MINISTRY OF EDUCATION
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY
COLLEGE OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

وزارة التعليم
جامعة الإمام عبدالرحمن بن فيصل
كلية علوم الحاسب وتقنية المعلومات

جامعة الإمام عبدالرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

Getting Started with Git and GitHub

@2026

Table of Contents

About GitHub and Git

About GitHub

GitHub is a cloud-based platform where you can store, share, and work together with others to write code.

Storing your code in a "repository" on GitHub allows you to:

- **Showcase or share** your work.
- **Track and manage** changes to your code over time.
- Let others **review** your code, and make suggestions to improve it.
- **Collaborate** on a shared project, without worrying that your changes will impact the work of your collaborators before you're ready to integrate them.

Collaborative working, one of GitHub's fundamental features, is made possible by the open-source software, Git, upon which GitHub is built.

About Git

Git is a version control system that intelligently tracks changes in files. Git is particularly useful when you and a group of people are all making changes to the same files at the same time.

Typically, to do this in a Git-based workflow, you would:

- **Create a branch** off from the main copy of files that you (and your collaborators) are working on.
- **Make edits** to the files independently and safely on your own personal branch.
- Let Git intelligently **merge** your specific changes back into the main copy of files, so that your changes don't impact other people's updates.
- Let Git **keep track** of your and other people's changes, so you all stay working on the most up-to-date version of the project.

How do Git and GitHub work together?

When you upload files to GitHub, you'll store them in a "Git repository." This means that when you make changes (or "commits") to your files in GitHub, Git will automatically start to track and manage your changes.

There are plenty of Git-related actions that you can complete on GitHub directly in your browser, such as creating a Git repository, creating branches, and uploading and editing files.

However, most people work on their files locally (on their own computer), then continually sync these local changes—and all the related Git data—with the central "remote" repository on GitHub. There are plenty of tools that you can use to do this, such as GitHub Desktop.

Once you start to collaborate with others and all need to work on the same repository at the same time, you’ll continually:

- **Pull** all the latest changes made by your collaborators from the remote repository on GitHub.
- **Push** back your own changes to the same remote repository on GitHub.

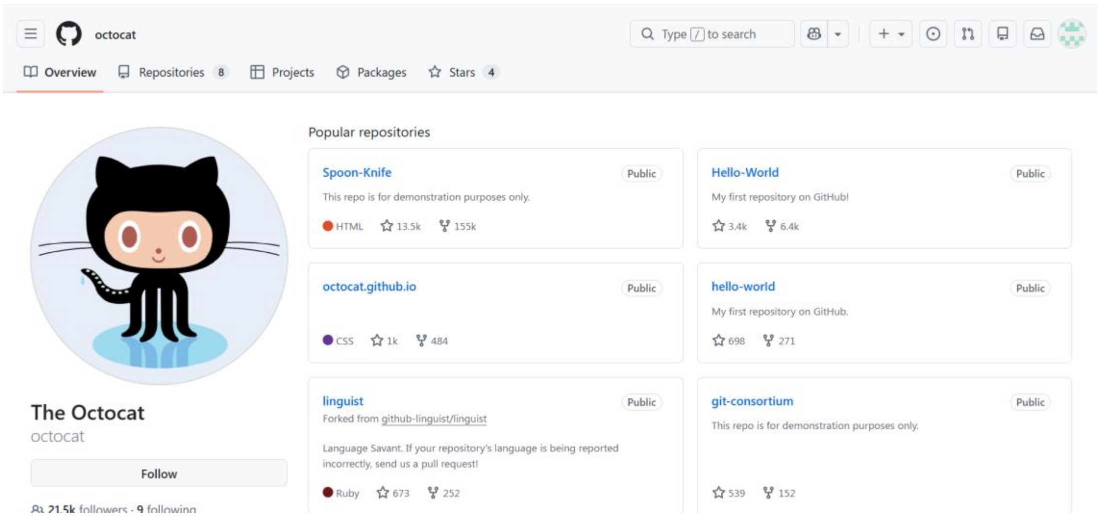
Git figures out how to intelligently merge this flow of changes, and GitHub helps you manage the flow through features such as "pull requests."

