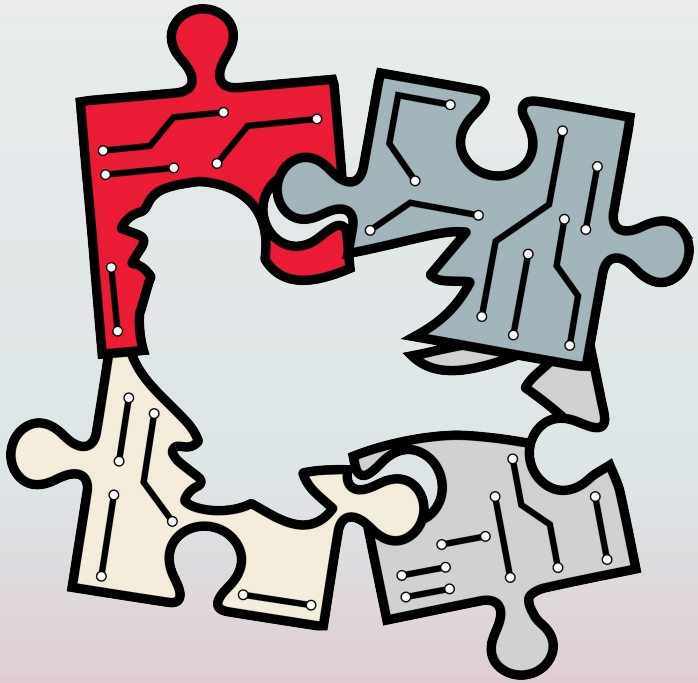
MCGILL PHYSICS

HACKATHON

Introduction to

git

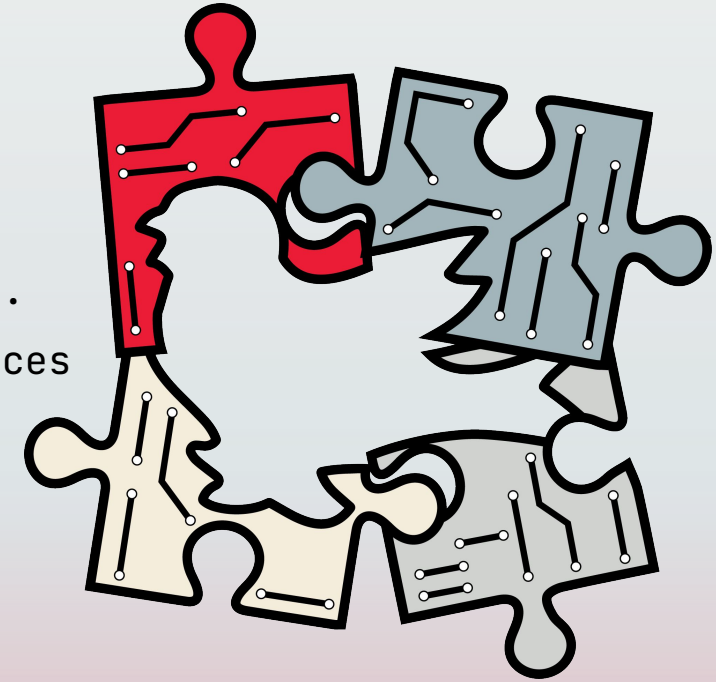


Simon Lavoie

February 12th, 2026

What is git?

- Created by Linus Torvalds to work on the Linux kernel.
- Free and open source.
- Designed to track changes in code.
  - Changes made by you.
  - Changes made by others.
- Supports branches.
- Allows you to *check out* your previous code.
- Allows you to *push* your code to remote places (e.g. github, gitlab, bitbucket, etc.)



