

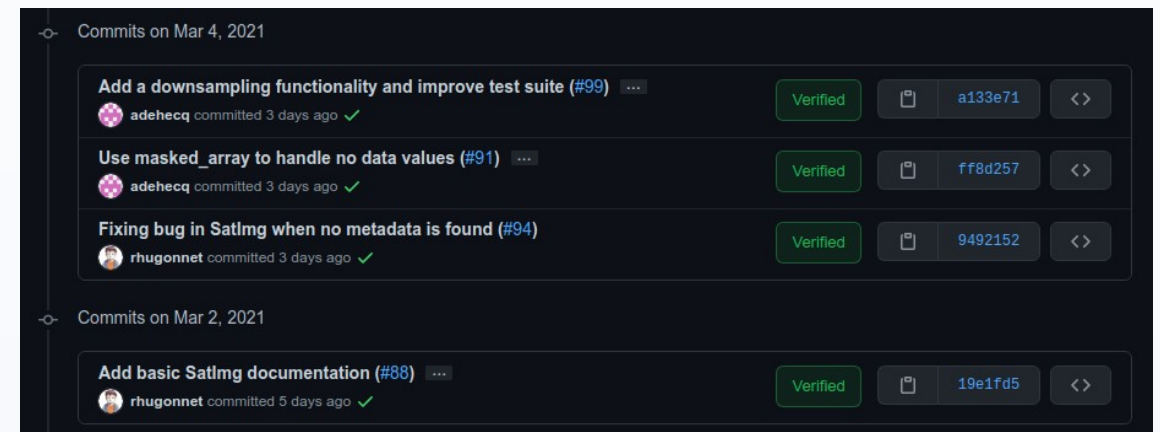
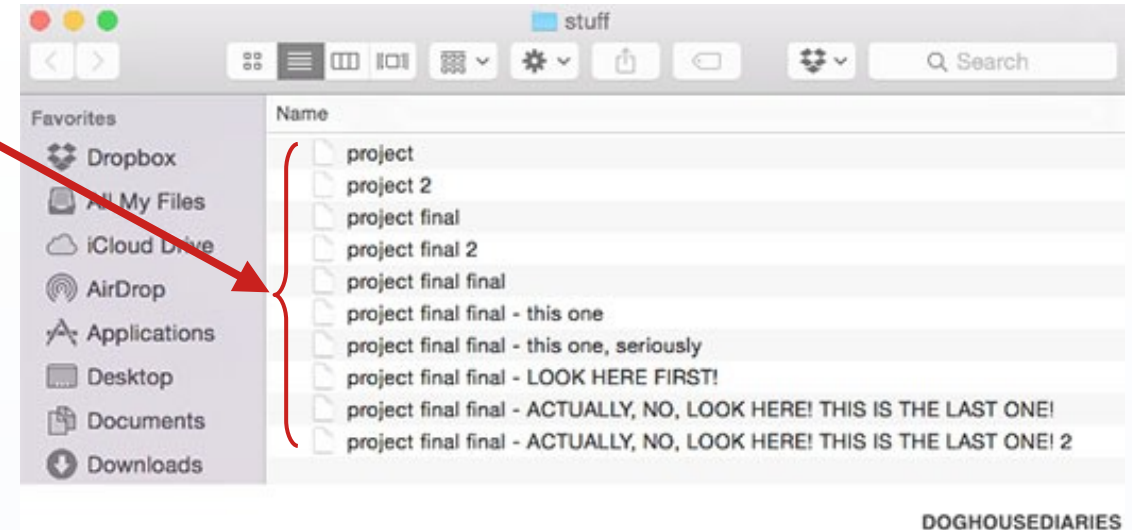
Programming skills for PhD Researchers