Hands-on with GitHub

Creating an account on GitHub

To get started with GitHub, you'll need to create a free personal account and verify your email address.

Every person who uses GitHub signs in to a user account. Your user account is your identity on GitHub and has a username and profile. For example, see [@octocat's profile].



Popular repositories

Spoon-Knife Public
This repo is for demonstration purposes only.
HTML 13.5k 155k

Hello-World Public
My first repository on GitHub!
3.4k 6.4k

octocat.github.io Public
CSS 1k 484

hello-world Public
My first repository on GitHub.
698 271

linguist Public
Forked from github-linguist/linguist
Language Savant. If your repository's language is being reported incorrectly, send us a pull request!
Ruby 673 252

git-consortium Public
This repo is for demonstration purposes only.
539 152

Signing up for a new personal account

1. Navigate to https://github.com/.
2. Click **Sign up**.
3. Alternatively, click on **Continue with Google** to sign up using social login.
4. Follow the prompts to create your personal account.

During sign up, you'll be asked to verify your email address.

Create your First GitHub Repository

This tutorial teaches you GitHub essentials like repositories, branches, commits, and pull requests. You'll create your own Hello World repository and learn GitHub's pull request workflow, a popular way to create and review code.

In this quickstart guide, you will:

- Create and use a repository.
- Start and manage a new branch.
- Make changes to a file and push them to GitHub as commits.
- Open and merge a pull request.

Step 1: Create a repository

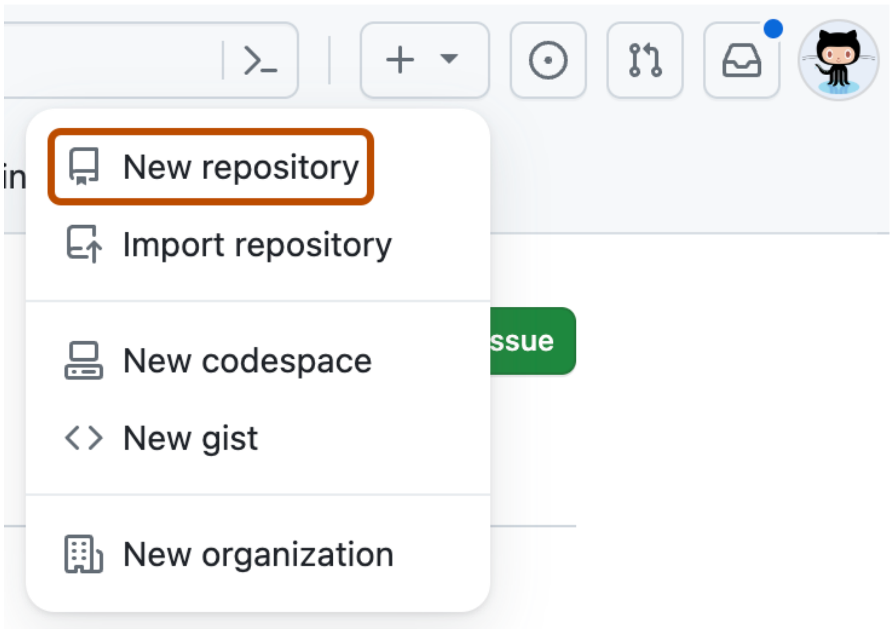
The first thing we'll do is create a repository. You can think of a repository as a **folder that contains related items, such as files, images, videos, or even other folders.** A repository usually groups together items that belong to the same "project" or thing you're working on.

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.

