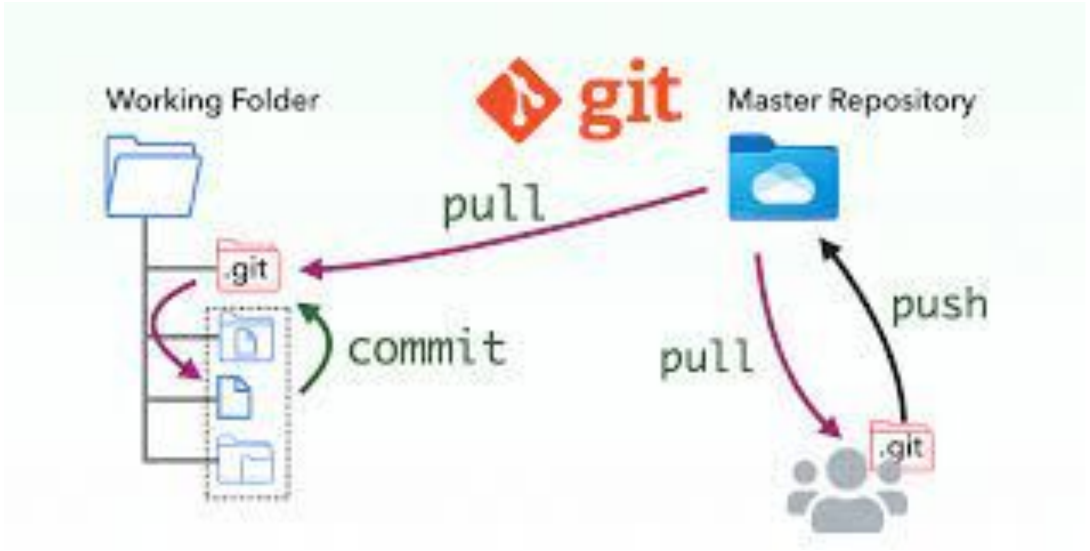
CSE 374 Programming concepts and tools

Winter 2024 Instructor: Alex McKinney

Today

We'll learn about:

- Version Control
- `git`



Version Control

User Story: Individual

Does any of the following sound familiar?

- Your code was working great! Then *you made a few changes* and now everything is broken and you **saved over the previous version**?
- You **accidently delete** a critical file and can't get it back.
- Your computer was broken or stole and now **all of your files are gone**!
- While writing a paper for one of your classes you save each version as **final_paper.doc**, **final_paper2.doc**, **final_paper_actually_this_time.doc**, **UGH.doc**

There has to be a better way to manage versions...

User Story: Team

Does any of the following sound familiar?

- My partner and I are paired up for a project for one of our classes. We usually work together in-person, but sometimes we have to work remotely.
    - **Who keeps the most up-to-date version of the project?**
    - How do we **share changes** with each other?
    - What if I want to compare the changes my partner made?
- Someone changed or removed all my work, so now I have to do it all over again.
- How do we **keep backups** of important files? Who stores them on their computer?

Version Control

Version Control: Software system that keeps track of changes to a set of files.

You likely use version control all the time:

- `Ctrl+Z` to undo changes
- In Google Docs you can see who made what changes to a file

Lots of people have a use-case for version control

  - Software systems
  - Law firms need to keep track of document changes over time

Examples: subversion, perforce, mercurial, cvs, sourcesafe, **git**

Git

Linus Torvalds

The creator of both Linux and Git!

- Linus → Linux

Git was originally created in 2005 for the sole purpose of continuing development of Linux.

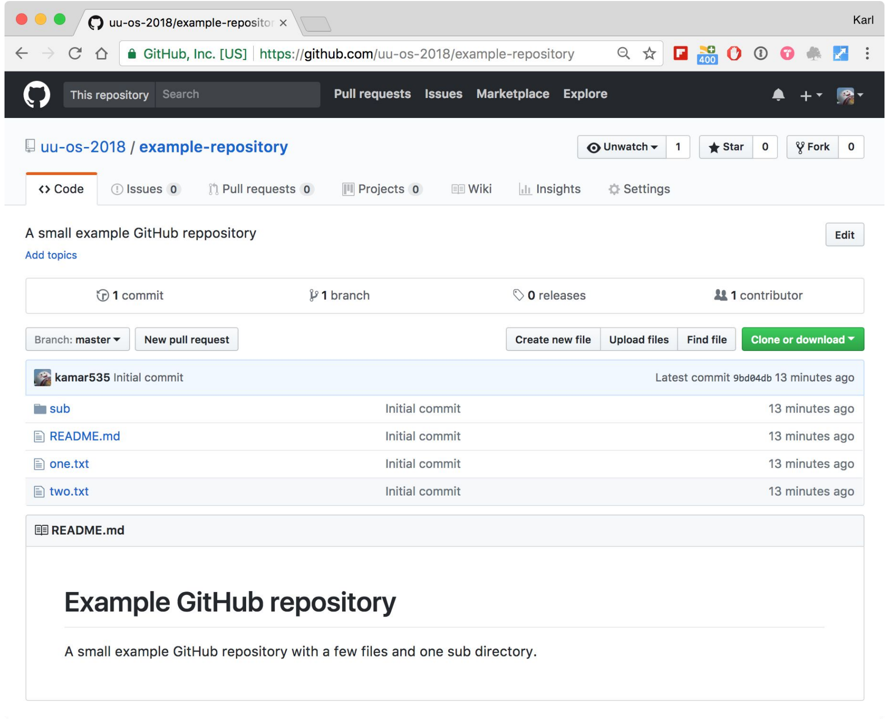
- Linux was originally written in 1991.
- The old version control system revoked its free license for Linux

Git is the most popular version control system, with approximately ~95% of developers using it.

Repository

A repository, commonly referred to as a repo is a location that stores a copy of all files.

