MONTANA
STATE UNIVERSITY

Version Control

Learning Git

What is Version Control

“Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.”

In the Beginning

- The Stone Age
  - cp and diff
  - emailing diff patches around
- Early Version Control Systems - Centralized
  - cvs - Concurrent Versioning System
  - subversion
  - perforce - commercial
- One source of truth, difficult to work “offline”

Today

- Distributed Version Control Systems (DVCS)
  - No locking!
  - No central repository!
  - Autonomy!
- git and mercurial (hg)
- git won
  - I don't know why

Today

- About that "No central repository!" Wellllllll…

- Github has become the central master repository:

  http://github.com

What Is a Github?

- Github is a site that hosts git repositories
  - It provides a nice web interface for git repos
  - Many web-based tools
    - Automated testing
    - Deploying
    - A whole eco-system...

What is a Github?

- Then Microsoft bought them :(
- There are alternatives!
  - Gitlab: https://about.gitlab.com/
  - My fav: https://sr.ht/
  - We aren't going to use them :(
- Github is the standard & has become important for a job

What Is a Github?

- Github is, ironically, very popular with open source
  - But not Richard Stallman!
- Github allows you to “fork” repositories easily
- You can contribute changes back via "pull requests"
- Contributing to Open Source projects is much, much easier

What Is a Github?

- “Forks” and "Pull requests" are a Github thing, **not** a git thing

- But we'll get to all that later. For right now, just make sure you have a github account set up: https://github.com/

Git

- Git is a command line tool for managing changes to source code
- Developed by Linus Torivalds, of Linux Fame
  - Linux Kernel was using a proprietary source control system for Linux, BitKeeper
  - BitKeeper guy got mad, pulled the free license
  - Linus declared he would begin working on git on April 3rd, 2005
  - First version shipped April 6th
  - Self-hosted April 7th
  - Managed the Linux 2.6.12 release on June 16th
  - Linus is a *really* good programmer...

Git Concepts

- Repository (repo) - A collection of code with a URL end point
  - e.g. https://github.com/bigskysoftware/htmx.git or git@github.com:bigskysoftware/htmx.git
- Clone - A clone (copy) of another repo (the “upstream” repo)
- Remote - A remote copy of a repo (e.g. the “upstream” repo)
- Branch - An independent copy within a given repo
  - A repo can have multiple branches, all with different content
- Merging & Rebasing - Ways to get changes from one repo into another
- It’s a lot, I know.

Git Intro - clone

- How to I get code from a remote repository onto my machine?
- The clone command - clones a repository to your local machine
- You can now make changes to the code locally

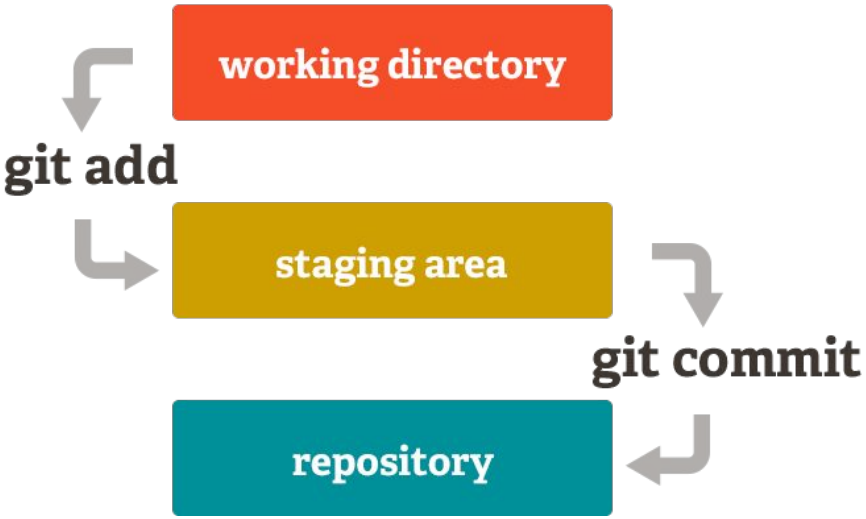
```
carson@BIGSKY-THINKPAD: ~
$ git clone https://github.com/bigskysoftware
/htmx.git
Cloning into 'htmx'...
remote: Enumerating objects: 3498, done.
remote: Total 3498 (delta 0), reused 0 (delta
 0), pack-reused 3498
Receiving objects: 100% (3498/3498), 2.18 MiB
 | 5.38 MiB/s, done.
Resolving deltas: 100% (2260/2260), done.
$ cd htmx/
$
```

