

Software Engineering for Spatial Data Analysis??

CEE 690

Software Engineering for Spatial Data Analysis??

CEE 690

Or “ESDA II: Now do it for real”

The ESDA experience



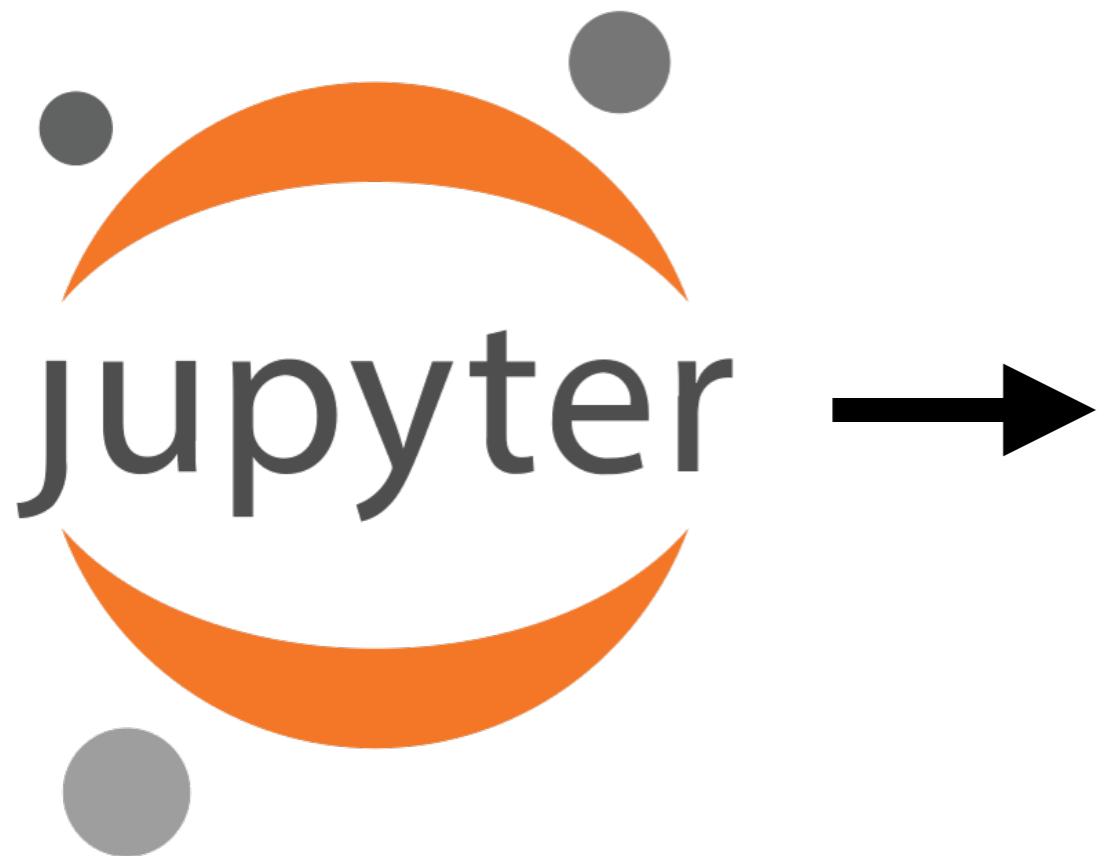
The ESDA experience

Jupyter is great... until it's not

Issues using Jupyter Notebooks for software development (and really any IDE)

- **Non-linear workflow**
 - Out-of-order execution
 - Variables remain in memory
- **Version control challenges**
 - Unreadable diffs
 - Merge conflicts
- **Poor software engineering**
 - Lack of testing
 - Difficult refactoring
- **Performance and memory issues**
 - Memory overhead
 - Blocking execution

Farewell to Jupyter



```
nathanielchaney — nc153@dcc-login-04:/hpc/home/nc153 — ssh nc153@dcc-login.oit.duke.edu — 50x15
(base) nc153@dcc-login-04 ~ $
```

A screenshot of a terminal window. The title bar shows the path "nathanielchaney — nc153@dcc-login-04:/hpc/home/nc153 — ssh nc153@dcc-login.oit.duke.edu — 50x15". The command prompt "(base) nc153@dcc-login-04 ~ \$" is visible at the top of the window. The main body of the terminal is entirely black, indicating it is currently empty or has no output to display.

Ultimately, it will be worth it



Ultimately, it will be worth it



But it is going to take a semester...

Tentative schedule

Part I: Python

Date	Topic
01/08	The headless environment
01/13	Text editors
01/15	Software architecture I
01/20	Software architecture II
01/22	Verification and profiling
01/27	Packaging, automation and documentation
01/29	Computing on spatial grids
02/03	GPU computing in Python
02/05	High-Speed kernels: Numba CPU & CUDA
02/10	Anatomy of an HPC cluster
02/12	Shared memory parallelism I
02/17	Distributed memory parallelism I
02/19	The scheduler
02/24	High-Performance I/O

Part II: Compiled languages

Date	Topic
02/26	Compiled languages I
03/03	Compiled languages II
03/05	The build process
03/17	Memory management and basic debugging
03/19	Tensors in memory and data layout
03/24	Computer architecture and cache
03/26	Code documentation and profiling
03/31	Code interoperability
04/02	Shared memory parallelism II
04/07	Distributed memory parallelism II
04/09	Containers
04/14	TBD

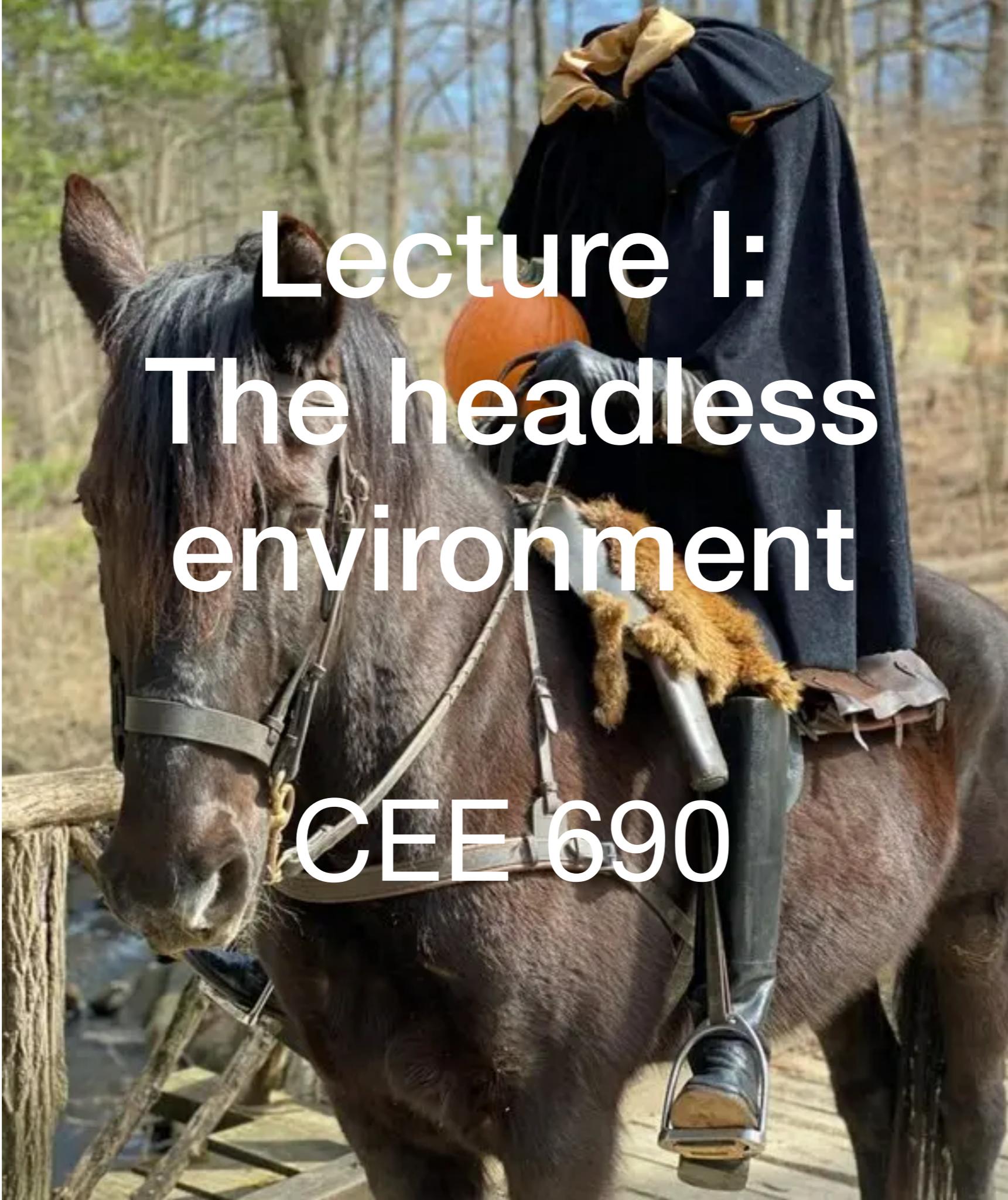
Grades and workload

The course grade will be based on:

- Homework: 30%
- Project I: 35%
- Project II: 35%

Homework

TBD

A close-up photograph of a dark brown horse's head and neck. The horse is wearing a dark blue saddle cloth with gold-colored trim and a matching bridle. It is leaning its head against a weathered wooden fence post. The background shows a field with some trees and a clear sky.