Synonymous to the root directory in a file system (i.e. '/').


uu-os-2018/example-reposito ×

Karl

GitHub, Inc. [US] | https://github.com/uu-os-2018/example-repository

This repository | Search | Pull requests | Issues | Marketplace | Explore

uu-os-2018 / example-repository

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

A small example GitHub reppository

Edit

Add topics

1 commit | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

kamar535 Initial commit — Latest commit 9bd04db 13 minutes ago

sub	Initial commit	13 minutes ago
README.md	Initial commit	13 minutes ago
one.txt	Initial commit	13 minutes ago
two.txt	Initial commit	13 minutes ago

README.md

Example GitHub repository

A small example GitHub repository with a few files and one sub directory.

Repository Do’s and Don’ts

What is stored inside of a repository?

- Source code files (i.e. .c files, .java files, etc)
- Build files (Makefiles, build.xml)
- Images, general resources files

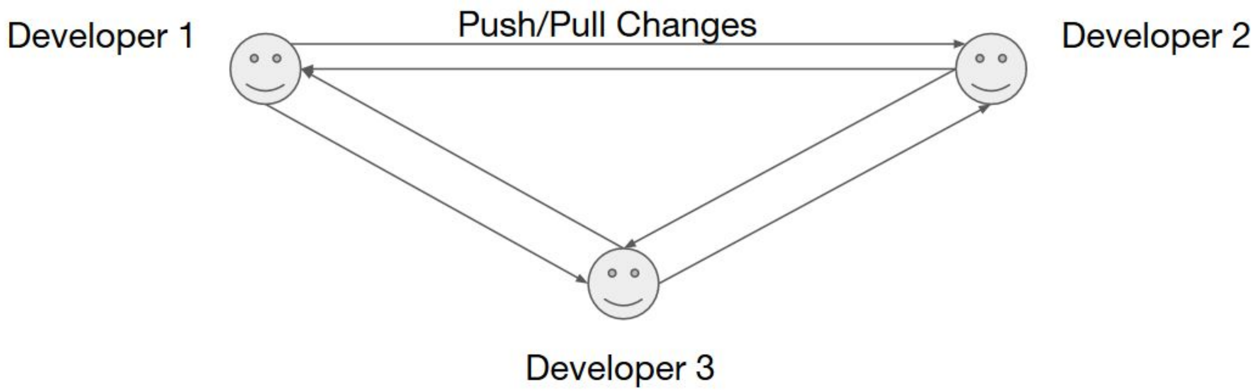
What should NOT be stored inside of a repository (generally)

- Object files (i.e. .class files, .o files)
- Executable binary files (executable scripts are OK!)

More on these types of files when we start with C next week!

Sharing Changes

- With git, everyone working on the project has a copy of the repository and its history
  - Everyone has a local copy of the repository, which is what we use to make our own changes.
  - We share changes by *pushing* and *pulling*



Central Git Repository

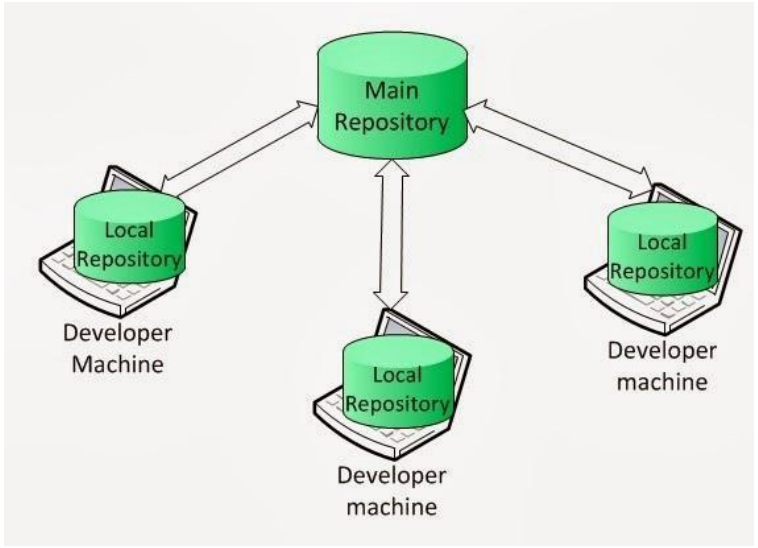
Keep an "origin" copy of the repo on a Git server

- The remote repository is the *defacto* central repository

Each user ***clone*** the repo to create a local copy

A user make changes, ***add*** and ***commit*** those to their copy and ***push*** to save them in the remote repository