More Git Concepts

- Working Directory - The current state of code on your machine
- Staging Area - Changes you have “added” to be committed to the repository
- Repository - What the repository knows about

Git Intro - add

- How do I get changes into my local repository?
- You “stage” them with “git add”
- You “commit” them with “git commit”



Git Intro - add

- Edit a file EXAMPLE.txt
- Run “git status”, note that it is “untracked”
- Use “git add” to stage your modification to EXAMPLE.txt
  - Again, stage means "ready to commit"
  - “git add” can also add new files, not just modified files

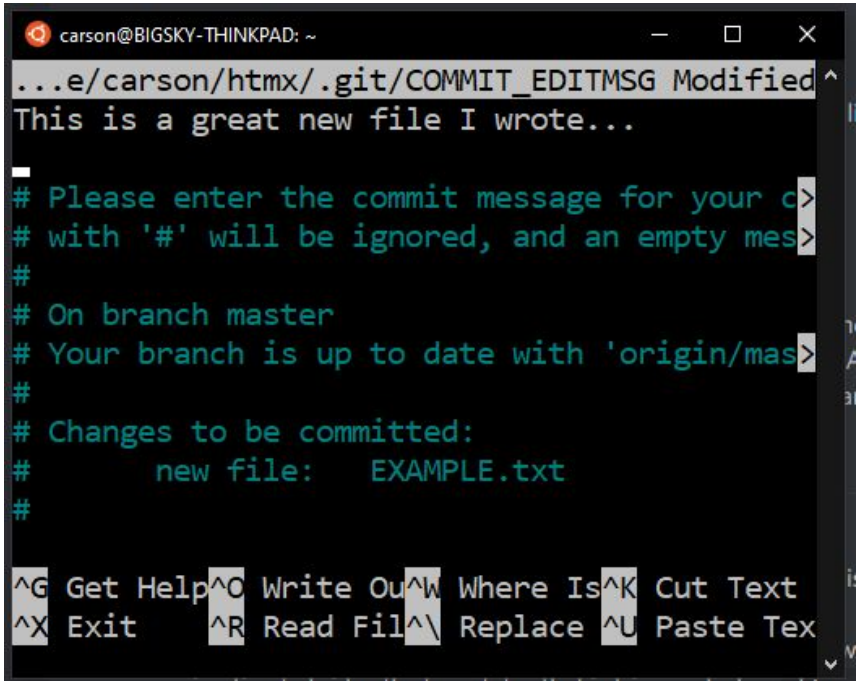
```
carson@BIGSKY-THINKPAD: ~
$ vi EXAMPLE.txt
$ git status
On branch master
Your branch is up to date with 'origin/master
'.

Untracked files:
  (use "git add <file>..." to include in what
 will be committed)
        EXAMPLE.txt

nothing added to commit but untracked files p
resent (use "git add" to track)
$ git add EXAMPLE.txt
$
```

Git Intro - commit

- OK, let's commit our change
- We are committing it to the *local* repository
  - We are NOT sending it up to the server we cloned from

Git Intro - commit

- What's this?
  - Remember pico?
- You need to create a "commit message"
- By default git will use the system editor, which is good old pico in our case
  - You can override this
- Write a message and save...