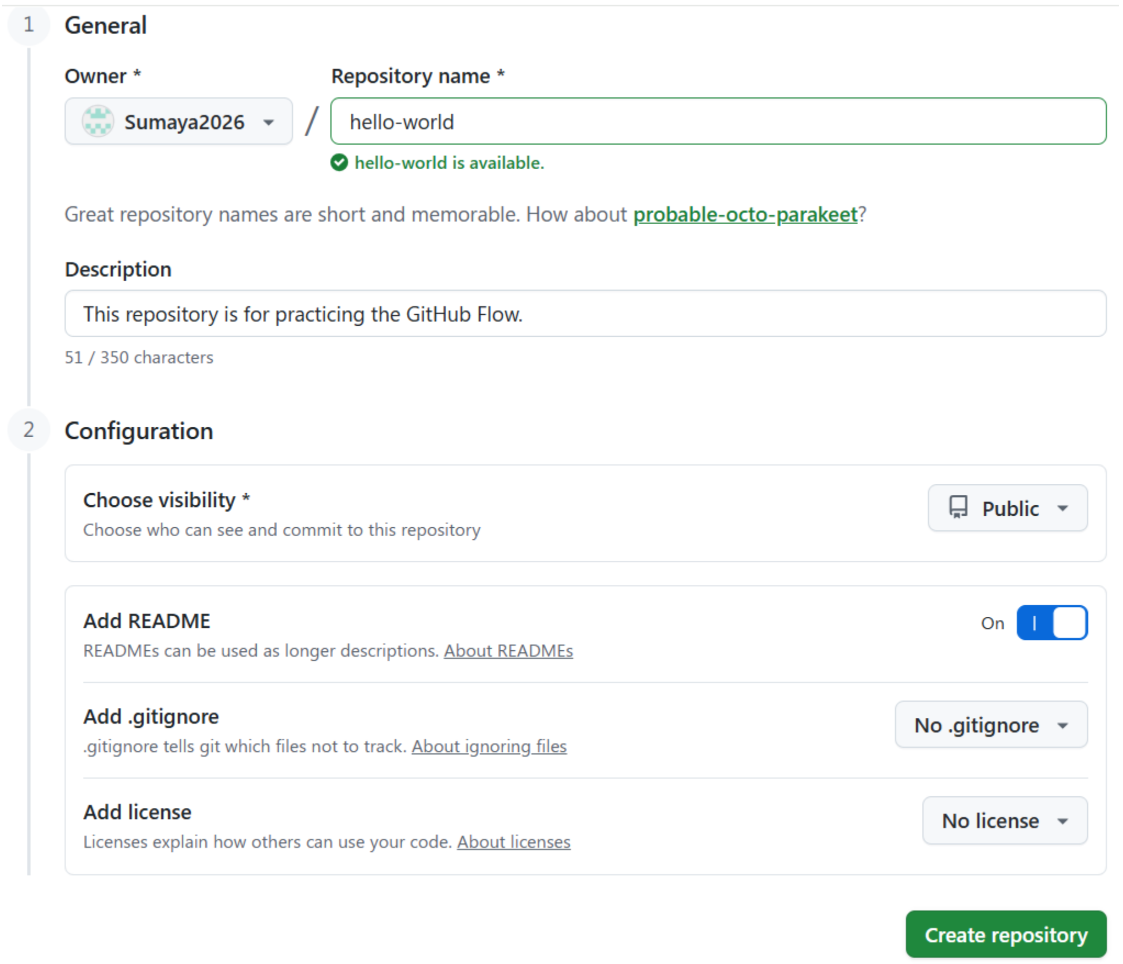
GitHub lets you add a README file at the same time you create your new repository.

Your hello-world repository can be a place where you store ideas, resources, or even share and discuss things with others.

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



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


1 General

Owner * Sumaya2026 / Repository name * hello-world

hello-world is available.

Great repository names are short and memorable. How about probable-octo-parakeet?

Description: This repository is for practicing the GitHub Flow.

51 / 350 characters

2 Configuration

Choose visibility * — Choose who can see and commit to this repository — Public

Add README — READMEs can be used as longer descriptions. About READMEs — On

Add .gitignore — .gitignore tells git which files not to track. About ignoring files — No .gitignore

Add license — Licenses explain how others can use your code. About licenses — No license

Step 2: Create a branch

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.