Session 1: A brief introduction to git

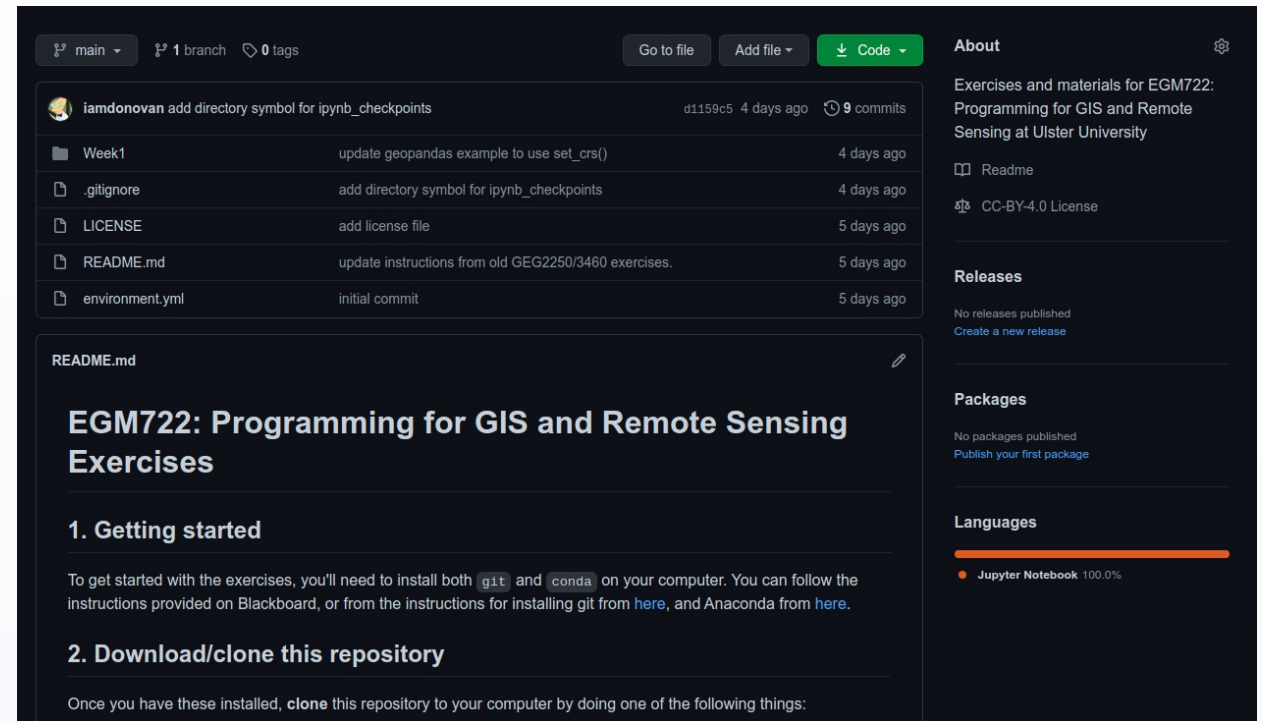
- Git is a **distributed version control system** (DVCS)
 - **Version Control System**: tool that records changes to file(s) over time
 - **Distributed**: each copy is independent & has complete history
- Started with software, but is not limited to software
- Enables you to:
 - See the entire timeline of your project
 - Keep track of changes made (and why!)
 - More easily collaborate with others

Why version control?

- Because this is nightmare fuel
 - With a version control system, we (hopefully) avoid this
- Version control:
 - Records snapshots of a project
 - Keep track of what/why changes are made
 - Can go back and undo changes if needed



