



Git Good: an Introduction to GitHub for Collaboration

Create a new GitHub account at <https://github.com/>

Exercise 1 - Commenting on an issue

GitHub issues are an accessible way to contribute to a project. They are often used by project teams to keep track of what they're working on and receive feedback from others.

Issues are like a message thread, which you can connect with other parts of the GitHub workflow (for example, they can be referenced in commit messages). You can tag other project members to get their input, and use emojis to show you care!

Good first issue

In The Turing Way, we use the label “good-first-issue” to identify feedback or tasks that can be actioned by folks who are new to that project.

- 1a) Find the issues in The Turing Way GitHub repo (<https://github.com/alan-turing-institute/the-turing-way>).

Find an issue labelled “good-first-issue”. Choose one of these issues and try to understand what action is being asked for. What information was useful to you? What information was missing?

Using issues for decision making

Issues can be viewed by anyone in the project if the repository is private, or anyone in the world if the repository is public! Anyone with a github account can comment on a public issue. This visibility is important for transparency (showing your working) and accountability (taking responsibility for what you do).

We often use issues to make decisions with transparency and accountability. For this exercise, we will use an issue to decide how you will contribute to a file in Exercise 2.

- 1b) Find the issue #11 in the repository for this workshop (<https://github.com/alan-turing-institute/github-introduction>) and follow the instructions in the issue.

Note that issues can be written in markdown. Take a look at this Markdown Cheat Sheet (<https://www.markdownguide.org/cheat-sheet/>). Use the “write” tab in the issue comments to write using markdown syntax, then the “preview” tab to see how it is rendered.



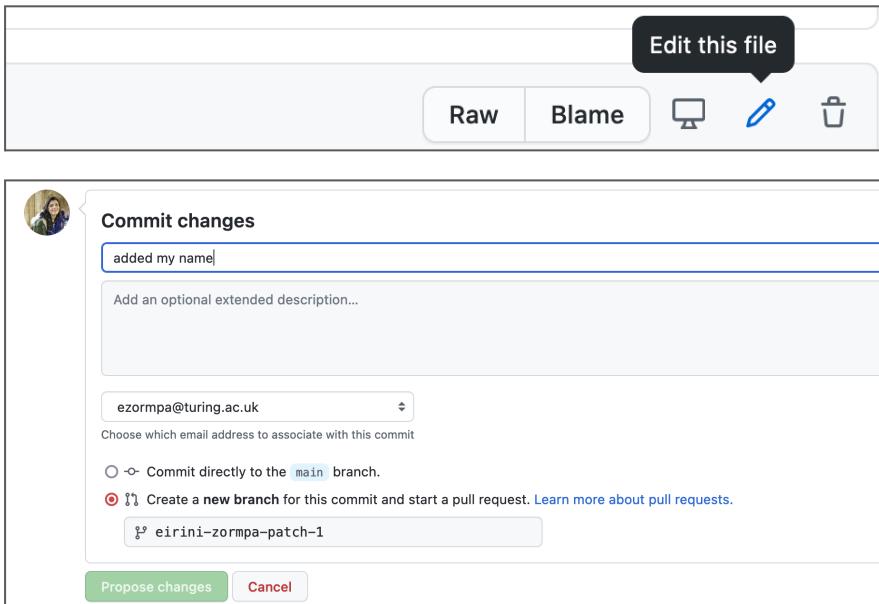
Exercise 2 - Edit a file and make a pull request

Go to the github-introduction repo - [alan-turing-institute/github-introduction: Repository to accompany the Introduction to GitHub workshop](https://github.com/alan-turing-institute/github-introduction)

To contribute to a collaborative project, you need to be able to edit the file you want to make changes to, keep that change in a separate branch and then ask your collaborators to review your work before bringing it into the main branch of the repository. In GitHub, this is done through creating a new branch and then a pull request.