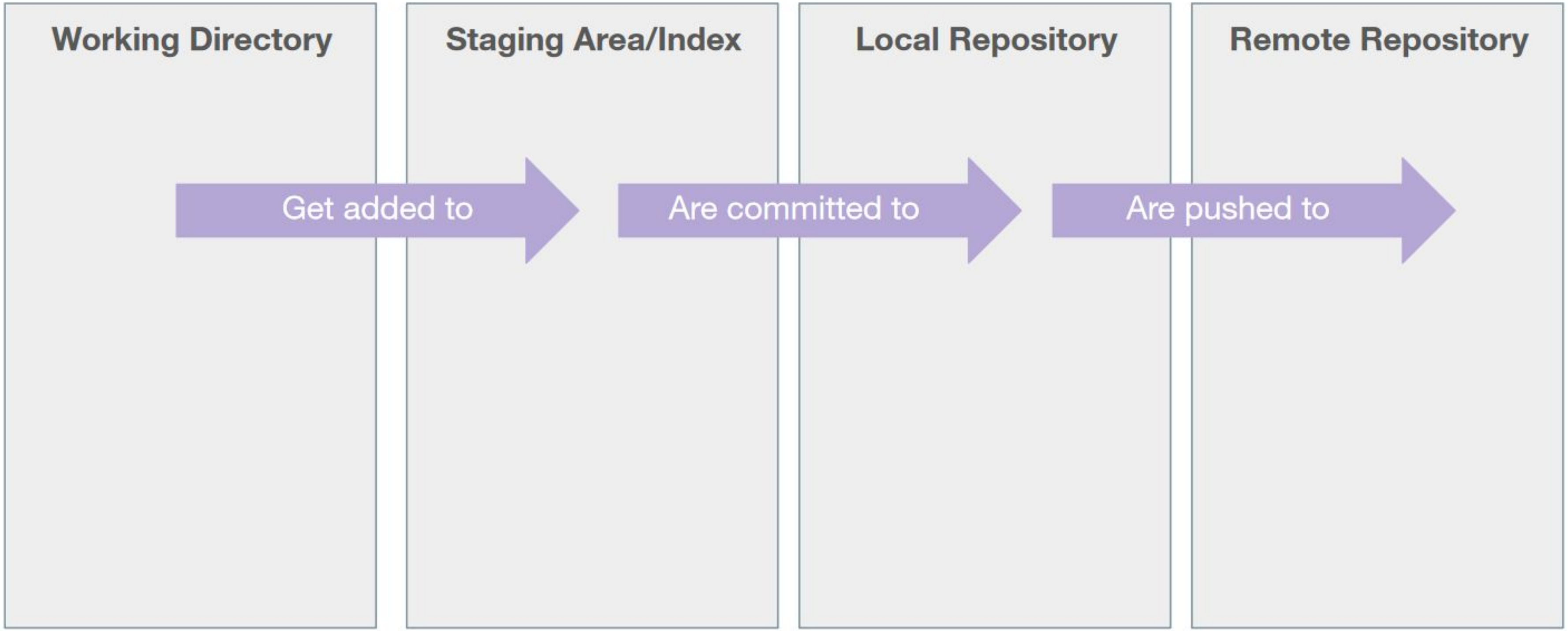
All users ***pull*** from the central server periodically to get changes (instead of from each other).



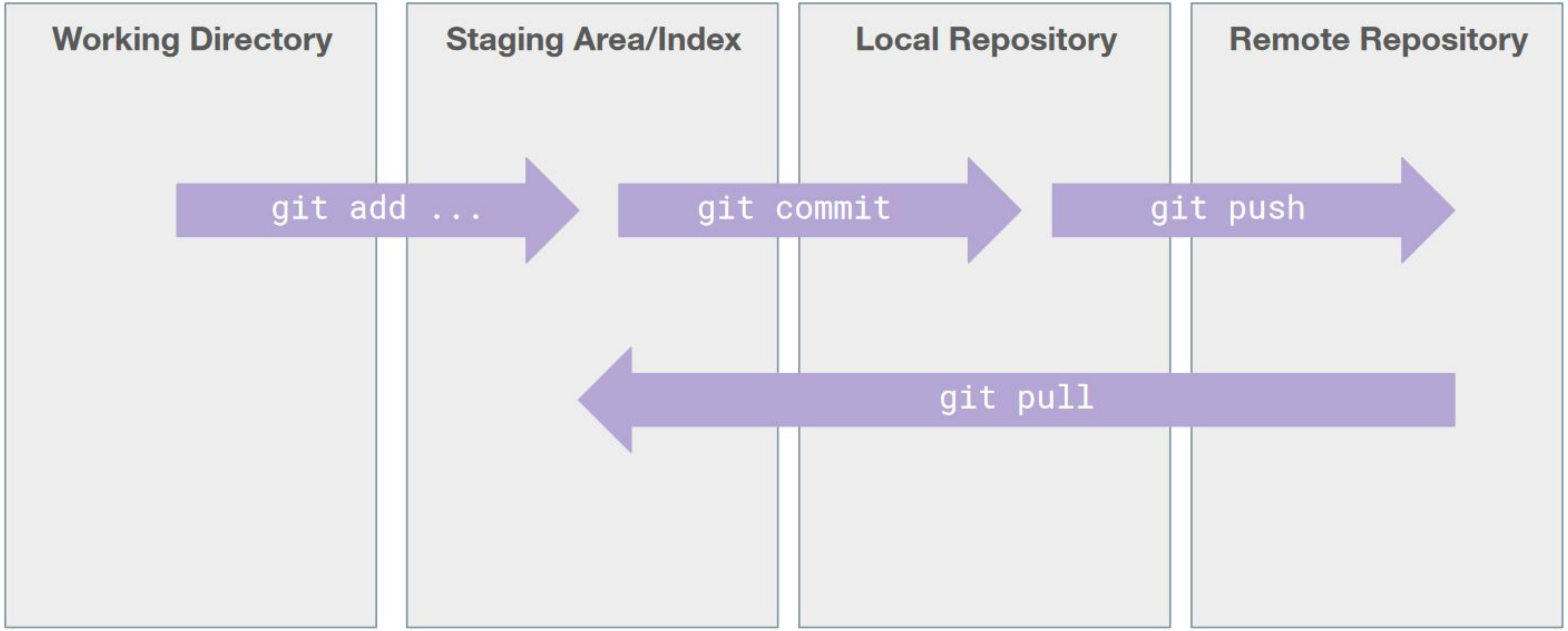
Git: Four Phases

Working Directory	Staging Area/Index	Local Repository	Remote Repository
Working changes	Changes you're preparing to commit	Local copy of the repository with your committed changes	Remote shared repository (Usually stored with a platform like GitHub/Gitlab)

