Git

November 26, 2025

1 Git Guide

Git workflows follow a simple sequence: install Git, clone the project, create a branch, make changes, open a pull request, and merge when approved.

- **Branches:** Think of this as being a 'sandbox' for development. You can mess around in a branch without destroying the project with bad code.
- **Pull Requests (PRs):** After messing around in a branch, if you think your changes are ready for the main project, a pull request signals to the owner of the project that you'd like for your code to be reviewed and added to the project.
- **Merging:** Incorporates approved changes from your branch into the main branch. After merging, branches are typically deleted.

We will learn to install Git, clone repositories, create and switch branches, open pull requests, merge safely, and use a simple command checklist for daily work.

1.0.1 + Notes

- `highlighted commands` should be run in a terminal.
- GPT is a very helpful tool for any OS-related tasks - use it!

1.1 Installation

1.1.1 Linux

- Check if Git is installed
  `git --version`
- Install Git using your distro's package manager (examples):
  - Debian/Ubuntu:
    ```
    sudo apt update
    sudo apt install git
    ```
  - Fedora:
    `sudo dnf install git`
  - CentOS/RHEL (with dnf/yum):
    `sudo dnf install git`

  - Arch/Manjaro:
    `sudo pacman -S git`

  - openSUSE:
    `sudo zypper install git`

- Configure your identity (once per machine):
  ```
  git config --global user.name "Your Name"
  git config --global user.email "you@example.com"
  ```

1.1.2 Mac

- Check if Git is installed
  - `git --version`
- Option 1: Xcode Command Line Tools (simple)
  - Run: `git --version`
  - If prompted, choose to install "Command Line Developer Tools" and follow the dialog.
- Option 2: Homebrew (recommended if you use brew)
  - Install Homebrew (if not already installed) – see brew.sh for the one-line install.
  - Then: `brew install git`
- Configure your identity (once per machine):
  ```
  git config --global user.name "Your Name"
  git config --global user.email "you@example.com"
  ```

1.1.3 Windows

Note: I **highly encourage you** to use linux if not on a mac, it is a much cleaner OS and working with it is a **lot** simpler than Windows. It is very easy to install Ubuntu, which is ready for research out of the box, is a far lighter OS (less telemetry, background processes etc.) and is completely open source. You will look back and think "why did I ever use windows in the first place".

If you would like to do this, I recommend looking for a guide on youtube or otherwise. You will need a USB. It is as simple as - Creating a bootable USB (OS runs on the USB) from the ubuntu .iso file using Balena Etcher or a similar tool - Plugging the USB into your computer and installing the OS using the automated installer. - Make sure to **back up important files on an external drive!** A full system overwrite will erase your previous OS and all of it's contents. - If you are feeling like a pro and want to be able to still run windows, look into dual booting. You will need to make sure you have enough space on your hard drive to partition the drive for both OS's.

We carry on to the instructions:

- Check if Git is installed (PowerShell or Command Prompt)
  - `git --version`
- Install Git for Windows
  - Go to the official site (search "Git for Windows download").