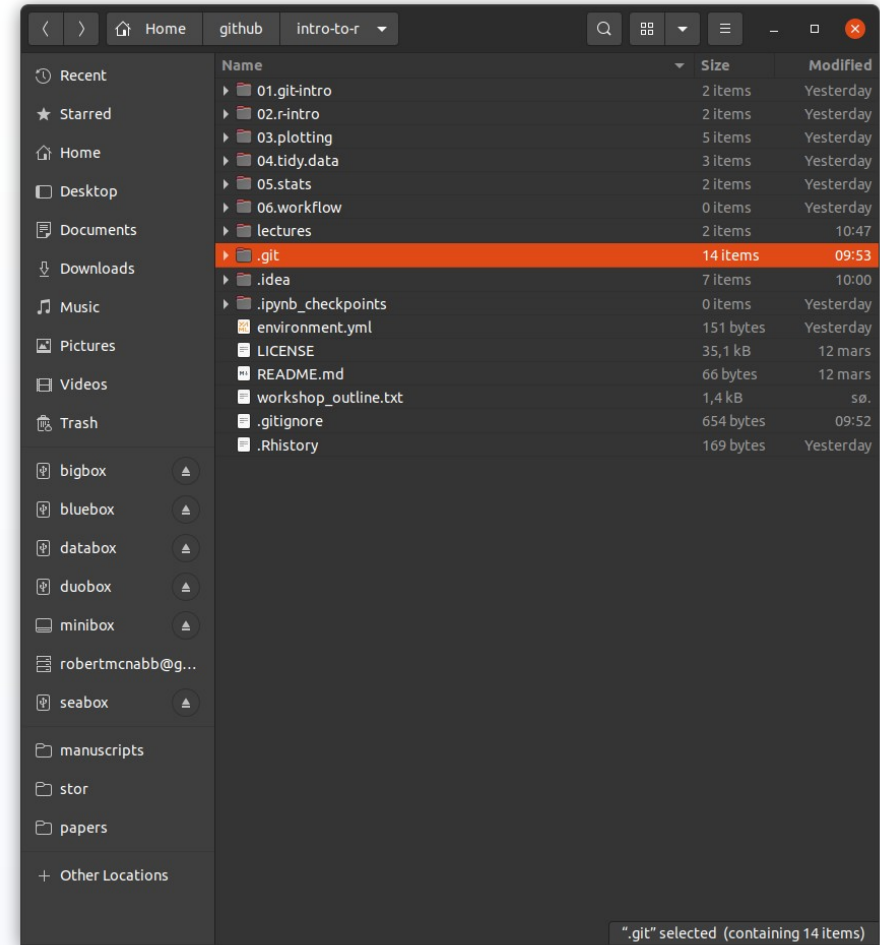
- A **repository** is the storage space for the project
 - Files
 - Past versions
 - All branches
- Any folder can be turned into a git repository:
 - `git init`



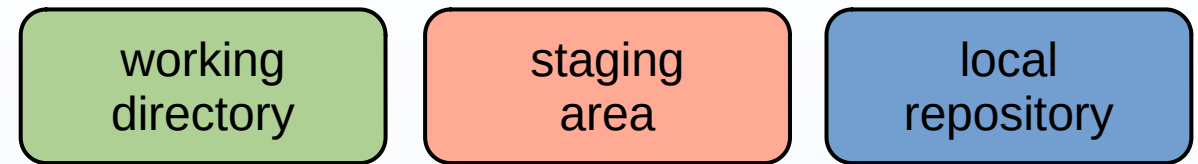
The screenshot shows a GitHub repository page for 'EGM722: Programming for GIS and Remote Sensing Exercises' by iamdonovan. The repository is on the 'main' branch and has 1 branch and 0 tags. The commit history shows a recent commit 'add directory symbol for ipynb_checkpoints' by iamdonovan 4 days ago. The file list includes 'Week1', '.gitignore', 'LICENSE', 'README.md', and 'environment.yml'. The README.md file is open, showing the title 'EGM722: Programming for GIS and Remote Sensing Exercises' and the first section '1. Getting started'. The right sidebar shows the repository's description, license (CC-BY-4.0), and no releases or packages published.

Any folder can be a git repository!

