Version control- GitHub

What is a ‘Version Control System?

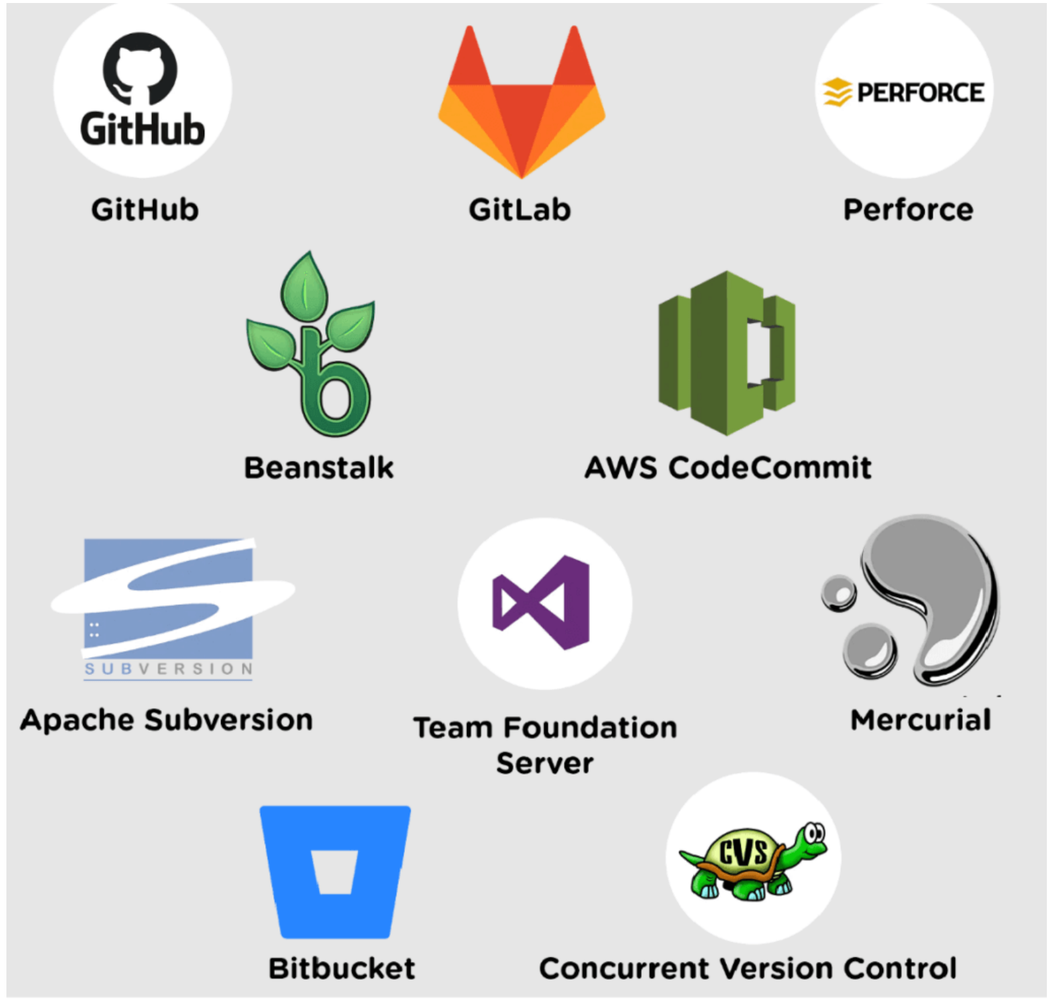
- A **version control system** allows users to **keep track of the changes** in software development projects, and collaborate easier on projects.

  - Collaborate on code and separate their tasks through **branches**
  - Combine the code changes when required
  - View the history of changes
  - Go back to the previous version(s)

Why are ‘Version Control Systems’ used?

- **Advantages include:**
  - Streamlining the development process
  - Management of code for multiple projects
  - Keeping a history of all changes within code
- Often, Version Control Systems can be integrated with an IDE (Integrated Development Environment) such as Visual Studio Code

‘Version Control Systems’



Top 10 Version Control Systems:

1. **GitHub**
2. GitLab
3. Beanstalk
4. PerForce
5. Apache Subversion
6. AWS CodeCommit
7. Microsoft Team Foundation Server
8. Mercurial
9. CVS Version Control (Concurrent Versions System)
10. Bitbucket

Types of GitHub accounts

With GitHub, you can store and collaborate on code. Accounts allow you to organise and control access to that code. There are three types of accounts on GitHub.