Git Intro - commit

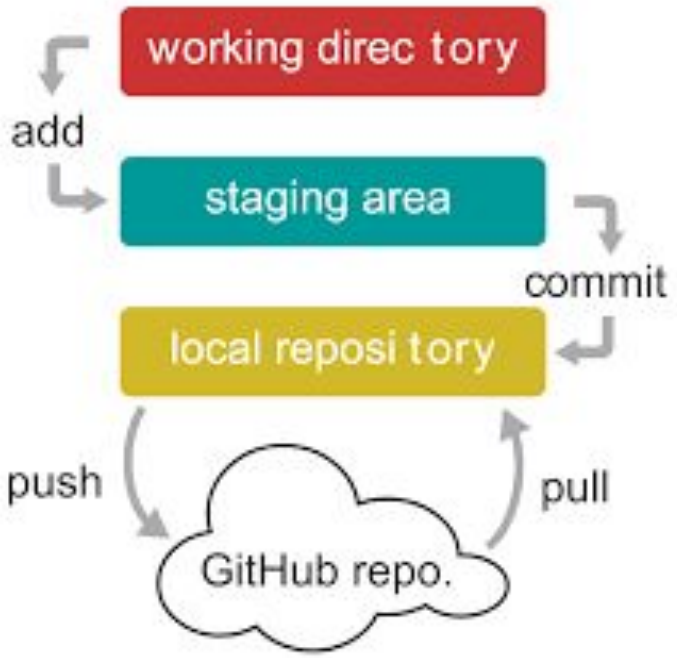
- And its saved!
- git now knows about your new file
- If you wanted, you could roll back to a version before your file was created.

```
carson@BIGSKY-THINKPAD: ~
$ git commit
hint: Waiting for your editor to close the fi
[master d23579b] This is a great new file I w
rote...
 1 file changed, 2 insertions(+)
 create mode 100644 EXAMPLE.txt
$
```

Git Intro - push

- OK, so how do I get changes into the remote repository?

- You “push” them to the remote repository with “git push”



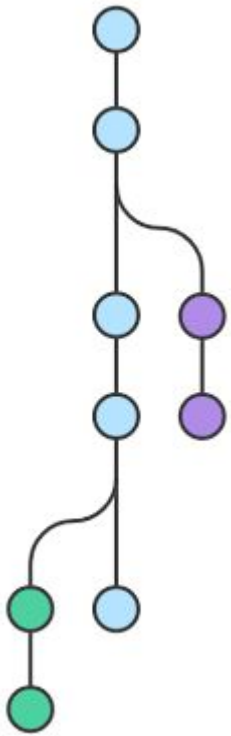
Git Intro - pull

- What about new changes *from* the remote repository

- You “pull” them from the remote repository with “git pull”

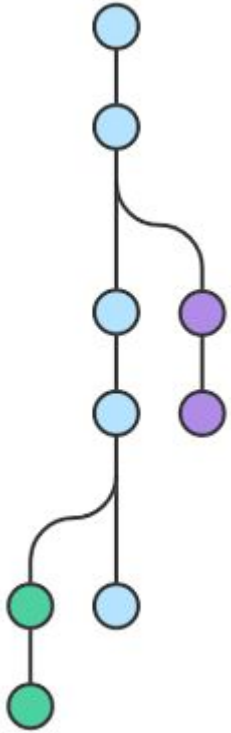
Git Intro - branch

- Think of a branch as a new local copy of the code
  - It can be edited alongside other branches
  - Changes in the branch won’t be visible in others until you merge
- Add a branch with:
  git branch <branch-name>



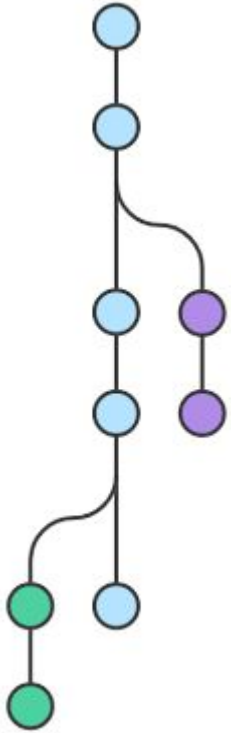
Git Intro - branch

- Git branches have lead to a large number of "git branch strategies"
  - Conventions for branches that facilitate development
- There is no standard
- You will have one “main” branch
  - Traditionally it is called the "master" branch



