5.1 GITHUB INTRODUCTION

GitHub is a cloud-based platform for version control and collaborative software development. It hosts Git repositories and provides a web-based interface and additional features for managing projects and collaborating with others.

Key aspects of GitHub include:

Version Control: GitHub leverages Git, a distributed version control system, to track changes to code and other files within a project. This allows developers to revert to previous versions, manage different branches of development, and understand the history of changes.

Collaboration: GitHub facilitates teamwork by enabling multiple developers to work on the same project simultaneously. Features like pull requests, issue tracking, and code reviews streamline the collaborative process, ensuring quality control and effective communication.

Repository Hosting: It provides a central location to store and share code repositories, making it accessible to team members and the wider open-source community.

Project Management: GitHub offers tools for managing projects, including features for bug tracking, task management, and wikis, which help organize development workflows.

Community and Open Source: GitHub is a prominent platform for open-source projects, fostering a large community of developers who contribute to and collaborate on a vast array of software.

In essence, GitHub acts as a central hub for developers to store, manage, and collaborate on code, making it an indispensable tool in modern software development.

5.1.1 CREATE GITHUB ACCOUNT AND CREATE REPOSITORY

Step 1: Open the web site www.github.com on some web browser of your choice. Enter the signup option.

Step 2: Enter your Email ID. Then create a password for this account. Click the continue button.

Step 3: Enter the username that will appear as a username in your Github account . Click on continue.

Step 4: Then a simple puzzle appears, to verify that you are a human being. Just solve it and continue.

Step 5: Then code will be sent to your email account (which you given in step 2), enter the code to get verified. Then click on continue for free. That’s it, the account gets created.

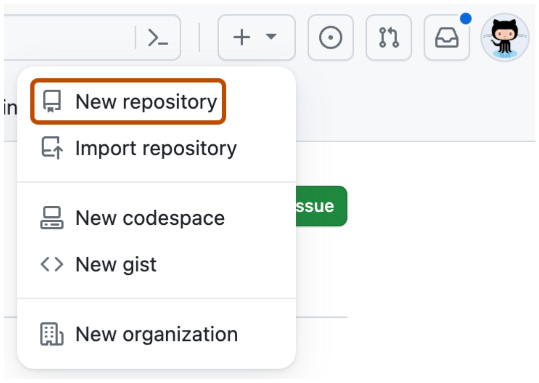
Create a Repository:

The first thing we'll do is create a repository. You can think of a repository as a folder that contains related items, such as files, images, videos, or even other folders. A repository usually groups together items that belong to the same "project" or thing you're working on.

Often, repositories include a README file, a file with information about your project. README files are written in Markdown, which is an easy-to-read, easy-to-write language for formatting plain text. We'll learn more about Markdown in the next tutorial, Setting up your profile.

GitHub lets you add a README file at the same time you create your new repository. GitHub also offers other common options such as a license file, but you do not have to select any of them now.Your hello-world repository can be a place where you store ideas, resources, or even share and discuss things with others.

- In the upper-right corner of any page, select, then click New repository.



1. In the "Repository name" box, type hello-world.

2. In the "Description" box, type a short description. For example, type "This repository is for practicing the GitHub Flow."
3. Select whether your repository will be **Public** or **Private**.
4. Select **Add a README file**.
5. Click **Create repository**.

Step 2: <u>Create a branch:</u>

Branching lets you have different versions of a repository at one time.