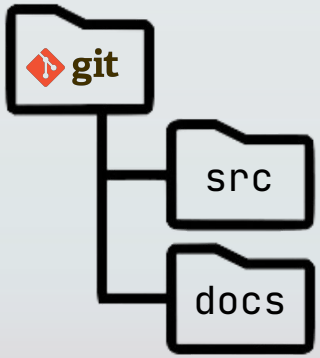
Gitting started

- Installed via your package manager or at **https://git-scm.com/install/**

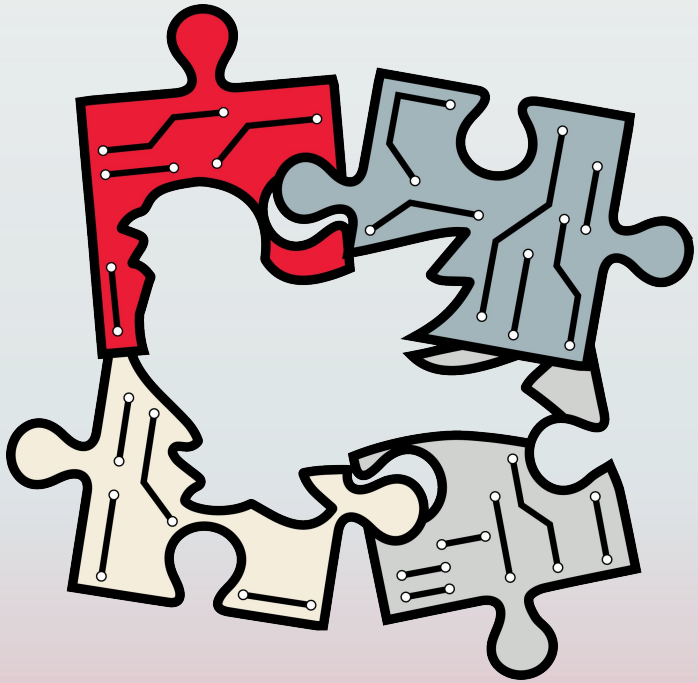
```
$ sudo apt install git
```

← if using WSL/Ubuntu

- Git tracks files at the root of your project directory.



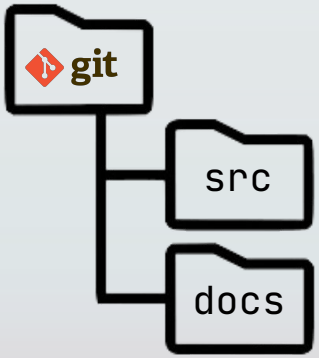
- Specifically, Git tracks *changes* between versions, i.e. it does not store copies of your files.



Continuing to gitting started

- Projects tracked by git are called repositories (repo)
- A git repo is created (in the CL) by navigating to your project root and running:

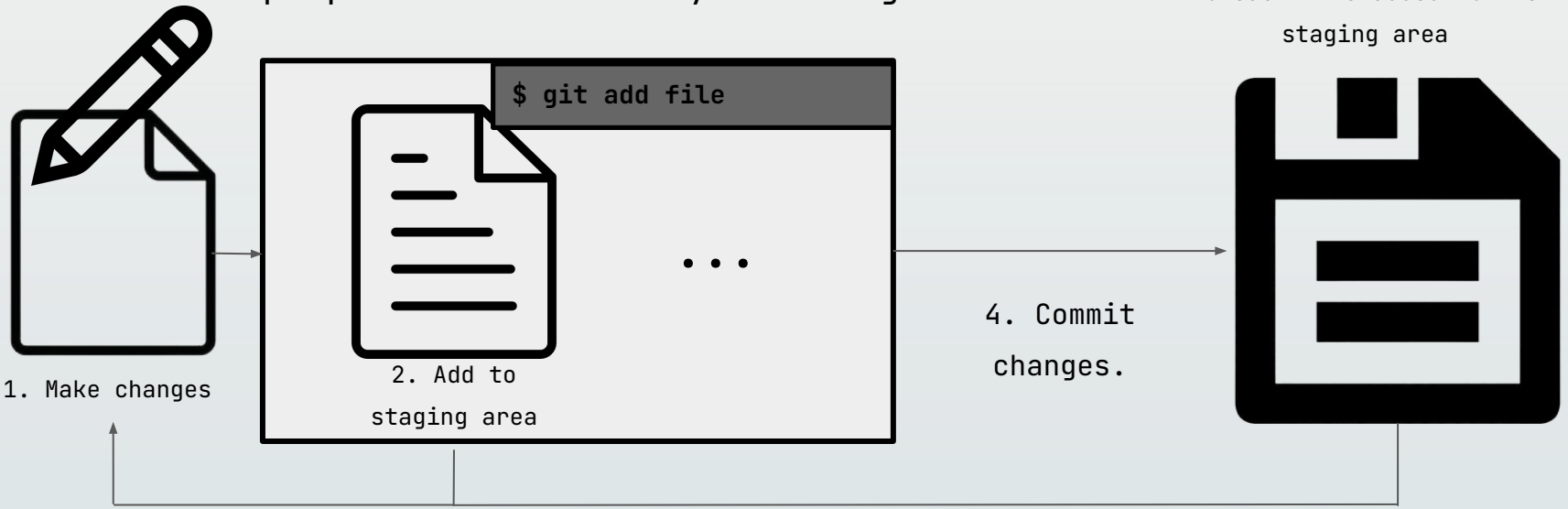
```
$ git init
```



- This creates a .git directory.
- Remember: “dotfiles” (i.e. files whose names start with a dot) are *hidden* files.