Git Intro - branch

- You often have one or more "development" branches
- you might also have "feature" branches
- You might have "release" branches
- It can get pretty crazy
- We will keep it simple for this class: you will have one branch

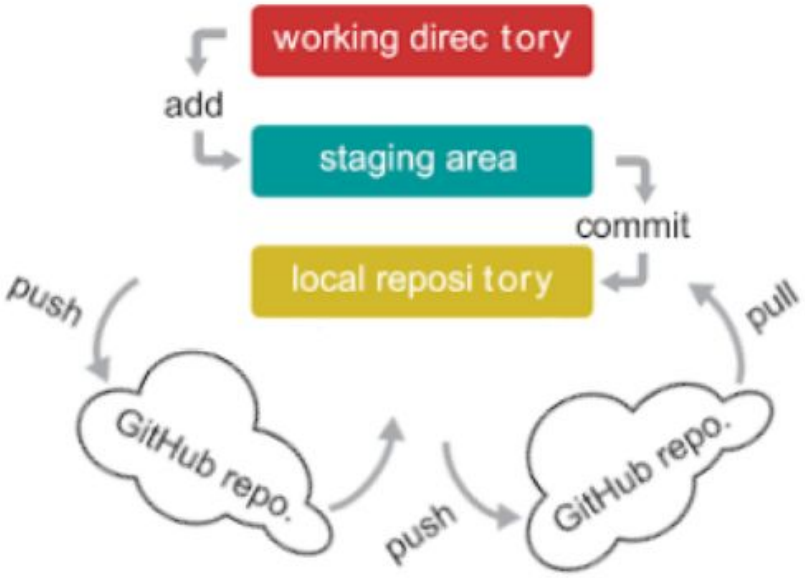


Git Intro - remote

- What if I want to pull from another remote repository?
  - For example, from the class repository? ;)

- Add the remote with:
  git remote add <name> <url>

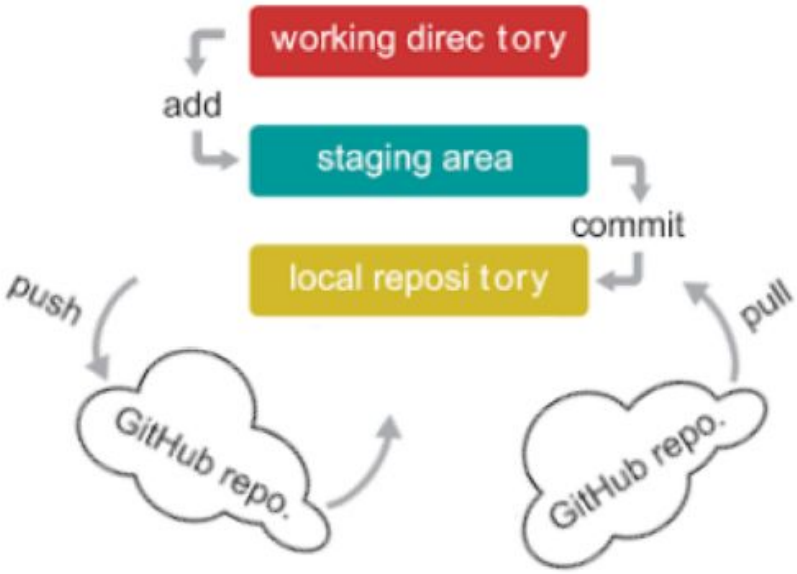
- Now you can pull from it with:
  git pull <name> <branch>



Git Intro - remote

- With remotes, you are usually only pushing to one

- The other remotes are “upstream” repos you only pull from



Other Git Commands

- git diff - compare objects
- git revert - undo changes but save as a new commit
- git cherry-pick - merge with specific commits
- git tag - add a tag to a commit
- git rebase - change base (HEAD^)
- git reset - undo changes to a previous commit

Whew!