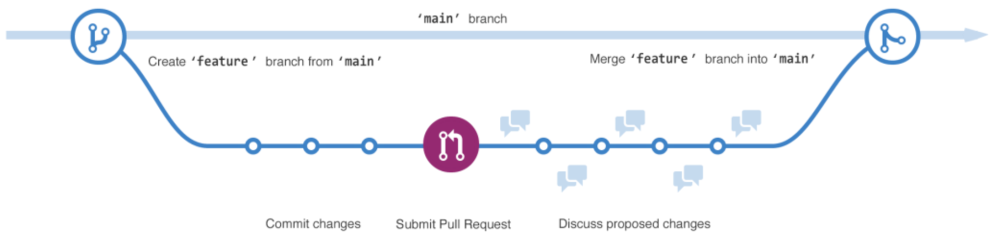
Branching is helpful when you want to add new features to a project without changing the main source of code. The work done on different branches will not show up on the main branch until you merge it, which we will cover later in this guide. You can use branches to experiment and make edits before committing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

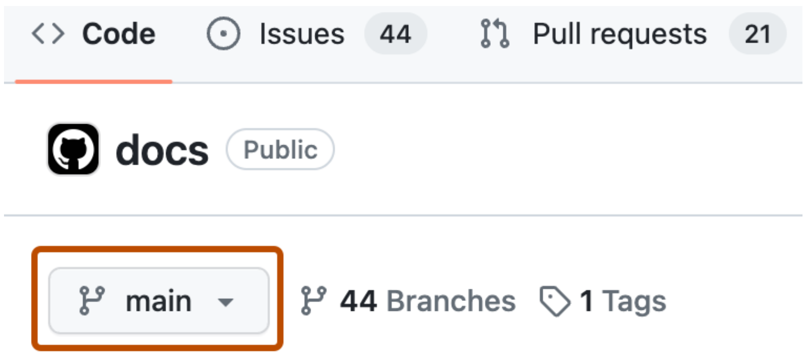
- The main branch

- 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

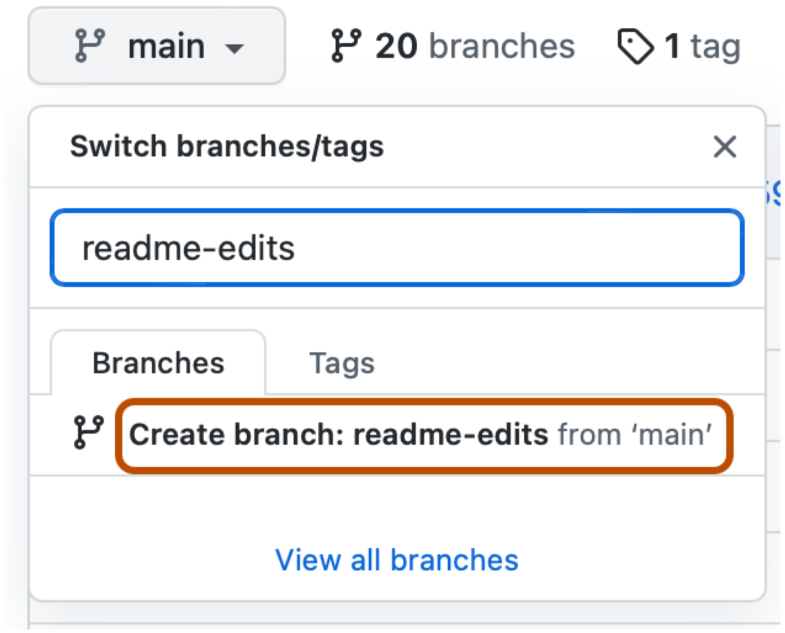


Creating a branch

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



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



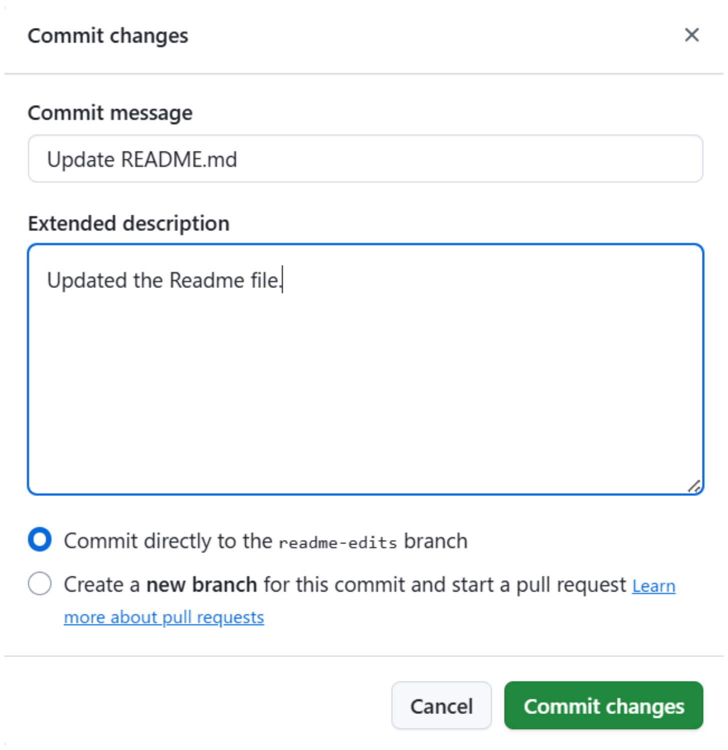
Now you have two branches, main and readme-edits. Right now, they look exactly the same. Next you'll add changes to the new readme-edits branch.

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** Commit changes... .
5. In the "Commit changes" box, write a commit message that describes your changes.
6. Click **Commit changes**.