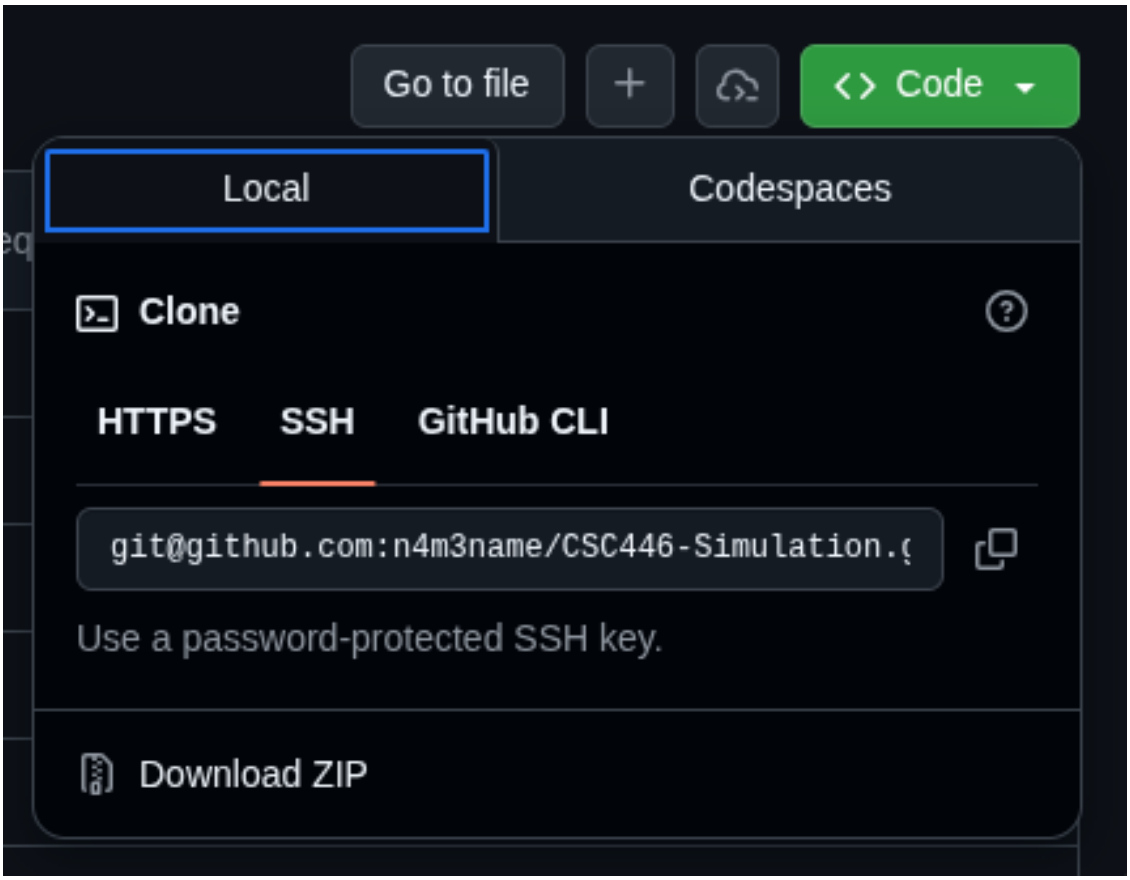
  - Download the installer `.exe`.
  - Run it and accept defaults unless you know you need something specific (this installs **Git Bash** and the `git` command).

- Use Git Bash (recommended)
  - Open "Git Bash" from the Start Menu for a Unix-like shell.
  - In Git Bash, run: `git --version` to confirm installation.

- Configure your identity (once per machine) in Git Bash or CMD/PowerShell:
```
git config --global user.name "Your Name"
git config --global user.email "you@example.com"
```

1.2 Cloning a repository

+ Note: Repositories can be found on github. To find the link (following `git clone` below), navigate to the repository and navigate to the `Code` dropdown menu:



Copy the link and paste into your terminal after `git clone` (this is usually done using `ctrl+shift+v` but may vary depending on system).

Examples: - Clone via HTTPS:
`git clone https://github.com/username/repo.git`

- Clone via SSH (requires SSH keys set up):
  `git clone git@github.com:username/repo.git`

This creates a new folder with the project files and its full Git history. After cloning, cd into the project directory:
`cd repo`

Common options:
- Clone into a specific directory name:
`git clone https://github.com/username/repo.git myfolder`

It is best to first navigate to the parent folder where you want the repo to be, e.g. if you keep your projects in `~/Projects`, you would first run `cd ~/Projects` then `git clone ...`.

Once you have cloned the repo, in order to run git commands you must be in the repository's main folder. - To find the folder from the folder in which you ran `git clone`: `ls` - This command will show all folders: The repo name should show up - To enter the repo folder: `cd repo-name`

1.3 Branches, Commits, Pushing and Pull Requests

Branches let you work on new features or fixes without touching the main project. Think of a branch as a safe copy of the code where you can experiment without breaking anything. When your work is ready, you bring it back into the main project by merging.

Commits are snapshots of your work. Each commit saves a set of changes along with a short message describing what you did. Commits let you track progress, undo mistakes, and review how the project changed over time. It is important to make commits regularly while you are working on the project.