The staging area (working alone)

- Two-step* process to track your changes



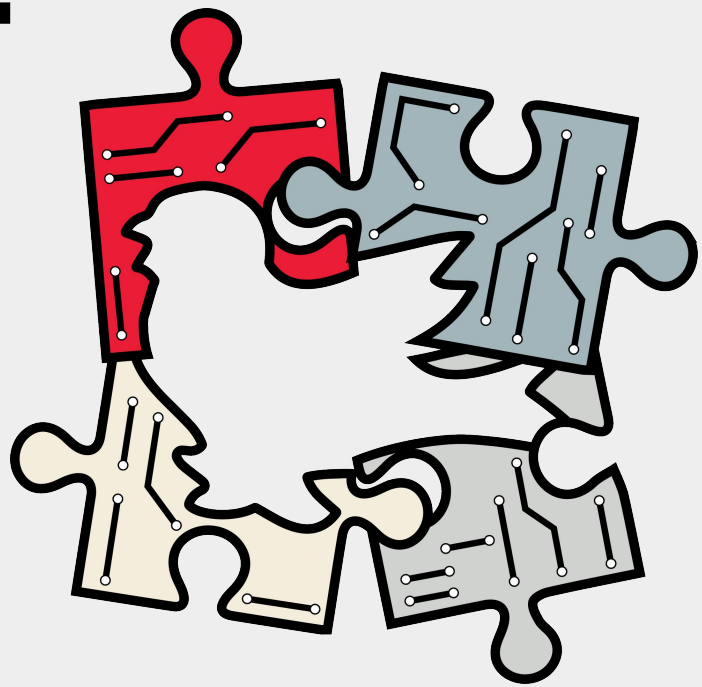
Tip: Your commit message is all you have access to when browsing past commits.

MCGILL PHYSICS HACKATHON

Simon Lavoie

EXAMPLE

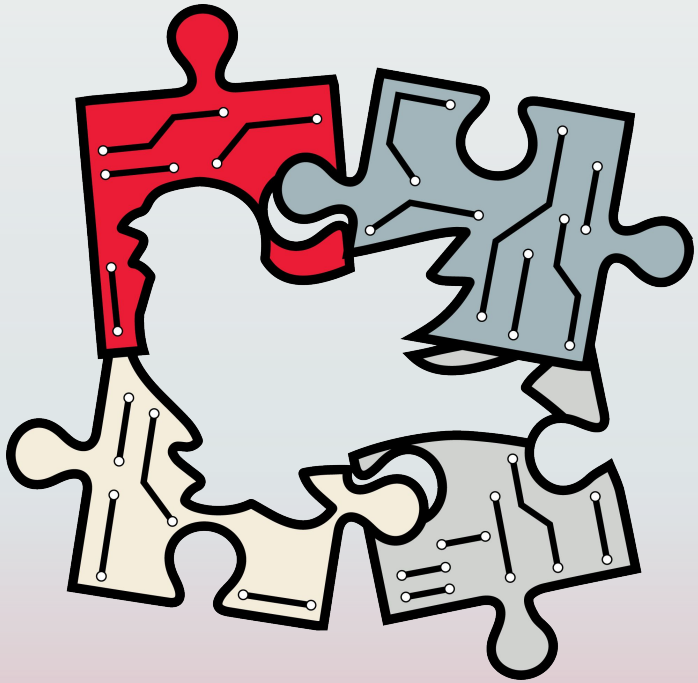
- ***Init***ializing a repo.
- Editing files.
- ***Add***ing to staging area.
- ***Commit***ting.
- Going back on commits with ***checkout***.
- Checking ***log***s.
- Checking ***status***.