- **Personal accounts** - Every person who uses GitHub signs into a personal account.
- **Organisation accounts** - Enhances collaboration between multiple personal accounts
- **Enterprise accounts** - allows central management of multiple organisations.

Personal accounts

- Using your personal account, you can own resources such as repositories, packages, and projects.
- Each personal account uses either GitHub Free or GitHub Pro.
- Most people will use one personal account for all their work on GitHub.com, including both open-source projects and paid employment.
- If you're currently using more than one personal account that you created for yourself, it is recommended that both accounts are combined.

Organisation accounts

- Organisations are shared accounts where an unlimited number of people can collaborate across many projects at once.
- Like personal accounts, organisations can own resources such as repositories, packages, and projects. However, you cannot sign into an organisation.
  - Instead, each person signs into their own personal account, and any actions the person takes on organisation resources are attributed to their personal account.
  - Each personal account can be a member of multiple organisations.

Enterprise accounts

GitHub Enterprise Cloud and GitHub Enterprise Server include enterprise accounts, which allow administrators to centrally manage policy and billing for multiple organisations and enable inner sourcing between the organisations.

About repositories

- A repository contains all of your project's files and each file's revision history. You can discuss and manage your project's work within the repository.
- You can own repositories individually, or you can share ownership of repositories with other people in an organisation.
- You can restrict who has access to a repository
- For user-owned repositories, you can give other people collaborator access so that they can collaborate on your project.
- If a repository is owned by an organisation, you can give organisation members access permissions to collaborate on your repository.

Permission levels for a repository

- A repository owned by a personal account has two permission levels:
  - Repository owner
  - Collaborators.
- Repositories owned by personal accounts have one owner.
- Ownership permissions can't be shared with another personal account.
- You can also invite users on GitHub to your repository as collaborators
- The repository owner has full control of the repository.

What can I use a repository for?

With GitHub Free for personal accounts and organisations, you can work with unlimited collaborators on unlimited public repositories.

- You can use issues to collect user feedback, report software bugs, and organise tasks.
- You can use GitHub Discussions to ask and answer questions, share information, make announcements, and conduct or participate in conversations about a project.
- You can use pull requests to propose changes to a repository (discussed later)
- You can use project boards to organise and prioritise your issues and pull requests.

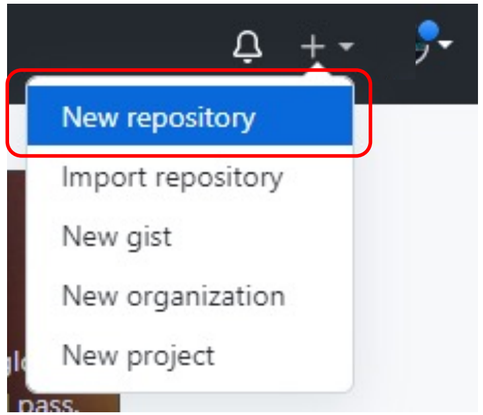
A repository visibility

You can restrict who has access to a repository by choosing a repository's visibility:

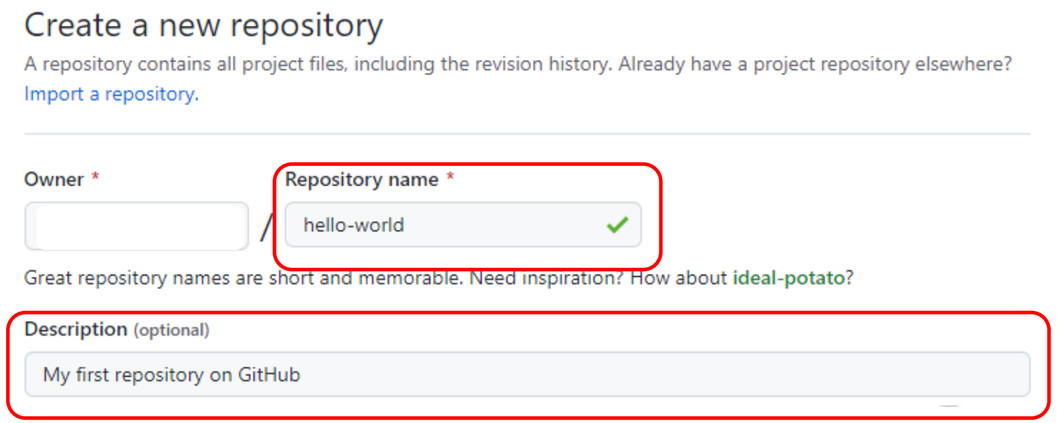
- **Public**
- **Private**.