Commit changes ×

Commit message

Update README.md

Extended description

Updated the Readme file.

[✓ Commit directly to the `readme-edits` branch]

Create a **new branch** for this commit and start a pull request Learn more about pull requests

Cancel | 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

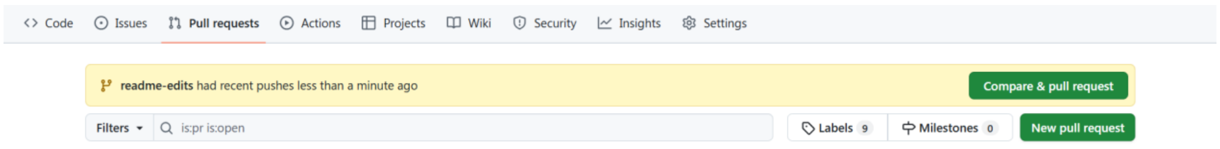
Now that you have changes in a branch off of main, you can open a **pull request**.

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.

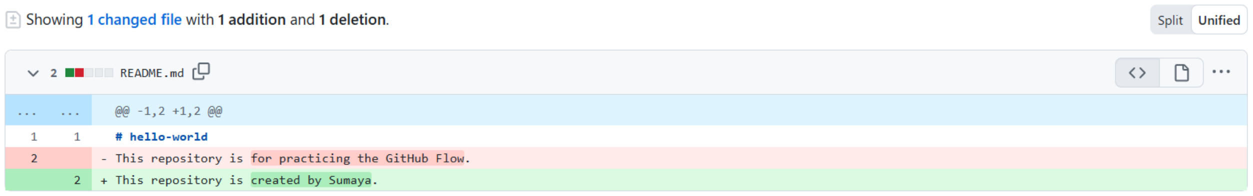
As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.

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 1 addition and 1 deletion.

```
2 README.md
...   ...   @@ -1,2 +1,2 @@
1     1     # hello-world
2           - This repository is for practicing the GitHub Flow.
      2     + This repository is created by Sumaya.
```

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**.

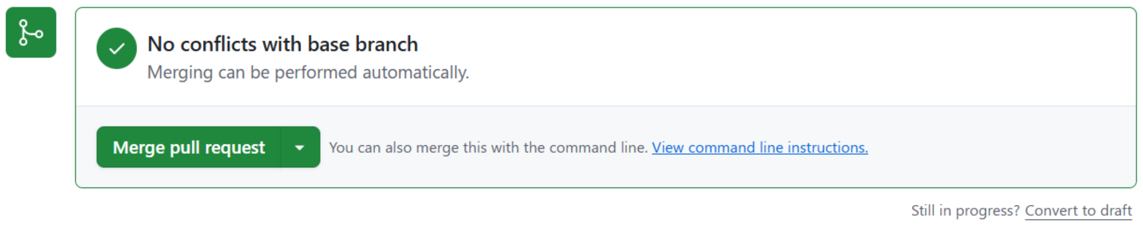
Step 5: Merge your pull request

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.