Git: Four Phases



Git: Four Phases with Remote

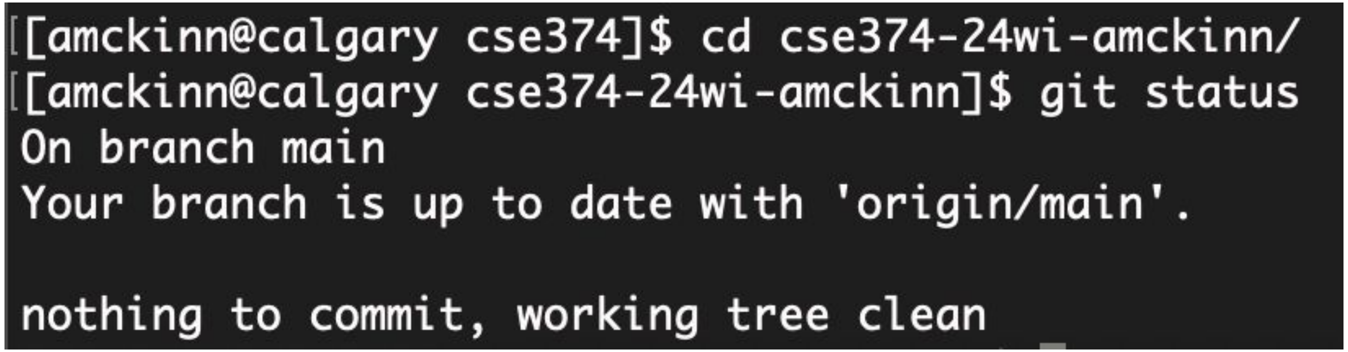


Git Commands

git init, git clone, git add, git commit, git pull, git push, git status, …

The “git" Command

- The **`git`** command is the primary way of interacting with git
- You must `cd` into the folder where your repo is stored, or any subfolder within it
- Used like any other shell command!



```
[amckinn@calgary cse374]$ cd cse374-24wi-amckinn/
[amckinn@calgary cse374-24wi-amckinn]$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Git Commands

Command	Operation
`git clone` *`url [dir]`*	Copy a Git repository so you can add to it
`git add` *`file`*	Adds file contents to the staging area
`git commit`	Records a snapshot of the staging area
`git status`	View the status of your files in the working directory and staging area
`git diff`	Shows diff of what is staged and what is modified but unstaged
`git help [`*`command`*`]`	Get help info about a particular command
`git pull`	Fetch from a remote repo and try to merge into the current branch
`git push`	Upload your local commits to the remote repository/server

Creating a Git Repo

Two common scenarios (only do one of these):

1. To create a new local Git repo in your current directory:

   ```
   git init
   ```

   - This will create a .git directory in your current directory.
   - Then you can commit files in that directory into the repo.

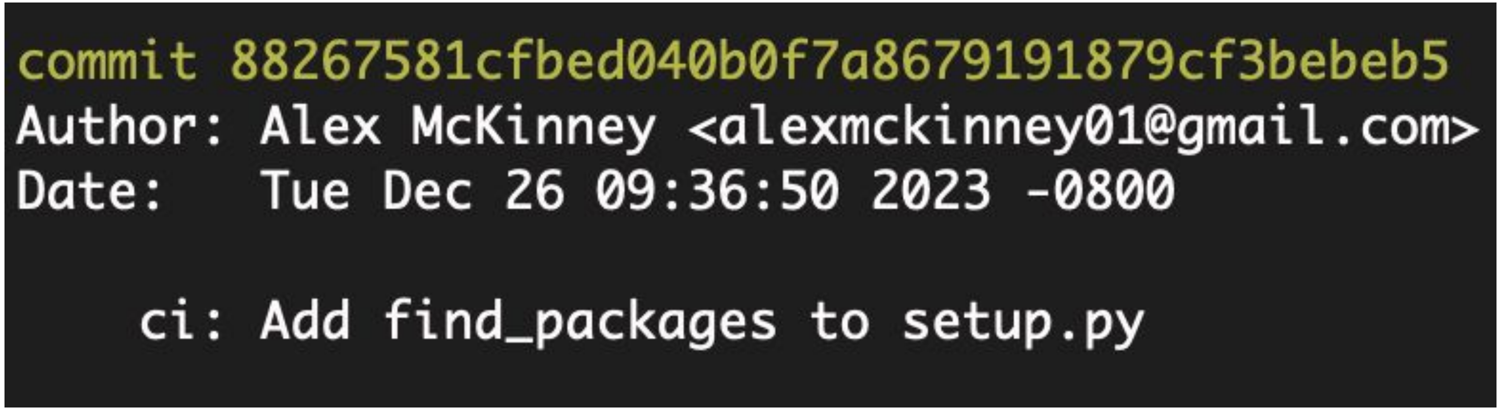
2. To clone a remote repo to your current directory:

   ```
   git clone url [localDirectoryName]
   ```

   - This will create the given local directory, containing a working copy of the files from the repo, and a .git directory.

What is a "commit"?

- A ***commit*** is a single set of changes made to your repository
- Also records:
  - The name of the author
  - The date and time
  - A commit message: short sentence describing what that commit did
- Identified by an ID, or "SHA"



```
commit 88267581cfbed040b0f7a8679191879cf3bebeb5
Author: Alex McKinney <alexmckinney01@gmail.com>
Date:   Tue Dec 26 09:36:50 2023 -0800