Branches

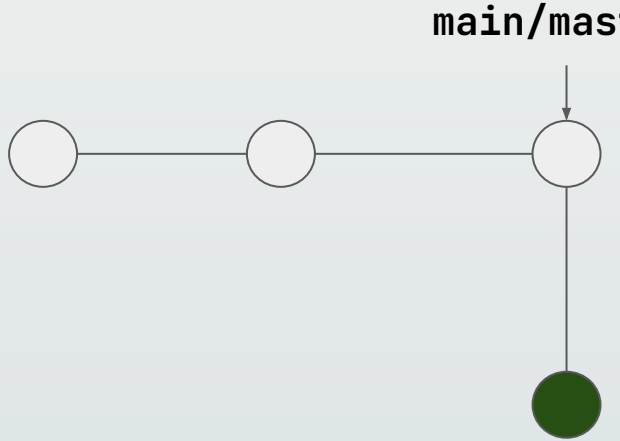
- **You** have a ***crazy*** idea that you want to implement.
- This could ~~ruin~~ change **everything!**

main/master:

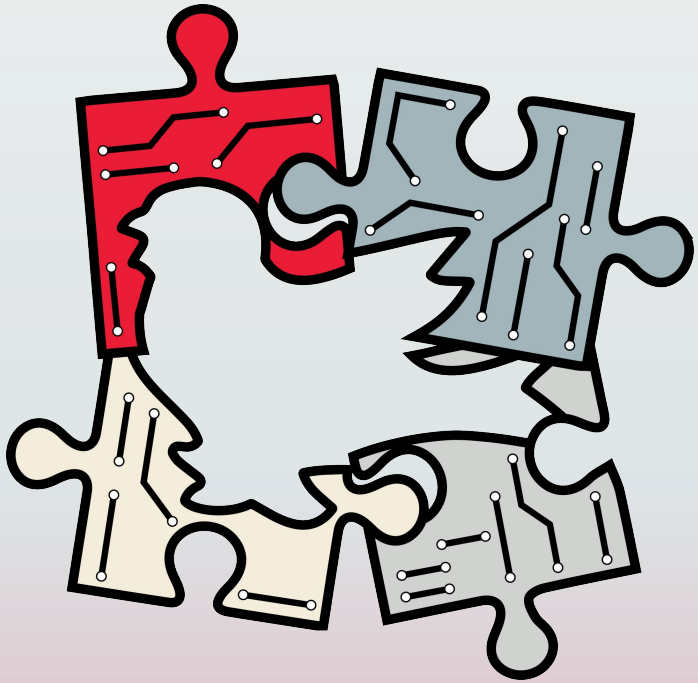


Branches

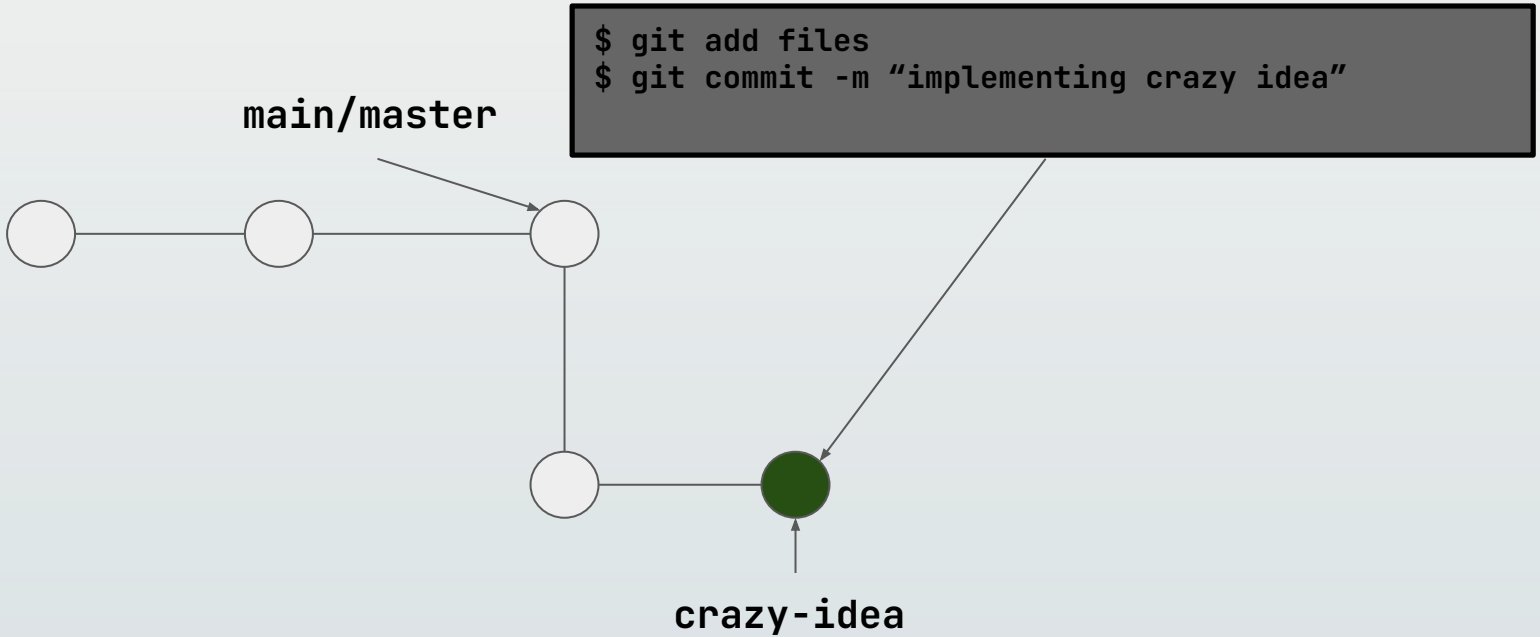
```
$ git checkout -b crazy-idea