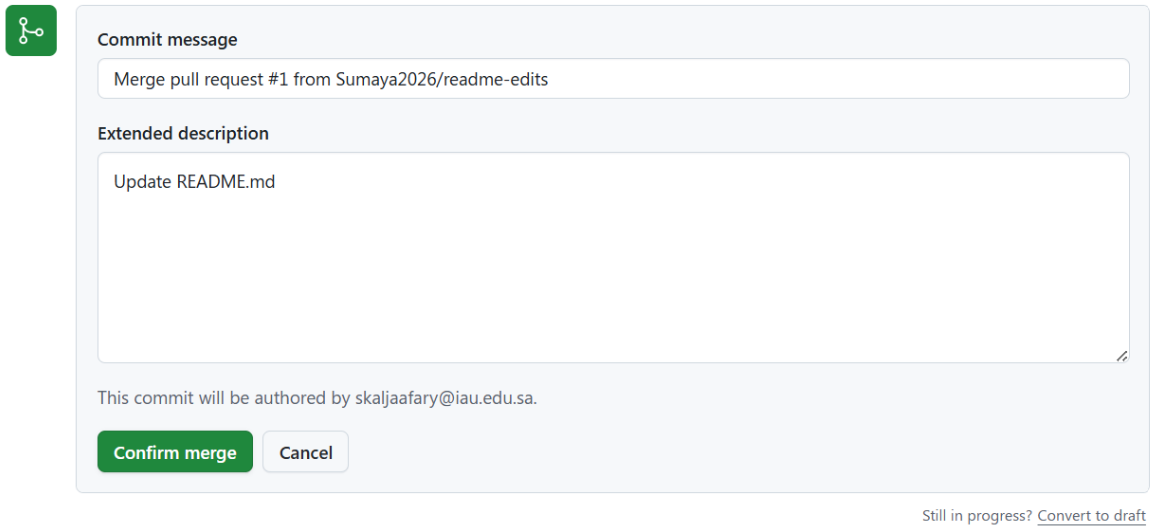
1. At the bottom of the pull request, click **Merge pull request** to merge the changes into main.


No conflicts with base branch
Merging can be performed automatically.

You can also merge this with the command line. View command line instructions.

Still in progress? Convert to draft

2. Click **Confirm merge**. You will receive a message that the request was successfully merged and the request was closed.


Commit message

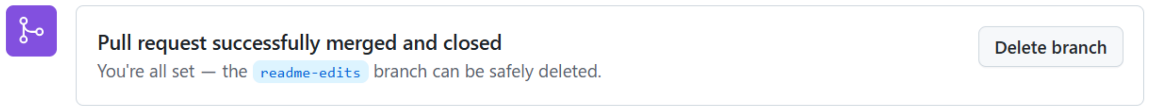
Merge pull request #1 from Sumaya2026/readme-edits

Extended description

Update README.md

This commit will be authored by skaljaafary@iau.edu.sa.

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.


Pull request successfully merged and closed

You're all set — the `readme-edits` branch can be safely deleted.

4. Click back to the **Code** tab of your hello-world repository to see your published changes on main.

Downloading files from GitHub

GitHub.com is home to millions of open-source software projects, that you can copy, customize, and use for your own purposes.

There are different ways to get a copy of a repository's files on GitHub. You can:

- **Download** a snapshot of a repository's files as a zip file to your own (local) computer.
- **Clone** a repository to your local computer using Git.
- **Fork** a repository to create a new repository on GitHub.

Each of these methods has its own use case, which we'll explain in the next section.

This tutorial focuses on downloading a repository's files to your local computer. For example, if you've found some interesting content in a repository on GitHub, downloading is a simple way to get a copy of the content, without using Git or applying versi on control.