- **Public** repositories are accessible to everyone on the internet.
- **Private** repositories are only accessible to you, people you explicitly share access with, and, for organisation repositories, certain organisation members.

Creating a repository



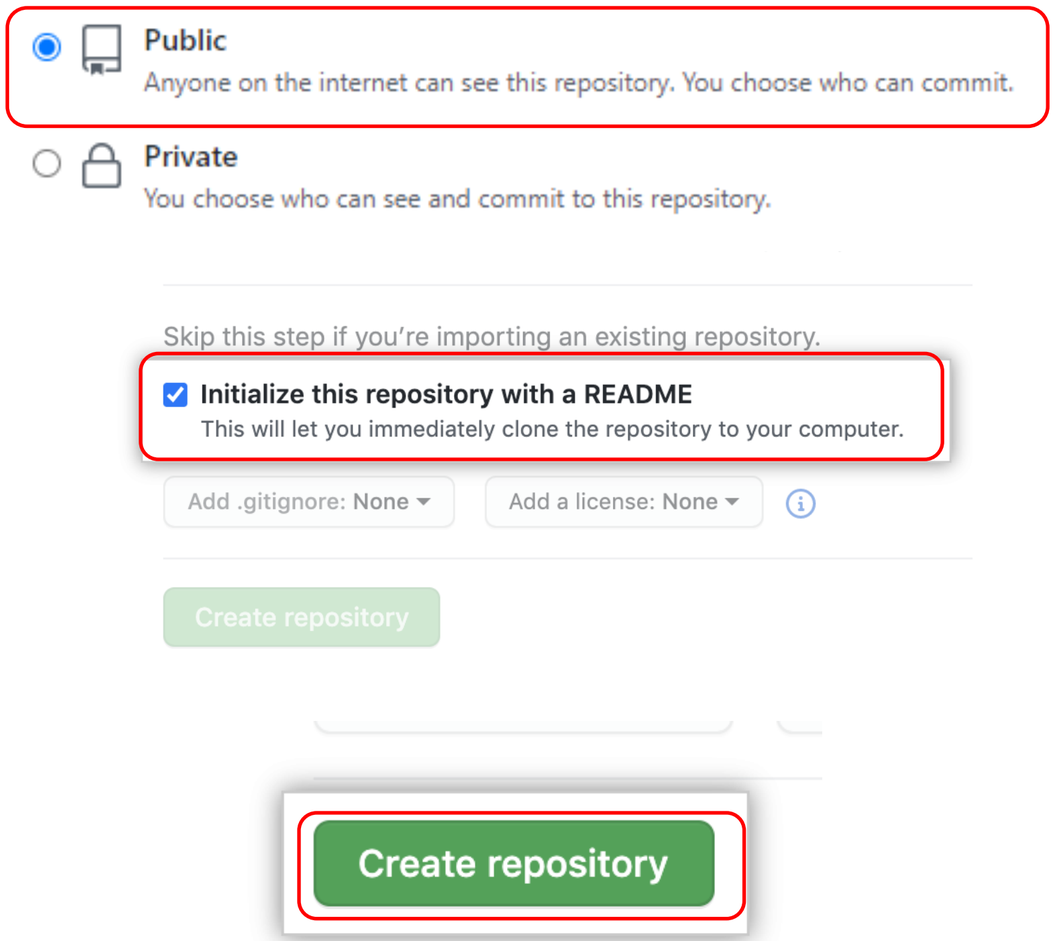
On GitHub.com, in the upper-right corner of any page, use the drop-down menu, and select **New repository**.



Type a short, memorable name for your repository. For example, "hello-world".

Optionally, add a description of your repository. For example, "My first repository on GitHub."

Creating a repository



Choose the repositories visibility to **Public**

Select **Initialize this repository with a README**.

(*Discussed on next slide*)

Click **Create repository**

About READMEs

A README file is often added to a repository to inform other people why your project is useful, what they can do with your project, and how they can use it.