```



crazy-idea:

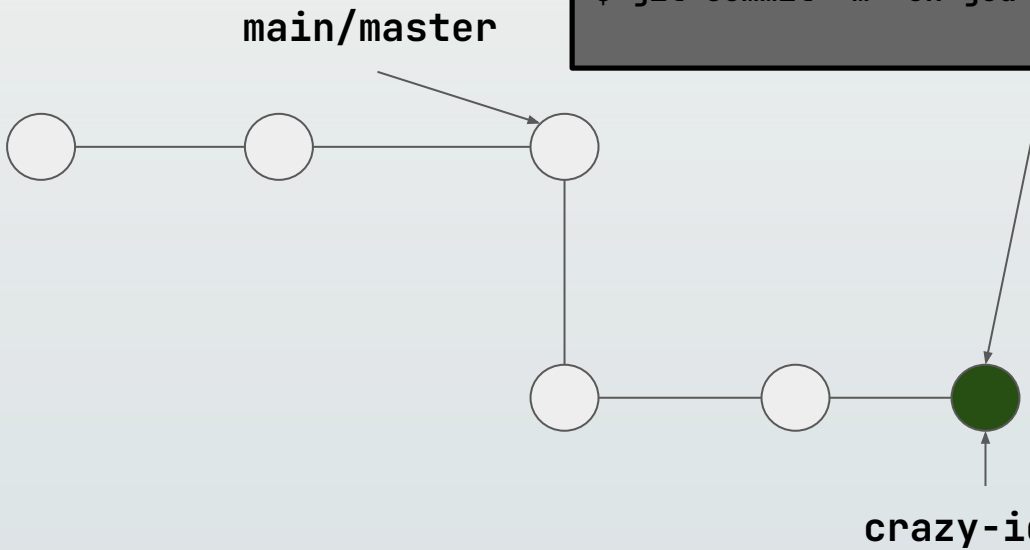


Branches


$ git add files
$ git commit -m “implementing crazy idea”
main/master
crazy-idea

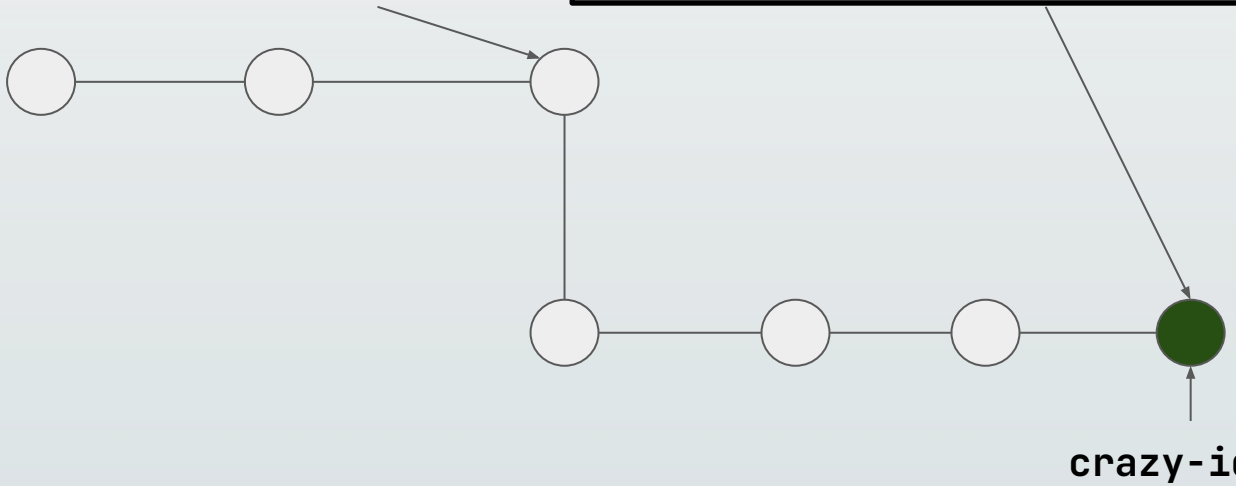
Branches

```
$ git add more-files
$ git commit -m “oh god I can’t stop”
```