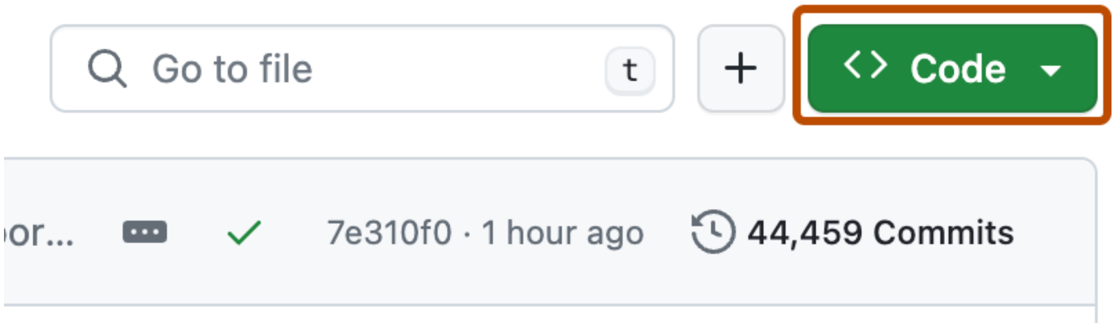
Understanding the differences between downloading, cloning, and forking

Term	Definition	Use case
Download	To save a snapshot of a repository's files to your local computer.	You want to use or customize the content of the files, but you're not interested in applying version control.
Clone	To make a full copy of a repository's data, including all versions of every file and folder.	You want to work on a full copy of the repository on your local computer, using Git to track and manage your changes. You likely intend to sync these locally-made changes with the GitHub-hosted repository. For more information, see Cloning a repository.
Fork	To create a new repository on GitHub, linked to your personal account, that shares code and visibility settings with the original repository.	You want to use the original repository's data as a basis for your own project on GitHub. Or, you want to use the fork to propose changes to the original repository. After forking the repository, you still might want to clone the repository, so that you can work on the changes on your local computer. For more information, see Fork a repository.

Downloading a repository's files

For the tutorial, we'll use a demo repository (octocat/Spoon-Knife).

1. Navigate to octocat/Spoon-Knife.
2. Above the list of files, click **Code**.



3. Click **Download ZIP**.

You now have a copy of the repository's files saved as a zip file on your local computer. You can edit and customize the files for your own purposes.

Uploading a project to GitHub

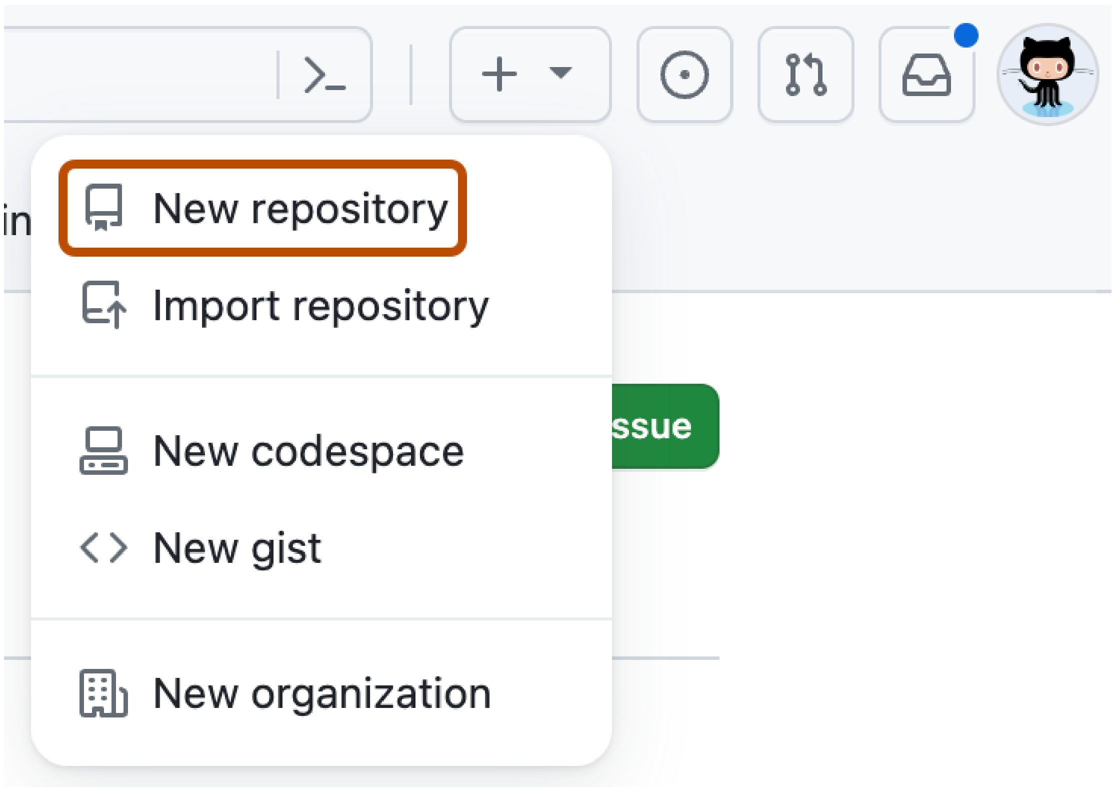
This tutorial will show you how to upload a group of files to a GitHub repository.