Lecture I: The headless environment

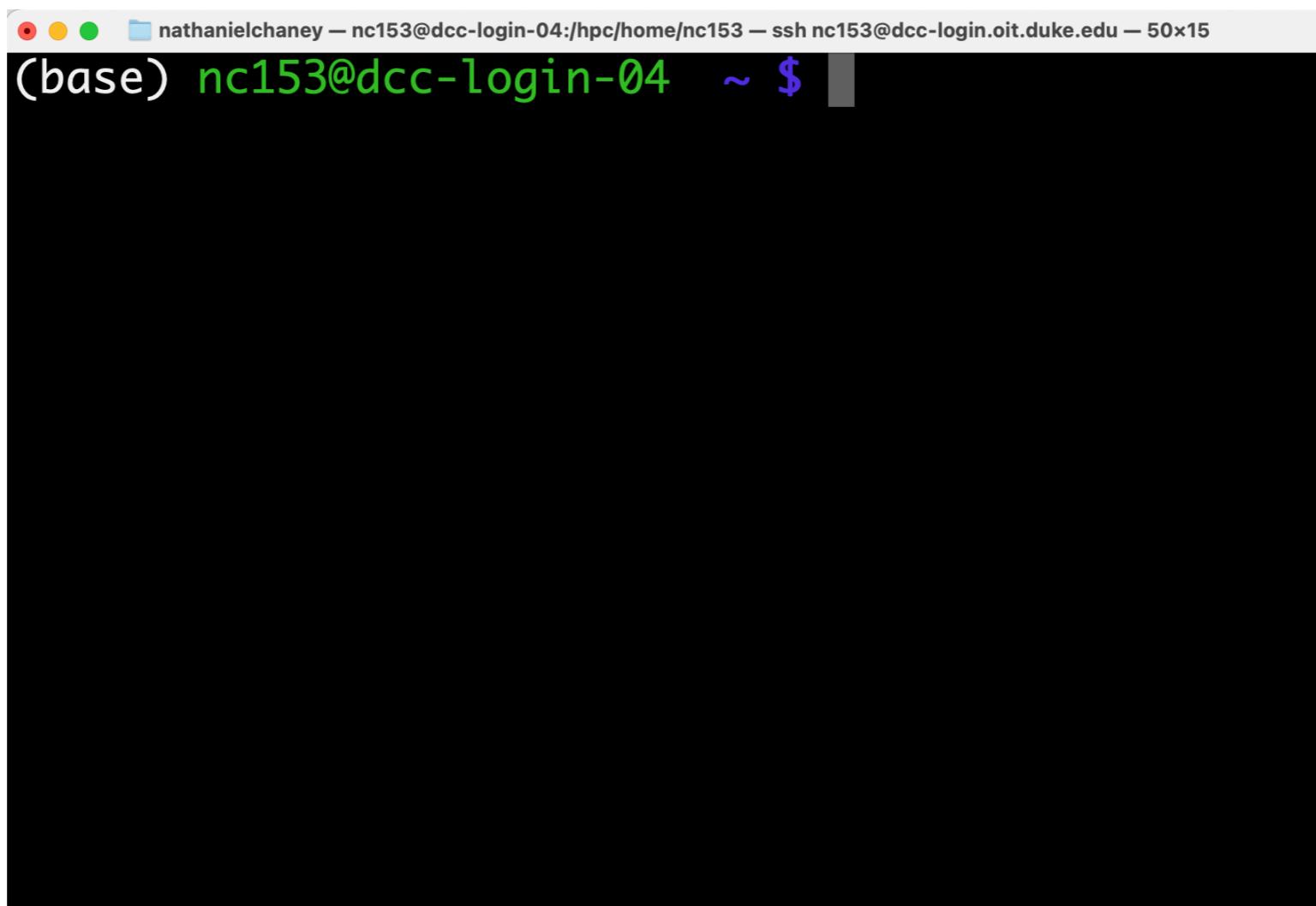
CEE 690

Headless computing



Headless computing

= No graphical user interface (GUI)

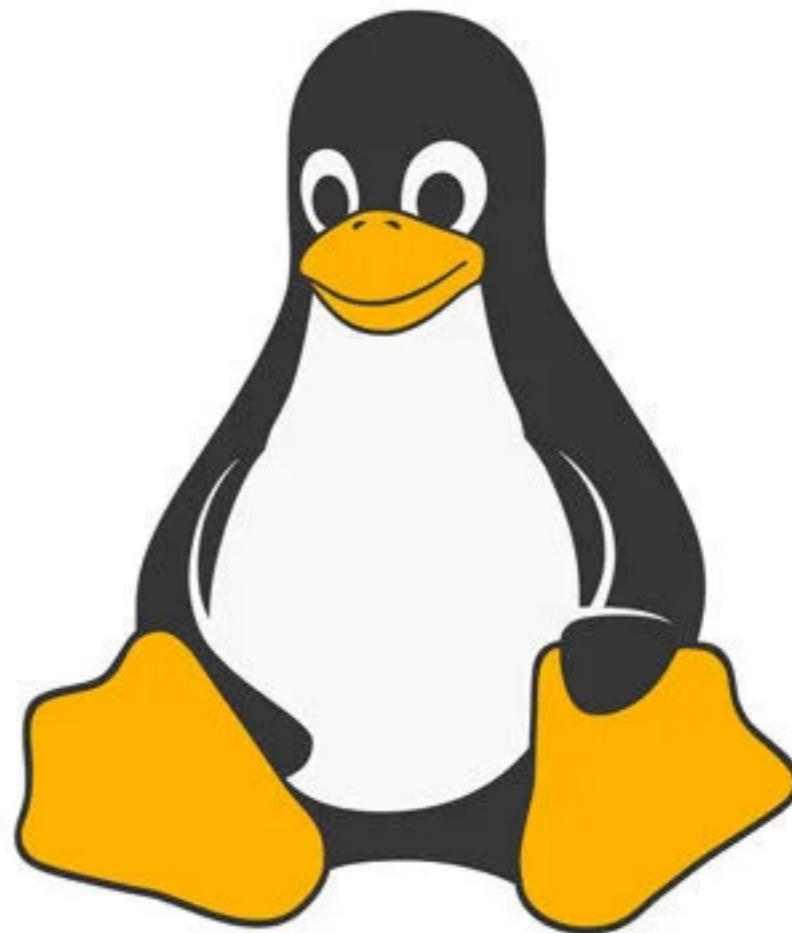


Why will this be critical?



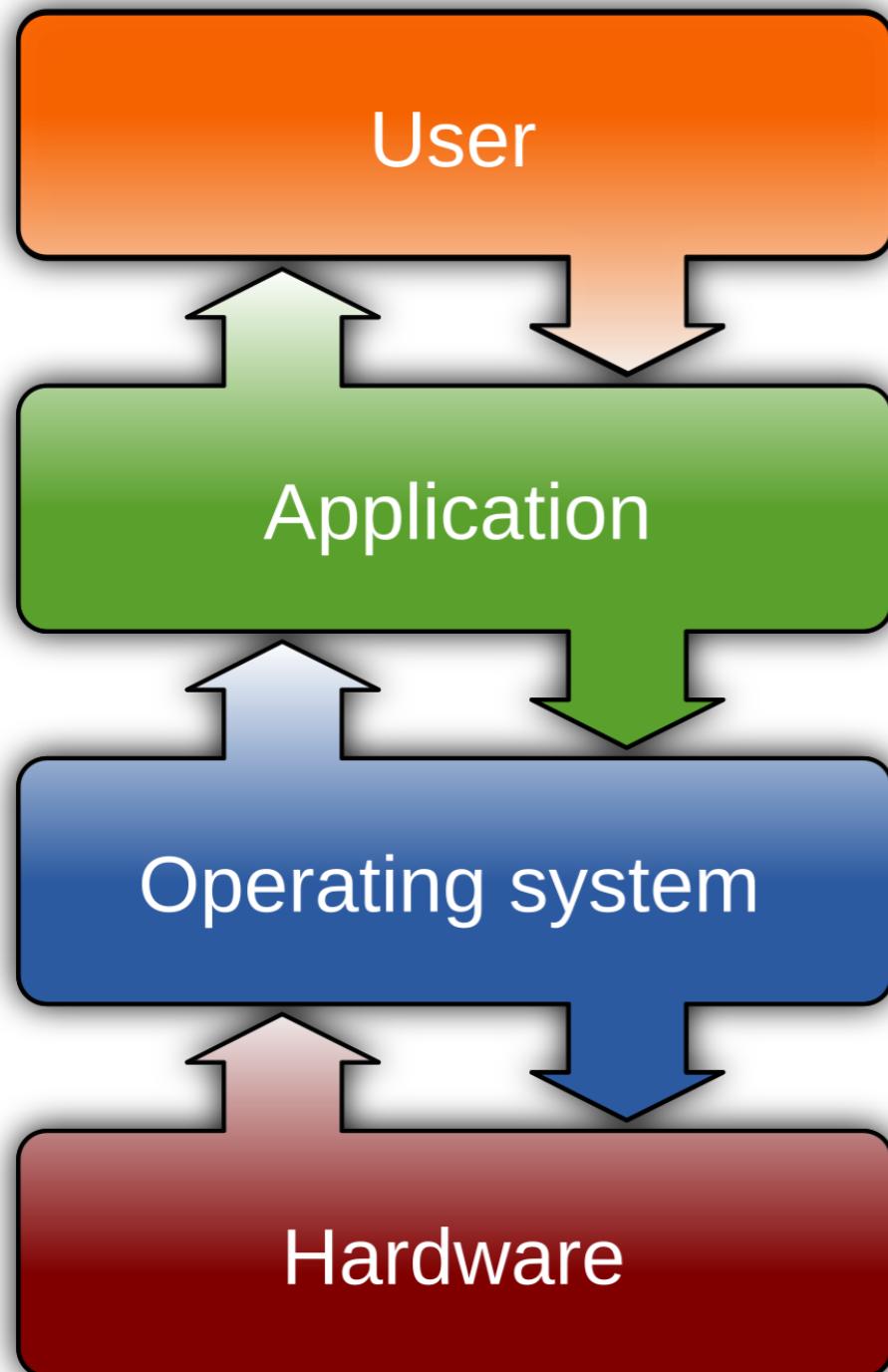
You'll need to embrace linux

- Stripped-down efficiency
- Specialized tuning
- Access to source code
- Community driven tools
- High uptime
- Scale capability
- Cost effectiveness
- Interoperability



