Building online profile using Git & Github with GitHub Desctop

School Of Social Sciences

The University of Manchester

Dr Tatjana Kecojevic

Introduction

About Me

I'm Dr Tatjana Kecojevic, a statistician with over 20 years of experience teaching data analysis at universities in England. I hold a PhD in Statistics from the University of Manchester, where I currently teach statistics in the social sciences. I'm also the founder of SisterAnalyst.org, an initiative focused on making data skills accessible and empowering women to use data to accelerate their careers. My work is about building confidence and practical skills so that everyone can benefit from using data in their everyday work and decision-making.

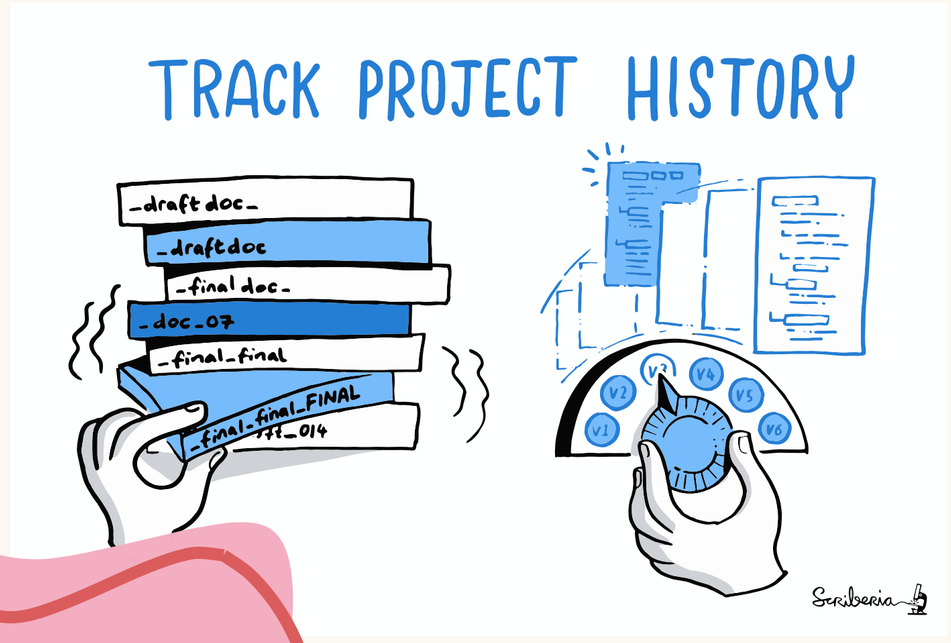
e-Portfolio Project

Today's plan:

Today's plan:

- **What is Git & GitHub?**
  - Understand the basics of version control and how GitHub supports collaboration.
- **Why version control?**
  - See why tracking changes matters in any digital project.
- **GitHub Desktop: download & set up**
  - Learn how to install and connect to GitHub without using the command line.
- **Hands-on practice:**
  - Fork & Clone a repository
  - Make & undo changes
  - Commit and Push updates
  - Create branches for edits
  - Open issues and pull requests
- **Wrap up:**
  - Recap Git flow
  - Explore GitHub Pages (optional publishing)
  - Your turn: build your own portfolio repo!

git



What is Git?

- A version control system
- Tracks changes to files
- Allows you to go back in time
- Essential for teamwork and project history

🔗 Learn more: What is Git? (Git SCM)

What is GitHub?

- A cloud platform that stores Git repositories
- Makes it easy to:
  - Back up your code
  - Share and collaborate
  - Track changes

GitHub

🔐 Don't have an account yet?
👉 Register for GitHub

What is GitHub Desktop?

- A free app that lets you use Git & GitHub without the command line
  - Great for beginners
  - Point-and-click interface
  - Download it here:

👉 https://desktop.github.com

git

GitHub

Installing GitHub Desktop

1. Visit: https://desktop.github.com
2. Click Download for Windows or macOS
3. Install the application
4. Sign in to your GitHub account

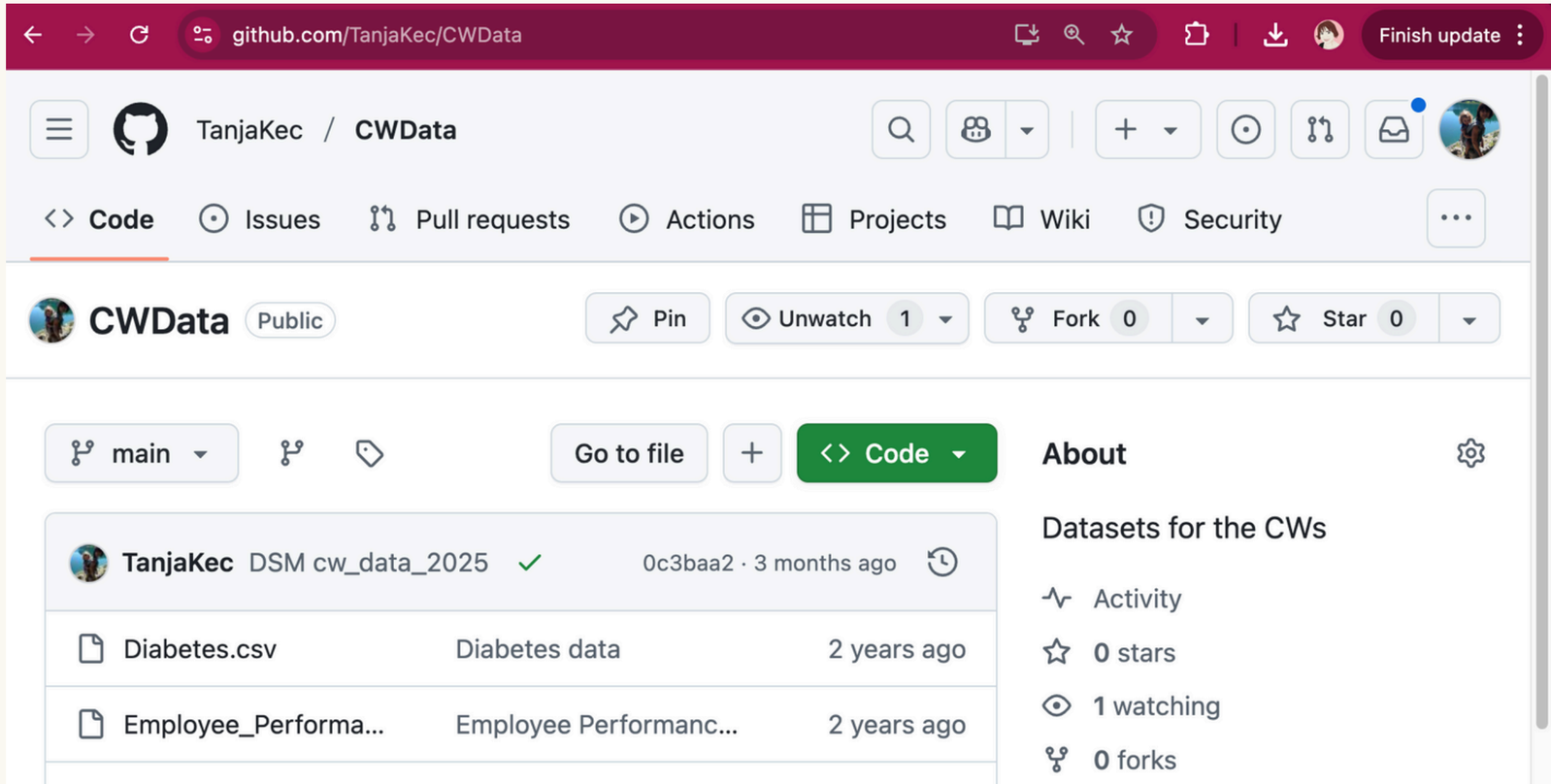
👉 Create GitHub account if you haven't yet