    ci: Add find_packages to setup.py
```

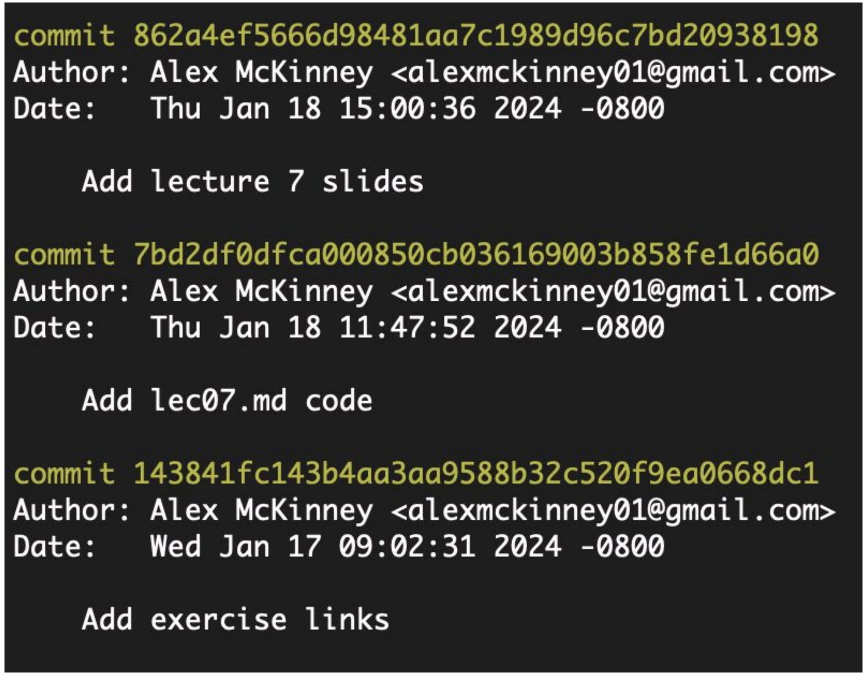
Commit Messages

- Commit messages are the way you remind yourself and tell others what you did
- Commit messages should be **descriptive**
    - E.g. "Added test for predicting null string"
    - *not* "changed test"
- Commit messages should be **short/medium length**
    - If you want to know *exactly* what code was changed, you can check the full changes.

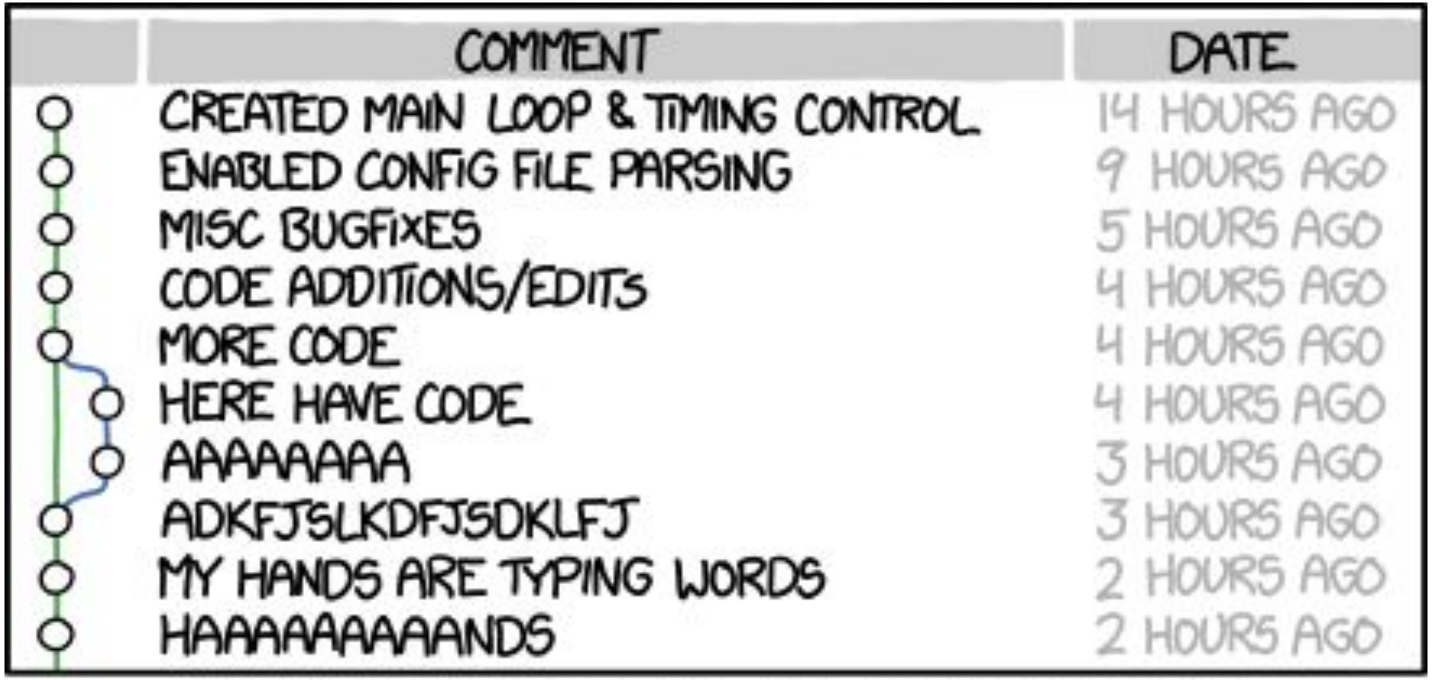
Commit History

- A repository's history is a series of "commits"
- Each commit makes changes to the files in the repo
- *Commit history* serves as a log of the changes everyone made
- `git log` to view the commit history

Commit early and often!



```
commit 862a4ef5666d98481aa7c1989d96c7bd20938198
Author: Alex McKinney <alexmckinney01@gmail.com>
Date:   Thu Jan 18 15:00:36 2024 -0800

    Add lecture 7 slides

commit 7bd2df0dfca000850cb036169003b858fe1d66a0
Author: Alex McKinney <alexmckinney01@gmail.com>
Date:   Thu Jan 18 11:47:52 2024 -0800

    Add lec07.md code

commit 143841fc143b4aa3aa9588b32c520f9ea0668dc1
Author: Alex McKinney <alexmckinney01@gmail.com>
Date:   Wed Jan 17 09:02:31 2024 -0800

    Add exercise links
```



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Source: XKCD

Add and commit a file

The first time we ask a file to be tracked, and every time *before* we commit a file, we must ***add*** it to the staging area:

```
git add path/to/file.txt
```

- Takes a snapshot of these files, adds them to the staging area so it will be part of the next commit.
- Example: `git add hello.c goodbye.c`
  - Adds / stages all of the files in the current directory : `git add .`

To move staged changes into the local repo, we commit:

```
git commit -m "<message>"
```

Viewing changes

To view status of files in working directory and staging area:

`git status` or `git status -s` (short version)

- Lists the files which you have changed but not yet committed
- Indicates how many commits have made but not yet pushed

To see what is modified but unstaged: `git diff`

To see a list of staged changes: `git diff --cached`

To see a log of all changes in your local repo: `git log` or `git log --oneline` (shorter version). Press q to exit.

GitLab

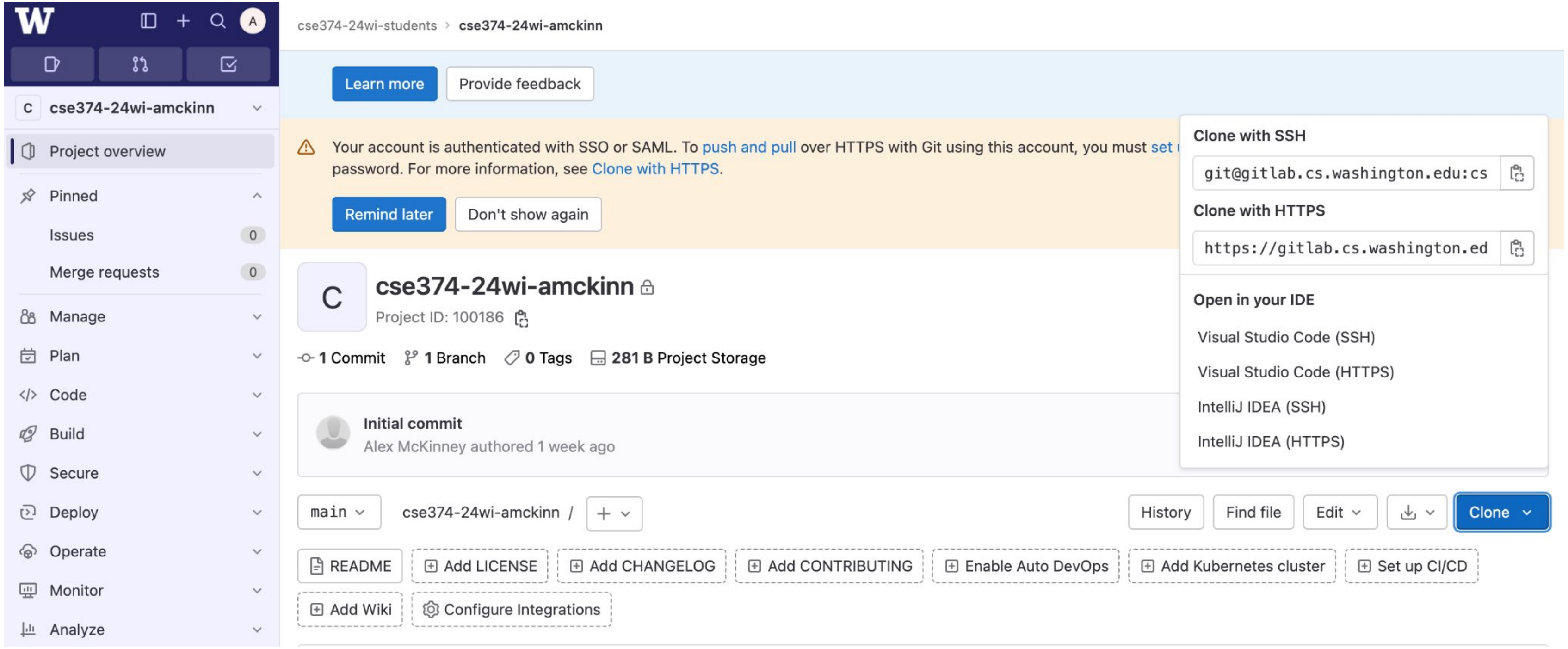
CSE Gitlab

- Github and Gitlab are just websites that store git repos
- You can create a repo on the website and `git clone` to edit it on your computer (e.g. laptop, calgary, etc.)
- CSE has its own version of Gitlab where you will be given a repository
    - https://gitlab.cs.washington.edu/cse374-24wi-students
    - We'll use this to distribute and submit homework assignments

Getting Started

1. Create local SSH keys (`ssh-keygen`) and add to your GitLab account
   a. Instructions in **Ex6**
2. Clone a working copy of your repository to your machine
   a. cd where-you-want-to-put-it
   b. `git clone git@gitlab.cs.washington.edu:path/to/repo`
   c. The URL is available from GitLab page for your project
3. Make sure you use the ‘Clone with SSH’ URL!
4. If git asks for password, your SSH keys aren’t set up right – fix it

Cloning From Remote


cse374-24wi-students > cse374-24wi-amckinn

Learn more | Provide feedback

Your account is authenticated with SSO or SAML. To push and pull over HTTPS with Git using this account, you must set
password. For more information, see Clone with HTTPS.

Remind later | Don't show again

cse374-24wi-amckinn

Project ID: 100186

1 Commit · 1 Branch · 0 Tags · 281 B Project Storage

Initial commit
Alex McKinney authored 1 week ago

Clone with SSH

```
git@gitlab.cs.washington.edu:cs
```

Clone with HTTPS

```
https://gitlab.cs.washington.ed
```

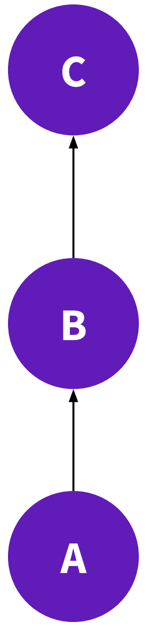
Open in your IDE
- Visual Studio Code (SSH)
- Visual Studio Code (HTTPS)
- IntelliJ IDEA (SSH)
- IntelliJ IDEA (HTTPS)

https://gitlab.cs.washington.edu/cse374-24wi-students/cse374-24wi-[uw_net_id]

Collaborating

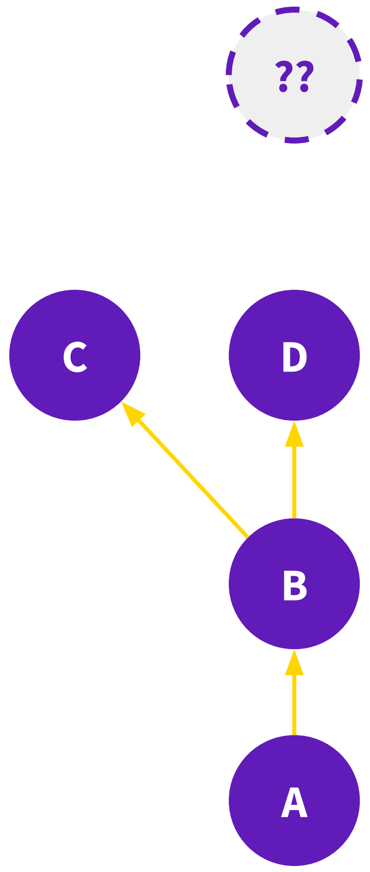
Collaboration: Ideal

- A "linear" history
    - **Alice** makes a commit and **pushes**
    - **Bob pulls**, makes a change, commits the change, and **pushes**
    - **Alice pulls**, makes a change, commits, and **pushes**
    - ...etc.



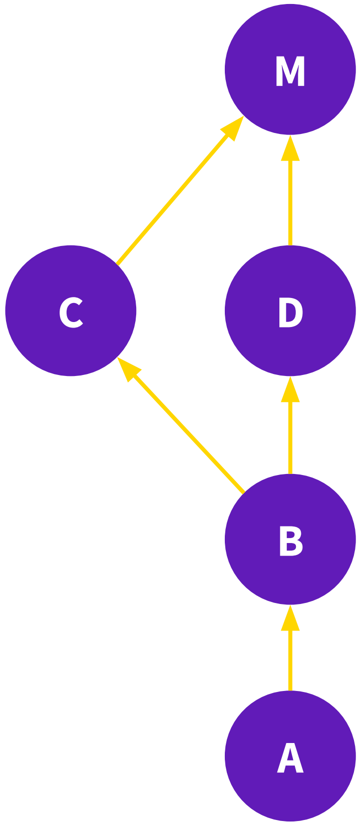
Collaboration: Reality

- We said the "commit history" is a list of commits, so what happens here?
  - **Charlie** makes a change and creates commit C, but **doesn't push**
  - **Diane** also makes a change and commit D, and **pushes**
  - **Charlie pulls** from the remote repo
  - It's no longer a list! The history has diverged
- Does Charlie just have to delete, pull and start over?



Merging

- A merge commit is a commit which has two "parents"
  - Combines the changes in each
  - Commit "M" includes all of Diane's changes, *plus* all of Charlie's



How do we merge?

`git pull`

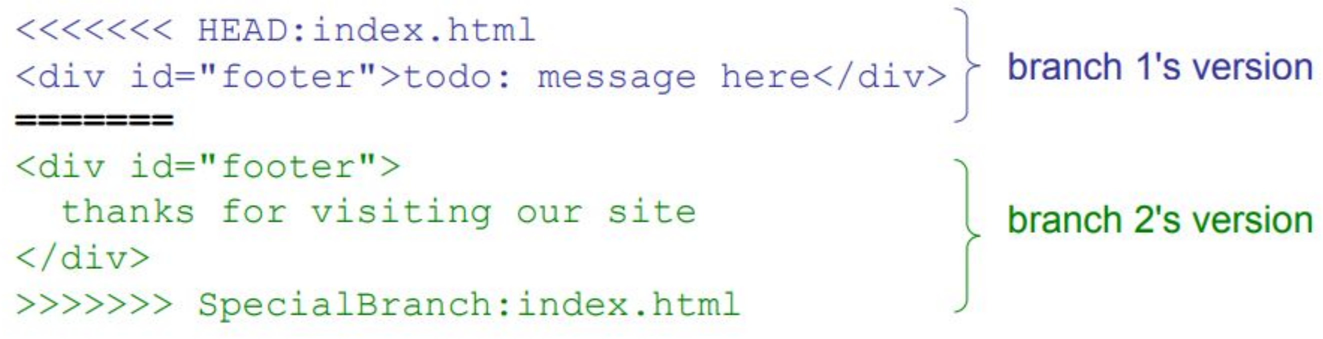
- Automatically fetches the changes and merges them into yours
- Then, `git push`
  - This push your local changes to the remote repo
  - Others can now work off of your combined changes

Sometimes, the changes you make will ***conflict*** with the changes others make

- e.g. you both edit the same line
- Resolving merge conflicts is more complicated; we will teach the basics here but it takes a lot of practice - come to OH or post on Ed!

Merge conflicts

Git will tell you which files had merge conflicts (use `git status` to see conflicts), and the files will be edited to identify the conflict:



```
<<<<<<< HEAD:index.html
<div id="footer">todo: message here</div>
=======
<div id="footer">
  thanks for visiting our site
</div>
>>>>>>> SpecialBranch:index.html
```

Find all such sections, and edit them to the proper state (whichever of the two versions is newer / better / more correct). You must modify the section to contain the code you want, then save, ***add***, and ***commit*** the merge.

.gitignore

Git may be used to store any types of files.

However…

Do not store files that are unnecessary.

- Backup files (like *.swp vim files)
- Files that can be recreated (such as .o files) should not be added.
- System specific files

‘.gitignore’ lists files not to upload to HEAD. Below is a sample .gitignore file content:

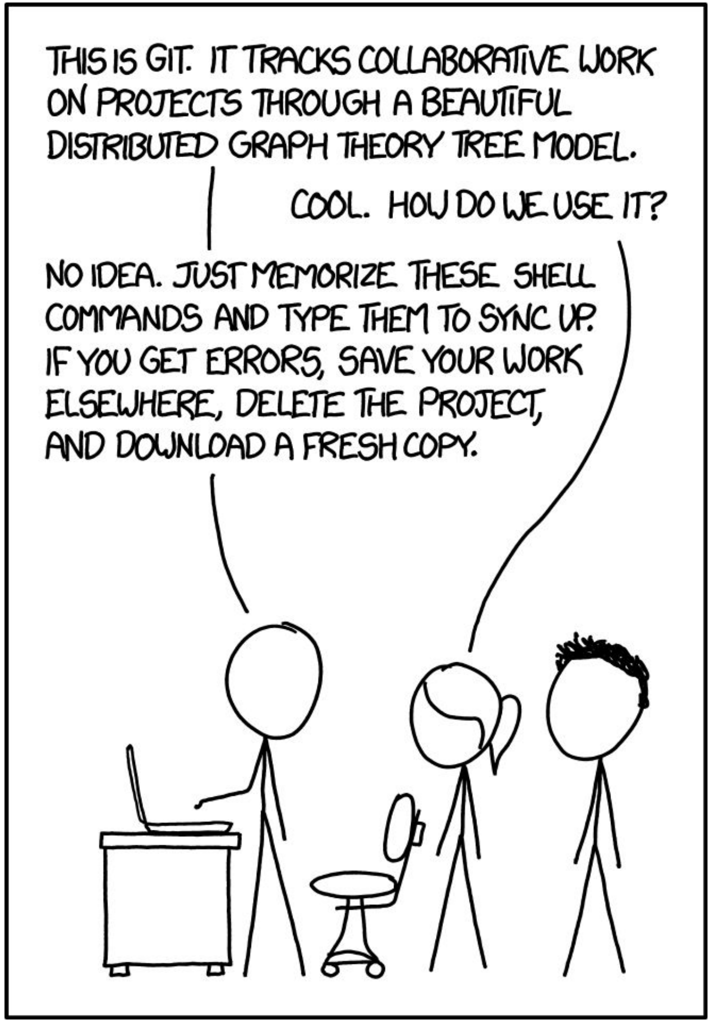
```
# Ignore vim temporary files
*.swp

# Ignore OS X finder info
files
.DS_Store

# Ignore built object files
*.o
```

Questions?

How do I fix git?!

You will inevitably run into frustrating situations with git

- Even for experienced users, sometimes you may accidentally get your git repo into an undesirable situation
- There is always a way to fix this, although it's not always obvious how
- Online resources are helpful
  - https://ohshitgit.com/


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 MEMORIZE 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.

Learning more

- Lots of interesting and useful topics, including:
  - Branching, checkout
  - Resolving merge conflicts, merge tools
  - "Merge requests" (a.k.a. "Pull requests")
  - Rebase
- The web is your friend!
  - Official documentation
  - "Git Book": https://git-scm.com/book/en/v2

Bonus Slides 🎁

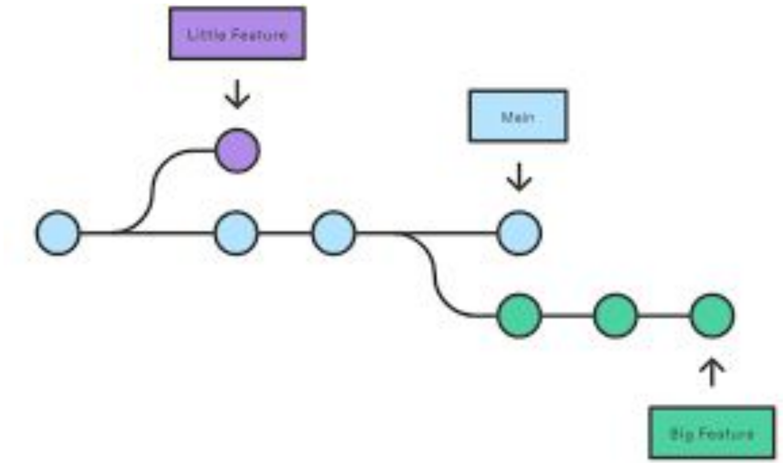
Branch

Git supports branches, which are used to split out a line of work.

- A branch is composed of one or more commits.
- A repository contains one or more branches.

The main branch is the primary source of truth!

- Default when a repository is created.



repository > branch > commit

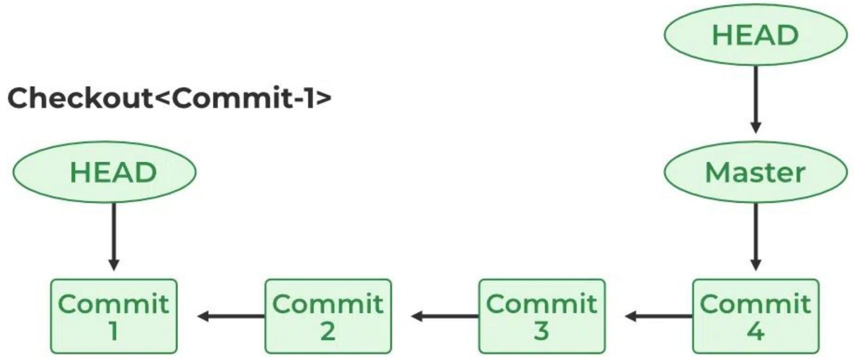
HEAD Commit

You will often see references to HEAD, which is the **latest commit**.

- Acts like a bookmark
- Every branch has its own HEAD

Display the content of the latest commit:

```
git show HEAD
```



Link