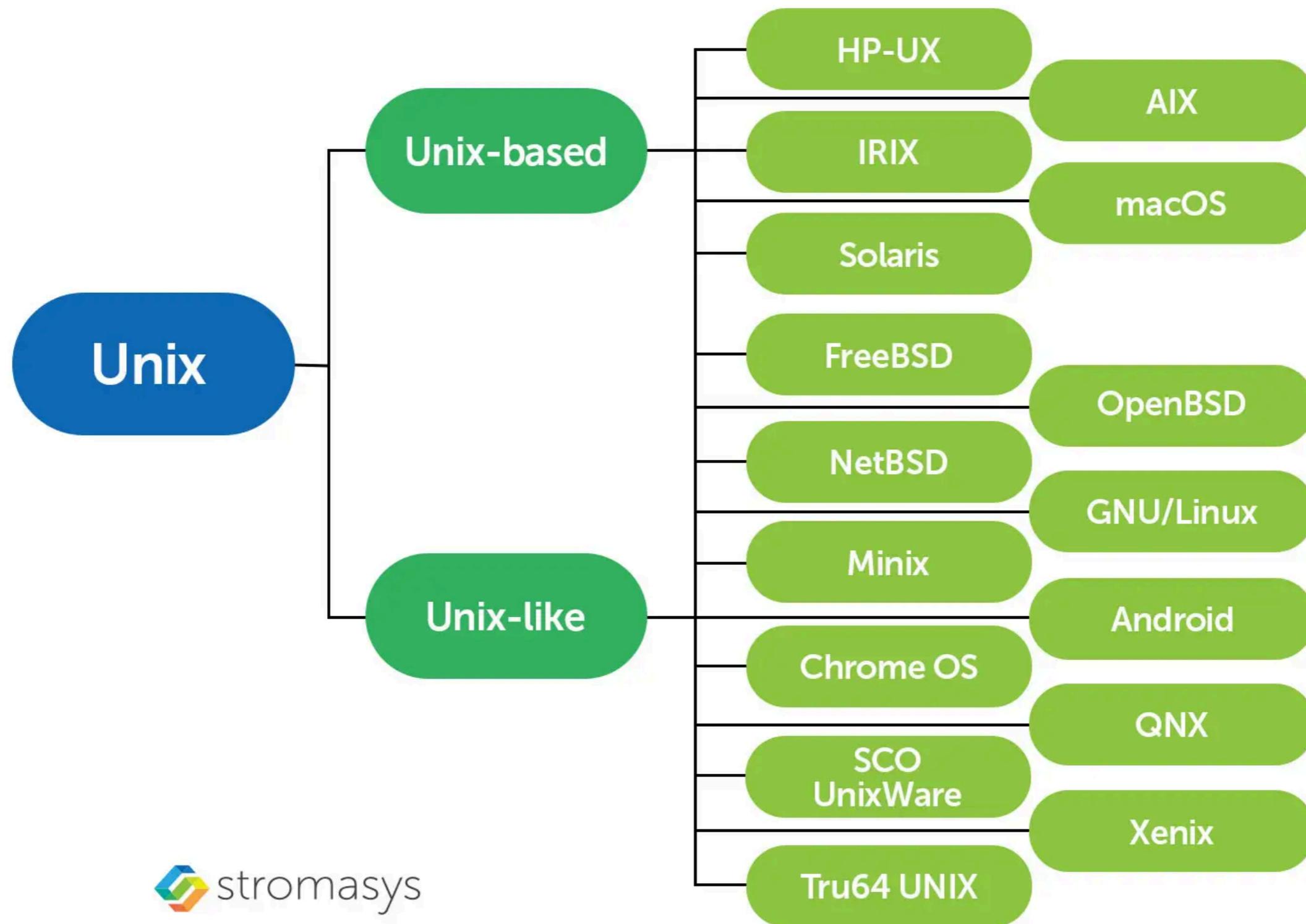
Linux™

What is an operating system?



- Conductor of the orchestra (computer)
- Complex software that sits between the raw physical hardware and the user/application
- The “kernel” is the heart of the OS. It is the only part that talks directly to the hardware.
- The shell (Bash) is how we interact with the OS (and thus hardware) in our course

Origins of Linux



The root directory (/)

Every single folder on the system resides in /

```
[(base) nc153@dcc-login-04 ~ $ cd /
[(base) nc153@dcc-login-04 / $ ls
admin  cifs          cwork      fastrs    lib64     nfs      run       sys   vwork
afs    core.dcc-login-04_1138_gssproxy_1764807312  datacommons  home     l stor    opt      sbin     tmp   work
bin    core.dcc-login-04_741_systemd-journal_1749584689  dev        hpc      media    proc     scratch  usr
boot   core.dcc-login-04_972_auditd_1764807315      etc        lib      mnt     root     srv      var
```

- /bin - basic commands (root access)
- /lib - code libraries (root access)
- /etc - system settings (root access)
- ...
- /home - your folder is in here (user-controlled directory)
- We will talk about more of these folders throughout the course

Home directory (~)

On the dcc it is /hpc/home/netid

You have full read, write, execute permissions here

```
(base) nc153@dcc-login-04 ~ $ cd ~
(base) nc153@dcc-login-04 ~ $ ls
CEE690 miniconda3 Miniconda3-latest-Linux-x86_64.sh ondemand README test.py text2.txt text.txt
(base) nc153@dcc-login-04 ~ $ pwd
/hpc/home/nc153
```

“All” Compute clusters/HPC use linux



Duke Compute Cluster

Overview

The Duke Compute Cluster is a general purpose high performance/high-throughput installation, and it is fitted with software used for a broad array of scientific projects. With a few notable exceptions, applications on the cluster are generally Free and Open Source Software.

Quick facts:

- 1360 nodes which combined have more than 45,000 vCPUs, 980 GPUs and 270TB of RAM
- Interconnects are 10 Gbps or 40 Gbps
- Utilizes a 7 Petabyte Isilon file system
- Runs Alma 9 and SLURM is the job scheduler

Need an account on the Duke Compute Cluster (or another machine)

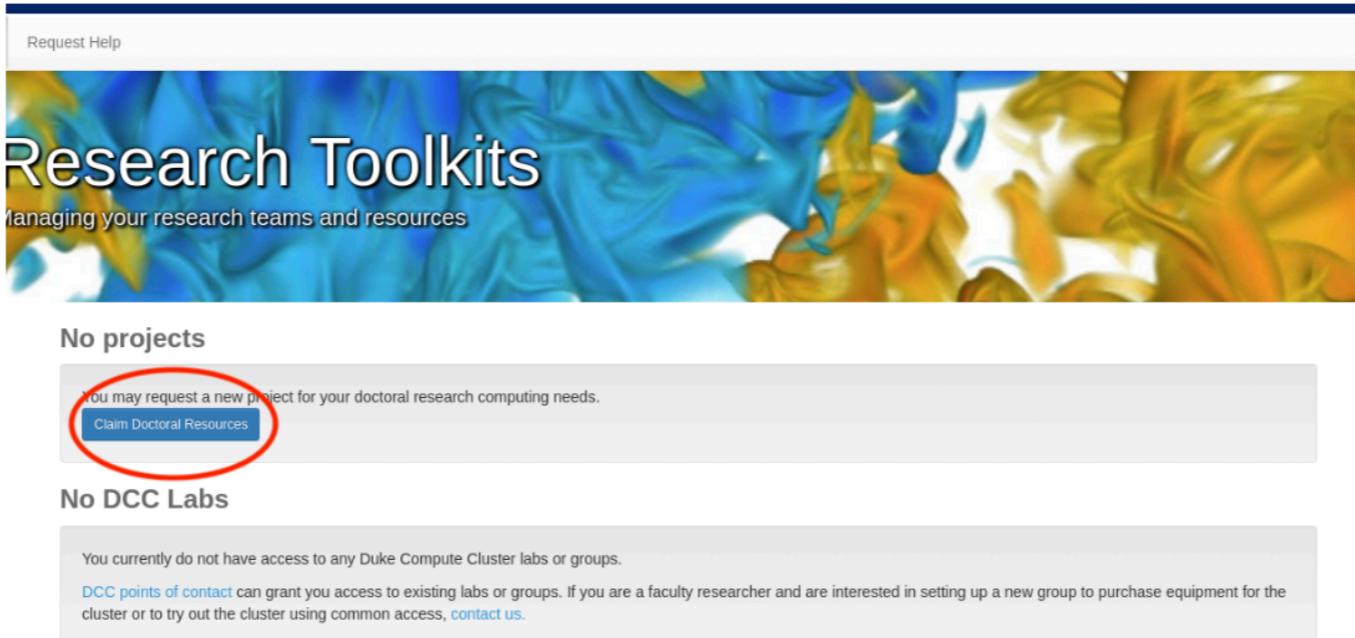
Claim PhD Student Allocation

Duke PhD students now direct allocations to the DCC for compute hours and 500GB of dedicated storage.

Dedicated storage can be found at: \hpc\dctrl\<netid>

To claim this allocation, PhD student should:

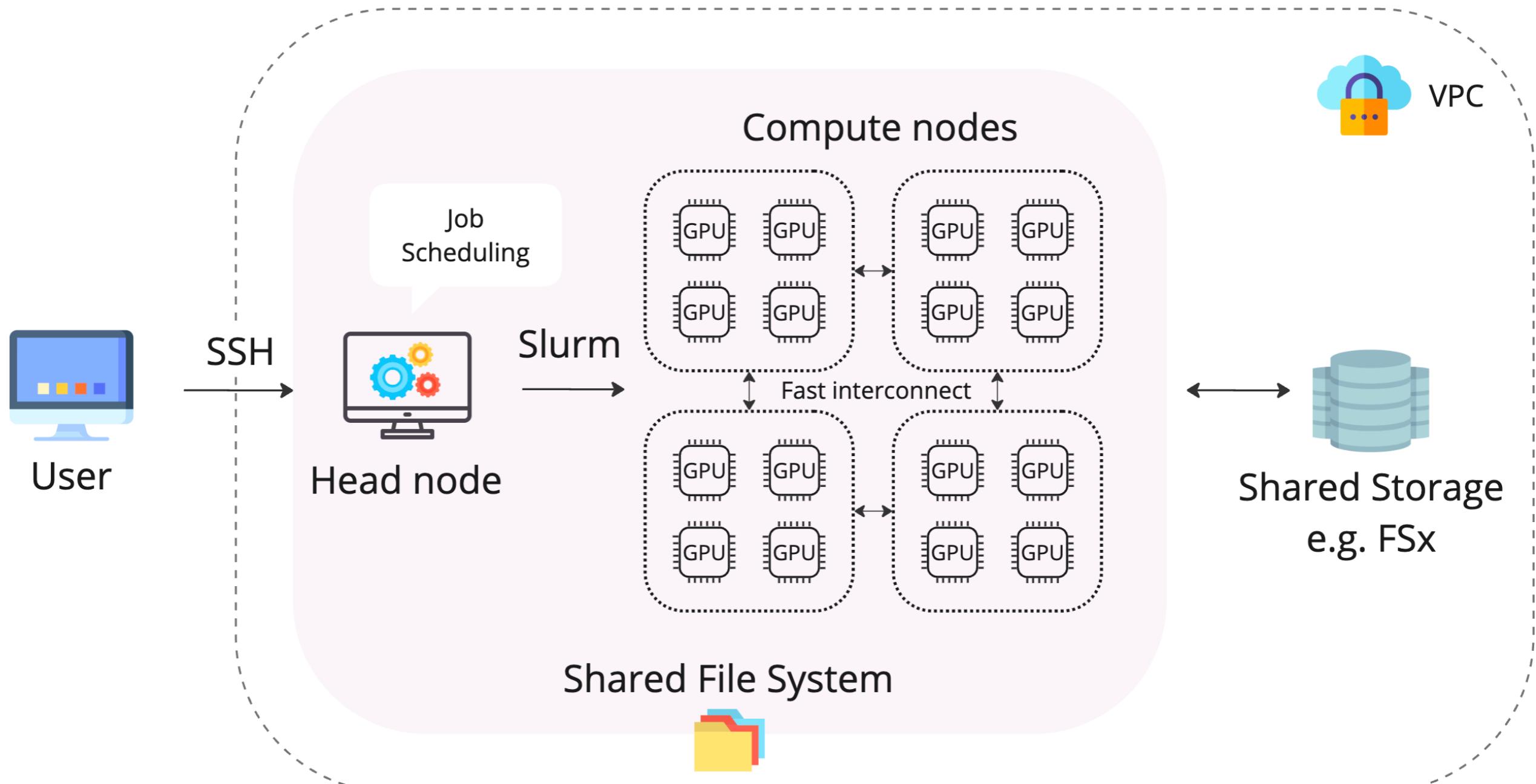
1. Login to [Research Toolkits](#)
2. Claim your Doctoral resources by selecting the [Claim Doctoral Resources](#) button



3. Once the project is created, open it and select the menu to [add a service](#), and choose [DCC Grant](#). Your project will now automatically update with status as your access is provisioned.

Once your access is provisioned check out our [training and quick-start guides](#).

How to connect to these machines



Source: <https://www.unitary.ai/>

SSH (secure shell): Portal to the cluster

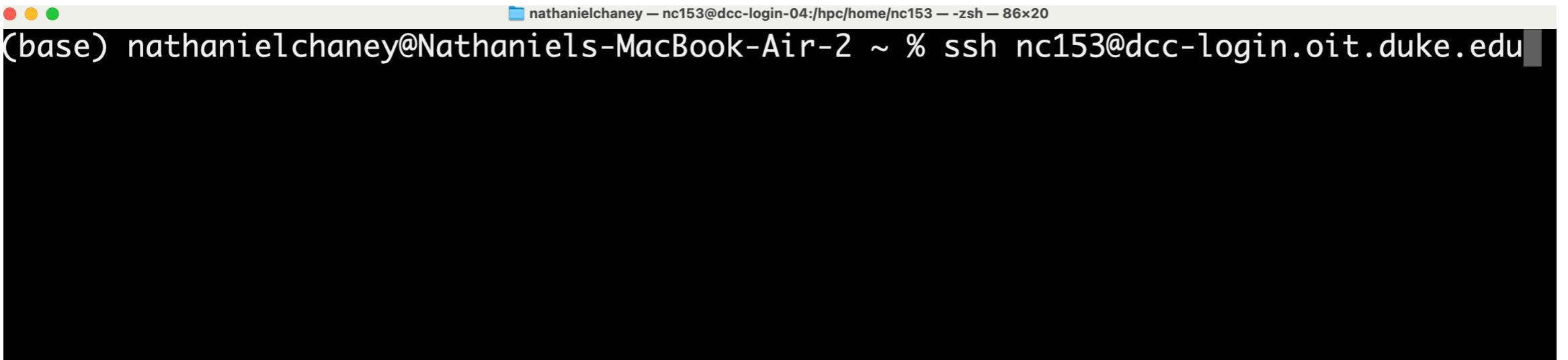
Tunnel to connect your laptop/desktop to a
compute cluster or HPC system

- **Components**

- Client (your local machine)
- Server (login node of HPC system)
- Handshake (two computers exchange
keys to establish encrypted connection)

SSH

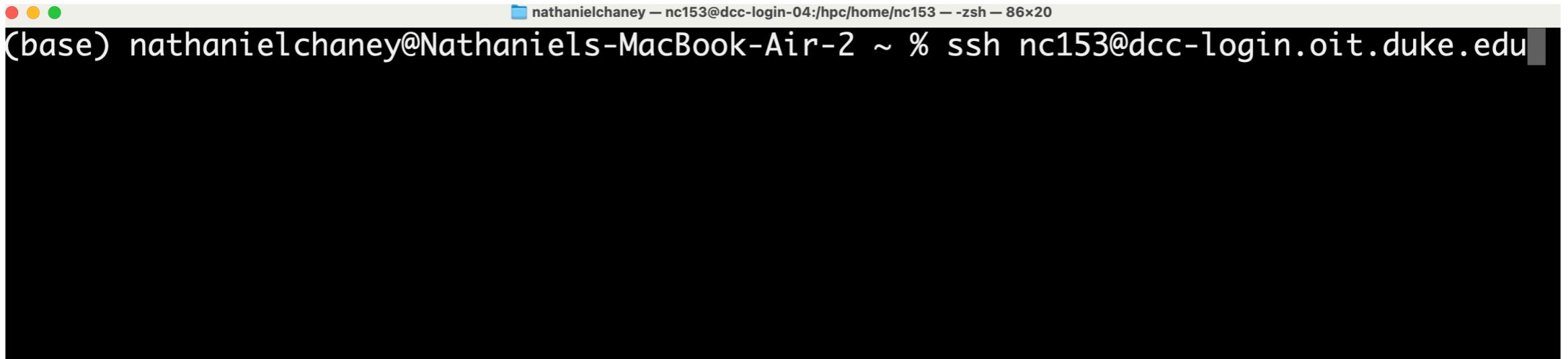
First open up a terminal/command line. This example is using MacOS.