GitHub Web Interface 🌐

When you visit a repository on github.com, you'll see:

- Files and folders in the project
- The `README.md` – homepage of your repo
- Tabs: Code, Issues, Pull requests, Actions
- A green Code button to copy the URL
- Project contributors and commit history

📎 **This is where others can see your work online!**



GitHub Desktop Interface 💻

When you open GitHub Desktop, you'll see:

- Current Repository dropdown (top left)
- Changes tab to review edits before committing
- History tab to see past commits
- Commit summary message box
- Push/Pull buttons to sync with GitHub

📌 GitHub Desktop helps you do version control without using the terminal.

✅ To add and switch to another repository:

1. Click the "Add" button (right next to the Filter bar)
2. Choose one of the following options:
   - **Clone Repository:** to download a repo from your GitHub account
   - **Add Existing Repositor:** if you already have a folder on your computer that is a Git repo
   - **Create New Repository:** to start a new one from scratch
3. Once added, it will appear in that list

Exercise: Create a New Repository

New Repo!

1. Open GitHub Desktop
2. Go to File -> New Repository
3. Fill in:

Name: my-first-repo

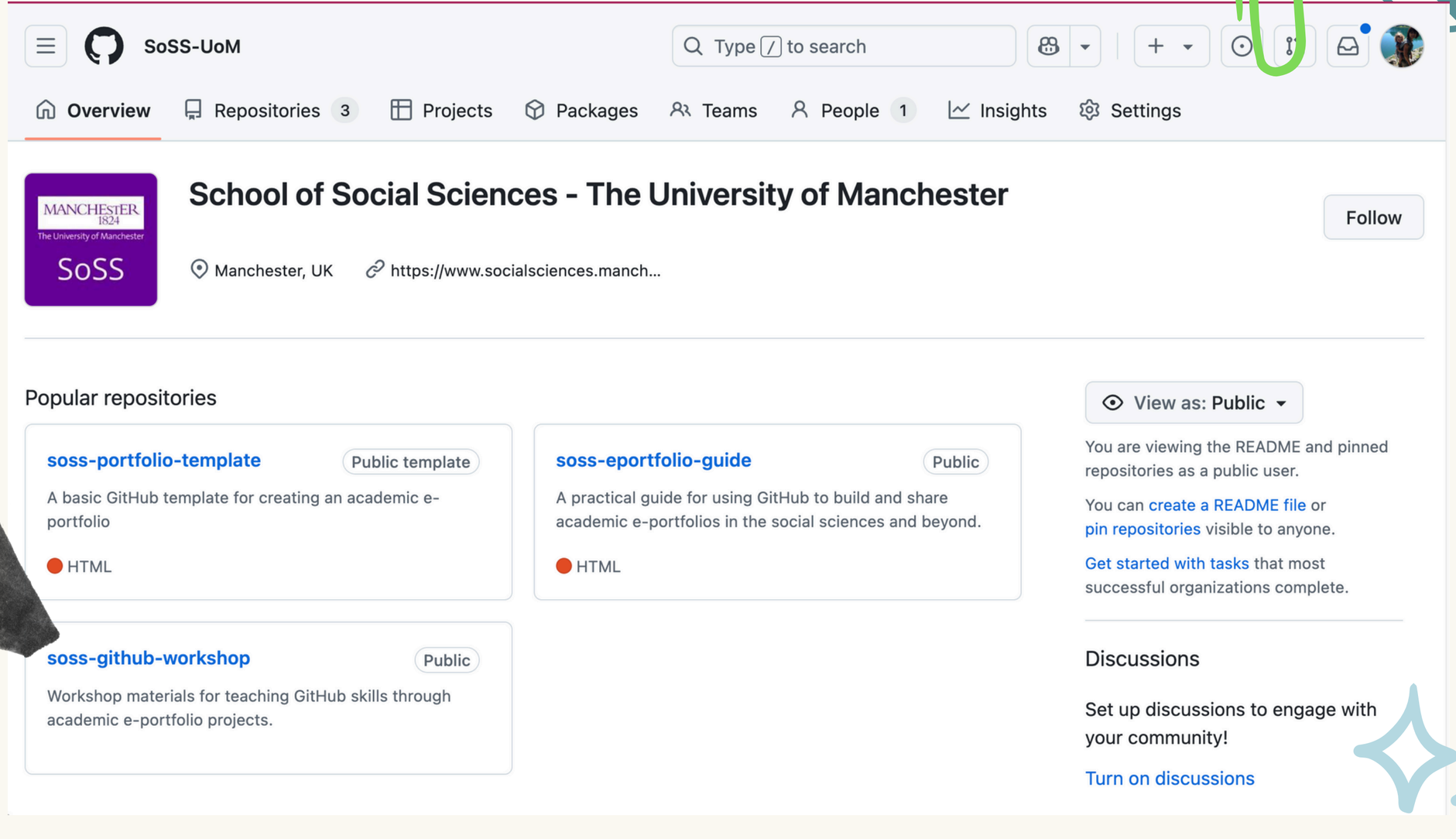
Local path (where it saves on your computer)

