Introduction to Unix Shell

Joshua Quan, Demography Computing Director

2024-09-19

Learning Goals

- Understand why Shell interfaces are important
- How to navigate and work with Files and Directories
- Pipes and Filters (combining commands)
- Finding Things
- Intro to version control with Git

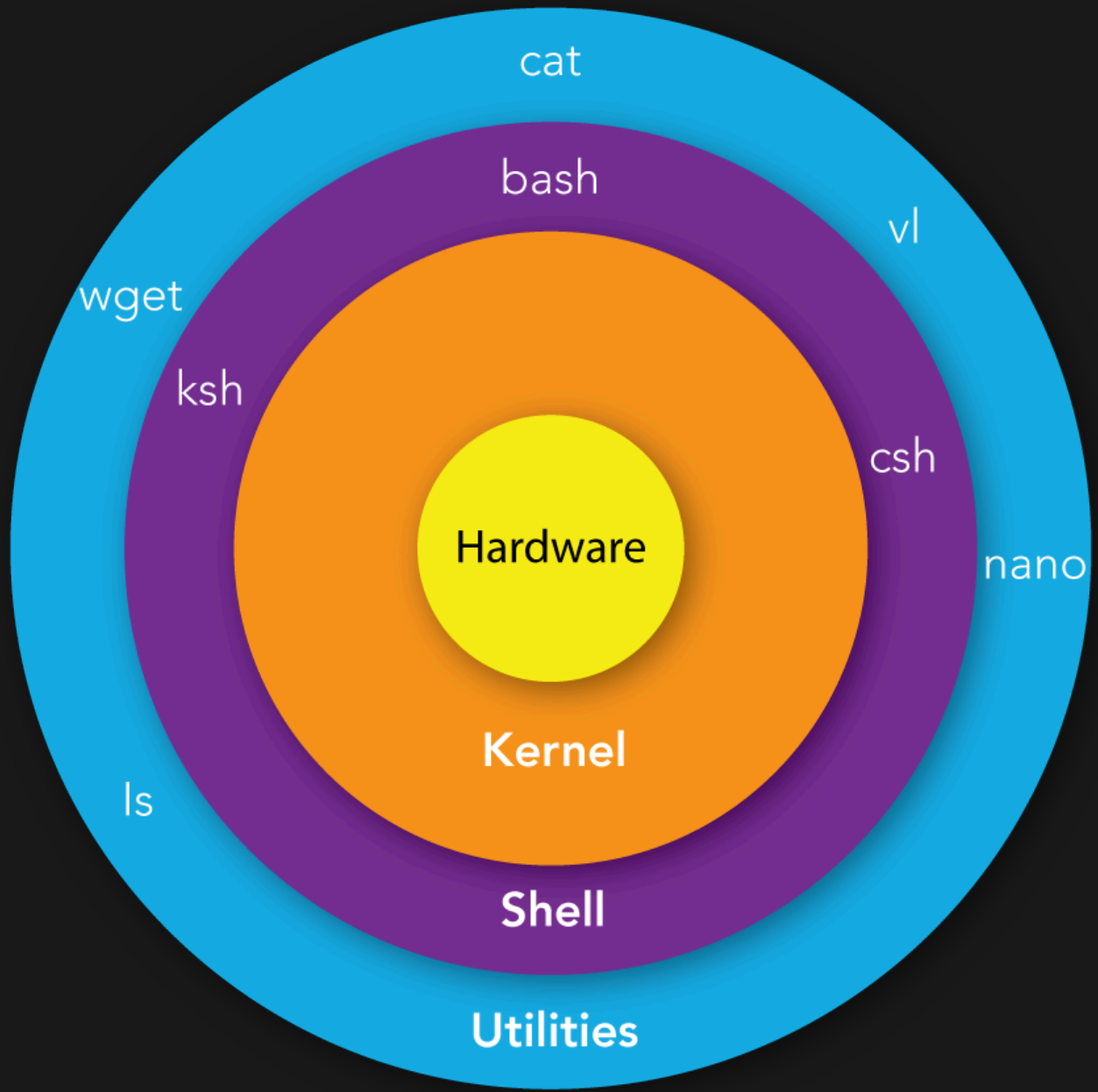
The Shell

- The interface to your OS, the “kernel”
- Almost any server you remotely access for compute, data access, websites, etc. are going to use the shell in some way to interface
- Bash (Born Again Shell), ZSH, KSH are the most common *nix shells.