Pushes upload your local commits to GitHub so the remote copy of your branch stays up to date. You must push your branch before you can create a pull request. Pushing also lets teammates see your changes and collaborate on your work.

Pull Requests (PRs) are how you propose your changes on GitHub. A PR shows the differences you made, lets teammates review them, and ensures everything is correct before merging. This keeps the main branch clean and prevents accidental mistakes.

We usually do the above in the order they are written. Sine this is a lot of info, we will include a short cheat-sheet/ command list at the end of the file that has all the main commands in the order that you should use them.

1.3.1 Branch

+ **Note:** Do this **before** you start working on a repository!

- To create a new branch and switch to it:
  `git checkout -b my-branch`
- If you already created a branch and just want to switch to it:
  `git checkout branch-name`

- To see all existing branches:
  `git branch`

The branch that holds the production code is created when the repo is initialized. It is usually called `main` or `master`, and you are usually in this branch after running `git clone`. **Do not change code in this branch!**

1.3.2 Commit

+ **Note:** Do this **while** you are working on a repository!

- To add all your changes to a commit:
  `git add .` or `git add -a`
- To create a commit:
  `git commit -m "description of changes"`

1.3.3 Push

+ **Note:** Do this **after** you are done working on a repository!

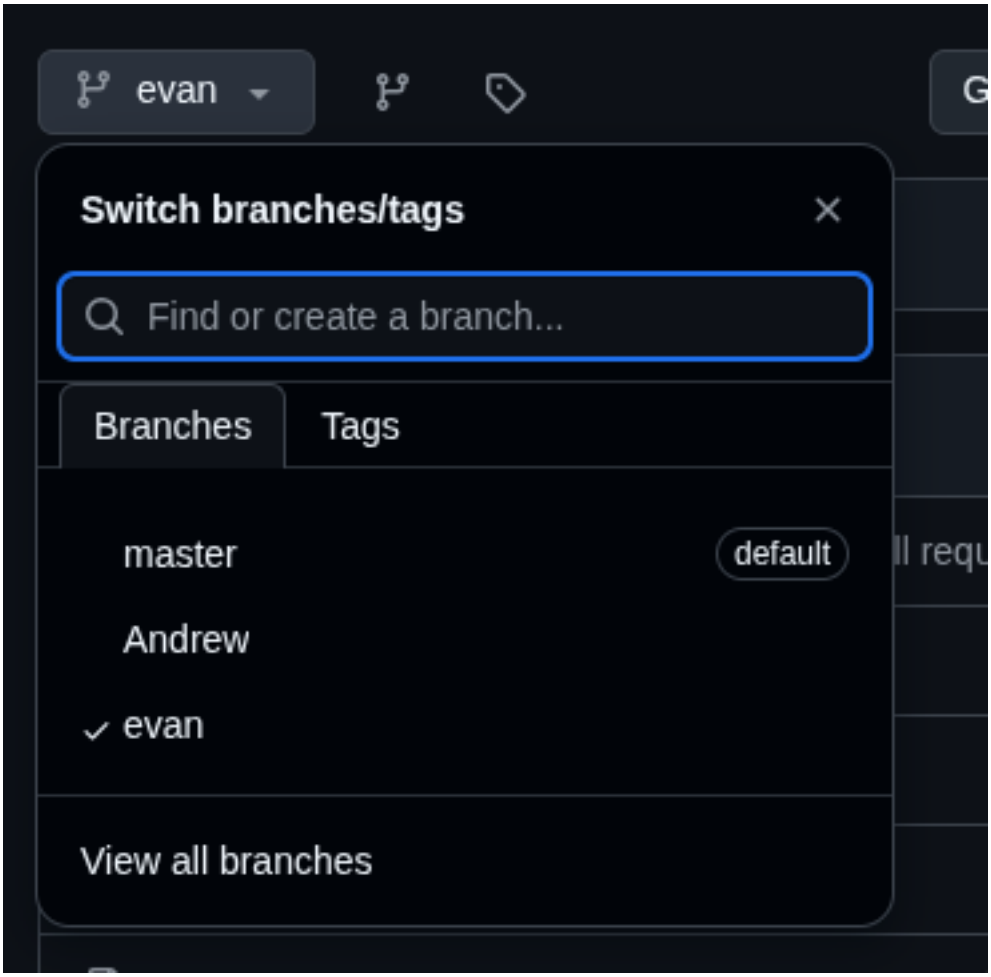
- To push your changes to the branch:
  `git push -u origin my-branch`