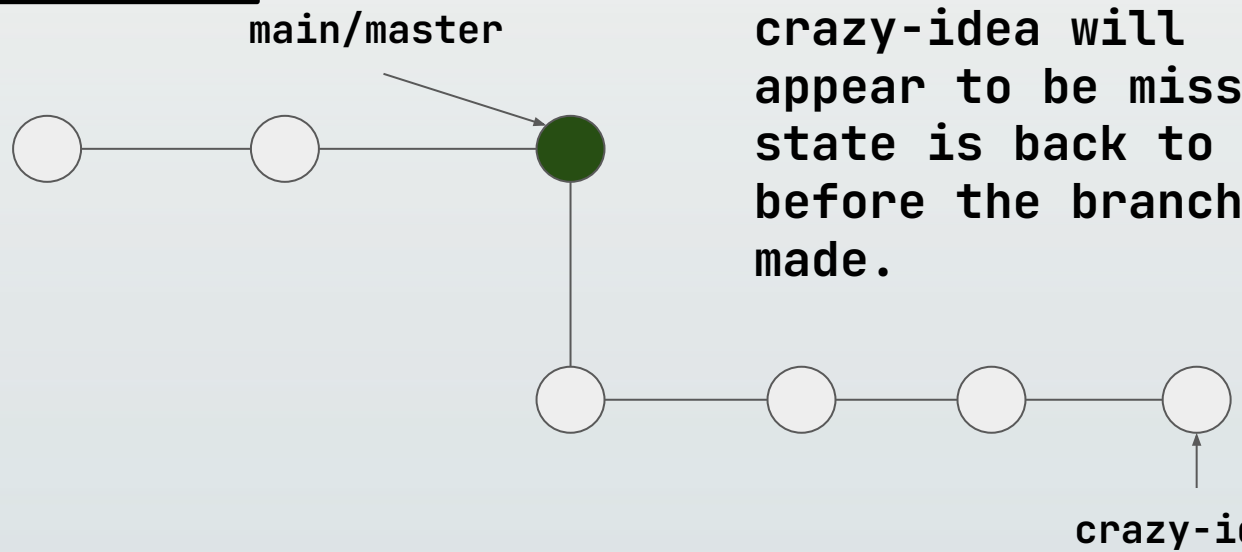
Branches

```
$ git add last-file
$ git commit -m “okay done. And it works!”
```