4. Click Create Repository

✅ You now have a local Git project

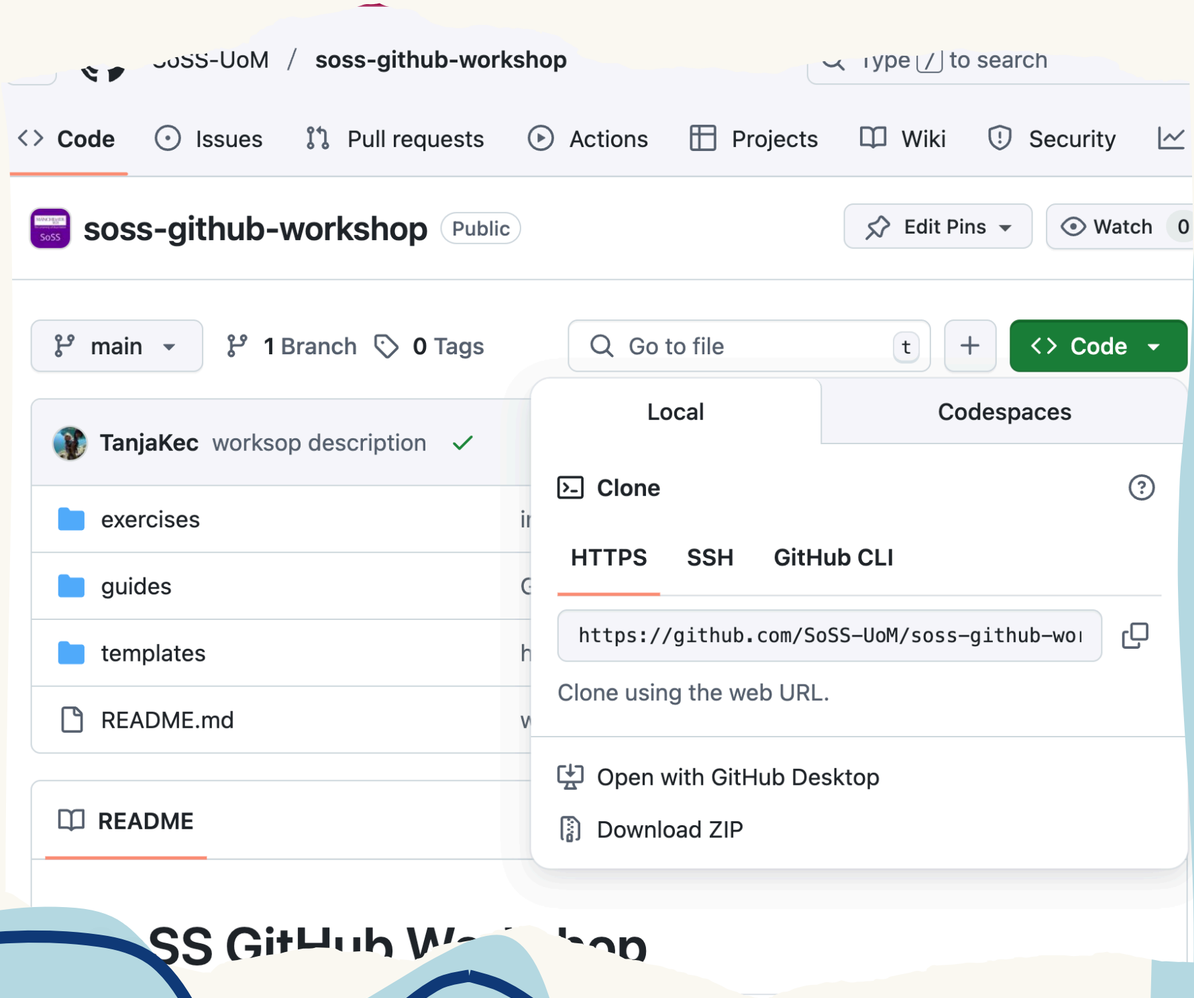
SoSS-UoM

https://github.com/SoSS-UoM



Exercise: Clone a repo!

Clone a Repo!



Use GitHub Desktop to:

- Clone the workshop repo
- Make edits locally
- Commit and push changes to GitHub

Repo: github.com/SoSS-UoM/soss-github-workshop

See: `exercises/intro-exercise.md`

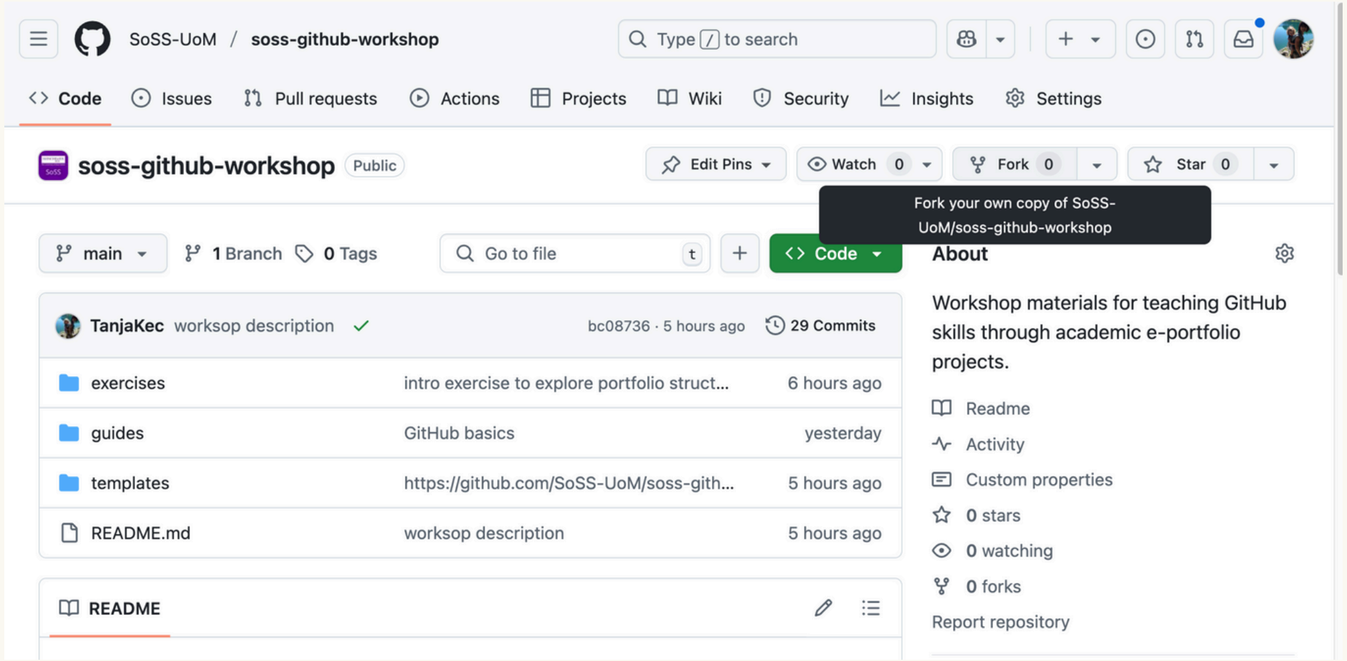
✅ You have now cloned a repository! It's on your computer and ready for editing!

BUT!

This repo is part of the **SoSS-UoM GitHub Organisation**, created to support digital skills and open collaboration in the School of Social Sciences.

Next Steps: Fork → Clone → Edit → Push

If you previously cloned the original SoSS-UoM/soss-github-workshop repo, delete that local folder to avoid confusion or editing the wrong version.



✅ **Step 1: Fork the repository**
On GitHub, click the Fork button (top right).

This creates your own copy under your GitHub account.

✅ **Step 2: Clone your fork**
In GitHub Desktop:

Go to File → Clone Repository
Choose your forked repo to clone it locally.

✅ **Step 3: Make edits locally**

Open the folder in a text editor like VS Code or RStudio.
Try editing:

- README.md
- or a file in /notes

✅ **Step 4: Commit & Push using GitHub Desktop**

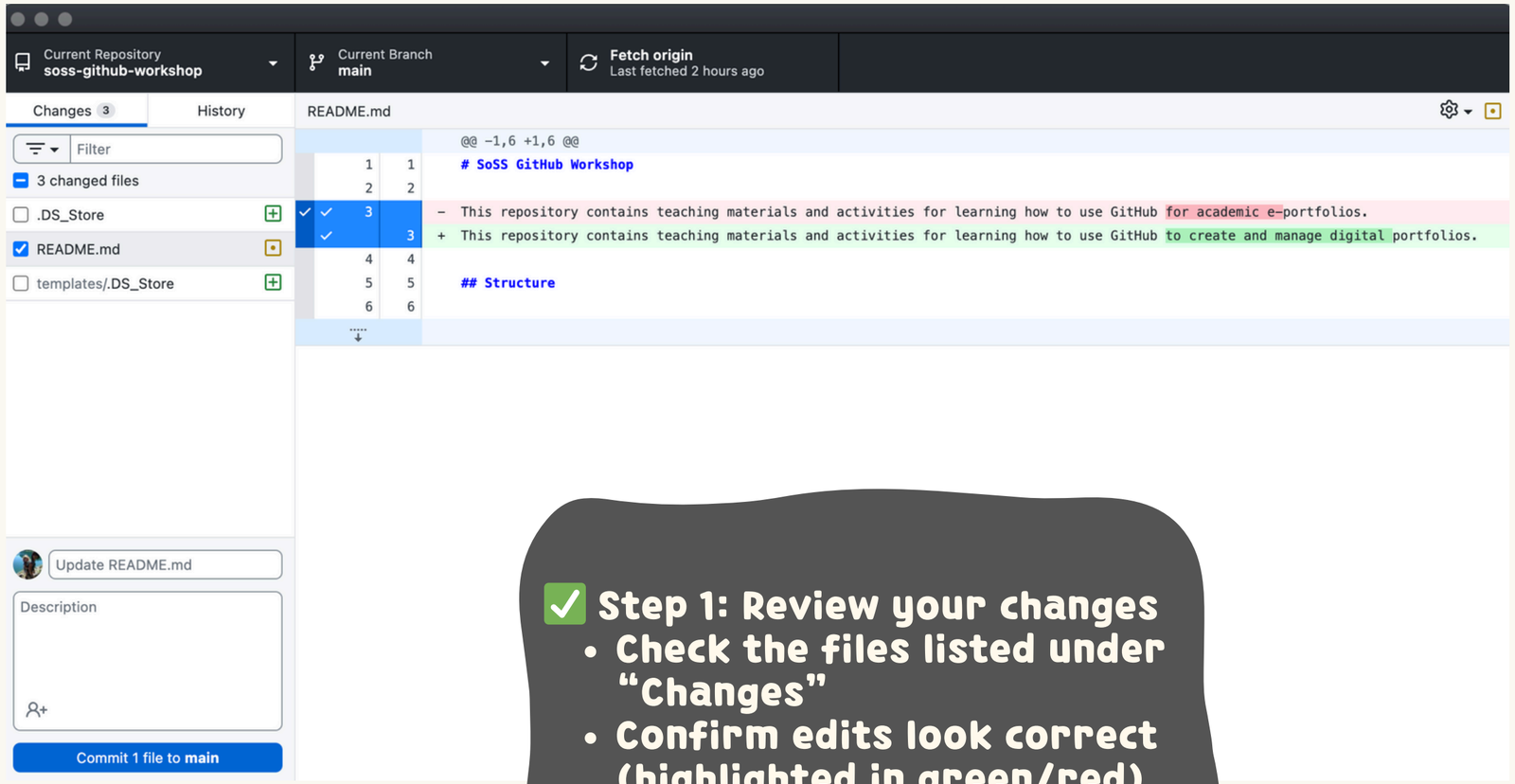
- Review changes
- Write a short commit message
- Click Commit to main, then Push origin

(see the following slide)

✅ Forking keeps the official repo safe
✅ Cloning puts it on your computer
✅ Small commits help you track your changes

Make a Change → Commit → Push

You've edited a file locally (e.g. README.md)
Now use GitHub Desktop to save and upload your changes:



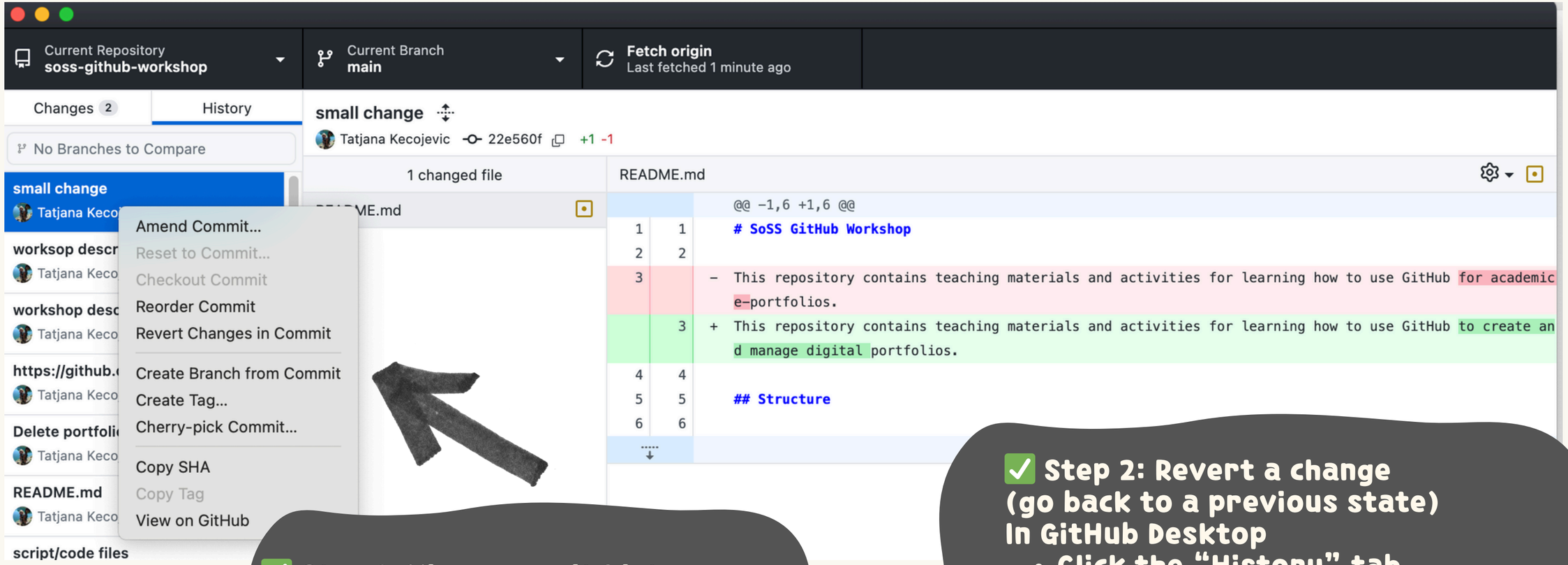
✅ **Step 1: Review your changes**
- Check the files listed under "Changes"
- Confirm edits look correct (highlighted in green/red)

✅ **Step 2: Write a commit message**
- Use a clear message like:
  - "Update README to broaden scope of portfolio use"
- This message helps track what each change does

✅ **Step 3: Commit and Push**
- Click "Commit to main"
- Then click "Push origin" (top bar)

✅ **Tip: Commit small, descriptive changes regularly so you can easily track and undo steps later.**

Undo a Change: Explore Commit History & Revert



✅ **Step 1: View commit history**

- In GitHub Desktop:
  - Go to the "History" tab to view previous commits.
- On GitHub.com:
  - Go to the repo → click on README.md → click "History".

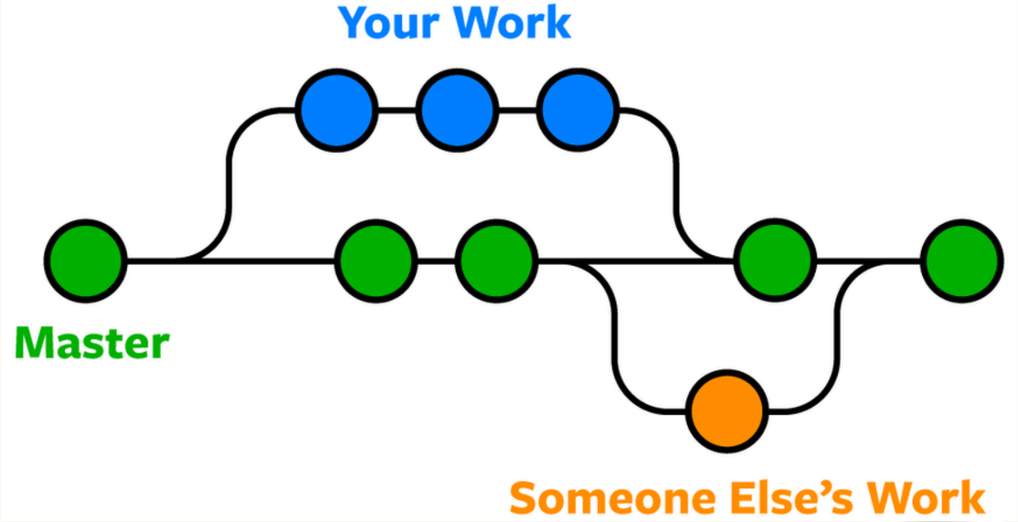
✅ **Step 2: Revert a change (go back to a previous state) In GitHub Desktop**

- Click the "History" tab.
- Right-click the previous commit → Select "Revert This Commit".
- This creates a new commit that undoes the previous one.

Branching: “What happens to my changes?”

Why Branch?

Branches let you work on different features or edits without affecting the main project. Great for collaboration, testing, or drafting ideas safely.



Why branch?

- Work on multiple things at once without conflict.
- Test or draft changes before sharing them.
- Collaborate safely with others.

What is a Branch?

A branch is a separate copy of your project where you can work on new ideas, fixes, or updates, without changing the original (main) version.

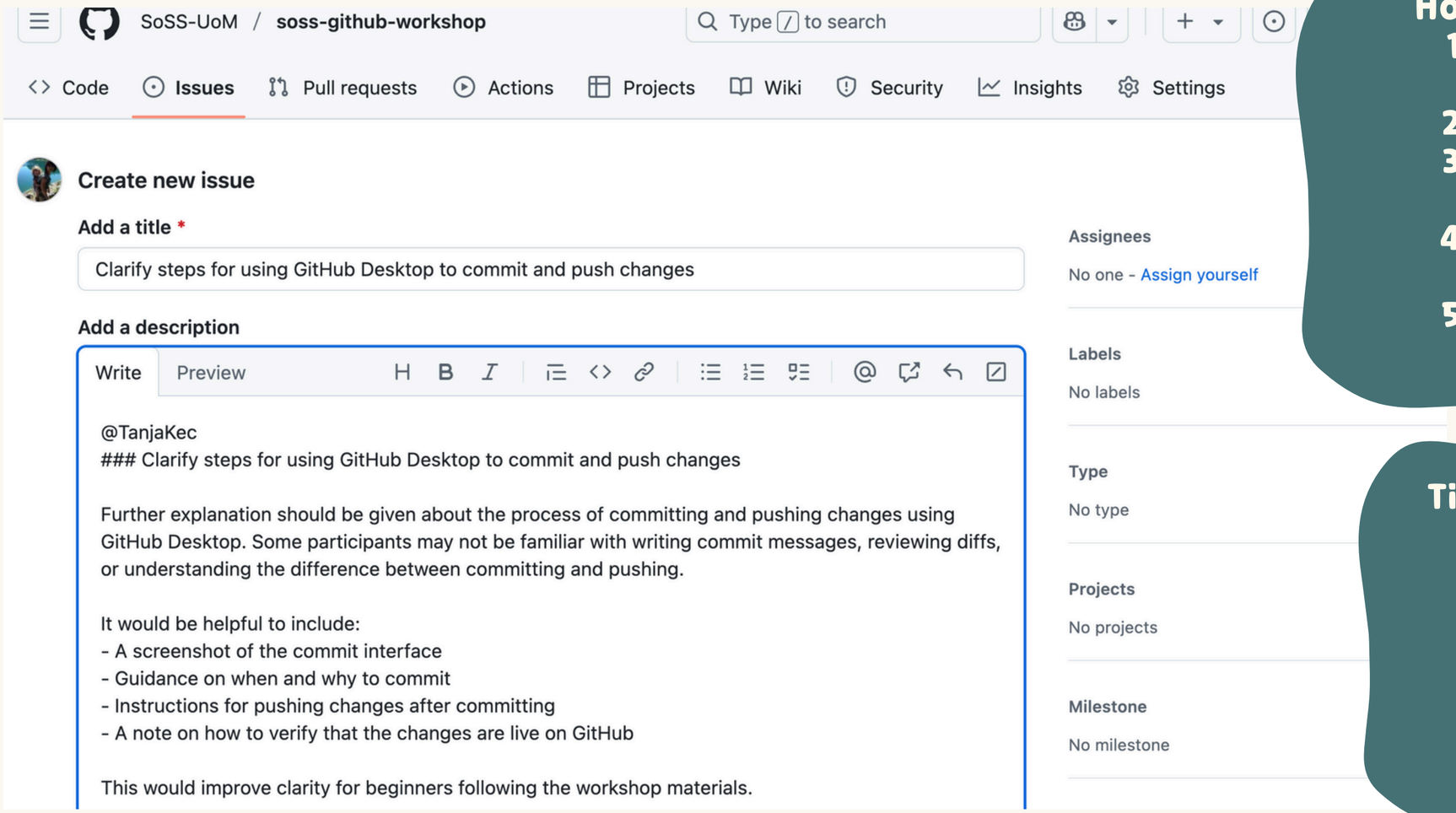
In this diagram:

- Green (Master/Main) is the official project version.
- Blue (Your Work) is a feature branch where you’re editing safely.
- Orange (Someone Else’s Work) is another person’s changes, also in a branch.

Collaborate with Issues

Why use Issues?

Issues help you track tasks, bugs, ideas, and questions. They keep collaboration organised and transparent.



How to create an Issue:

1. Go to the Issues tab in your repository.
2. Click New Issue.
3. Give your issue a clear title and description.
4. (Optional) Add labels like bug, enhancement, or question.
5. Submit the issue and tag collaborators if needed.