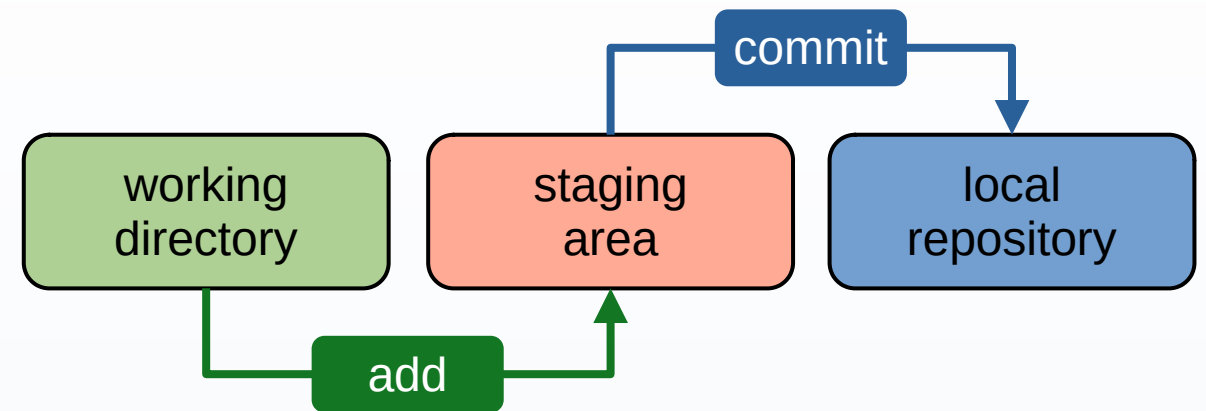
- `git init` creates a folder, `.git`, in the current directory
 - This is where the magic happens
 - Everything that git keeps track of is stored in here
 - Copying this folder gives you everything you need to re-create your project
- Handle with caution:
 - Changing/modifying files in here can break your project



- **local repository**: the `.git` folder
- **working directory**: the files that you work with
- **staging area**: an intermediate area where you can review changes before “saving” them



- **Commit:** a **snapshot** of the project
- Two-step process
 - Stage (`git add`)
 - Commit (`git commit`)
- Each commit has a unique identifier (**hash**)
- In general, make commits:
 - When you have finished something (e.g., a “feature”)
 - When you have changes you want to undo

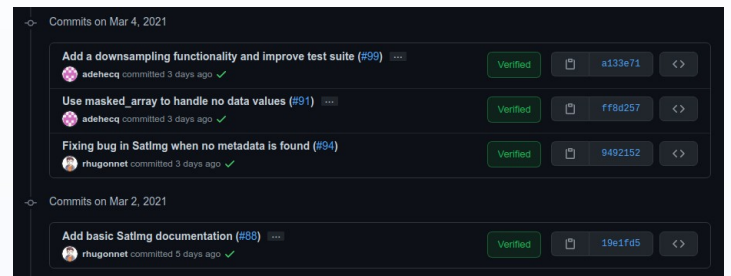


- Each commit should also have a **message** describing the commit
- Messages should:
 - Be short
 - Explain what was changed (title)
 - Explain why something was changed (body)
- Be specific!