A screenshot of a Mac OS X terminal window. The window title bar shows the path 'nathanielchaney — nc153@dcc-login-04:/hpc/home/nc153 — -zsh — 86x20'. The main pane of the terminal contains the command '(base) nathanielchaney@Nathaniels-MacBook-Air-2 ~ % ssh nc153@dcc-login.oit.duke.edu' followed by a blank black area where the command would be executed.

SSH

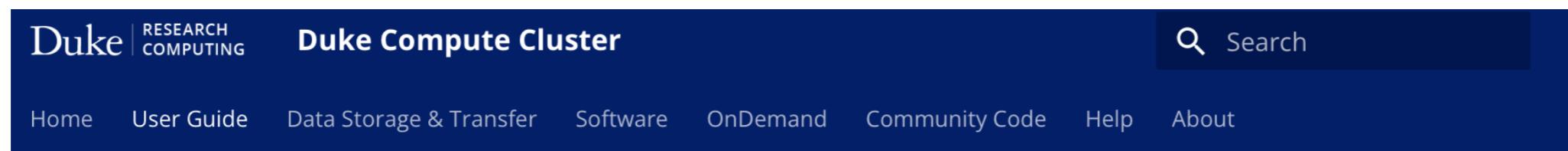
First open up a terminal/command line. This example is using MacOS.

A screenshot of a Mac OS X terminal window. The window title bar shows 'nathanielchaney — nc153@dcc-login-04:/hpc/home/nc153 — zsh — 86x20'. The main pane of the terminal contains the command '(base) nathanielchaney@Nathaniels-MacBook-Air-2 ~ % ssh nc153@dcc-login.oit.duke.edu' followed by a black rectangular redaction box covering the rest of the terminal's output.

```
(base) nathanielchaney@Nathaniels-MacBook-Air-2 ~ % ssh nc153@dcc-login.oit.duke.edu
```

Want passwordless entry? Learn about
public key authentication

Passwordless login on DCC?



User Guide

[Login](#)

Software

Partitions

H200

Scheduling Jobs

Usage Reports

Home > User Guide

Logging into the DCC

As a shared computing resource, all work done on the DCC must be done by submitting jobs to the [scheduler](#). Login nodes must not be used to execute computing tasks.

Acceptable use of login nodes include:

- lightweight file transfers,
- script and configuration file editing,
- job submission and monitoring,
- [BaseSpace BaseMount](#) - users can use basemount to mount BaseSpace into their home directories.

To minimize disruption and ensure a comfortable working environment for users, resource limits are enforced on login nodes, and processes started there will automatically be terminated if their resource usage (including CPU time, memory and run time) exceed those limits.

.config file

```
(base) nathanielchaney@Nathaniels-MacBook-Air-2 ~ % vi $HOME/.ssh/config
```

```
Host cluster
  HostName chaney.egr.duke.edu
  User nc153

Host dcc
  HostName dcc-login.oit.duke.edu
  User nc153
```

```
(base) nathanielchaney@Nathaniels-MacBook-Air-2 ~ % ssh dcc
```

More terminals!

```
...c153@dcc-login-05:/hpc/home/nc153 - ssh dcc ... ...c153@dcc-login-05:/hpc/home/nc153 - ssh dcc ...53@dcc-login-05:/hpc/home/nc153 - ssh dcc ...53@dcc-login-05:/hpc/home/nc153 - ssh dcc
Last login: Thu Jan  8 09:05:51 on ttys003
ssh dcc%
(base) nathanielchaney@NathanielsAir2 ~ % ssh dcc
Last login: Thu Jan  8 09:05:53 2026 from nathanielsair2.wired.duke.local
#####
# Please report any issues or questions to: oitresearchsupport@duke.edu      #
#                                     #
# Duke Research Computing info and documentation: https://rc.duke.edu      #
#                                     #
# My next patch run is Thursday (01-22-2026) at  3 AM                         #
#####
3.8GiB (15%) used of  25GiB in /hpc/home/nc153
179GiB (17%) used of 1.0TiB in /hpc/group/chaneylab
  0B (0%) used of 25TiB in /work/nc153
(base) nc153@dcc-login-05 ~ $
```

```
(base) nc153@dcc-login-05 ~ $ ps -aux | grep "sshd: nc153"
root      3810045  0.0  0.0  49600 14076 ?          Ss   09:05   0:00 sshd: nc153 [priv]
]
nc153     3810071  0.0  0.0  49600  7864 ?          S    09:05   0:00 sshd: nc153@pts/3
1
root      3810192  0.0  0.0  49600 14068 ?          Ss   09:05   0:00 sshd: nc153 [priv]
]
nc153     3810218  0.0  0.0  49600  7944 ?          S    09:05   0:00 sshd: nc153@pts/3
3
root      3810340  0.0  0.0  49600 14024 ?          Ss   09:05   0:00 sshd: nc153 [priv]
]
nc153     3810366  0.0  0.0  49600  7828 ?          S    09:05   0:00 sshd: nc153@pts/3
4
root      3810499  0.0  0.0  49600 14024 ?          Ss   09:05   0:00 sshd: nc153 [priv]
]
nc153     3810525  0.0  0.0  49600  7840 ?          S    09:05   0:00 sshd: nc153@pts/3
5
nc153     3810983  0.0  0.0   6420  2616 pts/35  S+   09:06   0:00 grep --color=auto
```

Terminal multiplexer

```
(base) nc153@dcc-login-05 ~ $ tmux
```

Once you are inside a tmux session, use these shortcuts with the **Prefix** (`Ctrl + b`).

Working with Panes (Splits)

- **Prefix + %** : Split the screen **vertically** (left and right).
- **Prefix + "** : Split the screen **horizontally** (top and bottom).
- **Prefix + Arrow Key** : Move your cursor between panes.
- **Prefix + z** : **Zoom** (maximize) the current pane. Press again to shrink it back.
- **Prefix + x** : Close the current pane.

Working with Windows (Tabs)

- **Prefix + c** : Create a **new window**.
- **Prefix + n / p** : Switch to the **next** or **previous** window.
- **Prefix + 0-9** : Jump directly to a window by its number.
- **Prefix + ,** : **Rename** the current window (makes it easier to find later).

Terminal multiplexer

```
p(chaneylab) 0B (0%) used of 25TiB in /work/nc153 | 1.0TiB in /hpc/gro|e/nc153  
53 | 0B (0%) used of 1.0TiB in /hpc/gro  
(base) nc153@dcc-login-05 ~ $ ^C | 25TiB in /work/nc|up/chaneylab  
(base) nc153@dcc-login-05 ~ $ |153 | 0B (0%) used o  
(base) nc153@dcc-log|f 25TiB in /work/nc  
(base) nc153@dcc-log|153  
|in-05 ~ $ |(base) nc153@dcc-log  
|in-05 ~ $ ls  
|CEE690  
|miniconda3  
|Miniconda3-latest-Li  
|nux-x86_64.sh  
|README  
|test.py  
|text2.txt  
|text.txt  
|(base) nc153@dcc-log  
|in-05 ~ $ |  
[3] 0: bash* "nc153@dcc-login-05:/h" 09:12 08-Jan-26
```

Terminal multiplexer

```
p(chaneylab) 0B (0%) used of 25TiB in /work/nc153 | 1.0TiB in /hpc/gro|e/nc153  
53 | 0B (0%) used of 1.0TiB in /hpc/gro  
(base) nc153@dcc-login-05 ~ $ ^C | 25TiB in /work/nc|up/chaneylab  
(base) nc153@dcc-login-05 ~ $ |153 | 0B (0%) used o  
| (base) nc153@dcc-log|f 25TiB in /work/nc  
| (base) nc153@dcc-log|153  
| in-05 ~ $ | (base) nc153@dcc-log  
| | in-05 ~ $ ls  
| | CEE690  
| | miniconda3  
| | Miniconda3-latest-Li  
| | nux-x86_64.sh  
| | README  
| | test.py  
| | text2.txt  
| | text.txt  
| | (base) nc153@dcc-log  
| | in-05 ~ $ |  
[3] 0: bash* "nc153@dcc-login-05:/h" 09:12 08-Jan-26
```

I personally never use this. But I know many people that do and are huge fans

Navigating the linux filesystem

Navigation

- CD

```
(base) nc153@dcc-login-03 ~ $ cd CEE690/
```

- ls

```
(base) nc153@dcc-login-03 ~ $ ls  
CEE690 miniconda3 Miniconda3-latest-Linux-x86_64.sh README test.py
```

- pwd

```
(base) nc153@dcc-login-03 ~ $ pwd  
/hpc/home/nc153
```

Finding data

- find

```
(base) nc153@dcc-login-03 ~ $ find . -name "*.py"
```

- grep

```
(base) nc153@dcc-login-03 ~ $ grep 'Remember' README  
# Remember, you only have 10 GB in your new home directory.
```

Regular expressions (Regex)

- . (Dot) - Matches any single character

```
(base) nc153@dcc-login-03 ~ $ grep '.y' README
# DON'T PANIC! Your old home directory still exists.
# The contents are located at /hpc/group/<groupname>/<your netid>
# Remember, you only have 10 GB in your new home directory.
```

- * (Asterisk) - Matches zero or more of the preceding element

```
(base) nc153@dcc-login-03 ~ $ grep 'a*' README
# DON'T PANIC! Your old home directory still exists.
# The contents are located at /hpc/group/<groupname>/<your netid>
# https://rc.duke.edu/working-with-new-dcc-storage-options/
# Remember, you only have 10 GB in your new home directory.
```