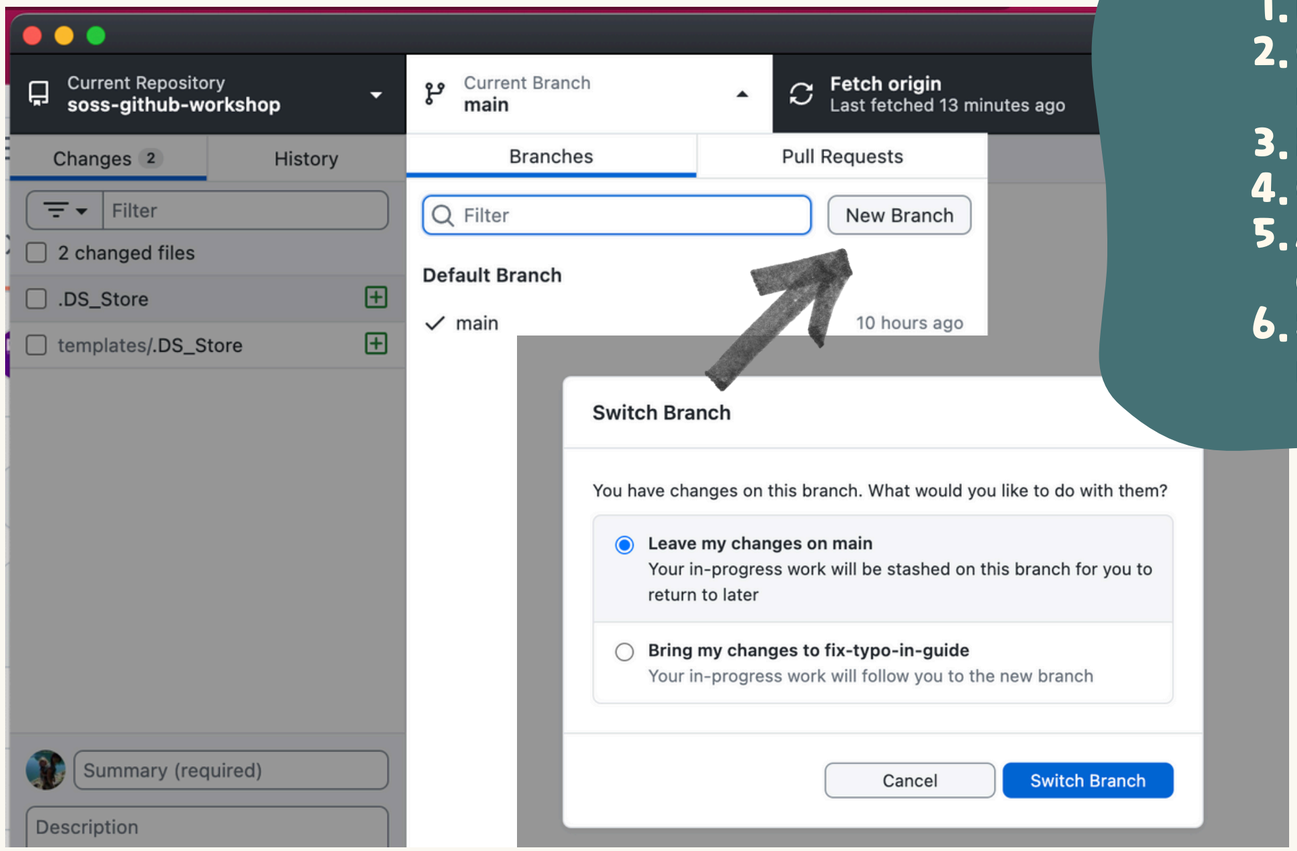
Tips:

- Use issues to ask for help or suggest changes.
- Reference specific files or lines using @ mentions or commit links.

Contribute with Pull Requests

What is a Pull Request (PR)?

A pull request lets you suggest changes to a repository. Others can review and discuss them before merging.



Steps to create a PR:

1. Fork and clone the repo.
2. Create a new branch and make your changes.
3. Push your branch to GitHub.
4. Click Compare & Pull Request.
5. Add a meaningful title and description of your changes.
6. Submit the PR and wait for review.

Best Practice:

- Keep PRs focused (1 task per PR).
- Reference issues by writing Fixes #issue-number in the PR description.

Since I'm working on Collaborate with a pull request, and I'm about to start a new feature or fix, I'll choose:
✅ Bring my changes to fix-typo-in-guide
(since the edits I already made are meant to be in that branch)
Then I'll click Switch Branch.

What if You Already Made Changes?

Switching Branches with Unsaved Work

What Happens If You Select “Leave My Changes on Main”?

- **Your uncommitted edits stay saved (but not lost) in the main branch.**
- **GitHub Desktop switches to the new branch with a clean slate.**
- **The edits will not appear in the new branch.**
- **You can return to main later to finish or commit those changes.**

✅ **Useful when you want to:**

- **Keep different tasks separate**
- **Avoid mixing edits between branches**
- **Start fresh work on a new feature or fix**

Summary

1. Fork
Make your own copy of the original repository on GitHub.

2. Clone
Bring that forked repo to your local computer using GitHub Desktop.

3. Branch
Create a new branch to work on a specific feature or edit, without affecting the main project.

3. Branch
Create a new branch to work on a specific feature or edit, without affecting the main project.

4. Code
Make your changes in your local files.

5. Commit
Save your changes with a clear message describing what you did.

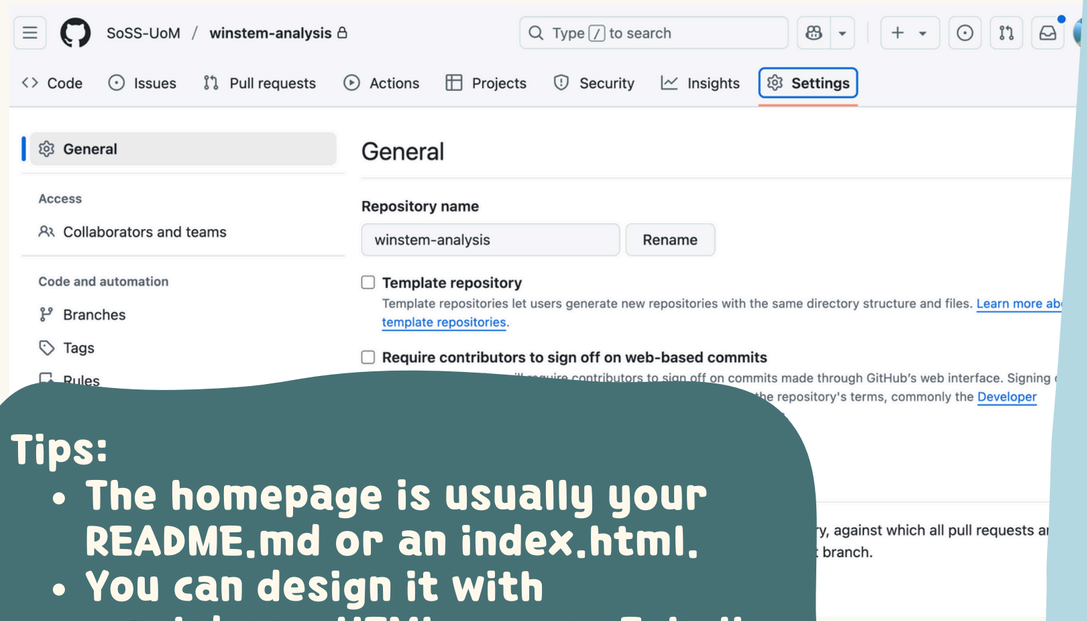
6. Push
Send your committed changes back to your fork on GitHub.com.

7. Pull Request
Propose your changes to the original project, this lets others review and merge your work.

Publish Your Project as a Website with GitHub Pages

What is GitHub Pages?

A simple way to turn your GitHub repo into a public website, great for showcasing projects, portfolios, or teaching resources.