Branches

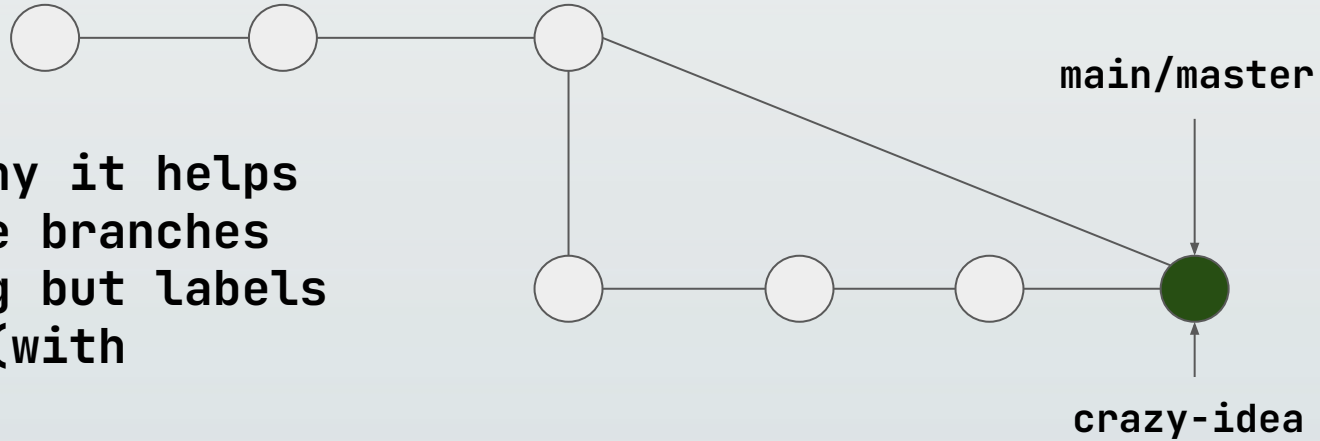
```
$ git checkout main/master
```



Any files created in crazy-idea will appear to be missing; state is back to before the branch was made.

Branches

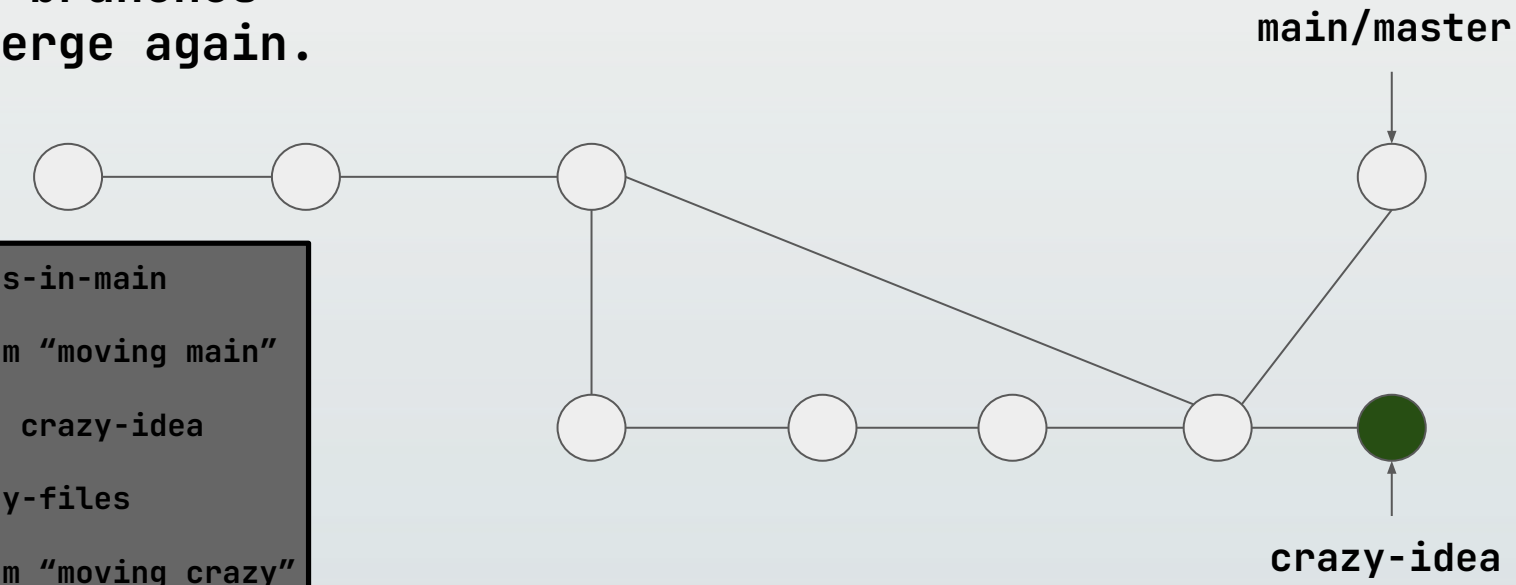
```
$ git merge crazy-idea
```



This is why it helps to imagine branches as nothing but labels to nodes (with history).

Branches

Final example showing how these branches could diverge again.




```
$ git add files-in-main

$ git commit -m “moving main”

$ git checkout crazy-idea

$ git add crazy-files

$ git commit -m “moving crazy”
```

EXAMPLE

- Making branches.
- Moving between them.
- Using git stash.
- Using git diff.
- Ignoring certain files.