- [] (Brackets) - Matches any one character inside the brackets

```
(base) nc153@dcc-login-03 ~ $ grep '[yuh]' README
# DON'T PANIC! Your old home directory still exists.
# The contents are located at /hpc/group/<groupname>/<your netid>
# https://rc.duke.edu/working-with-new-dcc-storage-options/
# Remember, you only have 10 GB in your new home directory.
```

Regular expressions (Regex)

- . (Dot) - Matches any single character

```
(base) nc153@dcc-login-03 ~ $ grep '.y' README
# DON'T PANIC! Your old home directory still exists.
# The contents are located at /hpc/group/<groupname>/<your netid>
```

There are many...

```
# https://rc.duke.edu/working-with-new-dcc-storage-options/
# Remember, you only have 10 GB in your new home directory.
```

- [] (Brackets) - Matches any one character inside the brackets

```
(base) nc153@dcc-login-03 ~ $ grep '[yuh]' README
# DON'T PANIC! Your old home directory still exists.
# The contents are located at /hpc/group/<groupname>/<your netid>
# https://rc.duke.edu/working-with-new-dcc-storage-options/
# Remember, you only have 10 GB in your new home directory.
```

Piping and redirection

- The pipe (|)

```
(base) nc153@dcc-login-03 ~ $ ls | wc -l  
5
```

- Redirection - Overwrite a file (>)

```
(base) nc153@dcc-login-03 ~ $ ls > text.txt
```

- Redirection - Append to a file (>>)

```
(base) nc153@dcc-login-03 ~ $ ls >> text.txt
```

Moving data

- cp - copy files within the host

```
(base) nc153@dcc-login-03 ~ $ cp text.txt text2.txt
```

- scp - copy files to/from client/server

```
(base) nathanielchaney@Nathaniels-MacBook-Air-2 ~ % scp test.py nc153@dcc-login.oit  
duke.edu: .
```

- Rsync - Checks which parts have changed and only sends those

Permissions

- Chmod - Modify permissions on a file

```
(base) nc153@dcc-login-03 ~ $ chmod 755 text.txt
```

Common Octal Values:		
Number	Permissions	Description
777	rwx rwx rwx	Anyone can read, write, and execute (generally not recommended for security).
755	rwx r-x r-x	Owner has full access; group and others can read and execute (common for public directories/programs).
644	rw-r-- r--	Owner can read and write; group and others can only read (common for data files).
700	rwx -----	Only the owner has full access; others have no rights (for private files/directories).

Understanding commands

```
(base) nc153@dcc-login-05 ~/miniconda3 $ grep --help
Usage: grep [OPTION]... PATTERN [FILE]...
Search for PATTERN in each FILE.
Example: grep -i 'hello world' menu.h main.c
PATTERN can contain multiple patterns separated by newlines.
```

Pattern selection and interpretation:

-E, --extended-regexp	PATTERNS are extended regular expressions
-F, --fixed-strings	PATTERNS are strings
-G, --basic-regexp	PATTERNS are basic regular expressions
-P, --perl-regexp	PATTERNS are Perl regular expressions
-e, --regexp=PATTERN	use PATTERN for matching
-f, --file=FILE	take PATTERN from FILE
-i, --ignore-case	ignore case distinctions in patterns and data
--no-ignore-case	do not ignore case distinctions (default)
-w, --word-regexp	match only whole words
-x, --line-regexp	match only whole lines
-z, --null-data	a data line ends in 0 byte, not newline

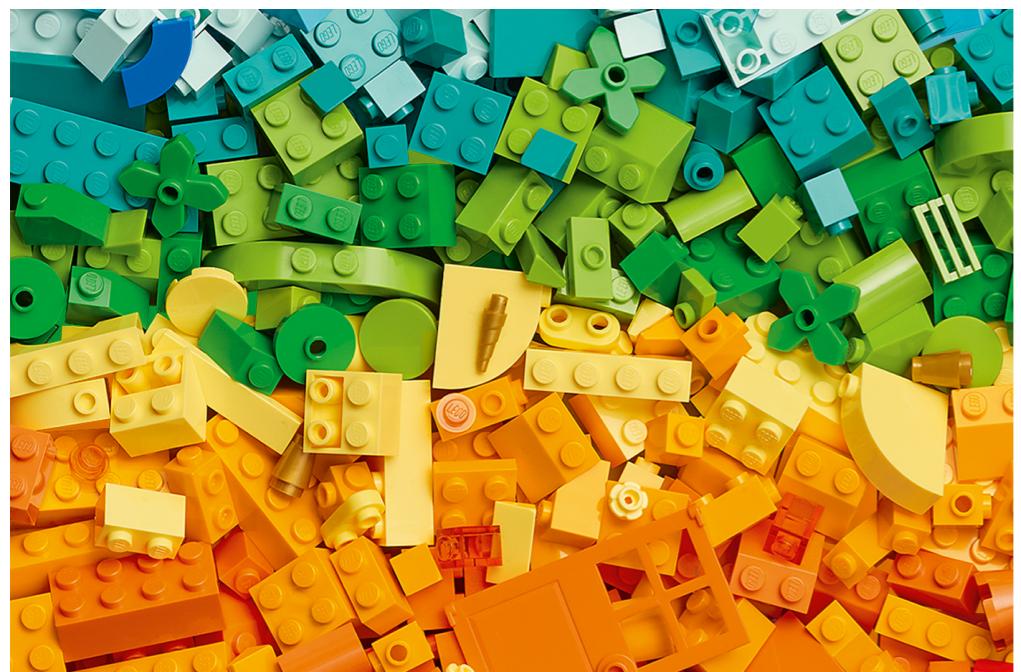
Linux commands you should know... maybe?



Top 50 Linux Commands you must know

- | | | | | |
|-----------|------------|-------------|----------------------|---------------------------|
| 1. ls | 11. cat | 21. diff | 31. kill and killall | 41. apt, pacman, yum, rpm |
| 2. pwd | 12. echo | 22. cmp | 32. df | 42. sudo |
| 3. cd | 13. less | 23. comm | 33. mount | 43. cal |
| 4. mkdir | 14. man | 24. sort | 34. chmod | 44. alias |
| 5. mv | 15. uname | 25. export | 35. chown | 45. dd |
| 6. cp | 16. whoami | 26. zip | 36. ifconfig | 46. wheris |
| 7. rm | 17. tar | 27. unzip | 37. traceroute | 47. whatis |
| 8. touch | 18. grep | 28. ssh | 38. wget | 48. top |
| 9. ln | 19. head | 29. service | 39. ufw | 49. useradd |
| 10. clear | 20. tail | 20. ps | 40. iptables | 50. passwd |

Lego brick nature of terminal commands



Lots of unique commands



Combine many to create a
“custom command”

Example I: User info

```
[base) nc153@dcc-login-05 ~/miniconda3 $ ps -aux | grep nc153
nc153 3808656 0.0 0.0 25368 15248 ? Ss 09:03 0:00 /usr/lib/systemd
/systemd --user
nc153 3808659 0.0 0.0 198040 8528 ? S 09:03 0:00 (sd-pam)
nc153 3809911 0.0 0.0 6016 3368 ? Ss 09:05 0:00 tmux
nc153 3809912 0.0 0.0 17664 5528 pts/32 Ss+ 09:05 0:00 -bash
root 3810045 0.0 0.0 49600 14076 ? Ss 09:05 0:00 sshd: nc153 [pri
v]
nc153 3810071 0.0 0.0 49600 7864 ? S 09:05 0:00 sshd: nc153@pts/
31
nc153 3810072 0.0 0.0 17784 5652 pts/31 Ss 09:05 0:00 -bash
nc153 3861915 0.0 0.0 19088 4504 pts/31 R+ 09:26 0:00 ps -aux
nc153 3861916 0.0 0.0 6420 2604 pts/31 S+ 09:26 0:00 grep --color=aut
o nc153
```

Example II: Count and save

```
(base) nc153@dcc-login-05 ~/miniconda3 $ ls -la | wc -l >& log.txt
```

```
(base) nc153@dcc-login-05 ~/miniconda3 $ cat log.txt  
19
```

Example III: Read and process

```
[base) nc153@dcc-login-05 ~/miniconda3 $ tail -10 LICENSE.txt | grep [acp]
A stand-alone version of pycryptodome.
A software library for encryption, decryption, signatures, password hashing and
more.
pynacl
A Python binding to the Networking and Cryptography library, a crypto library w
ith the stated goal of improving usability, security and speed.
Last updated September 28, 2020
```

Example III: Read and process

```
(base) nc153@dcc-login-05 ~/miniconda3 $ tail -10 LICENSE.txt | grep [acp]  
A stand-alone version of pycryptodome.
```

In practice, I just do most of these things directly in Python nowadays. However, you can save yourself a lot of time if you learn to use these well

Python on the system: Miniconda

Getting Started > Miniconda

Miniconda

Copy page ▾

Miniconda is a free, miniature installation of Anaconda Distribution that includes only conda, Python, the packages they both depend on, and a small number of other useful packages. See the full list of packages in Miniconda's [release notes](#).

If you need more packages, use the `conda install` command to install from thousands of packages available by default in Anaconda's public repo, or from other channels, like conda-forge or bioconda.