Make a commit on a branch

1. Navigate to the introductions file in the github-workshop folder -
<https://github.com/alan-turing-institute/github-introduction/blob/main/github-workshop/Introductions.md>
2. Click on the pencil icon to edit
3. Write on the line number assigned by the instructor from Exercise 1.
4. Scroll up to the 'Commit changes' green button at the top right
5. Write a good commit message! - *type a message that says what you have changed*
6. Create a new branch for the commit
7. Click on 'Propose changes' - *this creates a commit and sets up the pull request*
8. Don't close this page



The image shows two screenshots illustrating the GitHub commit process. The top screenshot is a preview of a file with a 'Raw' and 'Blame' button, and a 'Edit this file' button. The bottom screenshot is a 'Commit changes' dialog box. It includes a user profile picture, a title 'Commit changes', a message input field containing 'added my name', a description input field, an email dropdown set to 'ezormpa@turing.ac.uk', and a branch selection dropdown. The 'Create a new branch for this commit and start a pull request' option is selected, with the branch name 'eirini-zormpa-patch-1'. At the bottom are 'Propose changes' and 'Cancel' buttons.



Create a Pull request

1. Go back to the pull request page
2. You should see a green check at the top, saying 'Able to merge'
3. Add any additional information in the 'Leave a comment' section, if you like
4. Click on 'Create a pull request'
5. Assign the co-facilitator for the day as reviewer - *this is the person who gave you your number in Exercise 1.*

Open a pull request

The change you just made was written to a new branch named `eirini-zormpa-patch-1`. Create a pull request below to propose these changes.

base: main ▾ ← compare: eirini-zormpa-patch-1 ✓ Able to merge. These branches can be automatically merged.

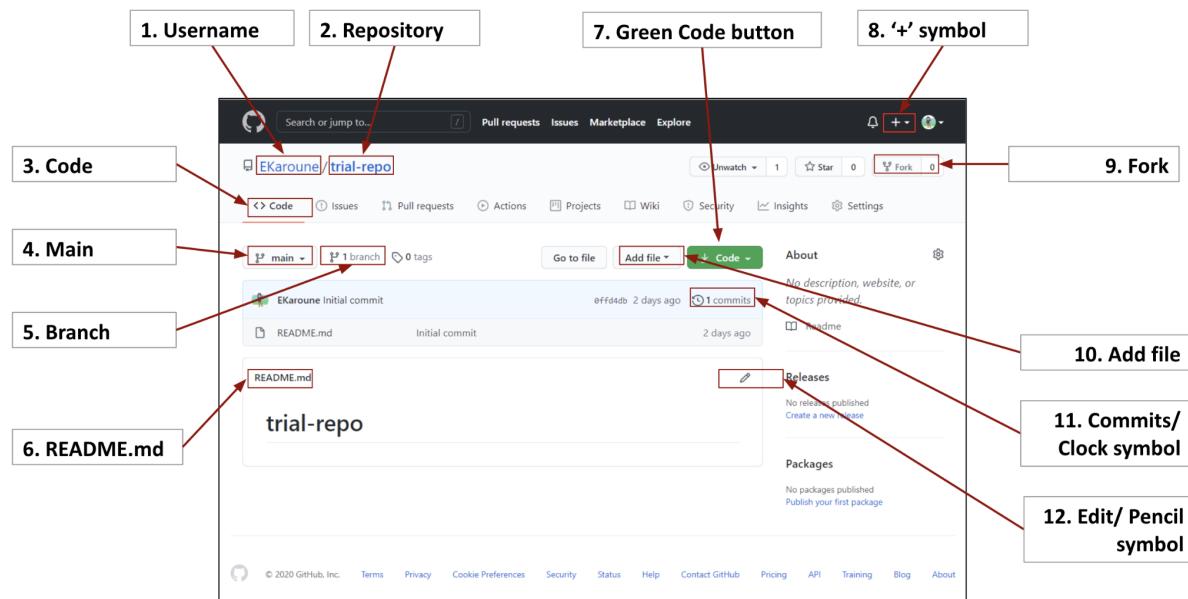


Stretch exercise: Creating a new repository

A repository or repo is the online space where you store all of the documents, data and other files for your project.

- To create the new repository, you need to click on the + sign in the top right corner (in the black band at the top of your window) and then click New repository. This will take you to a page that looks like a form.
- You will see the name of your account and you need to fill in a repository name next to it.
- Also, leave the box ticked for “public” (so your repository is open to all) or “private” (if you want to keep it close to just you) and then tick the box to create a “README file”.
- Then click the green create repository button at the bottom.

This is what you should see now. It is the landing page for your repository. The diagram below explains what all the buttons, tabs and other things do!