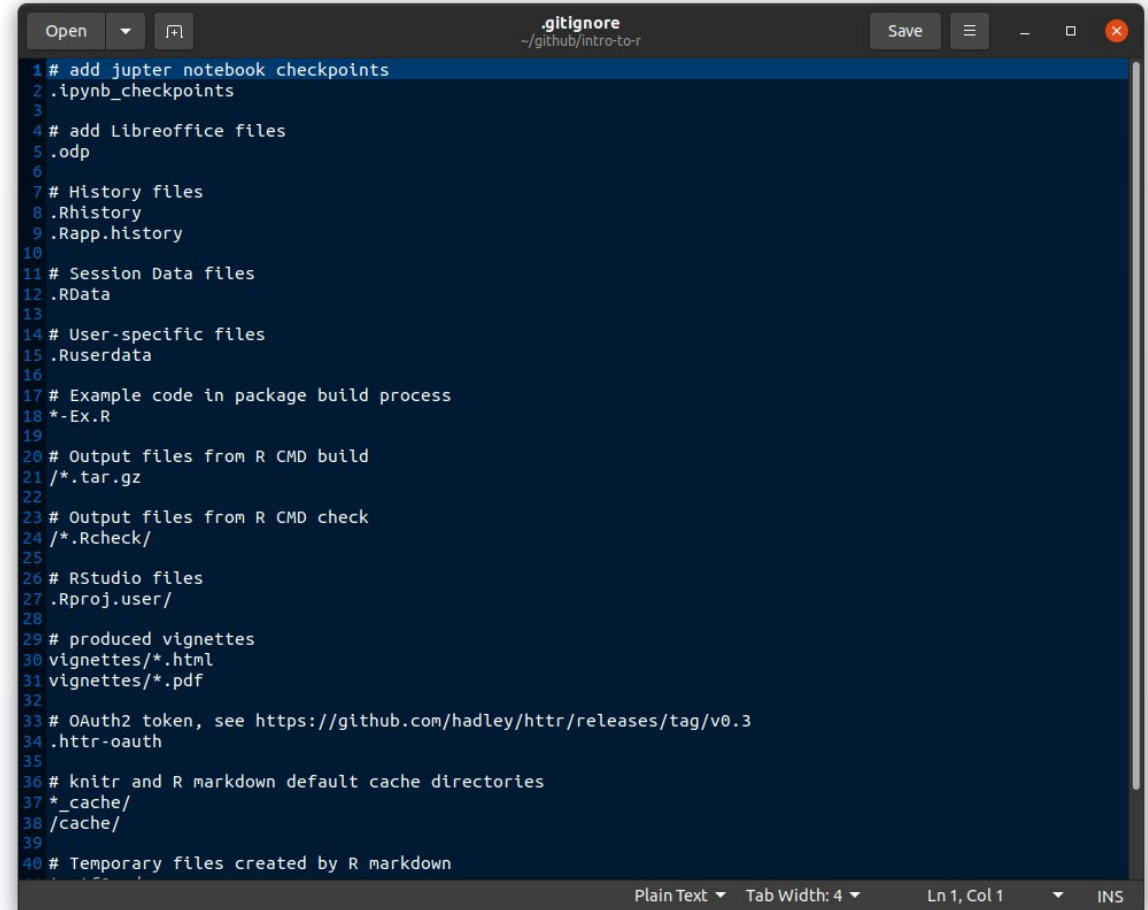
```
* 7d0fc3e typo
* 8fc509a more changes
* efe5fc5 add test
* 447c5a0 updates
* 1189cf0 update condition
* 68abca0 updates
* 29f73ed more changes
* cefaa18 add file
```

Jason McCreary



What do we want to keep track of?

- What files do we want to keep track of?
- .gitignore
 - Specific files/folders
 - Patterns
- Want to avoid:
 - Large (> 50MB) files
 - Automatically-generated files
 - User or OS-specific files
- <https://gitignore.io>: generate .gitignore files based on language, OS