1.3.4 Pull request

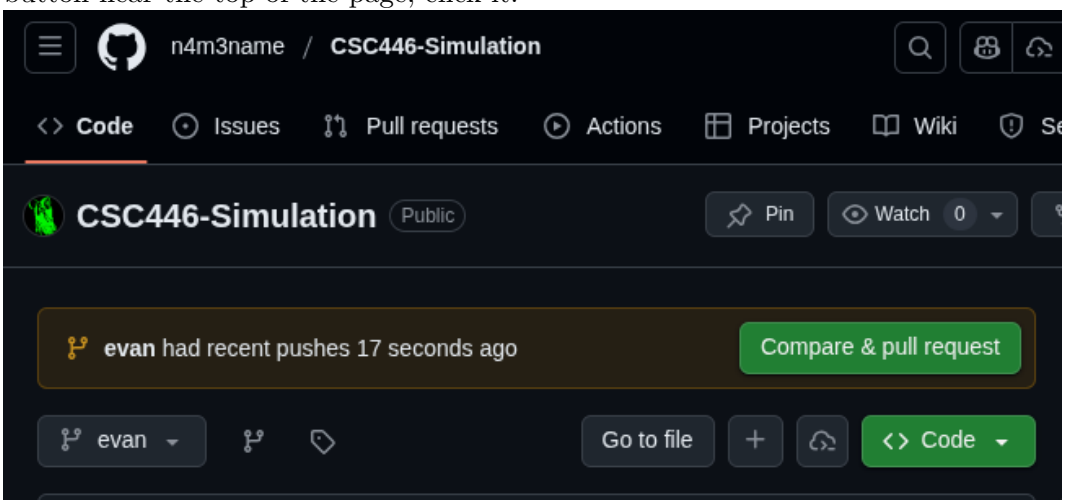
+ **Note:** Do this **after** you have pushed your work, on github.com!

Pull requests are made on github itself. Optionally, you can download a github CLI package such as `gh` or use vscode features - it is simplest to use the github web UI.

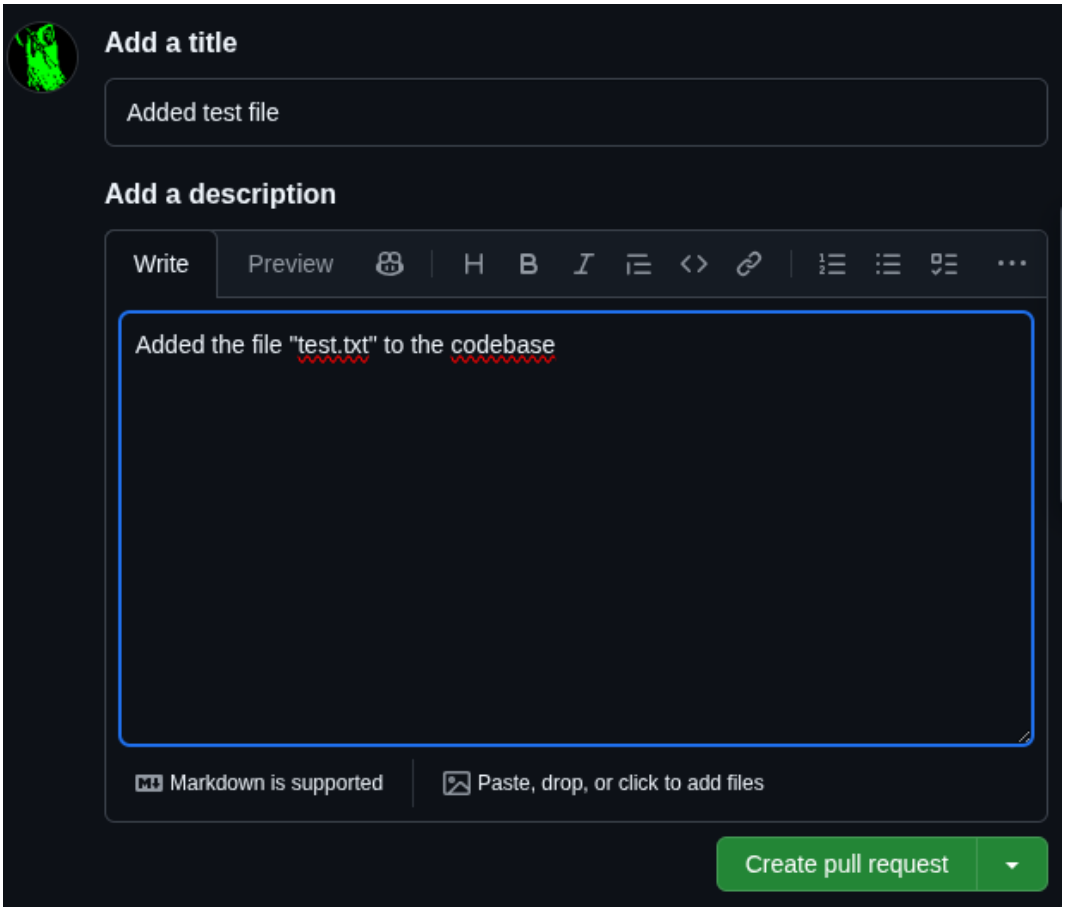
- Navigate to the branch where you have made using the branch dropdown menu on the main repo page:



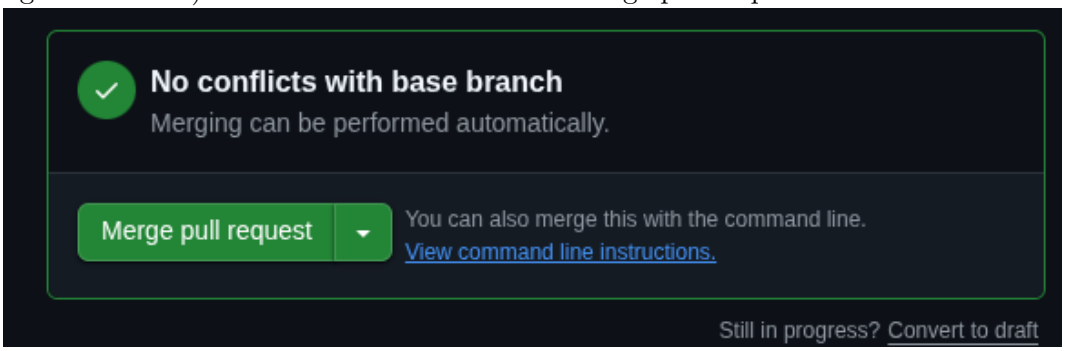
- If you have pushed changes to your branch, you should now see a “compare and pull request” button near the top of the page, click it:



- Here you will be prompted to title the change (use an appropriate, descriptive but concise title) and a description (describe only exactly what changes you made to the code). Once done, click “create pull request”:



- In most cases (class/research project), the merge will be good to go since nobody will be working on the same files. We will not cover the case where, for example, the same file has conflicting changes. (The UI is simple enough that a scientific researcher should be able to figure this out!) We should be able to click "merge pull request:



Note that if you are not the repo owner, you will not be able to merge your code. In this case you only need to create the pull request.

1.4 General use: Cheat sheet

```
# Clone a repo
git clone <link>
```

```
# Switch to the repo folder (replace repo-name with the folder name)
cd repo-name

# ----------------------------------

# Before coding:

## Make sure code is up to date
git checkout master
git pull origin master

## Create a branch for your work (only need to do if it doesnt exist)
git checkout -b my-branch

## Change to your branch & ensure code is up to date
git checkout my-branch
git rebase master

# You can now code safely.

# ----------------------------------

# After coding:

## Stage and push changes to your branch
git add .
git commit -m "one-line description of changes"
git push -u origin my-branch

## Now go to github.com and create a pull request!
```