Annotated diagram of repository after its basic creation, explaining the main features.

On the left side of the webpage we have the following features:

- 1. Username: GitHub user's name (account). In this example, the username is “EKaroune”.
- 2. Repository: project directory (also known as repo). In this example, the repository name is “trial-repo”.
- 3. Code: this tab brings you back to your landing page. It shows you the folders that you have made in the repo.
- 4. Main: this is your default development branch or active branch of your repository.
- 5. Branch: parallel version(s) of your repository.



- 6. `README.md` file: this file contains basic information about your project (in this case it only has the project name: “trial-repo”. When we plan to make a website, this will be rendered as a landing (front) page for your site.

On the right side of the webpage we have the following features:

- 7. Green Code button: click it to download your repository locally.
- 8. ‘+’ symbol: where you can create new repository, import repos and create new issues.
- 9. Fork: create a personal copy of another user’s repo. The number shows how many forks there are of your current repository.
- 10: Add file: create or upload a file to your repository.
- 11: Commits/clock symbol: click to see the history of this file as a list of all the edits (commits) saved at different time points.
- 12: Edit/Pencil symbol: click this pencil symbol to edit your `README.md` file.

Edit your `README.md` file

Unless you have added any other files or included a license file during repo creation, you should have one file in your repository now - `README.md`. We’ll need to edit this file to add information about the repo. This file is a Markdown file; you can see this because it has “.md” after the name of the file. This is where you start to use the Markdown formatting. Whatever you write in this file will be shown on the landing (front) page of your project on GitHub, so use it to tell people all about your project.

Navigating the GitHub editing interface

To edit your `README.md` file:

- You can click on the pencil symbol in the top right of the central box on your landing page.

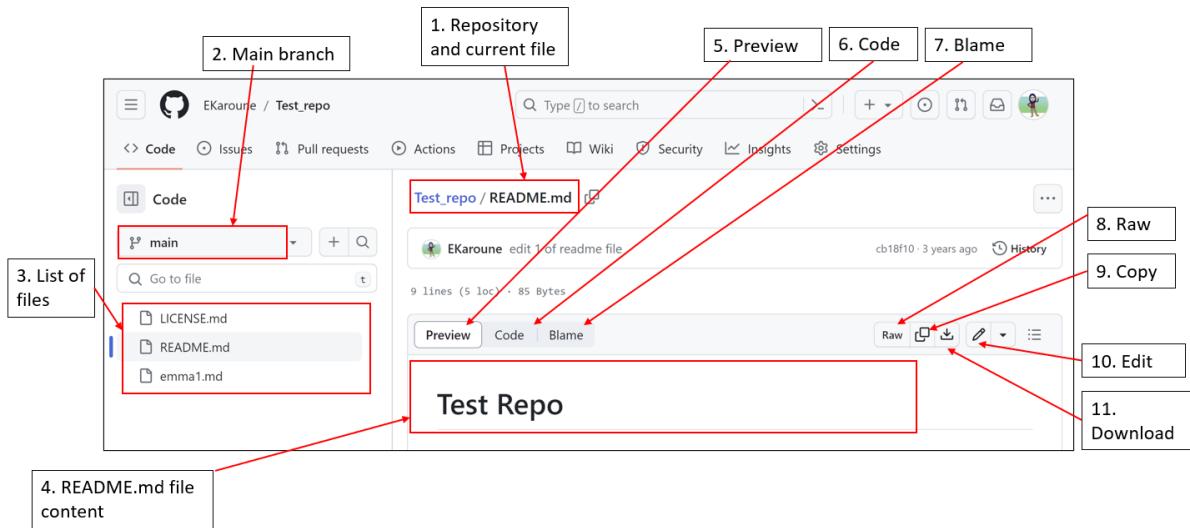
Or

- Click on the `README.md` file and then click the pencil symbol.

You can now edit the file. We’ll talk about how to save your changes after some pointers on writing a good `README`.

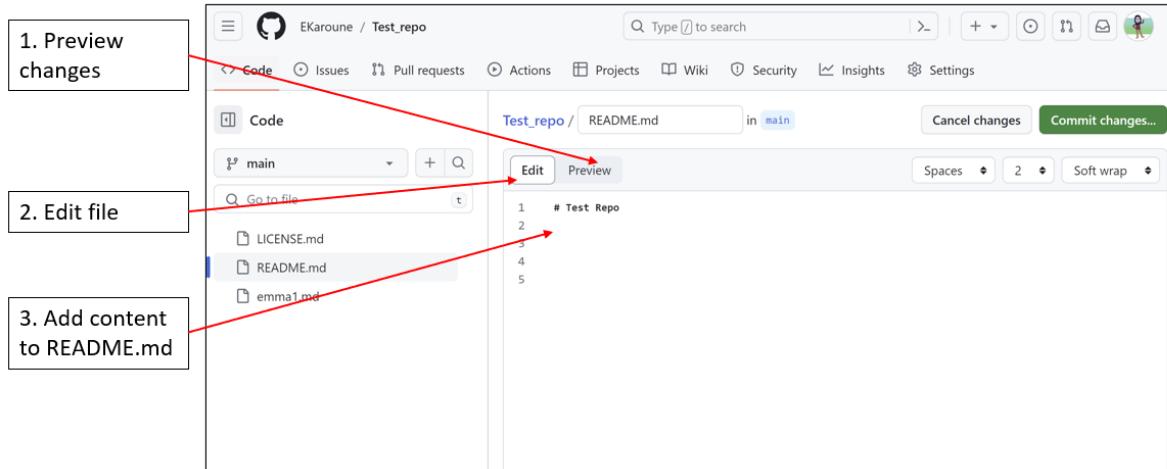
Remember! You can use a Markdown cheatsheet to help you edit the file.

- Markdown cheat sheet - <https://www.markdownguide.org/cheat-sheet/>



Annotated diagram of README.md file, if you click on the file name on your landing page.

- 1. Repository and current file: the repo name and the name of the file you are viewing.
- 2. Main branch: currently active branch (“main” is the default). Use to change to different branches of your repo (if there are more branches previously created).
- 3. List of files: list of files in the repo - you can click on these to go to the file.
- 4. README.md file content: the content of your README.md file appears here. This content will expand once we add more information.
- 5. Preview: shows a preview of the README.md file.
- 6. Code: shows the code in the README.md file.
- 7. Blame: view the last modification made to each line of the file. It can be used to track when and who made changes and go back to older versions of the file to fix bugs.
- 8. Raw file: view the raw markdown text file.
- 9. Copy: copies the file.
- 10. Edit file: click this pencil to edit your README.md file.
- 11. Download file: download file locally.



Annotated diagram of README.md file in edit mode – before editing.

- 1. Preview changes: press to see your text rendered (how it would appear on GitHub or on a web page).
- 2. Edit file: press this tab to edit the content of your README.md file.
- 3. Add content to README.md: write the Markdown text for your README.md file. You currently only have the repository title in this file.

Tips for writing your README file

- Keep it simple! When you're working in any field, whether it's software engineering or astrophysics, you'll learn and use jargon – terms that have a special meaning to your field but likely won't make sense to anyone who isn't part of that field. Too much jargon can confuse newcomers, so use simple language and define all potentially unfamiliar terms here.
- Share your project with others - describe what you are doing now and what you want to do in the future.
- Tell people who you are and how you can be contacted.

NOTE: If you're having trouble getting started, it's a good idea to look at other peoples' README .md files.

If you can't get your raw markdown content to render in the way you want, it is also a good idea to find a file that has what you want and then look at the raw file. You can copy and paste other people's raw file content into your README .md file and then edit it.