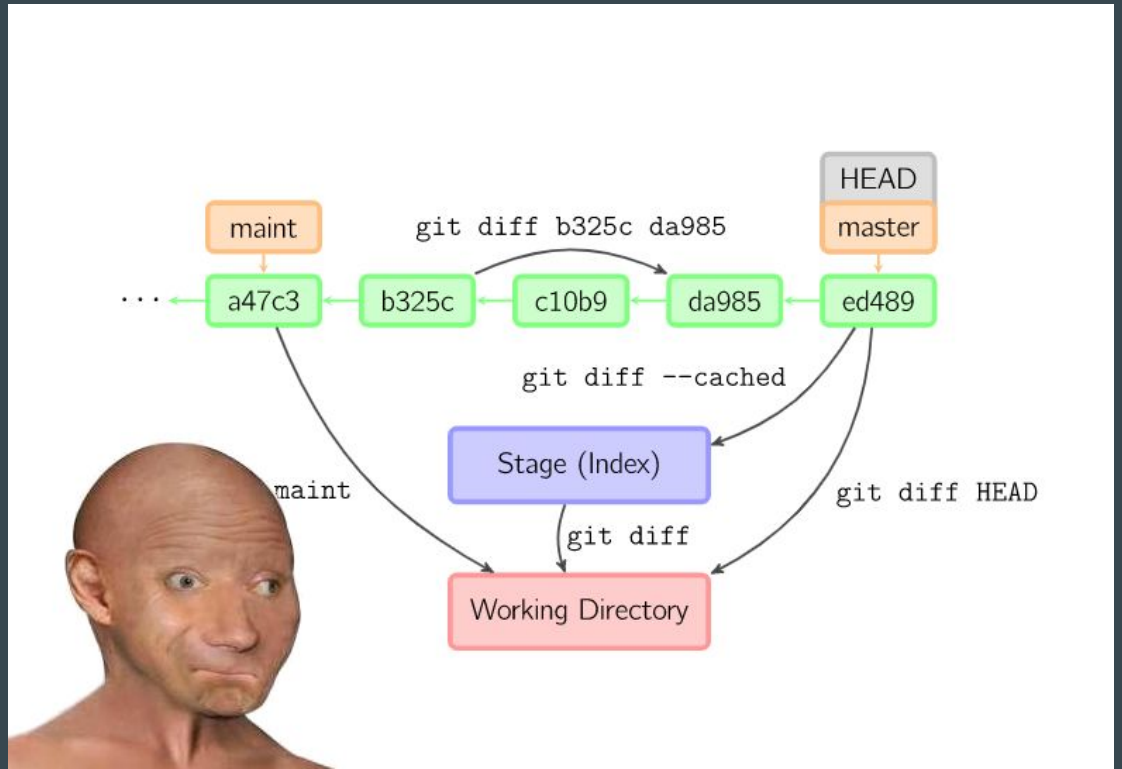
- Yes, it’s a lot
- Don’t worry, we will be available to help you get all set up
- The course will be set up so that you can avoid most of this
  - One upstream repo
  - One branch per student (your email as the branch name)
  - All you have to do is pull from the upstream master branch, and push to your fork
    - We will talk about forks next time
  - It won’t be that bad, I promise

Some Git Resources

- Git Cheat Sheet:
  https://github.github.com/training-kit/downloads/github-git-cheat-sheet

- Git - The Simple Guide:
  https://rogerdudler.github.io/git-guide/

Some Git Realtalk



Some Git Realtalk

- I find git painful to deal with on the command line, even after years of using it
- I usually only use the command line for simple stuff
- For complicated things like merges, JetBrains has a great UI that makes things much easier

IntelliJ Demo

•••

Avoiding Learning Git

MONTANA
STATE UNIVERSITY