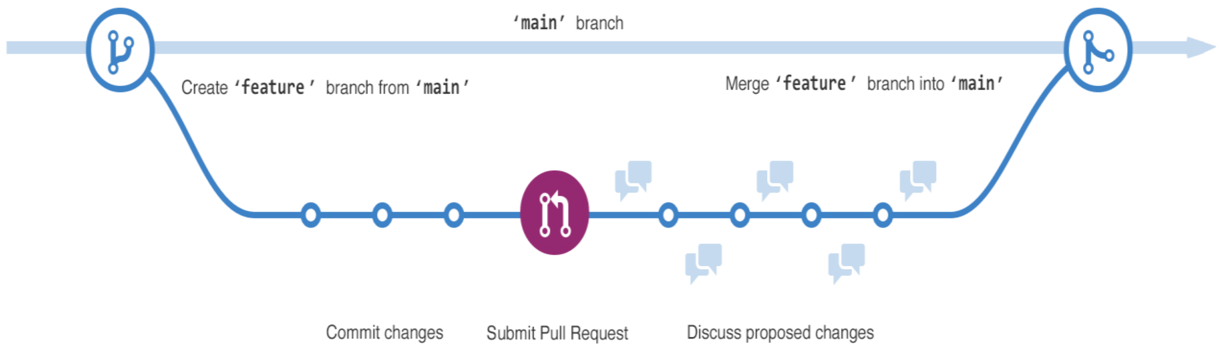
By default, your repository has one branch named main that is considered to be the definitive branch. You can create additional branches off of main in your repository.

Branching is helpful when you want to add new features to a project without changing the main source of code. The work done on different branches will not show up on the main branch until you merge it, which we will cover later in this guide. You can use branches to experiment and make edits before committing them to main.

When you create a branch off the main branch, you're making a copy, or snapshot, of main as it was at that point in time. If someone else made changes to the main branch while you were working on your branch, you could pull in those updates.

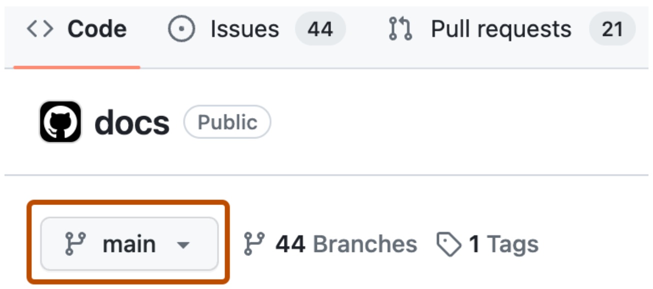
This diagram shows:

- The main branch
- A new branch called feature
- The journey that feature takes through stages for "Commit changes," "Submit pull request," and "Discuss proposed changes" before it's merged into main

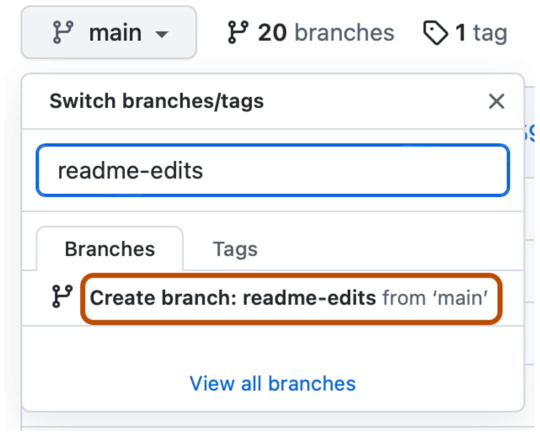


Creating a Branch:

1. Click the **Code** tab of your hello-world repository.
2. Above the file list, click the dropdown menu that says **main**.



1. Type a branch name, readme-edits, into the text box.
2. Click **Create branch: readme-edits from main**.



Now you have two branches, main and readme-edits. Right now, they look exactly the same. Next you'll add changes to the new readme-edits branch.

Step 3: Make and commit changes

When you created a new branch in the previous step, GitHub brought you to the code page for your new readme-edits branch, which is a copy of main.
You can make and save changes to the files in your repository. On GitHub, saved changes are called commits. Each commit has an associated commit message, which is a description explaining why a particular change was made. Commit messages capture the history of your changes so that other contributors can understand what you’ve done and why.

1. Under the readme-edits branch you created, click the README.md file.
2. To edit the file, click .
3. In the editor, write a bit about yourself.
4. Click **Commit changes**.
5. In the "Commit changes" box, write a commit message that describes your changes.
6. Click **Commit changes**.

These changes will be made only to the README file on your readme-edits branch, so now this branch contains content that's different from main.

