



Version Control



Show and Tell



- Take turns sharing what you built, tried, or discovered in the homework assignment.
- As you listen, ask questions, react, and share ideas. If you're not sure what to say, try:
 - ***I like...*** What stood out to you? What worked well or was interesting?
 - ***I wish...*** What could be improved, changed, or tried differently?
 - ***I wonder...*** What are you curious about? What do you want to know more about?
- You can also ask “how did you do that?” and “wait, *it can do that?*”

Outline for Today

- ***What is version control?***
 - A powerful tool for backing up work, sharing work, collaborating with others, etc.
- ***Using git***
 - Commits and branches.
- ***Using GitHub***
 - Pushing / pulling.

What is Version Control?

Vibecoding is Nonlinear

- Coding agents sometimes **one-shot**: they deliver exactly what you want on the first try.
- That's not the norm.
 - They create something that works, but which breaks something else.
 - They make something that does exactly what you asked for, and then you realize you didn't actually want that.
- Expect vibecoding to be a nonlinear process.
 - "Let me try something out and see how it goes."
 - "Oh, wait, I have to undo what the LLM just did."
 - "Yikes, a week ago we broke something and we just found out about it now, and we need to *unbreak* it now."
 - "I *think* I might want to keep this, but I'm not sure, and in the meantime I want to try something else out."

How do we try out experimental
changes on the software we're
vibecoding?

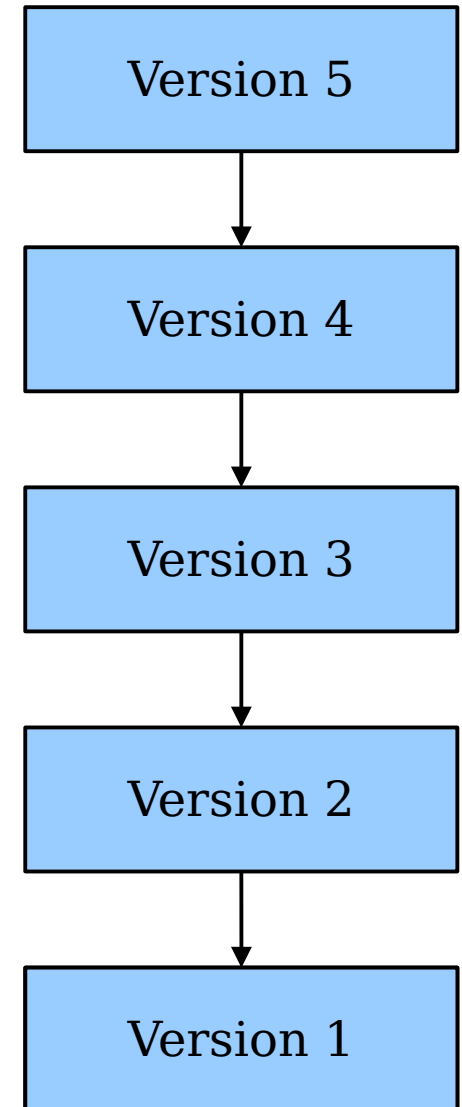
Introducing git

git

- git is a tool that provides **version control**.
- It lets you track different **versions** of files.
 - You have a “current” version of the files representing the version you like.
 - You can have other versions where you run experiments that you may or may not end up using.
 - You can jump backwards in time to see what things looked like earlier.
 - You can integrate the results of experiments.
- And you can do a *ton* of other things with git as well. Hit me up after class for details.

What Does git Do?

- At a basic level, git lets you take “snapshots” of your work that are saved for later.
- Those snapshots are called **commits**; the term comes from databases.
- Commits form an ordered history of what your files look like.
 - As in: first it looked like this, then it looked like this, etc.
- You can then switch back and forth between different commits.
- Those commits live in a **repository** stored in the folder.



