Announcements!

Attendance Raffle!

- Sign in at meetings
- Get entered for a chance to win:
  - Kindle Paperwhite
  - Amazon Echo Dot
  - ...and more!

signin.umlacm.org

Be an ACM member, get sweatshirt

- Membership to the national ACM (not our student chapter!) is $20/year
  - You get access to journals and textbooks
  - And probably some other academic stuff
- This is **not** necessary to be a member of the club!
- Show us you're a member and get one of our UML ACM sweatshirts!

Lightning Talks!

- November 13 (Wednesday, 5:30 pm)
- Talk about something cool for 5-10 minutes!
- Some Ideas:
  - Side projects or research
  - Internship/Co-op
  - Intro To [Software/Tool/Language]
  - Why emacs is better than vim
- Interested? Contact us!

Come Visit! DAN 302

- We have an office!
- Come stop by to:
  - Chat
  - Get help with homework
  - Learn how to navigate the CS curriculum at UML
- If the door’s open, stop by and say hi!

Career Fair Tomorrow!

- 4-7 pm, Tsongas Center
- Meet at the office (DAN 302) by 3:15 to head over together!
- Get a job!

Introduction to Git

By Amy Mazzucotelli

(Slides stolen from Joshua Hassler)

Version Control

- A way to keep track of code
  - Decent ways
    - Perforce
    - SVN
    - Git
  - Bad Ways
    - Dropbox
    - Copy of copy of copy of...

Why Use VC

- By yourself
  - Allows you to back up your work
  - Good for versioning and deployment
- Work with others
  - Concurrent developers without problems
  - Controls merging
  - Central place to get all code
- ALL COMPANIES USE SOME VC

What is Git?

- Git is a version control
  - Created by Linus Torvalds while developing Linux kernel
- Distributed Architecture
  - Each copy is it’s own Repository
- Git operates locally
- Takes snapshots called “commits”

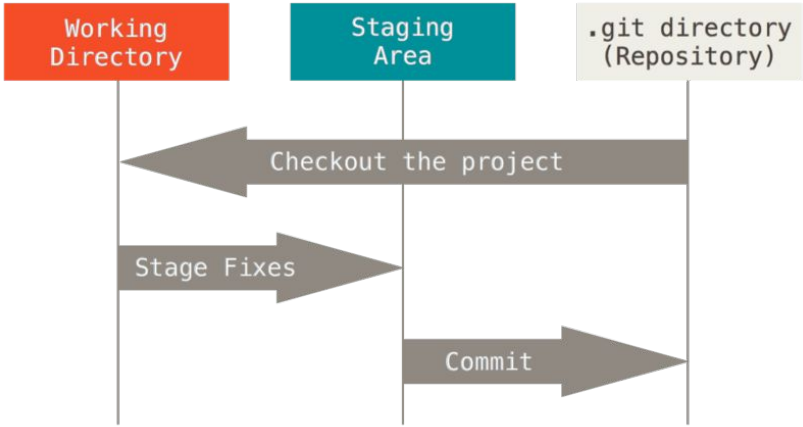
Lets Git some of the vocab

- **Commit** - Snapshot of work at a point in time
- **Repository (Repo)** - Container that stores the codebase, history, and metadata
- **Remote** - A copy of a repo stored on a server

What are remotes?

The Three Stages of Git

- **Modified** - The file has been modified
- **Staging** - I'm gonna commit it
- **Committed** - I committed it!



Demo Time
Setting Up a Repo and Committing

Demo Notes:

- Make a new repo on GitHub (browser)
  - It will show you how to get set up
- Locally: make a new folder, run “git init” from inside it
- Git remote add origin <repository>
- Git config user.email “your email”
- Git config user.name “your name”
- First commit: set upstream (origin=remote, master= your local branch)
  - Git push -u origin master
- Commit, pull, push

Some Reminders on Commits

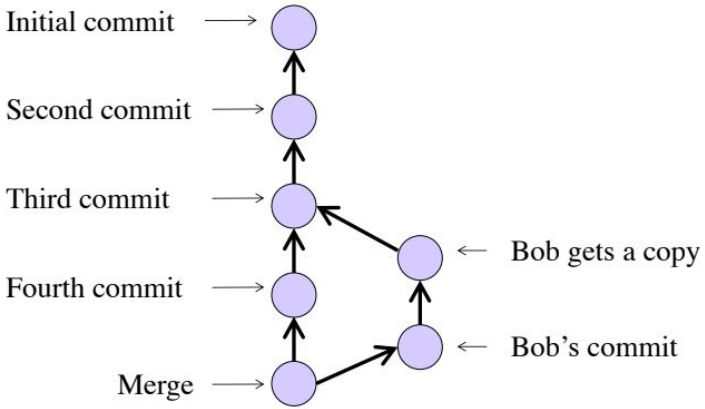
- Commits are cheap
  - Do them often
- Commit messages should be short and descriptive
  - Keep them in the present tense
    - Bad message: “Updated the code to do the thing”
    - Good message: “Update foo to handle negative values preventing crashes”
  - Code should show how, *messages tell why*
  - You will thank yourself in the future

Branching

- **Master** - The “main” branch and source of truth
- **Branch** - a copy of the code

What are Branches

- A split in the code
  - Multiple concurrent copies
- Split and go back
  - After working in your branch, you merge it back
  - Done through a **Pull Request**



Demo Time
Branching/Merging

Demo Notes

- Git branch <branchName> (makes the branch)
- Git checkout <branchName> (switches you to the branch)
- To merge your branch back into master:
  - Git checkout master
  - Git merge <branchName>
- Merge conflicts: easier with a UI!
  - <<<HEAD
  - (what you just tried to commit)
  - =======
  - (what someone else changed it to)
  - >>>>>commithash

Extras

- Use SSH instead of HTTP
  - Ssh-keygen
  - Copy the public (.pub) file to your GitHub Settings->SSH and GPG Keys
  - Checkout the repo with the git@github url
  - Never have to enter your password again!
- .gitignore
  - *.o (ignore any file ending with .o)
  - a.out (ignore a specific file)
  - temp/** (ignore everything in the temp folder, including subdirectories)