SIGDOC

What is GitHub and how do I get started with it?

Github is a platform for developers to host and share their code. It uses the Git control system (a command line tool) to enable many of its features. In addition to hosting code and providing access and control over it by using Git, GitHub also provides tools for tracking and managing bugs, creating automated deployments, creating built-in wikis for projects, and more recently, offers GitHub Copilot as an AI coding assistant.

While GitHub is popular for hosting open-source code and providing the means to collaborate on it easily, it can be used for other private, public, commercial, and educational projects as well. For example, GitHub pages can be used to host and deploy websites. You could even host all the resources for a class on GitHub and invite your students to collaborate on the curriculum resources. That being said, the layout and features are optimized for working on code.

GitHub accounts come in several tiers, including a free tier that offer all the basic functions of the platform, and paid tiers that offer additional features and storage. There are also options for commercial and educational accounts that offer additional options and control for companies or universities.

Since GitHub is designed to host code, it is suitable to host documentation as well if you use the docs-as-code approach. Many technical writers and documentation teams around the world already use GitHub to host their documentation. Writers who do not yet use GitHub to host their documentation will likely move in that direction, or at the very least, move toward a docs-as-code approach.

While you can learn more about docs-as-code by reading (insert link to docs-as-code resource), the takeaway is that GitHub is already an essential tool for many documentation teams and will only continue gaining popularity in the industry. Documentation hosted on GitHub commonly uses markup languages such as Markdown or Asciidoc.

Basic concepts and terms

When using GitHub, you might encounter several terms and concepts that are unfamiliar to you. The following list explains some of the most common terms and concepts you might encounter:

GitHub

- **Repository**: Where the source code, branches, and code history live. The repository (often shortened to repo) also contains the settings, issues, actions, and other information associated with your project.
- **Branch**: You can think of each branch as a different version of the code being hosted on a repository. A repository must have at least one branch (by default the main branch). Branches are usually a copy of an existing branch (such as the main branch) where you make changes to the code. Typically, the goal is to make changes on a separate branch to prevent problems on the main branch. Branches can also be used for other purposes. For example, you could host different versions of the same software in a single repo by using different branches for each version and maintaining them separately.
- **Commit**: Used as a verb or noun. When you apply changes, you are committing to your branch. The commit represents all your changes. This can include edits to existing files or even file creations and deletions. The commit history is tracked on GitHub so that every change that has ever been made to a file can be tracked.
- **Pull request**: If you have at least two branches, each with code that is different from each other, you can create a pull request (often shortened to PR). For example, this might be the case after you have created a new branch and made changes to the code. If you want your changes to be transferred to the main branch, creating a PR will show the differences between the two branches. In the PR, users can then review the code differences line by line, as well as comment and make suggestions.
- **Merge**: When a PR is approved by everyone involved, you can merge it. Merging a PR transfers all the changes from the source branch to the destination branch.

Git commands

- `git branch`: Check which branch you are currently on.
- `git diff`: Shows you changes.
- `git pull`: Download the latest updates for the branch that you are currently on.
- `git checkout <name>`: Switch to an existing branch.
- `git checkout -b <name>`: Create and switch to a new branch.
- `git add .`: Add all the changes you have made so far to your branch.
- `git commit -m "<message>"`: Commit your changes you made in the editor to your branch.
- `git push --set-upstream origin <name>`: Push your new branch.
- `git push`: Push your changes upstream.

Getting started

The following topics will guide you through creating an account and setting up a basic repository with a readme file. You can use GitHub by accessing the web UI with your browser, but most of the following steps can also be done by using the command line and Git.

Creating an account

You can create a GitHub account at https://github.com/join.

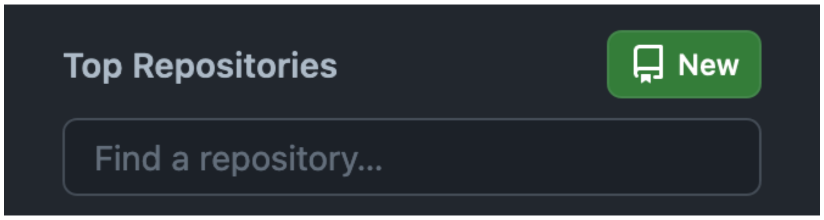
Pay special attention to your goals. What kind of projects are you working on and what do you plan to use GitHub for? Choose the type of account based on what you want to achieve.

Creating a repository

After creating and verifying your account, you are greeted with the GitHub homepage on https://github.com.

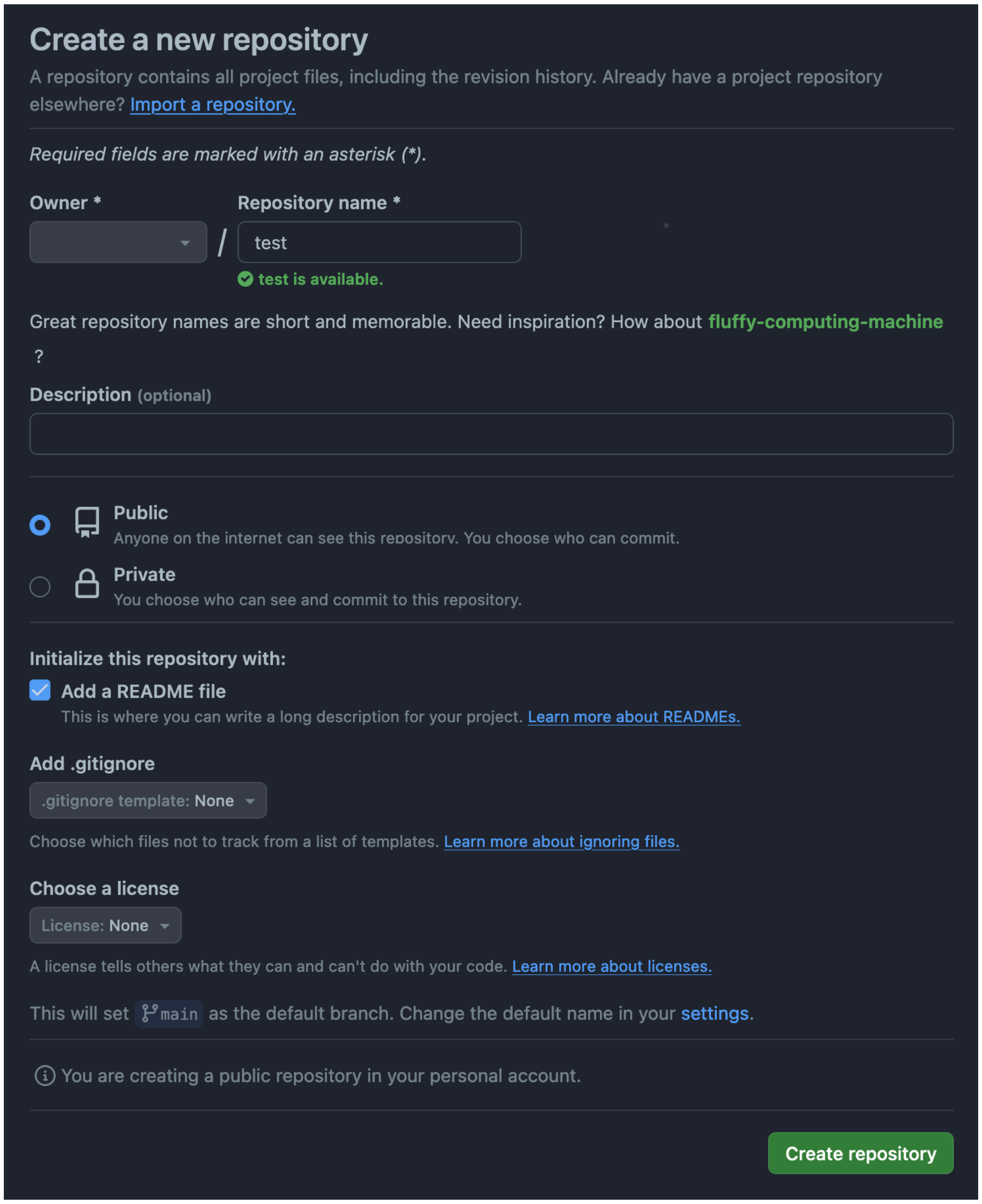
To create a new repository, complete the following steps:

1. On the homepage, click the **New** button in the left sidebar..



2. Fill out the required information:
   a. Add a repository name.
   b. Select **Add a README file**. You can ignore the other settings for this guide.

3. Click the **Create repository** button.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * / Repository name *: test

test is available.

Great repository names are short and memorable. Need inspiration? How about fluffy-computing-machine ?

Description (optional)

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

Initialize this repository with:

[✓] Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore
.gitignore template: None
Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license
License: None
A license tells others what they can and can't do with your code. Learn more about licenses.

This will set `main` as the default branch. Change the default name in your settings.

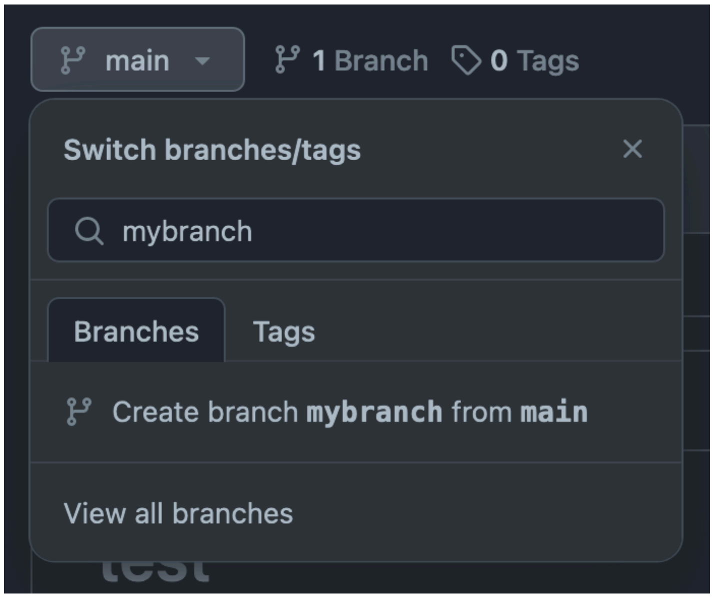
You are creating a public repository in your personal account.

Creating a branch

Complete the following steps to create a new branch:

1. In your repository, select the branch button. By default, the button shows the `main` branch.

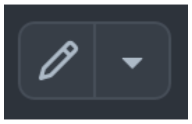
2. Type a name for your branch and click **Create branch <name> from main**. Your new branch is based on the branch that is currently selected.
3. Verify that you are on your new branch by checking the branch button.



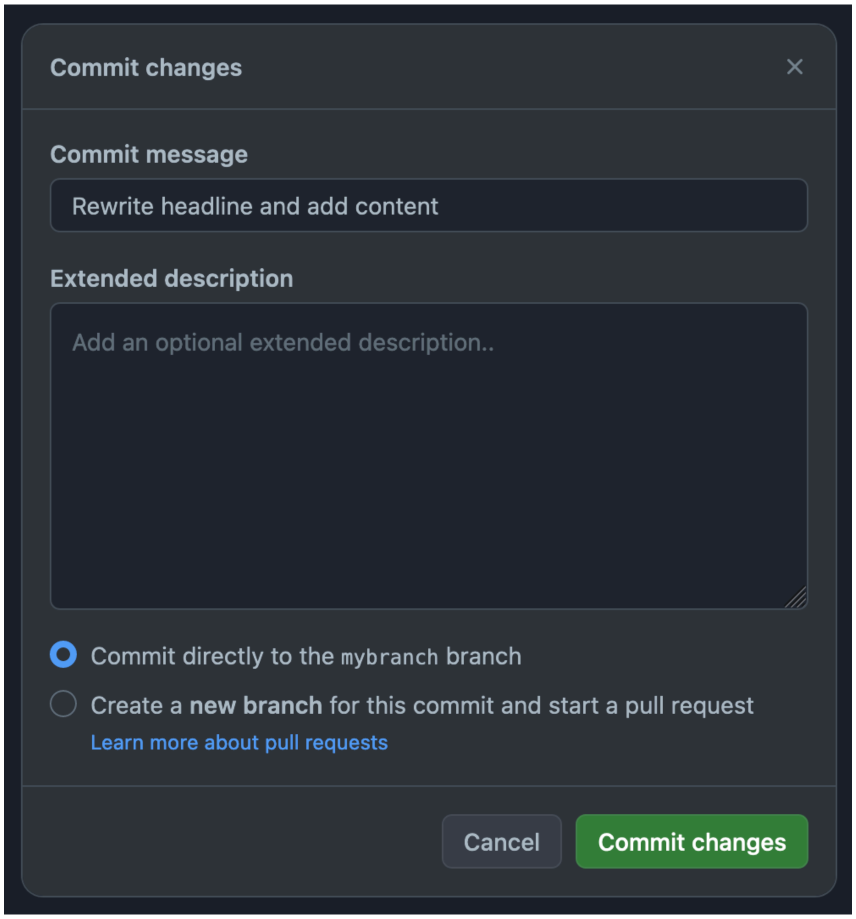
Commiting changes to a file

To practice creating a PR by adding content to the empty readme file, complete the following steps:

1. In the repository and on your new branch, click **README.md.**
2. Click the edit button and make changes to the file.



3. Click **Commit changes..** and add a commit message. Usually the commit message describes what changes you have made.
4. Click **Commit changes** to push your changes to the file on your branch.
5. Verify that your changes are committed by switching between your branch and the main branch while viewing the **README.md file**.



Commit changes

Commit message

Rewrite headline and add content

Extended description

Add an optional extended description..

[✓ Commit directly to the `mybranch` branch]

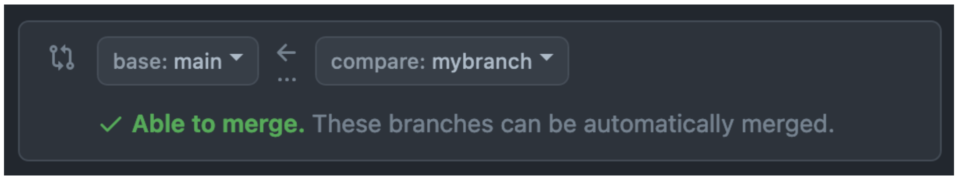
Create a **new branch** for this commit and start a pull request

Learn more about pull requests

Creating a PR

Creating a PR allows you to review changes before you merge them with your main branch. To create a PR a, complete the following steps:

1. Click the **Pull requests** tab. If you have recently pushed changes to a new branch, the **Compare & pull request** button appears automatically.
2. Click the **Compare & pull request** button. If it does not appear, you can manually
3. Make sure that the correct source and target branch are selected. The arrow points to the branch that receives the changes.

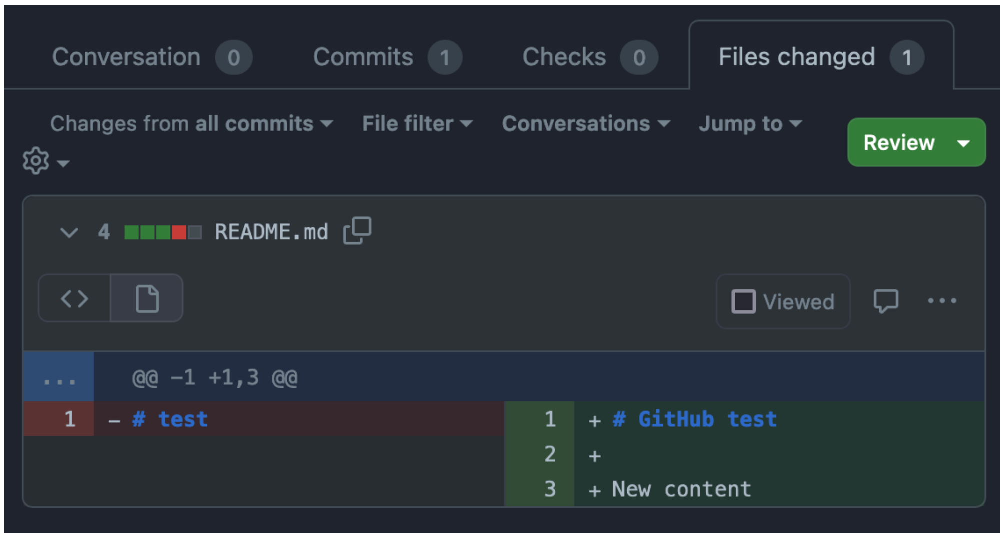


base: main ← compare: mybranch

✓ **Able to merge.** These branches can be automatically merged.

4. **Optional:** Add a title and description for your PR. In most cases, the title automatically uses the commit message that you previously used when working on your branch.
5. Click **Create pull request**.

6. Click the **Files changed** tab to view the diffs between the two branches you are trying to merge. You and other collaborators can also review, approve, leave comments, or make suggestions on the changes that you want to merge.



```
@@ -1 +1,3 @@
1  - # test          1  + # GitHub test
                    2  +
                    3  + New content
```

7. In the **Conversation** tab, click **Merge pull request > Confirm merge** when you are ready to push the changes from your branch to the main branch.

Example repo

(would be good to have an example repo with a basic doc structure and self-descriptive content in Markdown or similar, with branches, some open issues etc.)

Additional resources

https://docs.github.com/
https://git-scm.com/doc
https://www.lenovo.com/us/en/glossary/markup-language/
https://www.markdownguide.org/
https://asciidoc.org/
https://www.freecodecamp.org/news/git-cheat-sheet-and-best-practices-c6ce5321f52/
https://reflectoring.io/upstream-downstream/

Credit

Git commands: RHACM doc team
Github terms/concepts: RHACM doc team

Thank You!

See the Committee on Structured Authoring and Content Management page of the ACM SIGDOC website to learn more about committee activities, available resources, and volunteer opportunities.

See https://acm-sigdoc-structured.org to learn more about committee activities, available resources, and volunteer opportunities.