How to Enable GitHub Pages (Using WinStem Repo Example):

1. Go to your WinStem repository on GitHub.
2. Click on the Settings tab.
3. Scroll to “Pages” in the left-hand menu.
4. Under “Source”, select:
   - Branch: main (or another if needed)
   - Folder: / (root) or /docs
5. Click Save.

Your Site is Live!

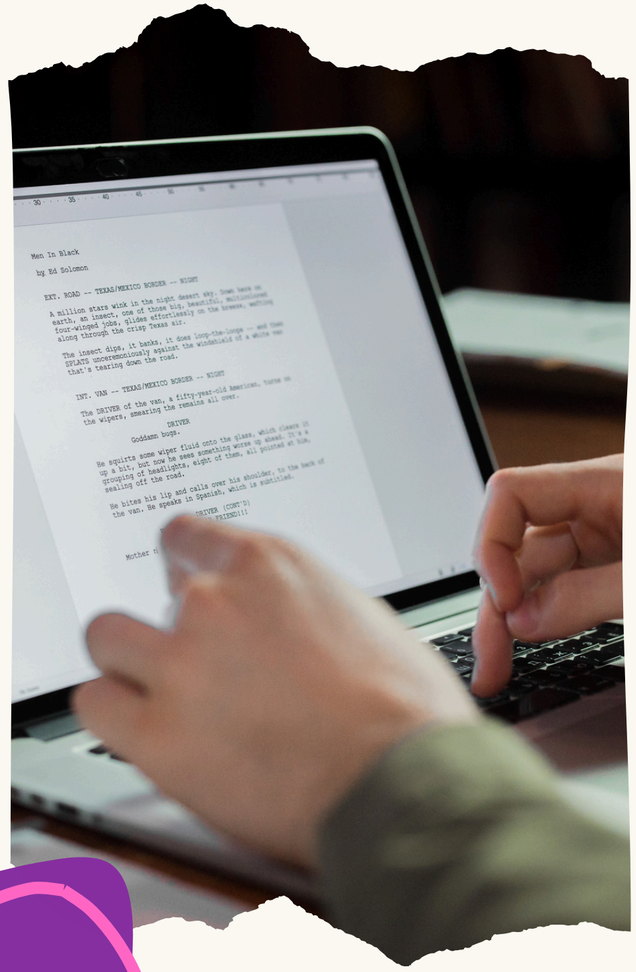
GitHub will give you a link like:
https://<your-username>.github.io/WinStem

It may take a minute or two to publish.

Tips:

- The homepage is usually your README.md or an index.html.
- You can design it with markdown, HTML, or use Jekyll themes.
- Perfect for sharing data projects or educational content.

GitHub Desktop: Summary

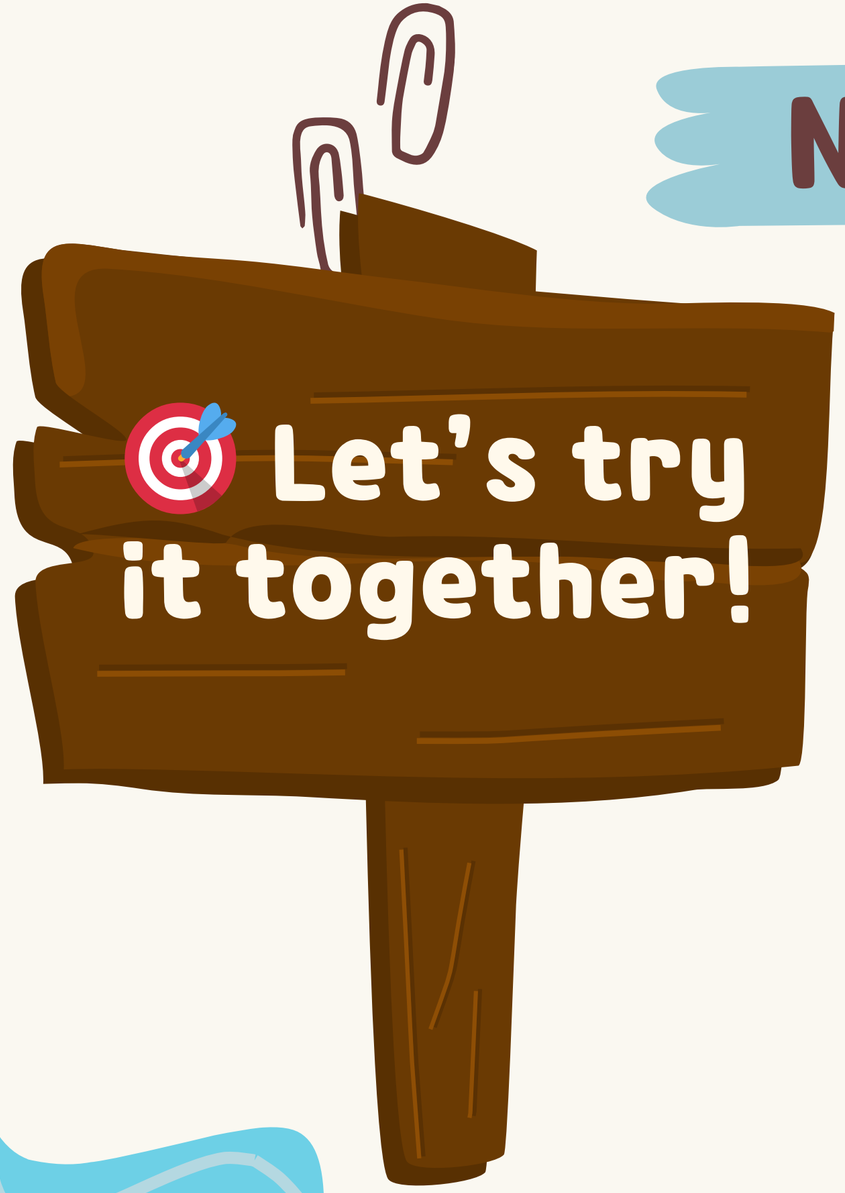


why?

a) Easy to use
b) No need for terminal commands
c) Good way to learn Git basics

You can always move to the command line later!

Next Steps



1. Create a new repo
2. Add a file
3. Commit and push to GitHub
4. Share your repo link with a collaborator!

Questions?

tatjana.kecojevic@manchester.ac.uk

Thank You