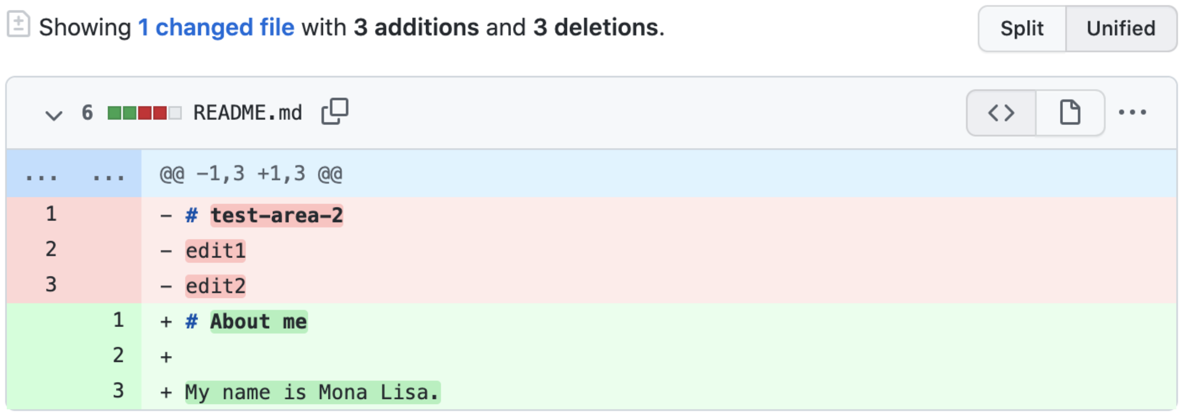
Step 4: Open a pull Request

Now that you have changes in a branch off of main, you can open a pull request.
Pull requests are the heart of collaboration on GitHub. When you open a pull request, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show diffs, or differences, of the content from both branches. The changes, additions, and subtractions are shown in different colors.

As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.

In this step, you'll open a pull request in your own repository and then merge it yourself. It's a great way to practice the GitHub flow before working on larger projects.

1. Click the **Pull requests** tab of your hello-world repository.
2. Click **New pull request**.
3. In the **Example Comparisons** box, select the branch you made, readme-edits, to compare with main (the original).
4. Look over your changes in the diffs on the Compare page, make sure they're what you want to submit.



Showing 1 changed file with 3 additions and 3 deletions.

```
@@ -1,3 +1,3 @@
- # test-area-2
- edit1
- edit2
+ # About me
+
+ My name is Mona Lisa.
```

5. Click **Create pull request**.
6. Give your pull request a title and write a brief description of your changes. You can include emojis and drag and drop images and gifs.
7. Click **Create pull request**.

Reviewing a Pull Request:

When you start collaborating with others, this is the time you'd ask for their review. This allows your collaborators to comment on, or propose changes to, your pull request before you merge the changes into the main branch.

We won't cover reviewing pull requests in this tutorial, but if you're interested in learning more, see About pull request reviews. Alternatively, try the GitHub Skills "Reviewing pull requests" course.

Step 5: Merge your Pull request:

In this final step, you will merge your readme-edits branch into the main branch. After you merge your pull request, the changes on your readme-edits branch will be incorporated into main.

Sometimes, a pull request may introduce changes to code that conflict with the existing code on main. If there are any conflicts, GitHub will alert you about the conflicting code and prevent merging until the conflicts are resolved. You can make a commit that resolves the conflicts or use comments in the pull request to discuss the conflicts with your team members.

In this walk-through, you should not have any conflicts, so you are ready to merge your branch into the main branch.

1. At the bottom of the pull request, click **Merge pull request** to merge the changes into main.
2. Click **Confirm merge**. You will receive a message that the request was successfully merged and the request was closed.
3. Click **Delete branch**. Now that your pull request is merged and your changes are on main, you can safely delete the readme-edits branch. If you want to make more changes to your project, you can always create a new branch and repeat this process.
4. Click back to the **Code** tab of your hello-world repository to see your published changes on main.