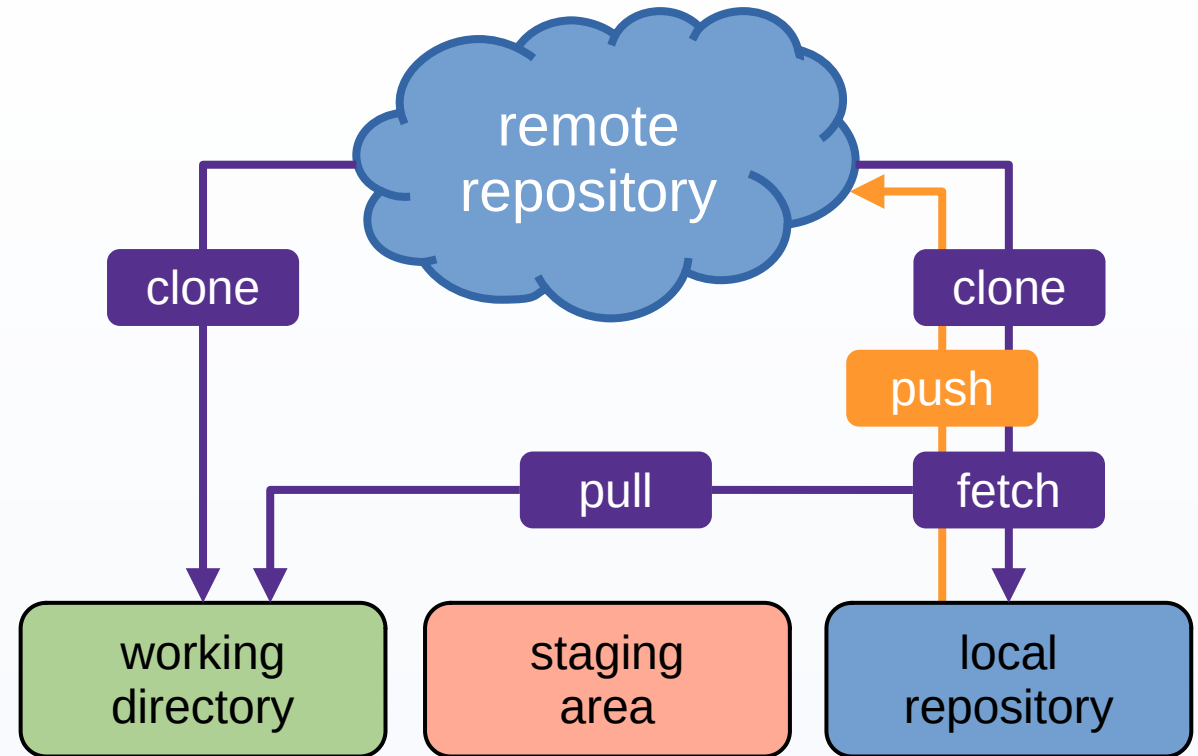
```

1 # add jupyter notebook checkpoints
2 .ipynb_checkpoints
3
4 # add Libreoffice files
5 .odp
6
7 # History files
8 .Rhistory
9 .Rapp.history
10
11 # Session Data files
12 .RData
13
14 # User-specific files
15 .Ruserdata
16
17 # Example code in package build process
18 *-Ex.R
19
20 # Output files from R CMD build
21 /*.tar.gz
22
23 # Output files from R CMD check
24 /*.Rcheck/
25
26 # RStudio files
27 .Rproj.user/
28
29 # produced vignettes
30 vignettes/*.html
31 vignettes/*.pdf
32
33 # OAuth2 token, see https://github.com/hadley/htr/releases/tag/v0.3
34 .htr-oauth
35
36 # knitr and R markdown default cache directories
37 *_cache/
38 /cache/
39
40 # Temporary files created by R markdown

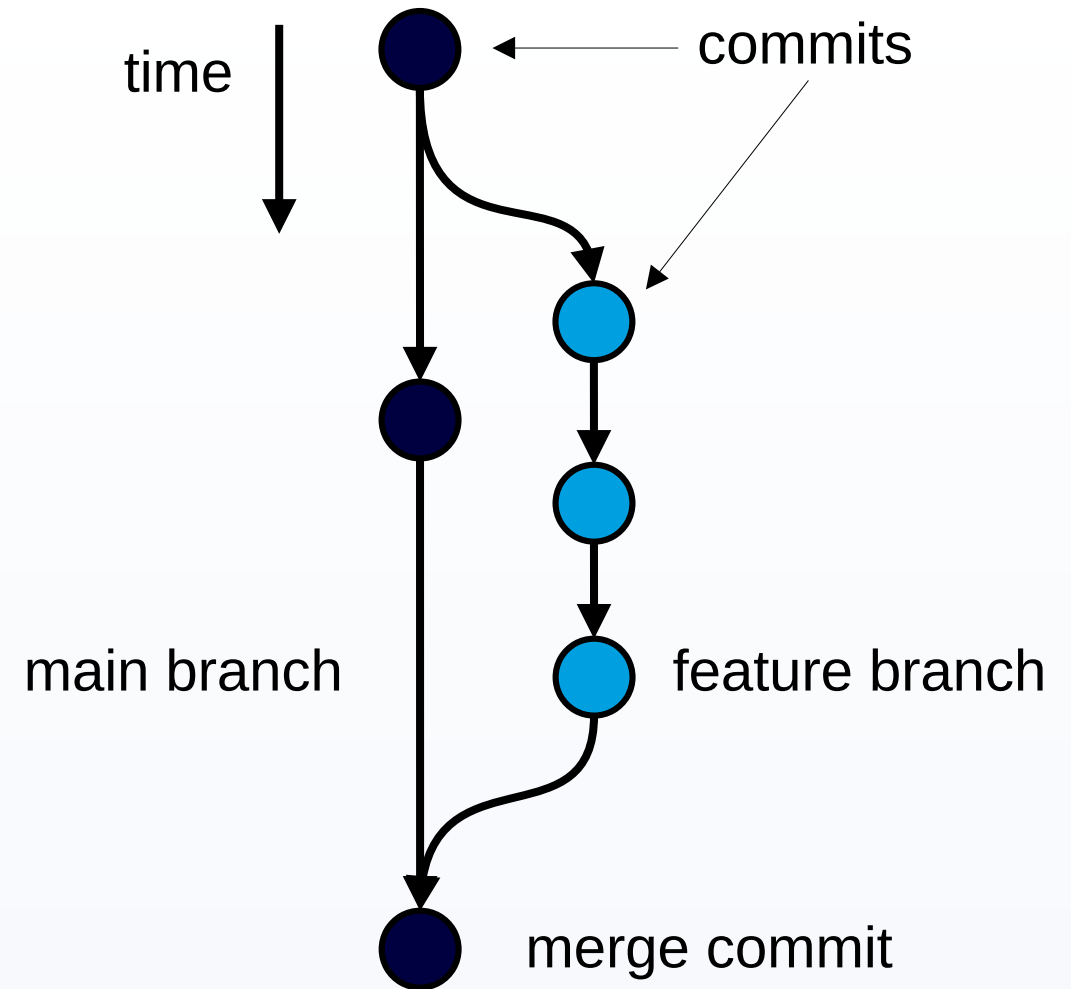
```

Remote repositories

- Can use git entirely locally
- Often, want to back up/share (**remote**)
 - e.g., GitHub
- git doesn't automatically send changes to/from remote repository
 - `git fetch`: get changes from remote (but don't update locally)
 - `git pull`: get changes from remote and update locally (**merge**)
 - `git push`: send changes to remote



- Often, we want to develop different things at the same time
- **Branches** are independent development lines
 - e.g., work on new feature without breaking everything
 - When feature is ready, **merge** back to main branch



GitHub != git

- **git**: a distributed version control system
- **GitHub**: a popular website for hosting git repositories
 - Others include GitLab, Bitbucket
 - **GitHub Desktop** provides a GUI for GitHub



git



GitHub



GitLab



Bitbucket

- Git is a tool to help us keep track of changes in files over time
- Each project is stored in a repository that includes all files and the history
- Keep track of changes using **commits** (savepoints)
- GitHub is a (very popular) website for hosting git repositories

- Git Handbook [[GitHub](#)]
- Understanding the GitHub flow [[GitHub](#)]
- GitHub Training & Guides [[YouTube](#)]
- [Learn Git Branching](#)
- GitHub without the command line [[CodeRefinery](#)]