A README file often includes the following information:

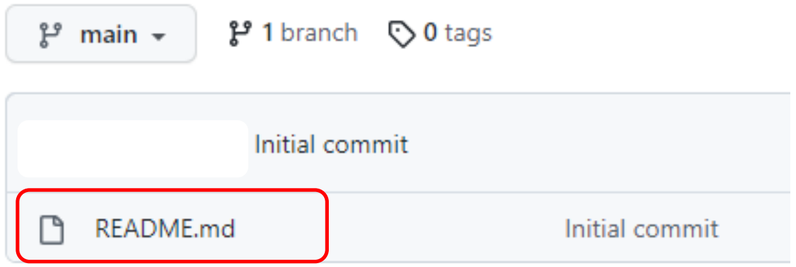
- What the project does
- Why the project is useful
- How users can get started with the project
- Where users can get help with your project
- Who maintains and contributes to the project

If the README file is saved in the hidden .github, root, or docs directory, it will automatically display the file to visitors.

Commit your first change

A commit is like a snapshot of all the files in your project at a particular point in time.

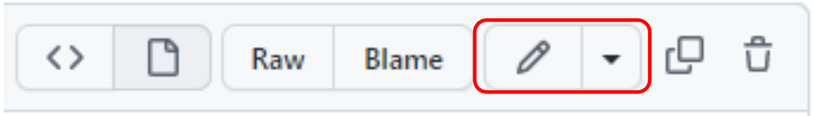
Changing the README file



Earlier, we created a new repository (hello-world) which included a README.ms file.

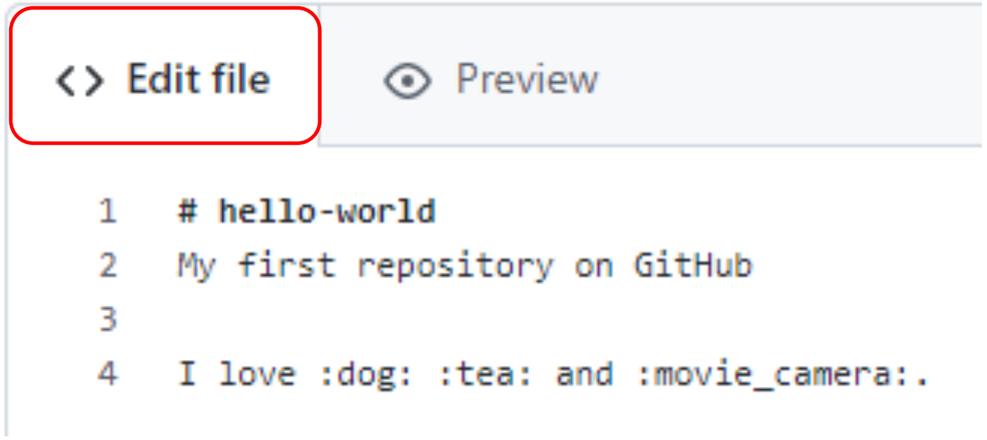
Let’s commit a change to the README file.

In your repository's list of files, click **README.md**.



Above the file's content, click the **edit icon**.

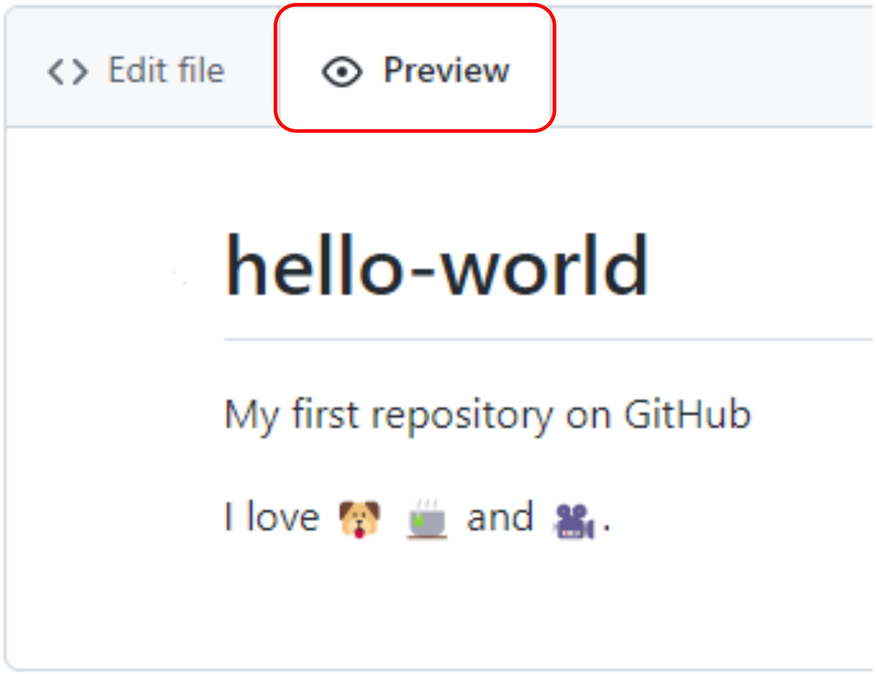
Changing the README file



Activity:

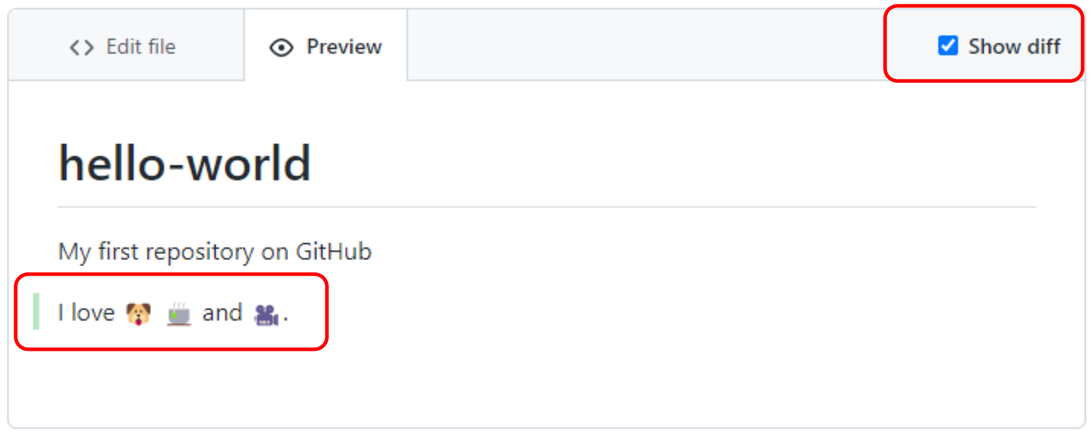
On the **Edit file** tab, type some information about yourself.

For a comprehensive list of GitHub emoji markup use:
https://gist.github.com/rxaviers/7360908



Above the new content, click **Preview**

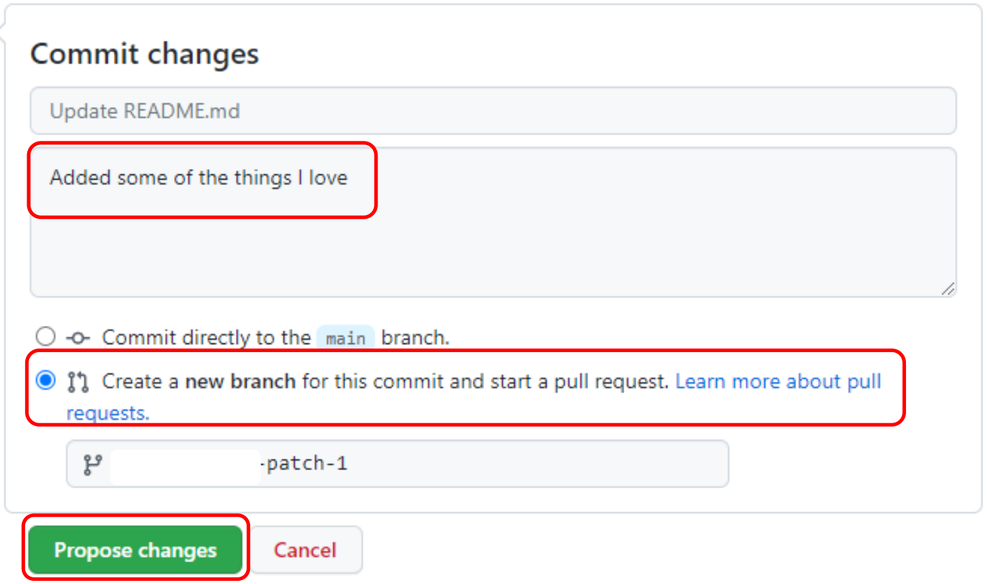
Changing the README file



Activity:

To review the changes, tick the **Show diff** button – changes will be highlighted in green.

Commit changes



Activity:

At the bottom of the page, type a short, meaningful comment that describes the change you made to the file.