The Shell

- Unix OS are the oldest and most commonly used *open source* kernels in computing. Origins from Bell Labs in the 70s
- More efficient than GUI which relies on pointing and clicking
- As a result automating tasks is common using the shell
- Windows implements Bash emulation: Git Bash; Windows Subsystem for Linux

The Shell

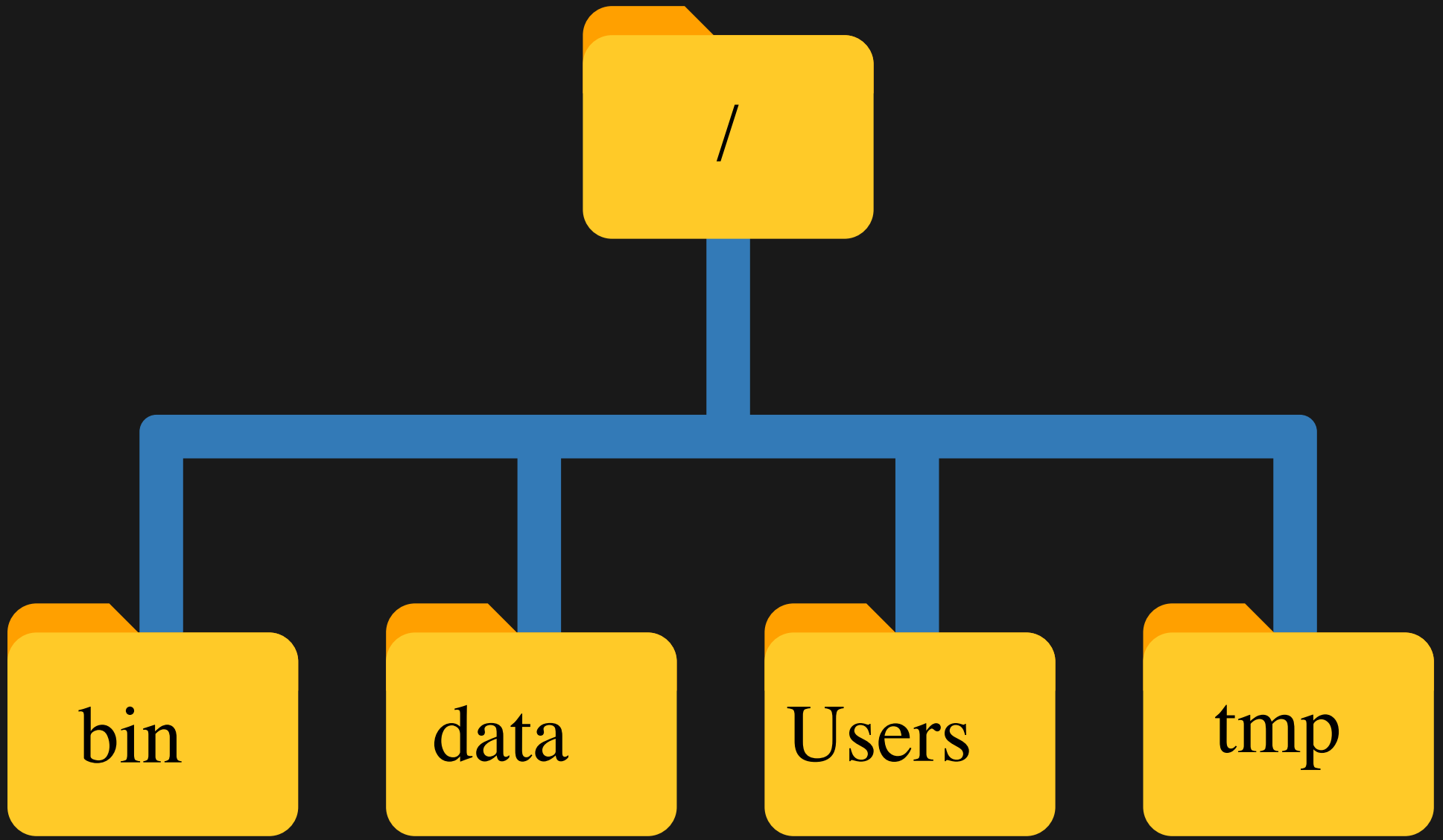

cat
bash
vl
wget
ksh
csh
Hardware
nano
Kernel
ls
Shell
Utilities

Terminal

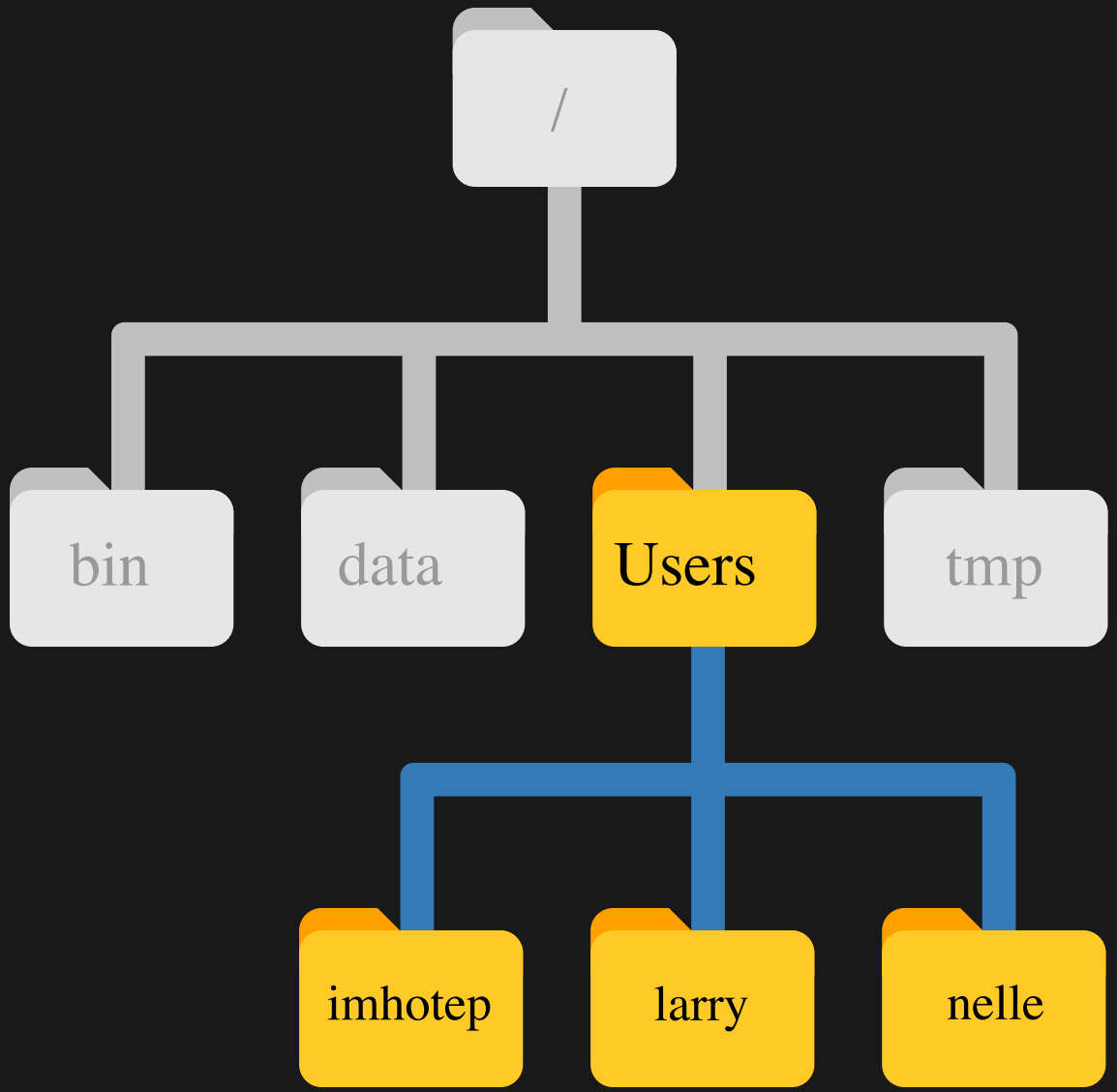
We will use the Datahub Terminal

- https://datahub.berkeley.edu/

Files and Directories



Files and Directories



Files and Directories

```
# print the current working directory, "where am i?"
pwd
```

```
# change to a directory; cd takes you home
cd
cd /
```

```
# list files and directories
ls
```

```
# what does "-l" mean here?
ls -l
```

```
# every command has a help page
man ls
```

```
# touch creates an empty file
touch {filename.txt}
```

```
# there are many file editors.
nano {filename.txt}
micro {filename.txt}
```

Files and Directories

```
# copies a file
cp {old} {new}
```

```
# creates a new directory
mkdir {path}
```

```
# moves (renames) a file or directory
mv {old} {new}
```

```
# remoters (deletes) a file. Be very careful with this!
rm {path}
```

```
# * is a wildcard that matches zero or more characters in a filename, so *.csv matches all files ending
*
```

Files and Directories

- Absolute paths vs. Relative paths
- A relative path specifies a location starting from the current location.
- An absolute path specifies a location from the root of the file system.