- ▶ Is Miniconda free for me?
- ▶ Should I install Miniconda or Anaconda Distribution?

Miniconda environment

```
(base) nc153@dcc-login-05 ~/miniconda3 $ conda create -n cee690  
Collecting package metadata (current_repodata.json): done  
Solving environment: done
```

```
(base) nc153@dcc-login-05 ~/miniconda3 $ conda activate cee690  
(cee690) nc153@dcc-login-05 ~/miniconda3 $ █
```

Miniconda environment

```
(base) nc153@dcc-login-05 ~/miniconda3 $ conda create -n cee690  
Collecting package metadata (current_repodata.json): done  
Solving environment: done
```

```
(base) nc153@dcc-login-05 ~/miniconda3 $ conda activate cee690  
(cee690) nc153@dcc-login-05 ~/miniconda3 $ █
```

Conda works really well until it doesn't

Install packages

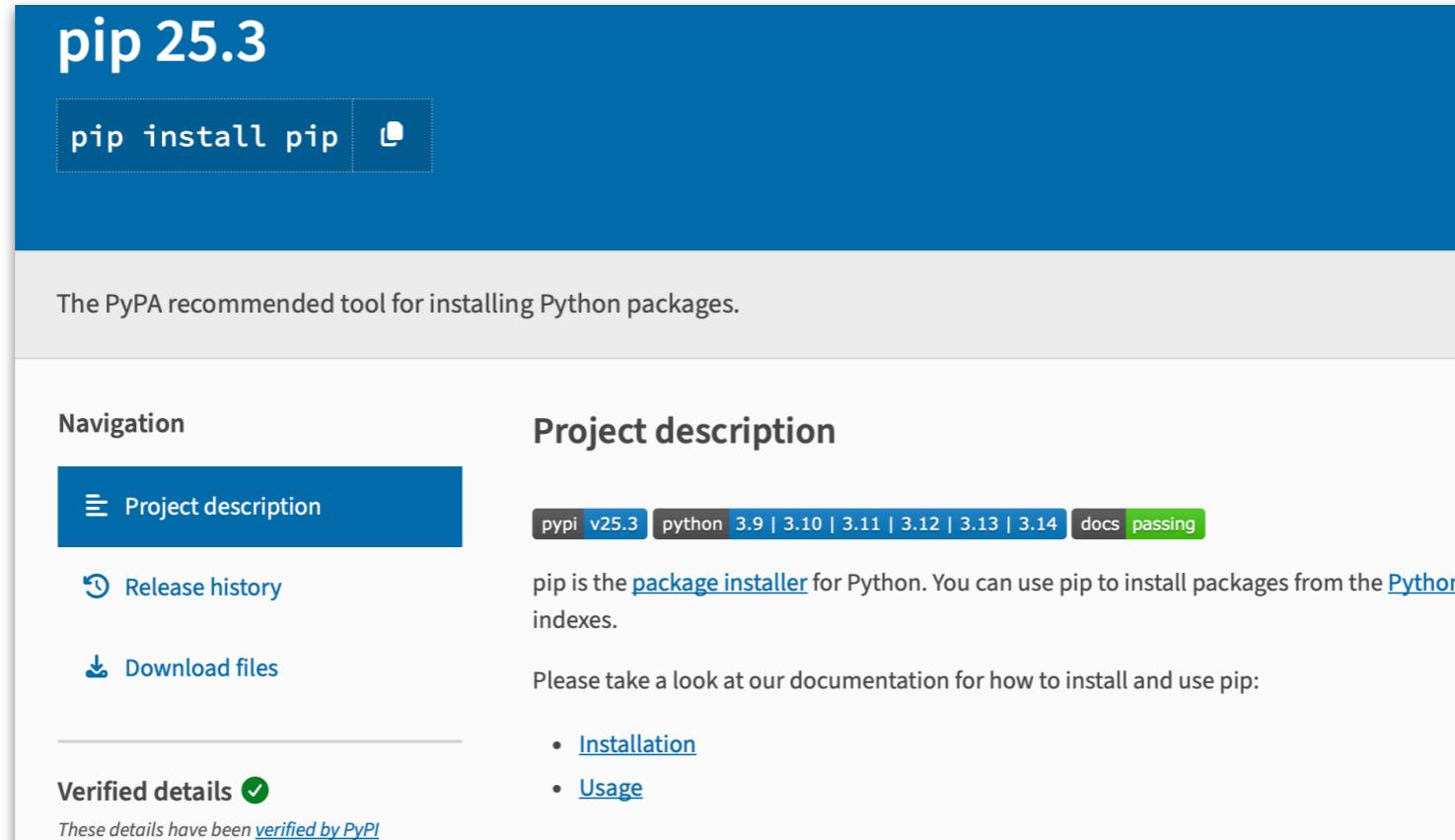
```
(cee690) nc153@dcc-login-05 ~/miniconda3 $ conda install -c conda-forge rasterio  
Collecting package metadata (current_repodata.json): done  
Solving environment: done
```

```
[Proceed ([y]/n)? y ]
```

Downloading and Extracting Packages

libxml2-2.15.1	44 KB	#####	100%
libcblas-3.11.0	18 KB	#####	100%
minizip-4.0.10	91 KB	#####	100%
ncurses-6.5	871 KB	#####	100%
libev-4.33	110 KB	#####	100%
libjxl-0.11.1	1.7 MB	#####	100%
certifi-2026.1.4	147 KB	#####	100%
libxml2-16-2.15.1	543 KB	#####	100%
liblapack-3.11.0	18 KB	#####	100%
libstdcxx-ng-15.2.0	27 KB	#####	100%
openssl-3.6.0	3.0 MB	#####	100%
liblzma-5.8.1	110 KB	#####	100%
libnghttp2-1.67.0	651 KB	#####	100%

More control? pip



The screenshot shows the PyPI project page for pip 25.3. The header is blue with the text "pip 25.3". Below it is a button labeled "pip install pip". The main content area has a light gray background. At the top left of this area, the text "The PyPA recommended tool for installing Python packages." is displayed. On the left, there's a "Navigation" sidebar with links for "Project description", "Release history", and "Download files". The "Project description" link is highlighted with a blue bar. To its right, under "Project description", there's a badge for "pypi v25.3" and another for "python 3.9 | 3.10 | 3.11 | 3.12 | 3.13 | 3.14" followed by a green "docs passing" badge. Below these badges, a paragraph of text describes pip as the package installer for Python. Further down, there's a section titled "Please take a look at our documentation for how to install and use pip:" with two bullet points: "Installation" and "Usage". At the bottom left of the main content area, there's a "Verified details" section with a green checkmark and a note: "These details have been [verified by PyPI](#)".

```
(cee690) nc153@dcc-login-05 ~/miniconda3 $ pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.10.8-cp314-cp314-manylinux_2_27_x86_64.manylinux_
  6_64.whl.metadata (52 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.3-cp314-cp314-manylinux_2_27_x86_64.manylinux_
  64.whl.metadata (5.5 kB)
```

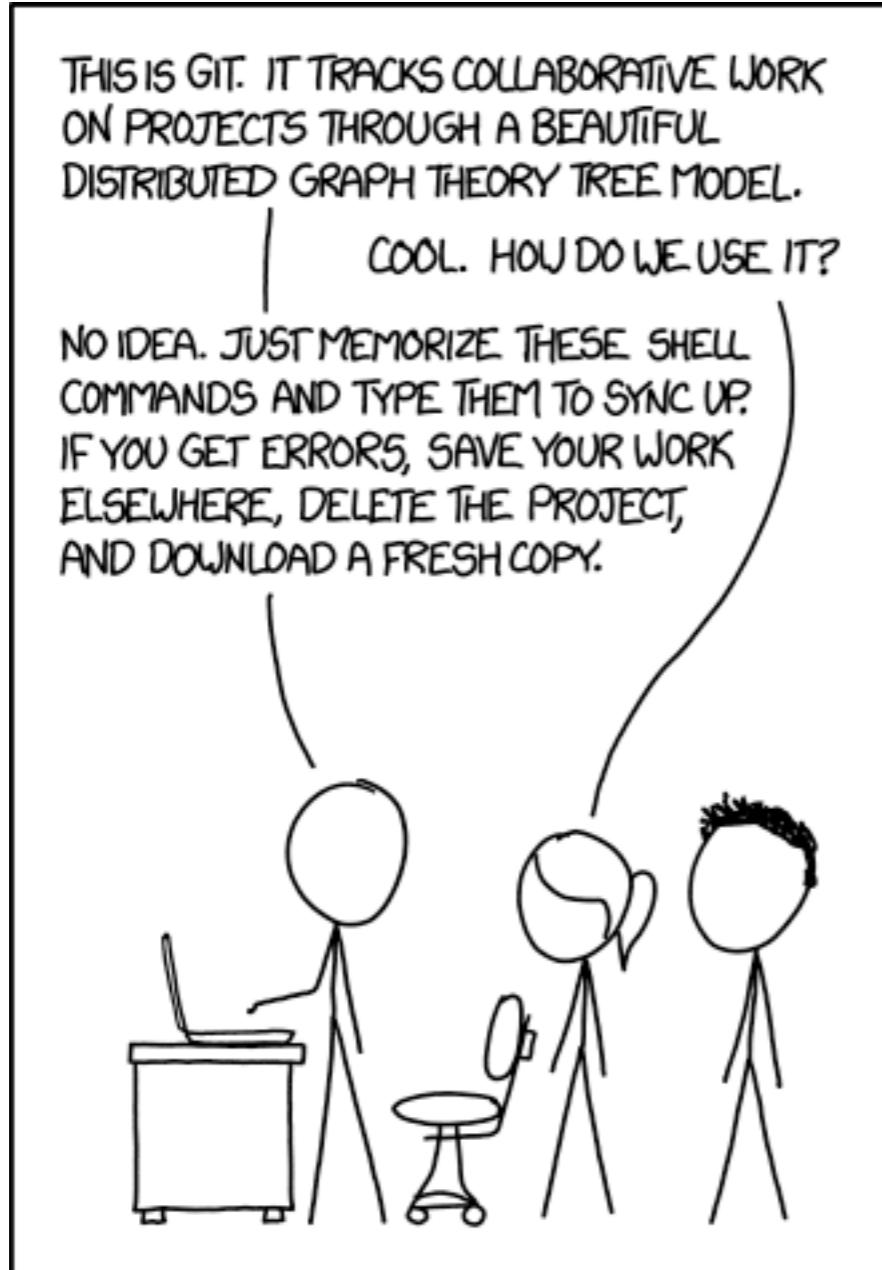
Python scripts

```
print("hello world")
```

```
~  
~  
~  
~  
~  
~
```

```
(cee690) nc153@dcc-login-05 ~/miniconda3 $ python test.py  
hello world
```

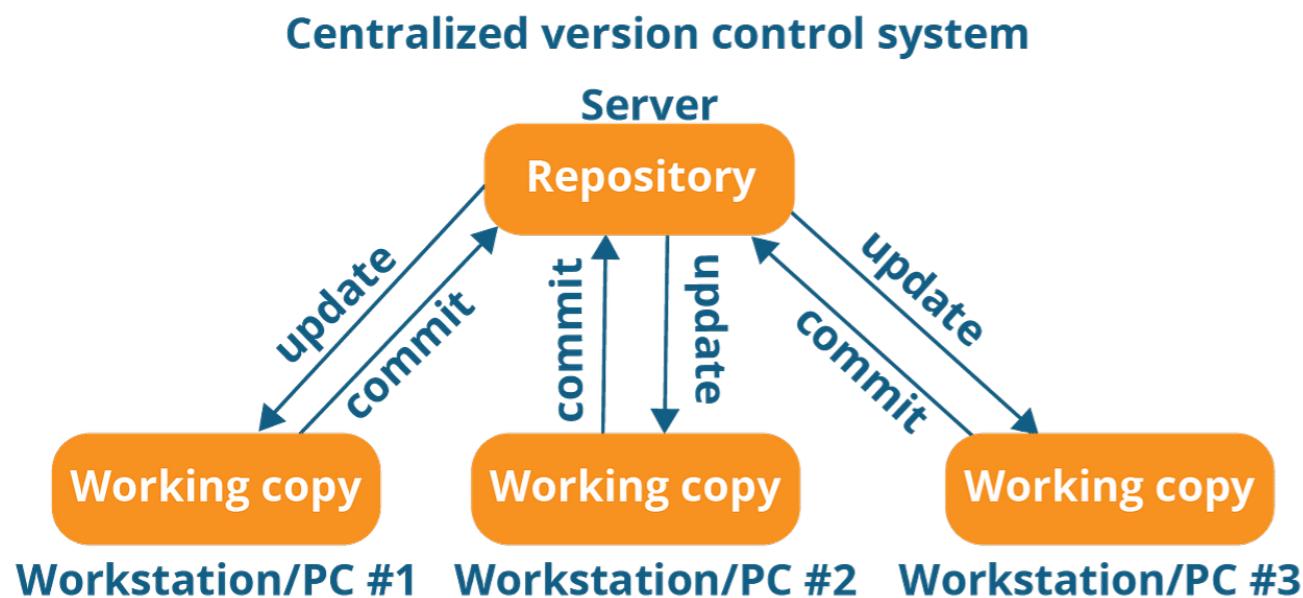
Version control



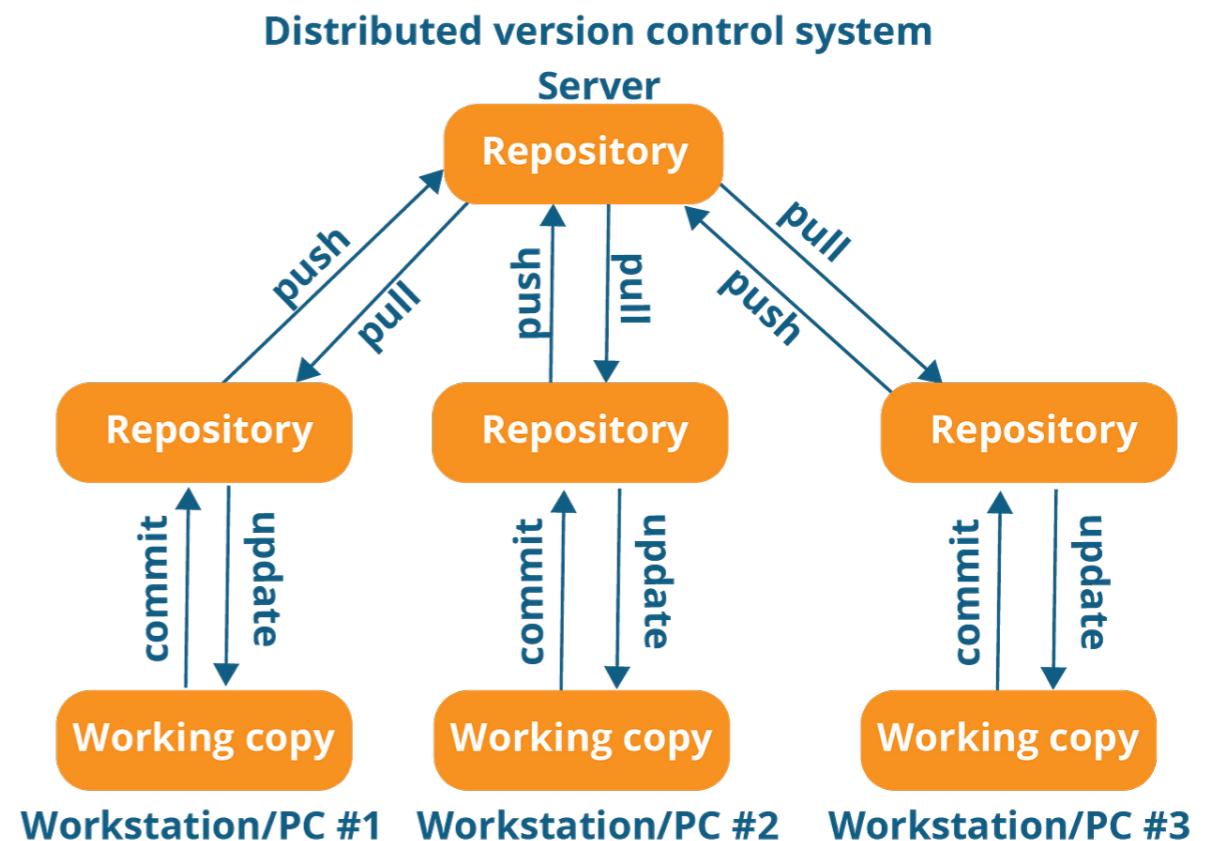
- Time Machine
- Scientific reproducibility (tag)
- Branching - experimentation
- Collaboration
- Commits “tell” the story (if commented appropriately)
- Conflict resolution (merge)
- Automation friendly
- Code review
- Portability

Distributed vs Centralized version control

E.g., SVN



E.g., Git



Git commands I

Getting started

- *git init* - Creates a new, empty Git repository
- *git clone <url>* - Copies an existing Git repository

Where you spend most of your time

- *git status* - What has been changed
- *git add <file>* - Add files to the staging area
- *git commit -m “message”* - Changes added to repository
- *git log* - History of all commits

Git commands II

Branching

- *git branch <name>* - Parallel timeline
- *git checkout* - File or branch
- *git merge <name>* - Combines branches (conflicts!)

Inspection

- *git diff* - Changes between current files and last commit
- *git stash* - Pauses and “hides” current work so you can retrieve old code or change branches

Git commands III

Remote synchronization

- *git remote add origin <url>* - Links local repo to remote server
- *git push origin <branch>* - Sends local commits to the remote server
- *git pull* - Fetches latest changes and merges them with your current branch
- *git fetch* - Downloads data but does not change your files

Git commands III

Remote sy

- *git remote* - Manage multiple Git servers
- *git push* or *git pull* - Work with a remote server
- *git pull* - Fetch changes from your current branch
- *git fetch* - Fetch changes from another branch



Hence...

THIS IS GIT. IT TRACKS COLLABORATIVE WORK
ON PROJECTS THROUGH A BEAUTIFUL
DISTRIBUTED GRAPH THEORY TREE MODEL.

| COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZIZE THESE SHELL
COMMANDS AND TYPE THEM TO SYNC UP.
IF YOU GET ERRORS, SAVE YOUR WORK
ELSEWHERE, DELETE THE PROJECT,
AND DOWNLOAD A FRESH COPY.



Homework

Figure out what remote system you will be using and get things working

LLMs are your friend and are encouraged