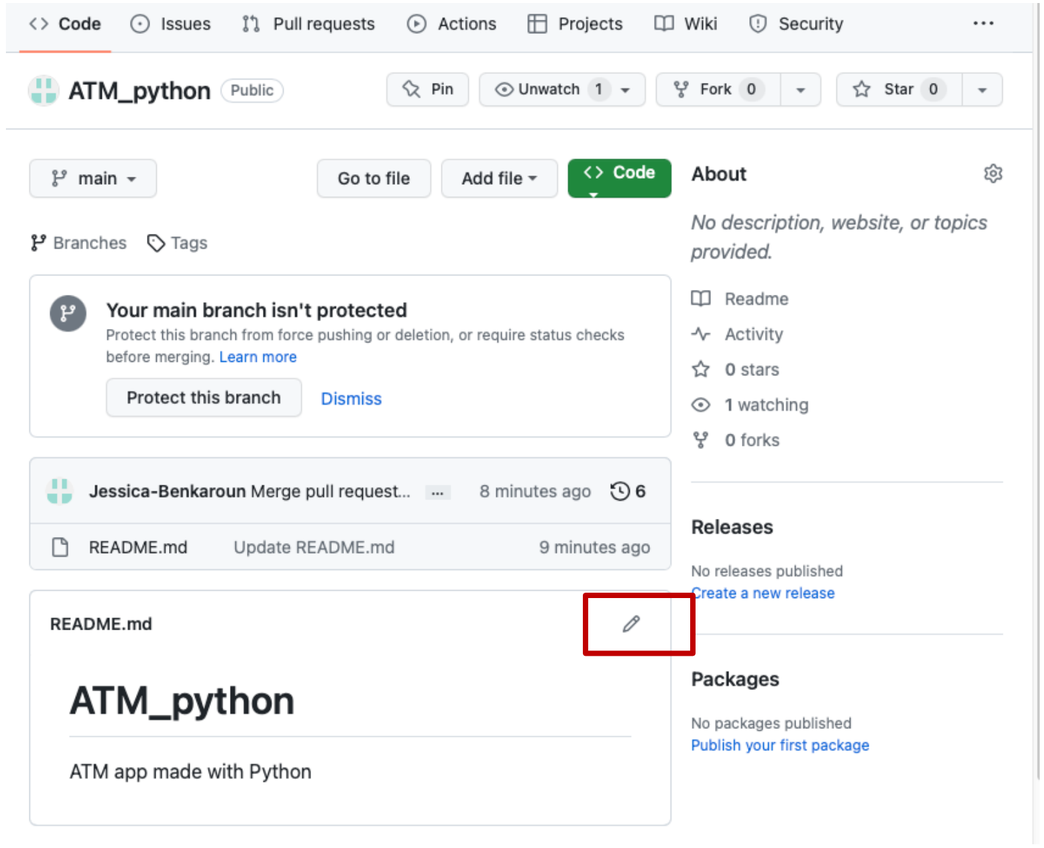
Below the commit message fields, choose to create a new branch for your commit and then create a pull request.
(Covered later in presentation)