Files and Directories

- Starting at `/home/jovyan/recipes` how would you make it back to the home directory, `/home/jovyan`?

```
cd .
cd /
cd /home/jovyan
cd ../..
cd ~
cd home
cd
cd ..
```

Files and Directories

```
cd . # no
cd / # no
cd /home/jovyan # Yes: jovyan's home directory is /home/jovyan.
cd ../.. # No: this goes up two levels, i.e. ends back in root /.
cd ~ # Yes: ~ stands for the user's home directory, in this case /home/
cd home # No: this would navigate into a directory home in the current
cd # Yes: shortcut to go back to the user's home directory.
cd .. # Yes: goes up one level.
```

Pipes and Filters

```
# displays the contents of its inputs.
cat
```

```
# displays the first 10 lines of its input.
head
```

```
# displays the last 10 lines of its input.
tail
```

```
# sort its inputs
sort
```

```
# counts lines, words, and characters in its inputs.
wc
```

Pipes and Filters

```
# | is a pipeline: the output of the first command is used as the input to the second.
first | second
```

```
sort data.csv | less
```

```
#  > allows you to take output of one command and write to a file
wc -l {file} > counts.txt
```

Finding Things

```
# finds files with specific properties that match patterns.
find ./recipes

find /bin/python*
```

```
# selects lines in files that match patterns.
grep -in dumplings recipes.txt
```

Unix Shell Summary

- Shell is interface to your kernel, there are many shells. *nix based shells are most common
- All *nix based filesystems have a similar structure
- Relative vs. Absolute paths
- pwd, cd, ls, man, touch, cp, mkdir, mv, rm, cat, head, tail, sort, wc, find, grep
- Soooooooo much more to learn! Awesome Bash

Introduction to Git

- Git is a free and open-source version control system (VCS) that helps you manage changes to your codebase, documents, or website content.
- It’s like having a “versioning” superpower for your projects.
- Origin is in Software Development in the early 2000s to manage complex code bases
- Imagine working on a document or a piece of code with multiple people. Without Git, you’d have to rely on manual tracking and renaming files (e.g., `document_v1.doc`, `document_v2.doc`, etc.). This can lead to confusion, lost changes, and frustration.

Introduction to Git

Introduction to Git

Git “solves” these problems by:

1. Tracking changes: Git records every modification, that you tell it to, made to your code or documents.

2. Creating snapshots: Git saves each version of your project as a separate snapshot (commit), allowing you to revert to any previous state.

3. Managing multiple versions: Git enables parallel development, making it easy to work on different features or branches without affecting the main codebase.

Introduction to Git

Key Git Concepts:

1. **Repository (Repo)**: The central location where all your project's history is stored.
2. **Commit**: A snapshot of changes made to your code or documents at a specific point in time.
3. **Branch**: A separate line of development that allows you to work on new features without affecting the main codebase.
4. **Merge**: Combining two branches, resolving conflicts and merging the changes.
5. **Push**: Uploading local commits to a remote repository (e.g., GitHub).
6. **Pull**: Downloading changes from a remote repository into your local workspace.

Introduction to Git

Difference between Git and Github?

- Git is the system/language that allows you manage your project history
- Github/Gitlab is an online repository that hosts your project and has many features to simplify Git.

Introduction to Git

Step 0:

Create your git profile

```
git config --global user.email "you@example.com"

git config --global user.name "Your Name"

git config --list
```

Introduction to Git

Step 1: Create a New Repository

Open your terminal or command prompt and run the following command:

```
git init

git init --initial-branch=main
```

```
ls -a
```

This will create a new directory called `.git` in your current working directory, which will hold all the metadata for your Git repository.

Introduction to Git

Step 2: Add Files to the Repository

Use the following command to add them to your repository:

```
git add recipes.txt
```

This will stage these files in preparation for committing them.

Use this frequently!

```
git status
```

Introduction to Git

Step 3: Commit Changes

Run the following command to commit your changes:

```
git commit -m "Initial commit"
```

This will save a new snapshot of your codebase, including all the changes made since the last commit.

Introduction to Git

Step 4: Push Updates

Create an account on GitHub or another remote hosting platform. Then, run the following command to link your local repository to a remote one:

```
git remote add origin https://github.com/your-username/repository-name.
git remote add origin git@github.com:your-username/repository-name.git
```

```
git remote -v
```

Next, push your changes to the remote repository using:

```
git push -u origin main
```

This will upload your committed changes to the remote repository.

Introduction to Git

Step 4.5: Authentication...

Starting in 2001, Github got rid of passwords for allowing pushes to a repo. They require you use

- ssh keys (secure shell), more conventional, my preference
- personal access token, exlusive to github

```
ssh-keygen -t rsa -b 4096
ls ~/.ssh
cat ~/.ssh/id_rsa.pub
```

copy the contents to your github account > Settings > SSH and GPG Keys

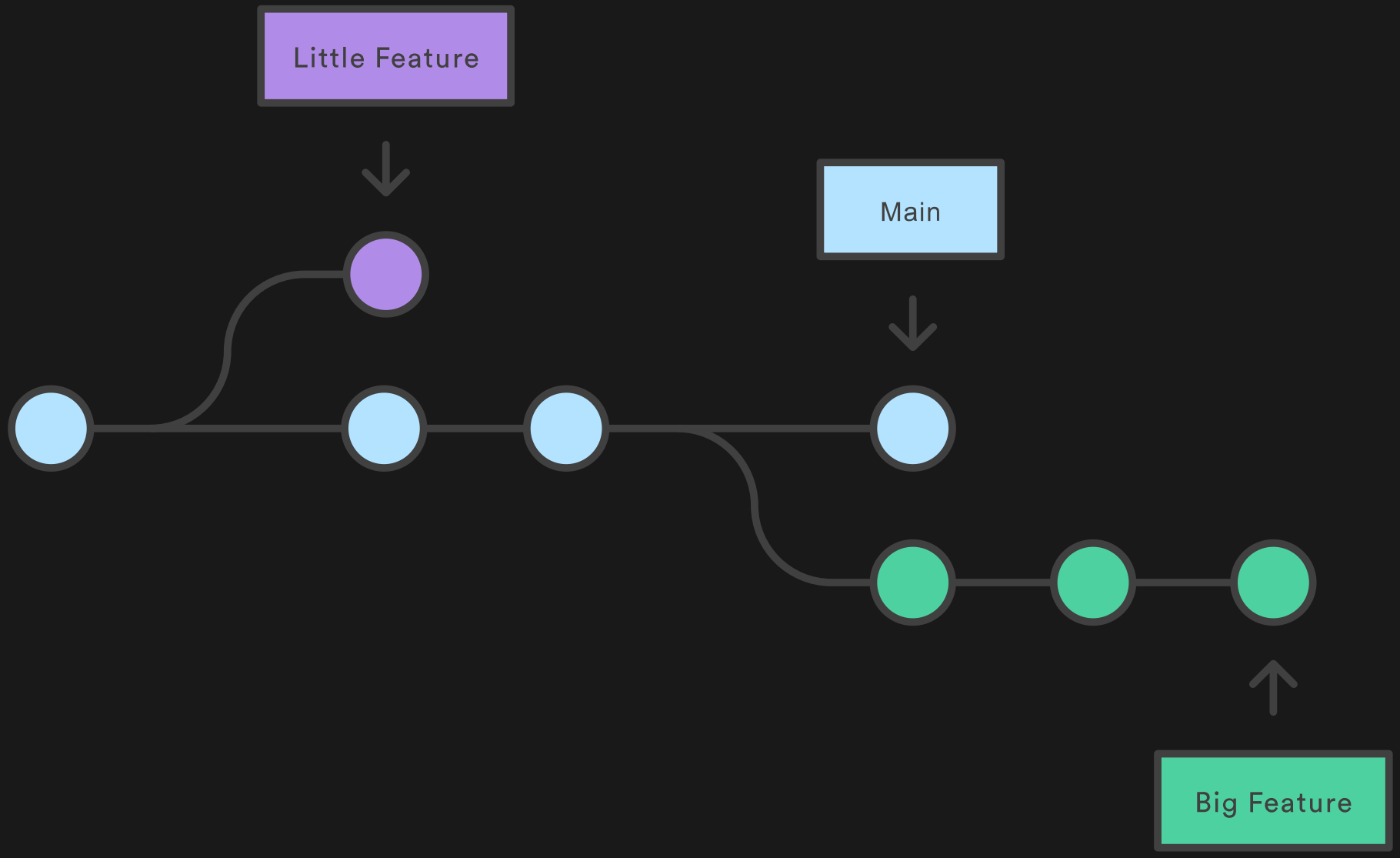
Introduction to Git

Step 4.5: Authentication...

```
eval `ssh-agent -s`
ssh-add
```

```
ssh -T git@github.com
```

Introduction to Git



Introduction to Git

To see history of commits

```
git log
```

To see what branch you are on

```
git branch
```

To go back to a specific commit, fool around with it, and be able to come back to where you are

```
git checkout {hash}
git checkout main
```

To create a new branch and then merge it with main branch

```
git branch {branch-name}
git checkout {branch-name}
git checkout main
git merge --no-ff {branch-name}
```

Introduction to Git

To go back *one commit* and erase local work since then

```
git reset --hard HEAD~1
```

To go back to a *specific commit* and erase local work since then

```
git reset --hard {hash}
```

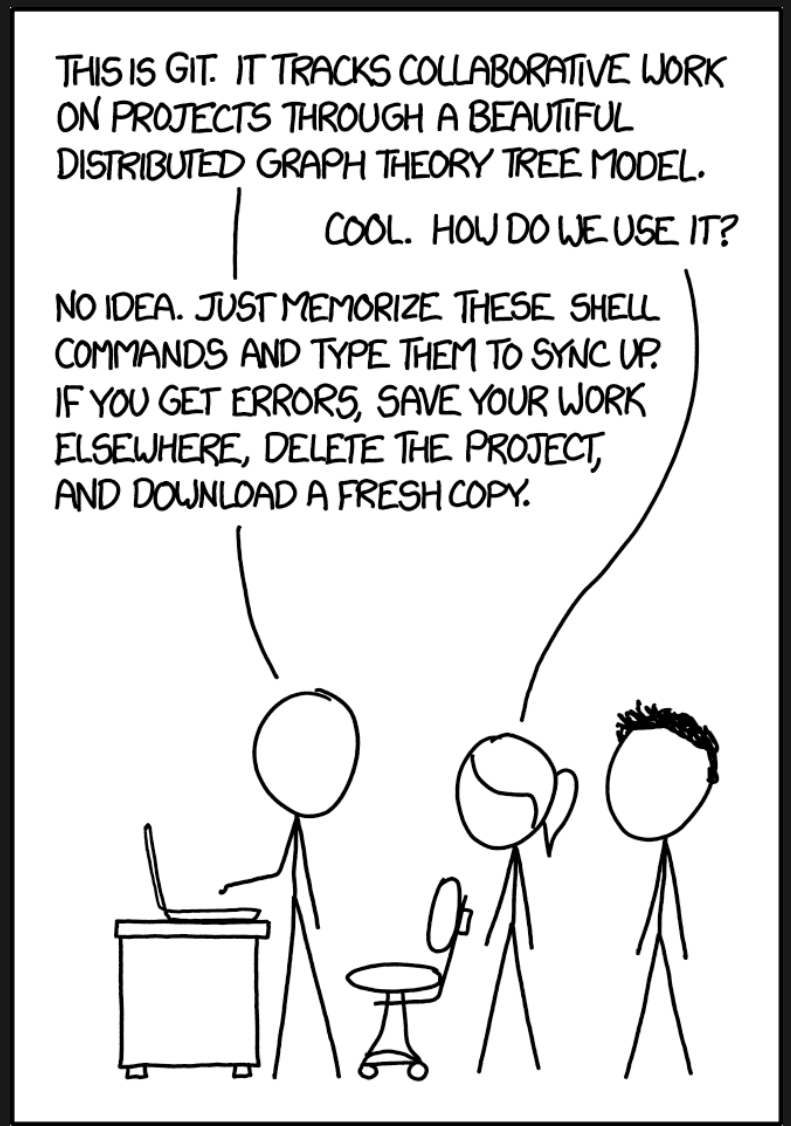
Introduction to Git

Making Pull Requests

- In working with others you might “fork” the original repo, make changes to it and submit your changes back “upstream” for review.

1. In Github, Fork the upstream repository
2. Clone it to your environment, make your changes, *add*, *commit*, *push* them.
3. In Github, click on *Pull Requests* and comment on what changes you’ve made and why.

Introduction to Git



Wrapping Up

- Git is a robust version control system suited for collaboration with yourself or hundreds of people.
- The basic workflow: git init, git add, git commit, git push, git pull
- For more complex workflows, you work with “branches” that can be merged into a main branch. You can go backward in time and revert changes.
- Pull Requests are copies of another repo that you contribute to that you fork, clone, submit for approval
- Github and Gitlab are great tools to enhance the Git experience.