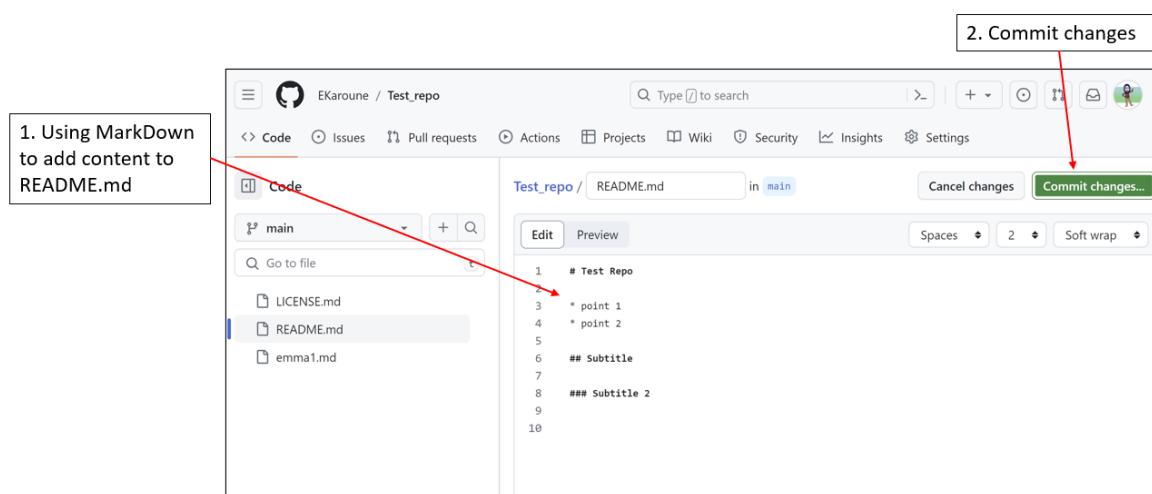
Here is an example of a really well formatted README .md file: [STEMM Role Models App](#)

If you click the link above, it will take you to their README file. You can use this as a template for your README .md file.

- To look at the raw markdown file you need to click on the raw button (top right of the white box).



- This takes you to the markdown raw file that is rendered into a nicely formatted `README.md` file on GitHub.
- Now just copy and paste it into your `README.md` edit tab. You can now edit this for your project.
- Remember to check what it looks like by clicking on the preview changes tab.
- When you have finished editing, you need to scroll down to the bottom of the page and press the green commit changes button.



Annotated diagram of `README.md` file in edit mode – with some markdown added.

- 1. Using MarkDown to add content to `README.md`: the Markdown (denoted by '.md' in the file extension) text for your `README.md` file. This example shows the template file that has different sections (headers and subheaders are created by using one or more of '#' symbol. See the [formatting consistency section of the Community Handbook](#) for some more information on using Markdown.

Committing - or saving - your changes

Committing your changes is like hitting the “save button” for a file. GitHub will not automatically save your changes, so it’s important not to skip this step.

Whatever changes you have made in the file will be deposited into your repository.

It is good practice to write a descriptive commit title and a short description of what you have done in the commit changes box. So something like - commit title: ‘first edit of the readme file’; description: ‘copied template from ... and edited it with the details of this project’. This information about the commit is called a “commit message”, and the commit title will enable you to quickly look through the history of changes for a file (which is why making them descriptive is so important - it’s like leaving a helpful note to your future self).

You can see a list of your commits (or your “commit history”) by clicking the clock symbol on your landing page or within the page for each file.



Add a license to your repository

It is important that all of your work has a license from the very beginning or no one can reuse it. Licenses tell other researchers how they are able to reuse, modify and remix your work. No license implies that others are *not* allowed to use your work, even with attribution. So it is better to include a license that lets people know what they can and can't do and how to give you credit for your work.

Depending on your field, much of your work may be documents with only some data or code. The standard licenses offered on GitHub are most appropriate for software and won't really be the right kind for documents.

Creative commons licenses are the best to use for this purpose, and the most open of these is the CC BY 4.0.

To add a license to your repository, the first thing to do is create a `LICENSE.md` file:

- To do this, click on the Add file button, and click create file. This will give you a blank file.
- Then, you need to name the file, so call it `LICENSE.md`. This makes it into a markdown file.
- You can find all the creative commons licenses in the link above so copy the text of the license you want and then paste it into this file.
- Don't forget to press the green commit new file button at the bottom and write a commit message to describe what you have done.
- You can also add a link to the license to the bottom of your `README.md` file. Here is a link to a repository that you can copy to add in a [CC BY 4.0 license](#). It has a text file for your `LICENSE.md` file and also a shield (or badge) that you can put at the bottom of your `README.md` file.

You can find more information about licenses in the [Licensing](#) chapter of The Turing Way.

Licence

Find this resource on Zenodo at DOI: <https://doi.org/10.5281/zenodo.8221283>

These materials (unless otherwise specified) are available under the Creative Commons Attribution 4.0 Licence. Please see the [human-readable summary](#) of the CC BY 4.0 and the full [legal text](#) for further information.