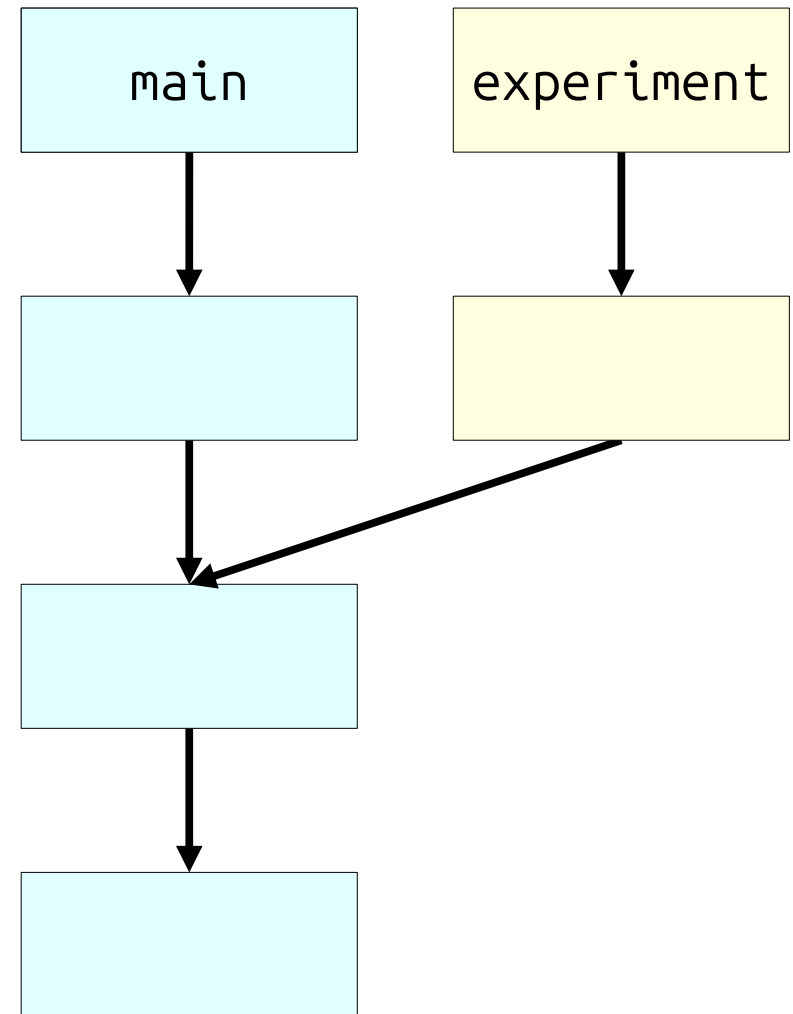
To Recap

- **git init** tells git to initialize a *repository* (snapshot database) in the current folder.
- Use **git status** to see what's changed since the last commit (snapshot).
- When ready to commit, do the following:
 - **git add** all the files to commit, then
 - **git commit** them to take the snapshot.
- Use `git log` to see the commit history and the *hashes* (IDs) of the commits.
- Use **git switch --detach hash** to see what previous versions looked like.
- Use **git switch main** to jump to the most recent version.

Branches

Branches

- By default, everything in git forms a tidy, linearly-ordered history.
- Sometimes we want to try an experiment to see what happens.
- We can **branch** from the main history. Work we do on a branch stays separate from the main branch.



To Recap

- Use the command **git switch -c *name*** to create a new branch with the given name.
- You can then commit as usual on the branch.
- If you want to accept the changes you made on the branch, ask your coding agent to “merge via rebase.”
 - Want to know what those magic words mean? Hit me up after class.
- This is great for running experiments.

Time-Out for Announcements!

Assignments

- Assignment 2 goes out today and is due on Thursday.
 - Build something artistic!
 - And learn git in the process!
- We're excited to see what you come up with! Come prepared for more Show and Tell. 😊

Tutorial Sessions

- Tutorial sessions start this week.
- We will email you shortly after class with your tutorial session time and location.
 - We do *not* use Axess enrollments for this; we've tried to make things as fair as possible.
- If you can't make your time slot, post on EdStem seeing if someone wants to swap slots with you.

Check Your Email

- Check your email later this evening for a time-sensitive surprise.
- That's all I'm going to say. 😊

Back to CS193V!

GitHub

Sharing Your Work

- Right now, all the files I've showed you are on my computer.
- What if...
 - ... I want to back up my work elsewhere?
 - ... I want you to see what I made?
 - ... I want you to add code to my project?
- For that, we need our repository to live somewhere other than my laptop.

GitHub

- **GitHub** is a commercial service that hosts git repositories in the cloud.
- Important note:

 **git ≠ GitHub** 

- git is the version control tool.
- GitHub is a site that hosts git repositories and provides collaboration and project management tools.

Cloning a Repository

- To get your own local copy of a repository on GitHub, use this syntax:
`git clone https://github.com/repo-name`
- This grabs all commits and branches from GitHub. You'll begin on the `main` branch.
- Recognize this command from anywhere else in this course?

Refreshers

- Use **git switch *branch*** to switch to a named branch.
 - You'll see many names of the form `origin/branch`. To switch to a branch of that form, leave the `origin/` bit off. (e.g. `git switch tetris` rather than `git switch origin/tetris`)
- Use **git switch --detach *hash*** to visit a particular commit.
- Play around. What do you find?

To Recap

- To get a copy (“clone”) of a repository on GitHub, use the **git clone** command.
- To send commits you’ve made locally back to GitHub, use **git push**.
- To receive commits made since you last did a clone, use **git pull**.

Next Time

- ***Software Testing***
 - How do we make sure vibecoded software works as intended?
- ***Automated Tests***
 - Letting agents close the loop.
- ***Reporting Bugs***
 - Helping coding agents fix their issues.