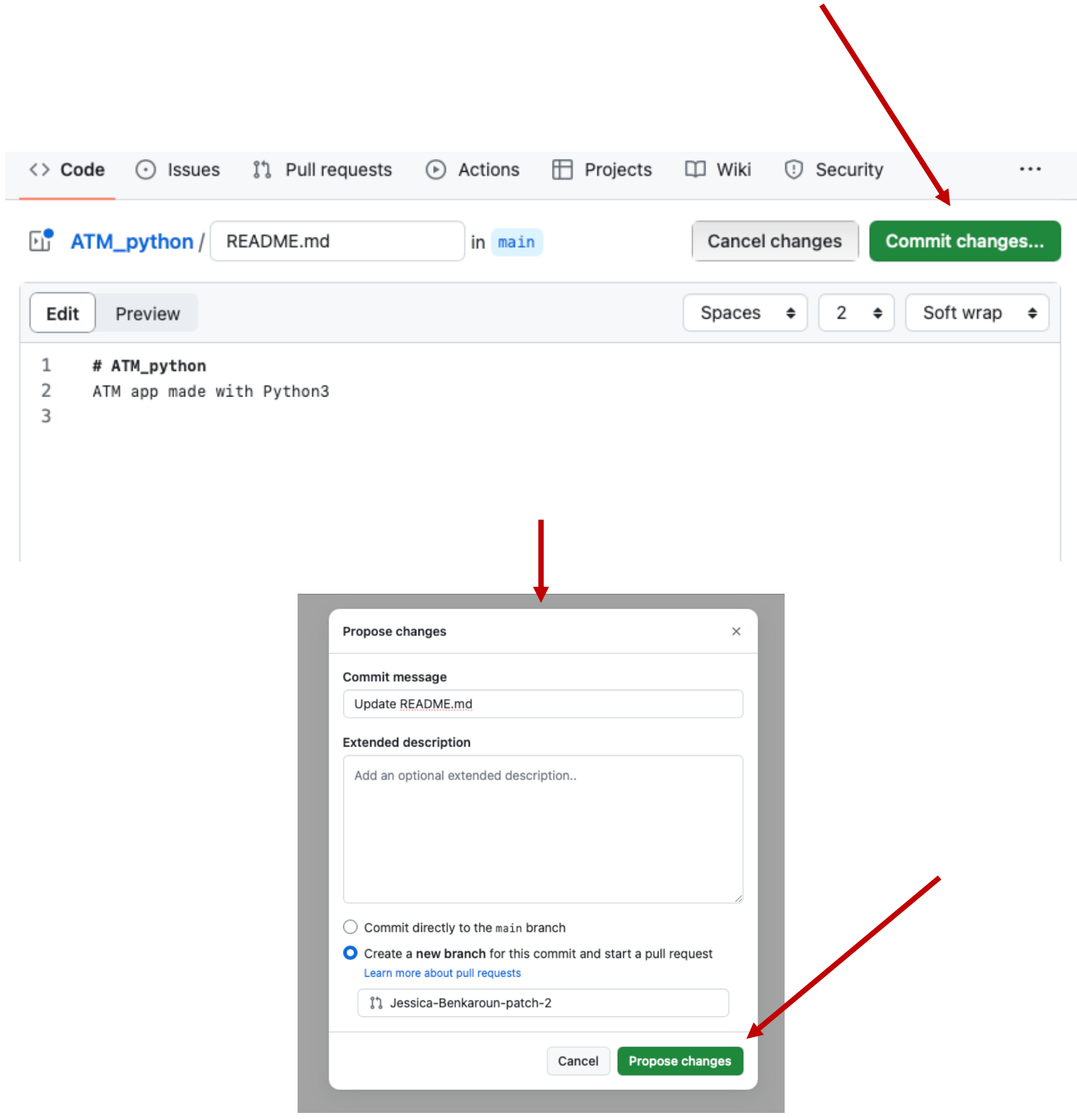
Finally, click **Propose changes**

Add and commit changes

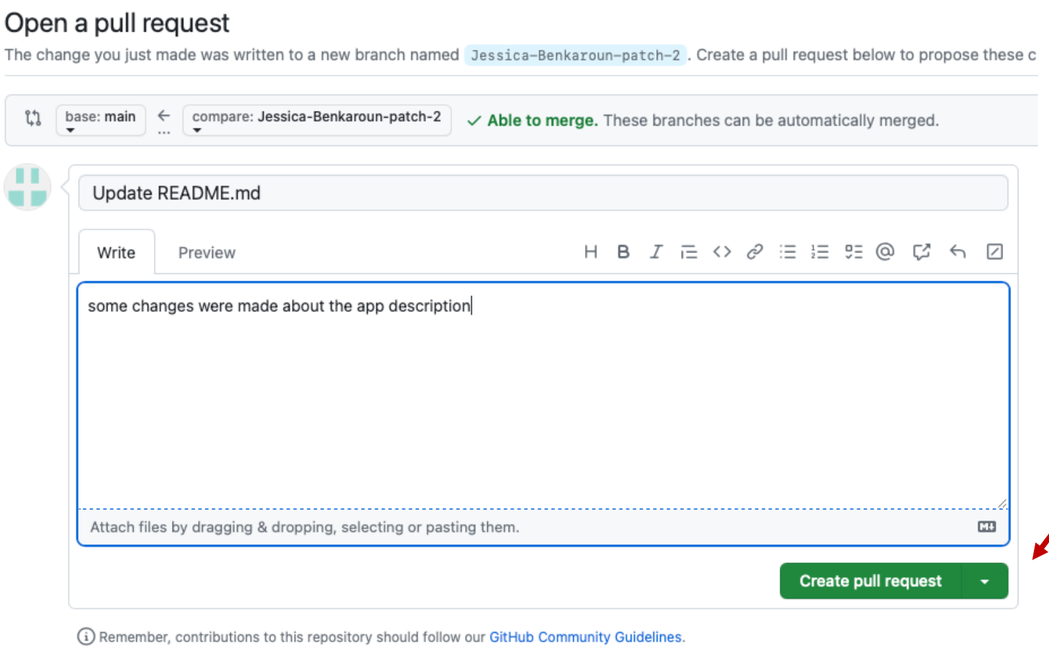
Commit changes

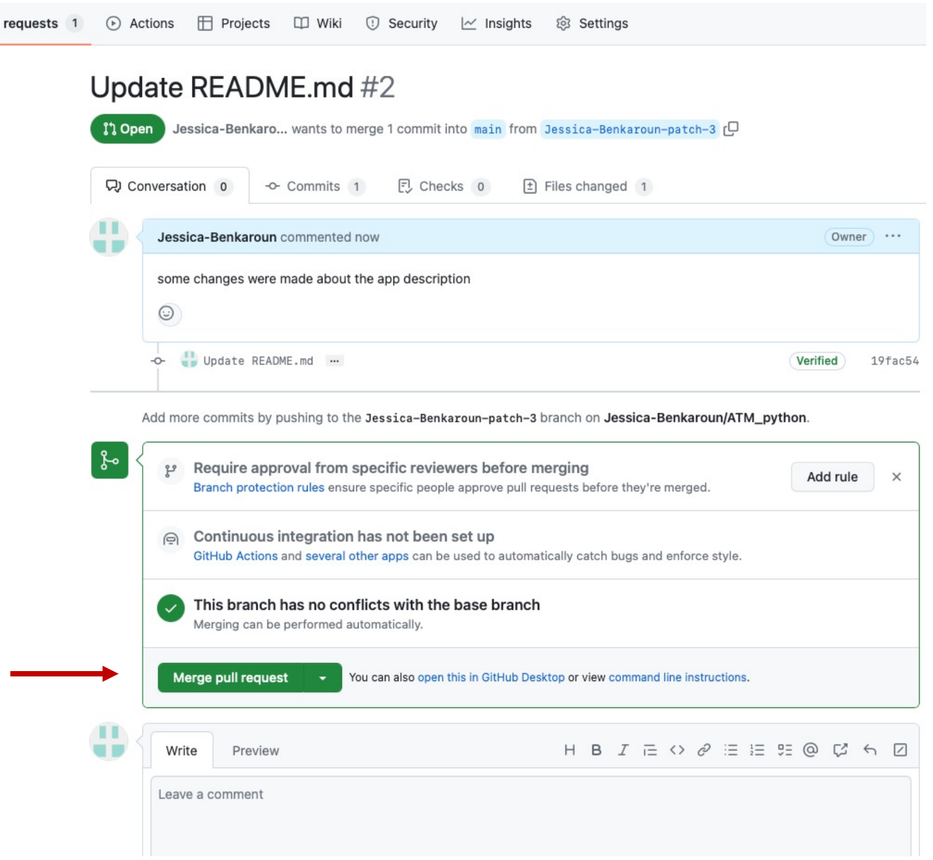
Edit the README file. Do a change in the file and Click on Commit changes.





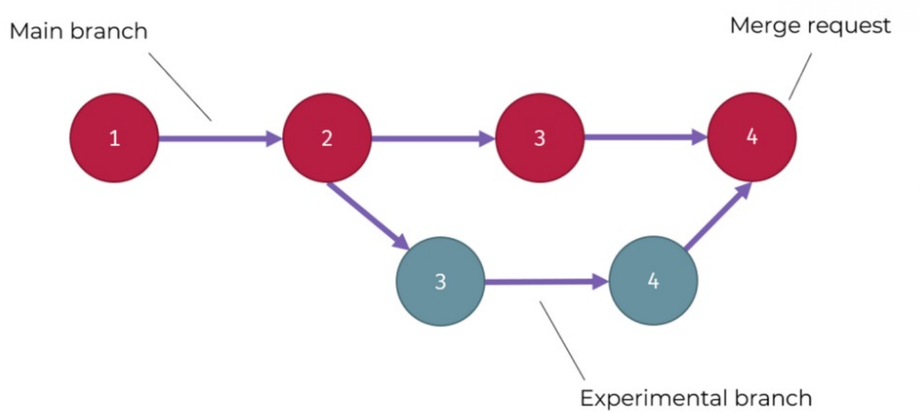
Commit changes





Merging

Merging Branches



A branch is a version of the repo that diverges from the main or master branch.

You can use this to create experimental branches to work on new features without messing with your original source code.

Once you're happy with those changes, you can merge those branches together.

Markdown text

- **Markdown** is a lightweight markup language that is commonly used to format plain text documents in a way that is easy to read and write.

- GitHub supports Markdown for formatting text in various parts of the platform, such as README files, issues, pull requests, and comments.

- See documentation here to use Markdown text on Github:
https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax