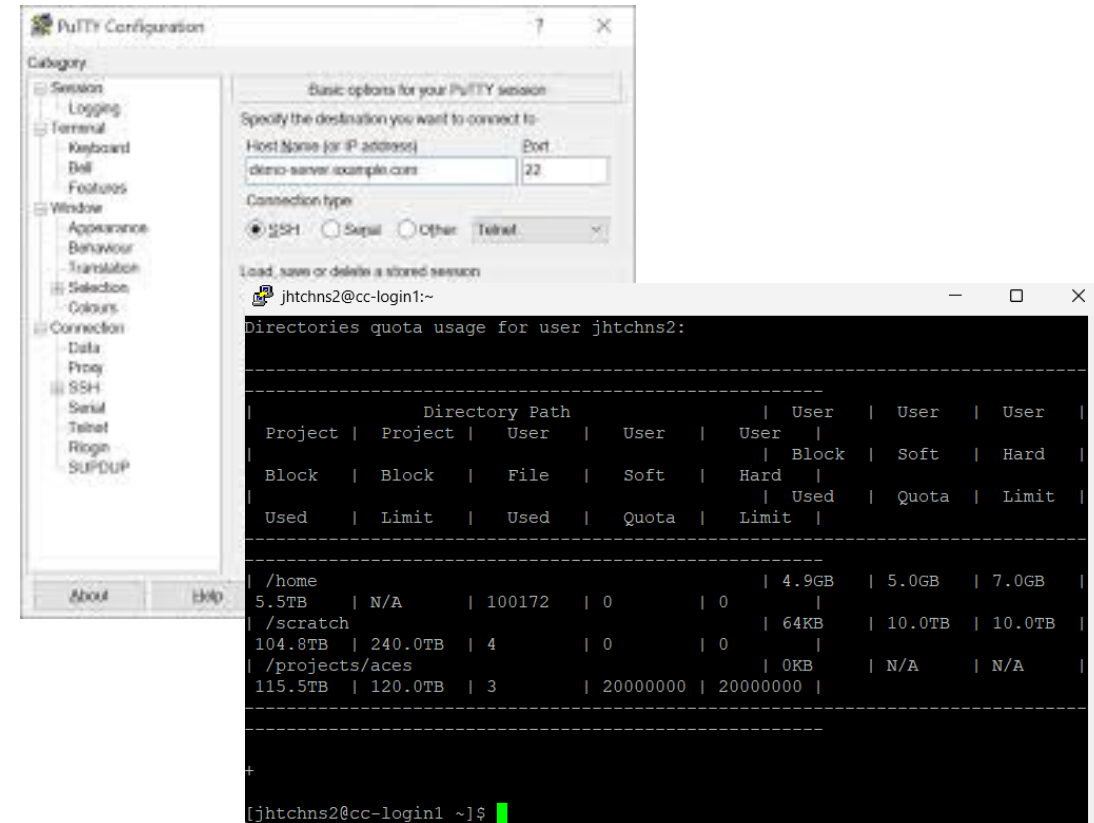
# Steps we will go over:

1. Login to the head node
  - I will demonstrate using the terminal and Windows SSH client "PuTTY"
2. Move around the cluster
  - I will demonstrate some basic bash commands and important folders.



# Steps we will go over:

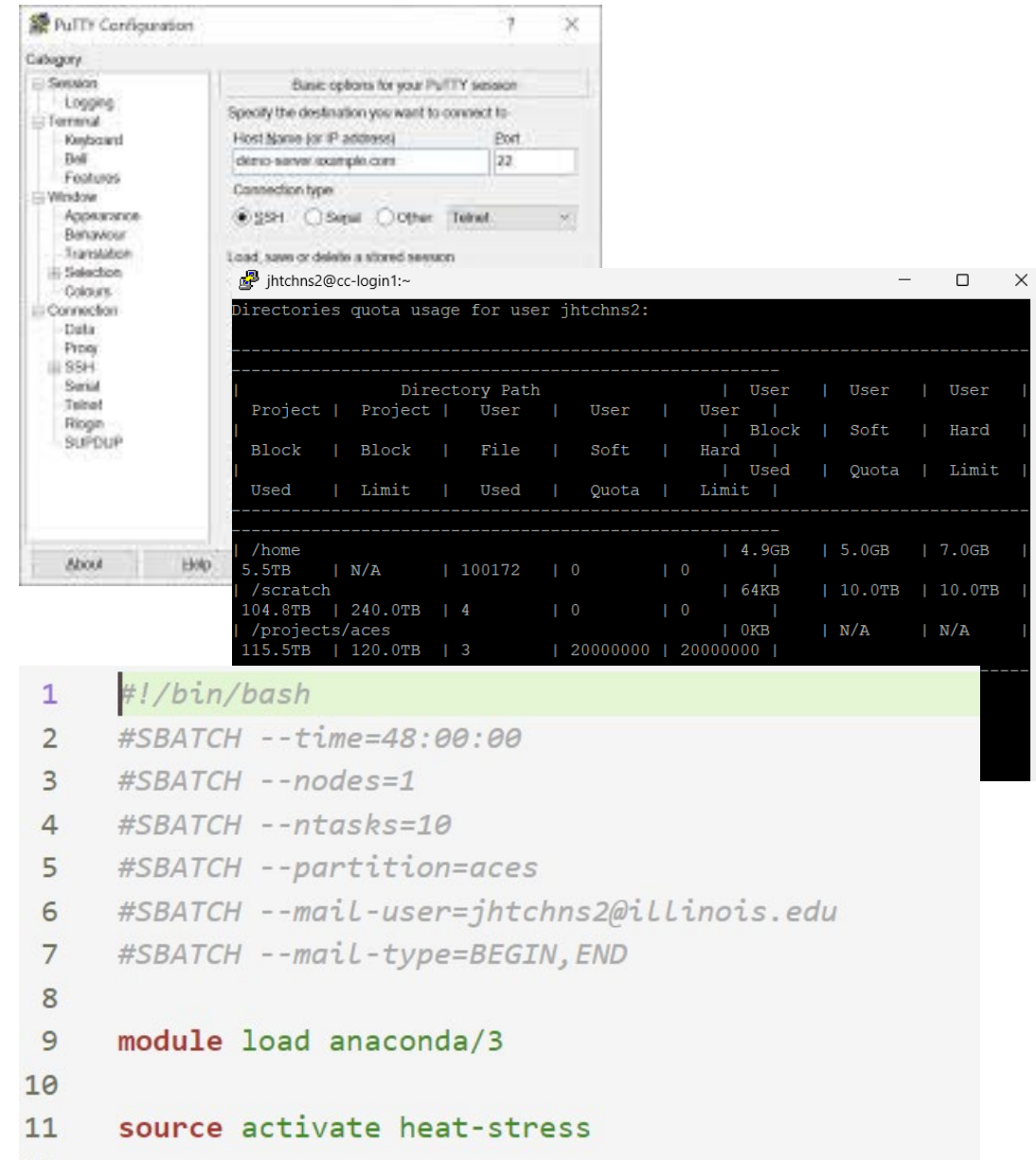
1. Login to the head node
  - I will demonstrate using the terminal and Windows SSH client “PuTTY”
2. Move around the cluster
  - I will demonstrate some basic bash commands and important folders.
3. Testing code
  - I will demonstrate how to load modules and do a little interactive programming





# Steps we will go over:

1. Login to the head node
  - I will demonstrate using the terminal and Windows SSH client “PuTTY”
2. Move around the cluster
  - I will demonstrate some basic bash commands and important folders.
3. Testing code
  - I will demonstrate how to load modules and do a little interactive programming
4. Writing and running the batch script
  - I will demonstrate a simple batch script



The image shows two windows. The top window is the PuTTY Configuration dialog, with the 'SSH' connection type selected. The bottom window is a terminal showing the output of the 'du' command for user 'jhtchns2'.

**PuTTY Configuration**

Category: Session, Logging, Terminal, Keyboard, Bell, Features, Window, Appearance, Behaviour, Translation, Selection, Colours, Connection, Data, Proxy, SSH, Serial, Telnet, Rlogin, SUDPUP

Basic options for your PuTTY session

Specify the destination you want to connect to:

Host Name (or IP address): demo-server.example.com Port: 22

Connection type:

☒ SSH ☐ Serial ☐ Other: Telnet

Load, save or delete a stored session

jhtchns2@cc-login1:~

Directories quota usage for user jhtchns2:

Project	Project	User	User	User	User	User	User
Block	Block	File	Soft	Hard	Used	Quota	Limit
Used	Limit	Used	Quota	Limit			
/home					4.9GB	5.0GB	7.0GB
5.5TB	N/A	100172	0	0	64KB	10.0TB	10.0TB
/scratch							
104.8TB	240.0TB	4	0	0	0KB	N/A	N/A
/projects/aces							
115.5TB	120.0TB	3	20000000	20000000			

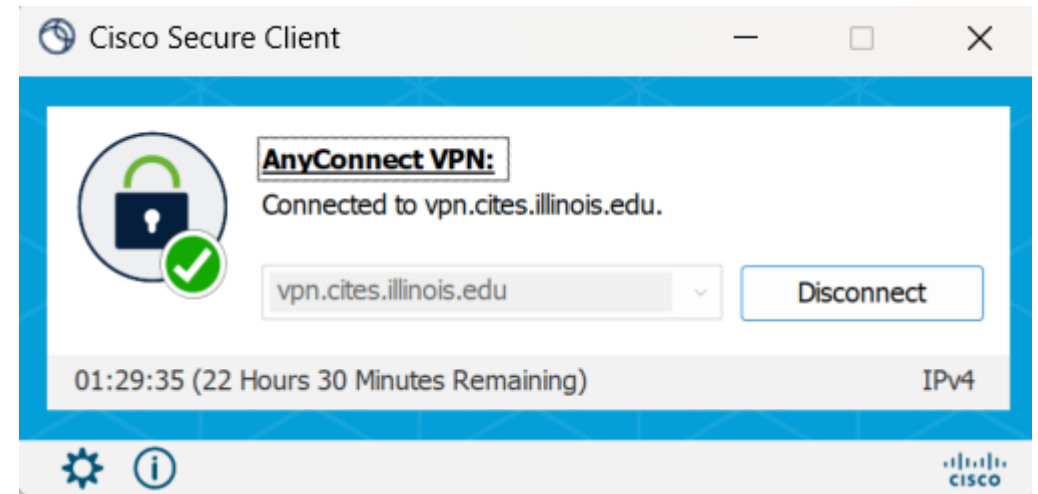
```
1 #!/bin/bash
2 #SBATCH --time=48:00:00
3 #SBATCH --nodes=1
4 #SBATCH --ntasks=10
5 #SBATCH --partition=aces
6 #SBATCH --mail-user=jhtchns2@illinois.edu
7 #SBATCH --mail-type=BEGIN,END
8
9 module load anaconda/3
10
11 source activate heat-stress
```

# Step 1: Logging in

***NOTE: if you are not on campus, you have to be VPN'ed into our campus network.***

Connect to “vpn.cites.illinois.edu” and enter netid and password.

Use **Cisco Secure Client** or else ask our IT folks how to do this.



# Step 1: Logging in with the Terminal

In Windows/Linux/MacOSX, open the application “Terminal” and type in:

```
ssh <NetID>@cc-login.campuscluster.illinois.edu
```

And then type in your password.

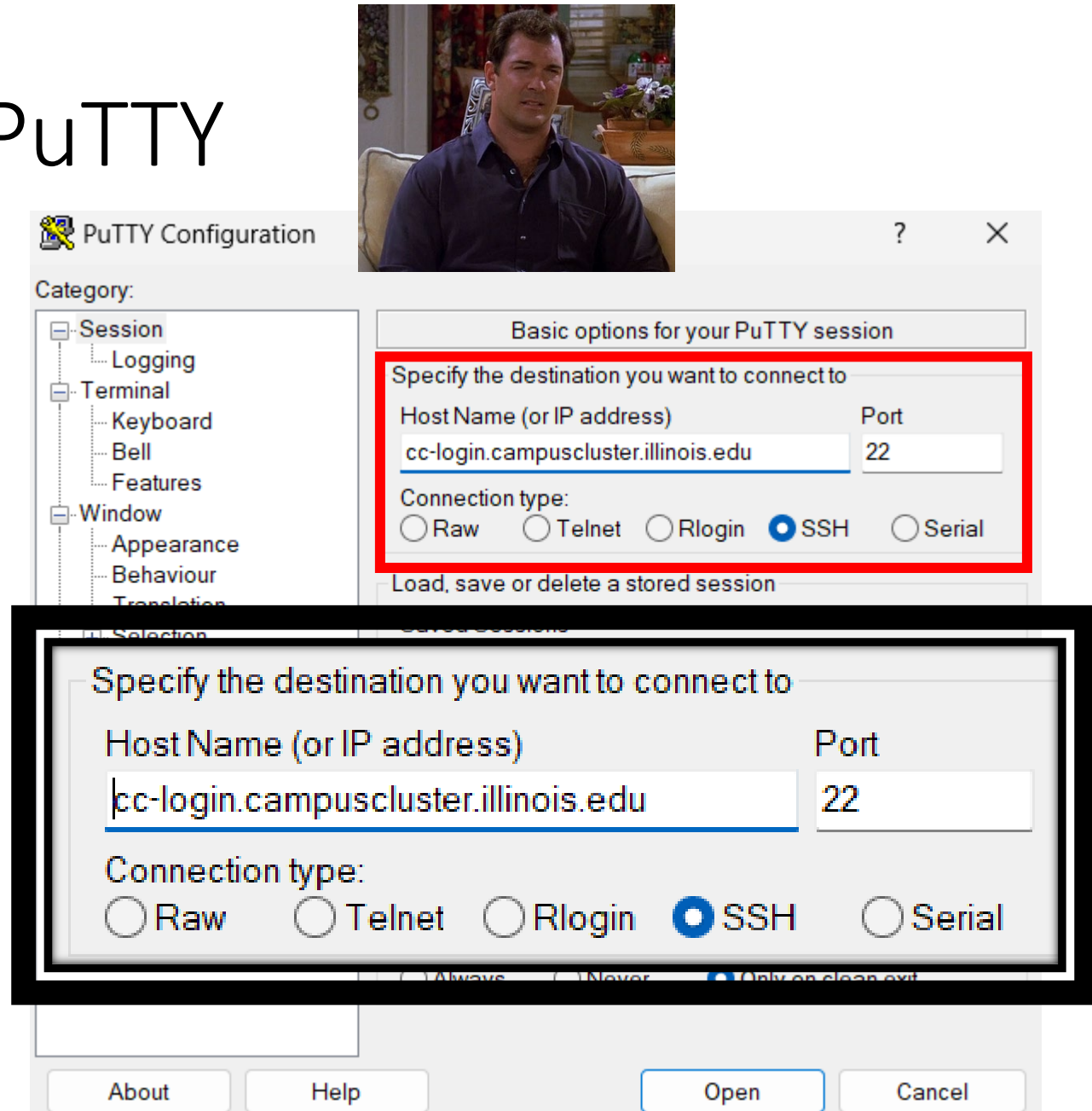


# Step 1: Logging in with PuTTY

The other point and click way is to use an SSH client like PuTTY.

## *Most important fields:*

- Host Name, which is always ***cc-login.campuscluster.illinois.edu***
- Port number, which for SSH is always **22**



# Do I have to type my password every time??

No, this is the purpose of creating an *ssh key pair*.

Basic steps:

1. Use “ssh-keygen” in the terminal
2. Find the file “id\_rsa.pub” in `~/.ssh/`
3. Copy over text in that file into the “authorized\_keys” file, located in `~/.ssh/` but on the cluster.

[Tutorial from Purdue](#)

## Step 1: *Demonstration*

*So I can login without having to type my password every time?*





## Step 2: Navigating

Now that you are in the door, how do you move around?

```
>Ye find yeself in yon dungeon. Ye see a FLASK.  
Obvious exits are NORTH, SOUTH, and DENNIS.  
  
What wouldst thou deau?  
>Get ye flask  
  
You can't get ye flask!
```

Bash command	What it does
<b>cd</b> <directory>	“Move” from where you are now to another directory
<b>ls</b> (<directory>)	“Look around,” lists files in directory (default: current)
<b>pwd</b>	“Where am I?” prints the current directory
<b>cp</b> <file> <destination>	Copy <file> to another directory <destination>
<b>rm</b> <file>	Delete <file> <i>(careful with this one!)</i>

# Step 2: Navigating

Other stuff you can do to files

```
>Ye find yeself in yon dungeon. Ye see a FLASK.  
Obvious exits are NORTH, SOUTH, and DENNIS.  
  
What wouldst thou deau?  
>Get ye flask  
  
You can't get ye flask!
```

Bash command	What it does
<b>mv</b> <file> <directory>	Copies the file to a new place and deletes old one.
<b>less</b> <file>	“Look at this file” prints contents to terminal.
<b>mkdir</b> <name>	“Make directory” creates a new directory (folder)
<b>vim</b> <file>	Open file using the <b>vim</b> <i>text editor</i>

# Step 2: The Geography

What are some places I can go?

```
>Ye find yeself in yon dungeon. Ye see a FLASK.  
Obvious exits are NORTH, SOUTH, and DENNIS.  
  
What wouldst thou deau?  
>Get ye flask  
  
You can't get ye flask!
```

Directory name	What it is	Example
<code>./</code>	Shorthand for “current directory”	<code>cp ./example.py ./new_directory/</code> (copy “example.py” in current directory to “new_directory” one in the same directory)
<code>../</code>	Shorthand for “previous directory”	<code>cd ../</code> (go back one) <code>cd ../other_directory</code> (go back one, then go into other_directory)



# Step 2: The Geography

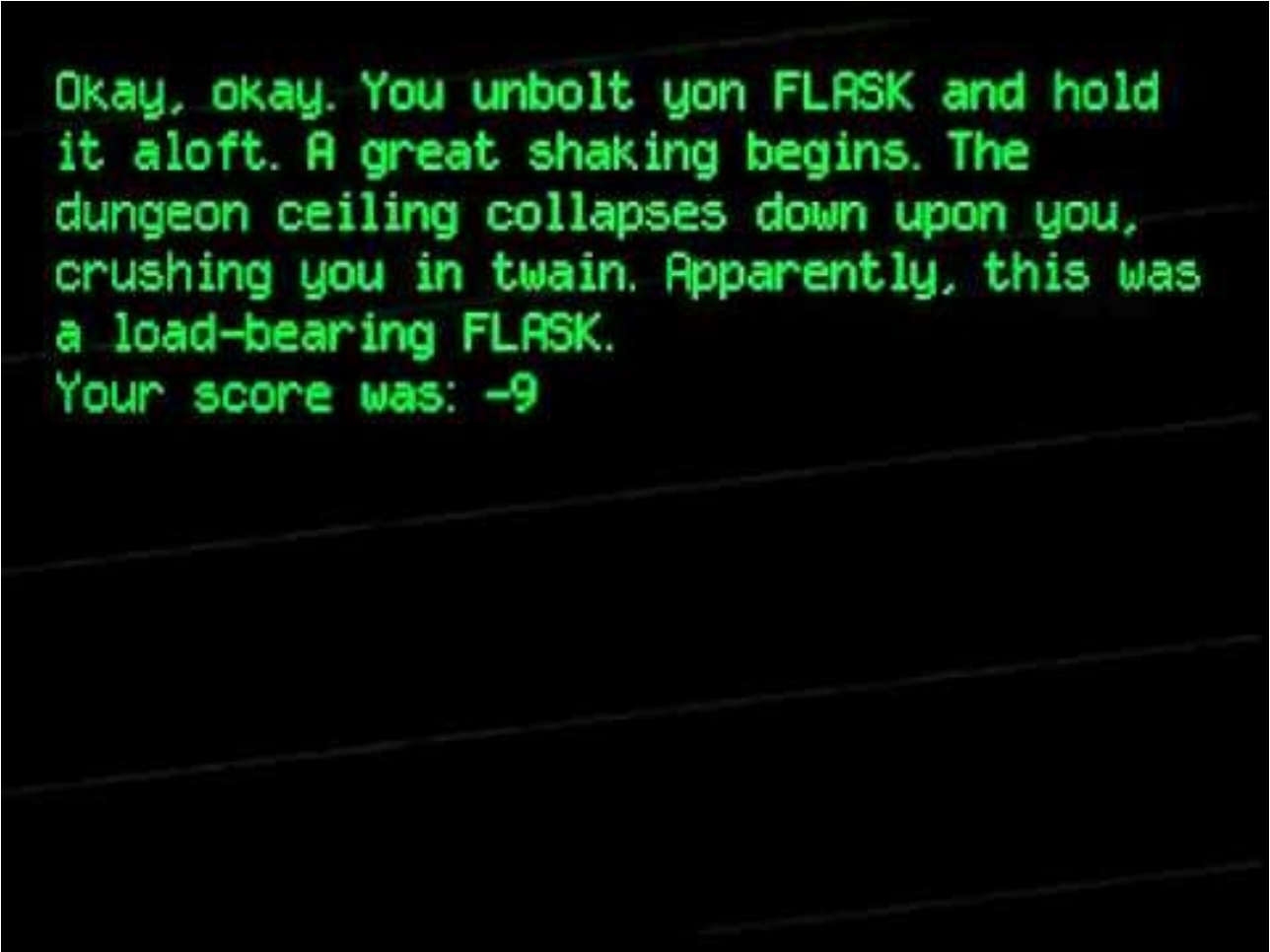
What are some places I can go?

[User Guide](#)

```
>Ye find yeself in yon dungeon. Ye see a FLASK.  
Obvious exits are NORTH, SOUTH, and DENNIS.  
  
What wouldst thou deau?  
>Get ye flask  
  
You can't get ye flask!
```

Directory name	What it is	Storage
~	“Home directory” the entryway you first walk into.	5GB Soft/7GB Hard; 7 Day Grace;
~/scratch	Your personal storage space on the cluster	10TB, but <i>cannot keep things for more than 30 days.</i>
~/project-aces	Storage specific to ACES, space for each user	However much ACES paid for, <i>shared resource</i>

## Step 2: Demonstration



Okay, okay. You unbolt yon FLASK and hold  
it aloft. A great shaking begins. The  
dungeon ceiling collapses down upon you,  
crushing you in twain. Apparently, this was  
a load-bearing FLASK.  
Your score was: -9

# Step 3: Programming Environment

## [User Guide section](#)

Now that you are in the cluster, you may want to run some code, right? To do that, you must first “load” the correct “modules.”

Bash command	What it does
<code>module avail</code>	lists all available modulefiles
<code>module list</code>	lists currently loaded modulefiles
<code>module load modulefile</code>	load <b>modulefile</b> into current shell environment

# Step 3: Programming Environment

[User guide for R and Python](#)

[User guide for matlab](#)

Bash command	What it loads
<code>module load R</code>	The latest version of R (or specify version)
<code>module load python/3</code>	The latest version of python 3
<code>module load anaconda/3</code>	The anaconda suite for python



# Step 3: How to program in a cluster

Two options:

1. Interactively: start an instance of R or python and type in commands.
2. Batch: write a script, run it using the command **python** or **Rscript**

You are probably more used to the first way, but to submit jobs on the cluster you must know how to do it the second way.

# Step 3: Differences in STATA

## Interactively

```
(R)
 _ _ _ _ _
  _ _ _ _ _ 16.1 Copyright 1985-2019 StataCorp LLC
  _ _ _ _ _ Statistics/Data analysis StataCorp
                4905 Lakeway Drive
                College Station, Texas 77845 USA
                800-STATA-PC https://www.st
                979-696-4600 stata@stata.co
                979-696-4601 (fax)

Stata license: 22-user 2-core network perpetual
Serial number: 501606210108
Licensed to: Dept of ACE
            University of Illinois Urbana-Champaign

Notes:
  1. Unicode is supported; see help unicode_advice.
  2. More than 2 billion observations are allowed; see help obs_advi
  3. Maximum number of variables is set to 5,000; see help set_maxva
  4. New update available; type -update all-

Checking for updates...
(contacting http://www.stata.com)

Update status
Last check for updates: 11 Apr 2024
New update available: 13 Jun 2023 (what's new)
Current update level: 14 Feb 2022 (what's new)

Command
Zhu Li, do the thing
```

## Batch

```
Do-file Editor - dcreate_sheep.do
File Edit View Language Project Tools

dcreate_sheep.do x Untitled

1 clear all
2 set more off
3
4 *Step 1: Generate full factorial design matrix
5 matrix levmat = 5,5,3
6 genfact, levels(levmat)
7 list, separator(4)
8
9 *Step 2: Change variable names and recode levels
10 forvalues i = 1/3 {
11   rename x`i' A`i'
12 }
13
14 /// Attributes ///
15 *Price
16 recode A1 (1=0) (2=1) (3=2) (4=3) (5=4)
17 *Color
18 recode A2 (1=0) (2=1) (3=2) (4=3) (5=4)
19 *Weight
20 recode A3 (1=0) (2=1) (3=2)
21
22
23 *Step 3: Run dcreate
24 matrix b = J(1,10,0)
25 matrix optout = J(1,3,1)
26 matrix V = I(10)
27
28 dcreate i.A1 i.A2 i.A3, nalt(2) nset(20) bmat(b) fixedalt(optout) vmat(V)
29
30 blockdes block, nblock(4)
31
32
33 *Reshape to use it as a matrix
34 order choice alt, before(A1)
35
36 sort choice alt
37
38 forvalues i = 1/3 {
```

# Step 3: Differences in R

## Interactively

```
Console Terminal x Background Jobs x
R 4.0.2 · C:/Users/jhtchms2/

R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> Zhu Li, do the thing|
```

## Batch

```
60_R_Analysis.r x
Source on Save Run
1 data_dir <- readLines("data_dir.txt")
2
3 setwd(data_dir)
4
5 package_list = c("IRkernel", "plm", "MASS", "stargazer",
6                 "lfe", "dplyr", "caret", "data.table", "nlwaldtest",
7                 "msm", "car", "rlist", "stringr", "stringi", "fixest")
8
9 install.packages(package_list)
10
11 # Read in data
12 df <- read.csv("./clean_data/fcs_final.csv")
13
14 # Require packages
15 lapply(package_list, require, character.only = TRUE)
16
17 nomvars = c("expend_fert", "expend_equip", "equip_value",
18            "total_crop_val", "farm_value", "expend_labor",
19            "expend_feed", "expend_labor", "total_livestock_val", "animal_revenue")
20
21 for(var in nomvars){
22   df[df$year == 1920, var] <- (df[df$year == 1920, var]/.2)
23   df[df$year == 1925, var] <- (df[df$year == 1925, var]/.175)
24   df[df$year == 1930, var] <- (df[df$year == 1930, var]/.167)
25   df[df$year == 1935, var] <- (df[df$year == 1935, var]/.137)
26   df[df$year == 1940, var] <- (df[df$year == 1940, var]/.14)
27 }
28
29
```

# Step 3: Differences in Python

## Interactively

```
(base) C:\Users\jhtchns2>ipython
Python 3.9.12 (main, Apr 4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.2.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: Zhu Li, do the thing|
```

## Batch

```
12_THI_Calculation.py 1 X
C: > Users > jhtchns2 > Box > Dairy-Heat-Stress > code > 12_THI_Calculation.py > {} pd

1 import pandas as pd
2 import sys
3
4 if __name__ == "__main__":
5     year = sys.argv[1]
6     in_dir = sys.argv[2]
7     out_dir = sys.argv[3]
8
9     rmax = pd.read_csv(in_dir + "rhmax_" + str(year) + ".csv").iloc[:, 1:].sort_values(['date'])
10    rmin = pd.read_csv(in_dir + "rhmin_" + str(year) + ".csv").iloc[:, 1:].sort_values(['date'])
11    tmax = pd.read_csv(in_dir + "tempmax_" + str(year) + ".csv").iloc[:, 1:].sort_values(['date'])
12    tmin = pd.read_csv(in_dir + "tempmin_" + str(year) + ".csv").iloc[:, 1:].sort_values(['date'])
13
14    W = tmax.merge(tmin).merge(rmin).merge(rmax)
15
16    W['tempmax'] = W['tempmax'] - 273.15
17    W['tempmin'] = W['tempmin'] - 273.15
18
19    W['THI_max'] = (.8 * W['tempmax'] + ((W['rhmax'] / 100) * (W['tempmax'] - 14.3)) + 46.4)
20    W['THI_min'] = (.8 * W['tempmin'] + ((W['rhmin'] / 100) * (W['tempmin'] - 14.3)) + 46.4)
21
22    W.to_csv(out_dir + "THI_" + str(year) + ".csv", index=False)
```



# Step 3: Installing your packages

Since this is not your computer, you only have write permissions in the home directory (~).

If you need specific packages, you must install them somewhere in the home directory.

[User Guide for R](#)

[User Guide for Python/Anaconda](#)

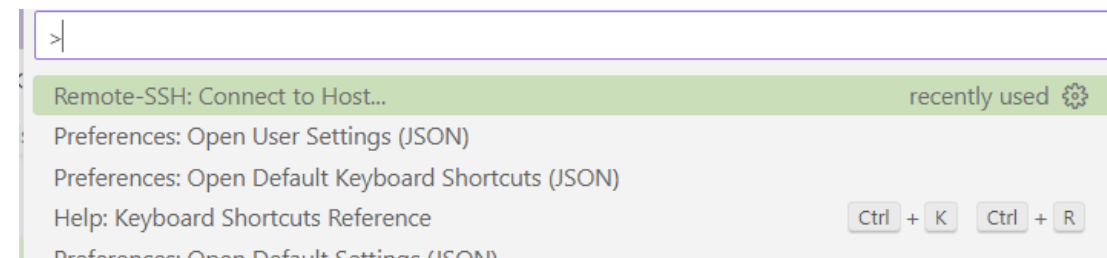
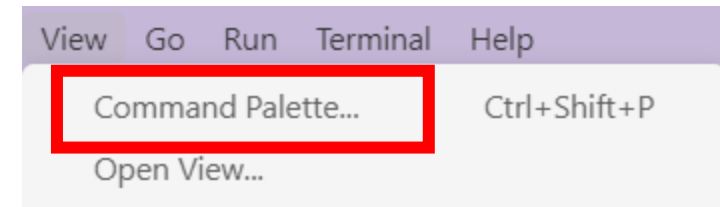
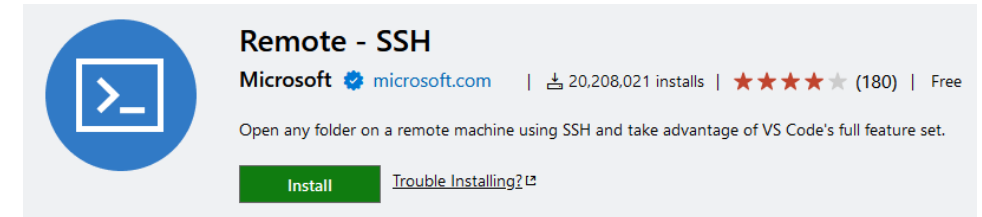
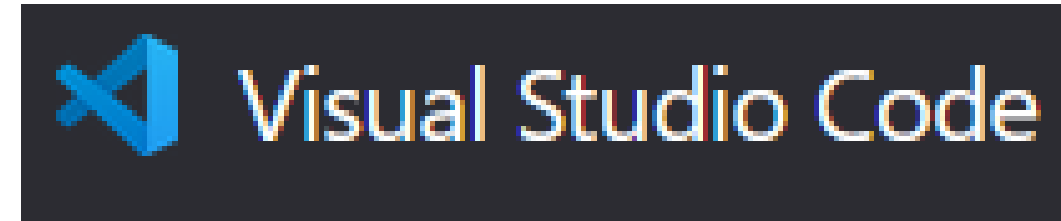
For Anaconda, **I highly recommend creating a conda environment every time.**

# Step 3: Writing a Script

I would recommend having some kind of SSH add-on for your text editor to remotely edit scripts.

In this case, I'll show you [Visual Studio Code](#) which can access the cluster with the add-on [Remote-SSH](#)

Then you can edit scripts on the server with VS Code.



## Step 3: Transferring a File (2 ways)

### Use scp (easy but Terminal way)

Open the terminal and navigate to your file. Then type this:

```
scp <file>  
<netid>@<clusterurl><dir_on_cluster>
```

Example:

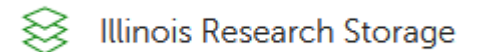
```
scp example.txt jhtchns@cc-  
login.campuscluster.illinois.edu:~/A  
CE592
```

### Use Globus endpoint (point and click)



Connect to Globus using [Globus Connect Personal](#)

Log in to Globus using your Illinois credentials and find “Illinois Research Storage”



# Step 3: Running a Script

Once you have written your script, you can run it using one of these commands:

Bash command	What it does
<code>python example_script.py</code>	Runs the script “example_script.py” using the loaded python module
<code>Rscript example_script.R</code>	Runs the script “example_script.R” using the loaded R module.



## Step 3: Demonstration

## Step 4: Submitting a job

Now that you have your script, the cluster needs information on how to run it.

- Where to run it
- What resources it needs (time, CPU, cores, etc.)
- What commands to run when it gets the resources.

```
home > jhtchns2 > ACE592 > $ test_job_array_par.sh
1  #!/bin/bash
2  #SBATCH --time=0:05:00
3  #SBATCH --output=out.txt
4  #SBATCH --nodes=1
5  #SBATCH --array=1-10
6  #SBATCH --partition=aces
7  #SBATCH --mail-user=jhtchns2@illinois.edu
8  #SBATCH --mail-type=BEGIN,END
9
10 module load anaconda/3
11
12 python hello_parallel.py $SLURM_ARRAY_TASK_ID
```

## Step 4: What is slurm?

To allocate resources across nodes efficiently, there needs to be a scheduler (think central planner).

Our cluster (and many others) uses the **Slurm Workload Manager**.

Basically, you tell slurm what you need and it finds the resources on the cluster.



## Step 4: Some *slurm* commands

Bash Command	What it does
<code>squeue -u &lt;your-netid&gt;</code>	Find out what jobs you are running
<code>squeue -j &lt;job_id&gt;</code>	Shows where your job is in the queue
<code>squeue -p aces</code>	Look at what jobs are being run on the “aces” partition (“-p”)
<code>sacct -j &lt;job_id&gt;</code>	Get information on this job once it’s done.
<code>sinfo -p &lt;partition_name&gt;</code>	Look at what’s available on the partition

# Step 4: Asking for resources

On the right is an example of what should be in every job script for the *slurm* system.

The objective of this part is to tell the cluster how much resources your job needs.

The script should end with “.sh” to tell the computer it’s a bash shell script.

```
#!/bin/bash
#SBATCH --time=48:00:00
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=2
#SBATCH --partition=aces
#SBATCH --mail-
user=jhtchns2@illinois.edu
#SBATCH --mail-type=BEGIN,END
```



## Step 4: Asking for resources

- “this is a bash script”  
`#!/bin/bash`
- “I want 48 hours to run it”  
`#SBATCH --time=48:00:00`
- “I want one compute node”  
`#SBATCH --nodes=1`
- “Run one task”  
`#SBATCH --ntasks=1`
- “Make it have 2 CPUs per task”  
`#SBATCH --cpus-per-task=2`
- “Run it on the ACES partition”  
`#SBATCH --partition=aces`
- “email Jared with updates”  
`#SBATCH --mail-`
- “... when it begins and ends”  
`user=jhtchns2@illinois.edu`  
`#SBATCH --mail-type=BEGIN,END`

# Step 4: Ok, but how do I know what to ask for?

## User Guide

### Time (hh:mm:ss):

- Usually your best guess at how long the script is going to take.
- If you ask for too much time, ***it may never run*** (it won't find a node with enough spare time).

### Partition:

- Usually using the ACES partition, but you may want to use another one.
- When ACES is too crowded, you can use **secondary**. However, this one has a walltime limit of 4 hours.

### Account:

- Specify this when using someone's account (e.g. if your PI has Illinois Computes space).

# Step 4: Ok, but how do I know what to ask for?

## User Guide

### **--mem or --mem-per-cpu:**

- User guide suggests **not specifying memory unless absolutely required**, as it will only run your job if it has that **specific amount available**.
- Memory configurations: 64GB, 128GB, 192GB, 256GB or 384GB.

### **--nodes:**

- Almost always “1”, unless using something like MPI to do things across nodes.

### **--ntasks:**

- The number of ***separate jobs*** you want to run at the same time (will run the script this many times).  
**(NOT THE SAME AS CORES)**

# Step 4: Ok, but how do I know what to ask for?

## [User Guide](#)

### **--cpus-per-task**

- This is where you ask for the number of cores that you want your program to use if parallelizing.

### **--array**

- This option allows you to run an analysis multiple times in parallel using an array of numbers.

### **--gres**

- This is how many GPUs you want. ([User Guide](#))

# Step 4: Submitting the Job

## **sbatch**

Once you've written your script ("job.sh"), run:

```
sbatch job.sh
```

After a while, it will let you know when it has found resources and is running the job.

## **srun**

To run an interactive job, specify all the options with "srun"

```
srun --partition=aces --nodes=1  
--time=00:30:00 /bin/bash
```

Once it gets your resources, you can code in the terminal.



# Step 4: Designing a parallel job

## Using a job array

Specify the sbatch option “--array” and give it a sequence of numbers.

In the script, you can then refer to `$SLURM_ARRAY_TASK_ID` which will be that sequence of numbers.

This submits *multiple jobs to slurm*.

```
home > jhtchns2 > ACE592 > $ test_job_array_par.sh
1  #!/bin/bash
2  #SBATCH --time=0:05:00
3  #SBATCH --output=out.txt
4  #SBATCH --nodes=1
5  #SBATCH --array=1-10
6  #SBATCH --partition=aces
7  #SBATCH --mail-user=jhtchns2@illinois.edu
8  #SBATCH --mail-type=BEGIN,END
9
10 module load anaconda/3
11
12 python hello_parallel.py $SLURM_ARRAY_TASK_ID
```

# Step 4: Designing a parallel job

## Using tasks

Specify the sbatch option “--ntasks” and give it a number

Then, allocate steps ***within*** the job to your requested resources using ***srun***.

Note: the “&” sign at the end of a bash command allows you to run a command after that one.

```
#!/bin/bash
#SBATCH --time=0:05:00
#SBATCH --output=out.txt
#SBATCH --nodes=1
#SBATCH --ntasks=2
#SBATCH --partition=aces
#SBATCH --mail-user=jhtchns2@illinois.edu
#SBATCH --mail-type=BEGIN,END

module load anaconda/3

srun --exclusive --ntasks 1 python hello_parallel.py 1 &
srun --exclusive --ntasks 1 python hello_parallel.py 2 &

wait
```

# Step 4: What's the difference between them?

## Using tasks

This tells slurm to create ***one job*** and then allocates resources within that job in “job steps”

This allows you to flexibly allocate things you need.

After running this, you can use ***srun*** to change the inputs to your program.

## Using arrays

This tells slurm to create ***multiple jobs*** and allocate them across the cluster (depending on needs).

This works best **if the resource needs are identical in each run**

Here you would write your program to accept an integer and use that to run tasks.

# Step 4: My jobs aren't running!! WHY??

When the queue is too busy, you may find that your jobs have to stand in line for a while before running.

First: check whether you are requesting too many resources.

Second: use the ***secondary queue*** (max walltime: 4 hours)

## Note to Faculty:

[Illinois Computes Program](#) offers faculty free space, and your students can use that partition instead!

### Get Started with Illinois Computes

Whether you're an expert HPC user or completely new to using computing and storage resources in your research, NCSA and its expert staff are ready to help.

[Request Resources Now](#)



## Step 4: Demonstration



# What you should check before using the cluster

- Can my code be made more memory efficient?
- Can I fix a speed bottleneck to make the code run faster?
- Can I vectorize any operations?
- Can I run parallel on my own computer?



# Advanced topics you can learn how to do:

- Running jobs on multiple nodes  
(Look into things like [OpenMPI](#) which will pass info between nodes)
- Parallelizing code efficiently  
(Learn to use a workflow engine like [makeflow](#))
- Make your code memory efficient on its own  
(learn to use chunks, vectorize operations, etc.)