Uploading your files to a GitHub repository lets you:

- **Apply version control** when you make edits to the files, so your project's history is protected and manageable.
- **Back up** your work, because your files are now stored in the cloud.
- **Pin** the repository to your personal profile, so that others can see your work.
- **Share** and discuss your work with others, either publicly or privately.

Step 1: Create a new repository for your project

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



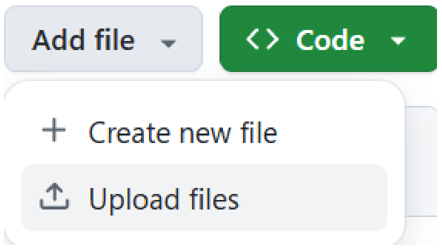
2. In the "Repository name" box, type a name for your project. For example, type "my-first-project."
3. In the "Description" box, type a short description. For example, type "This is my first project on GitHub."

4. Select whether your repository will be **Public** or **Private**. Select "Public" if you want others to be able to see your project.
5. Select **Add a README file**. You will edit this file in a later step.
6. Click **Create repository**.

Step 2: Upload files to your project's repository

So far, you should only see one file listed in the repository, the README.md file you created when you initialized the repository. Now, we'll upload some of your own files.

1. To the right of the page, select the **Add file** dropdown menu.
2. From the dropdown menu, click **Upload files**.



3. On your computer, open the folder containing your work, then drag and drop all files and folders into the browser.
4. At the bottom of the page, under "Commit changes", select "Commit directly to the main branch, then click **Commit changes**.

Getting started with Git

Have you ever wished you had a time machine for your code? Well, Git is exactly that, and so much more!

If you aren't familiar with Git, it's a **version control** system that helps you keep track of changes to your code. You can save a snapshot of your project at a particular point in time, then make experimental changes without risking your work, since you can always go back to your snapshot.

GitHub itself is a platform built around Git, letting you save your Git projects to the cloud and work on them with other developers.

While Git can be complicated, it's a powerful and necessary tool for any developer. This article will give you all the tools you need to use Git in your day-to-day workflow.

Prerequisites

To follow this tutorial, you need to install Visual Studio Code.

Learning Git basics with GitHub Desktop

For standard Git operations, we recommend GitHub Desktop, an app that lets you interact with Git visually instead of through written commands. In this section, we'll learn how to use GitHub Desktop to quickly perform the most common Git operations.

Setting up GitHub Desktop

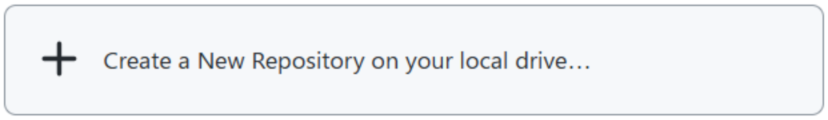
If this is your first time using GitHub Desktop, you need to install it and connect your GitHub account.

1. Download GitHub Desktop.
2. Open GitHub Desktop, then click **Sign in to GitHub.com** and authorize GitHub Desktop to access your account.
3. Back in GitHub Desktop, click **Finish**. This will add your name and email from your GitHub account to Git.

Creating a local repository

Now, you can take your first steps into Git by creating a **repository**. Think of a repository as a project folder that tracks changes and stores history. First, we'll create a **local** repository:

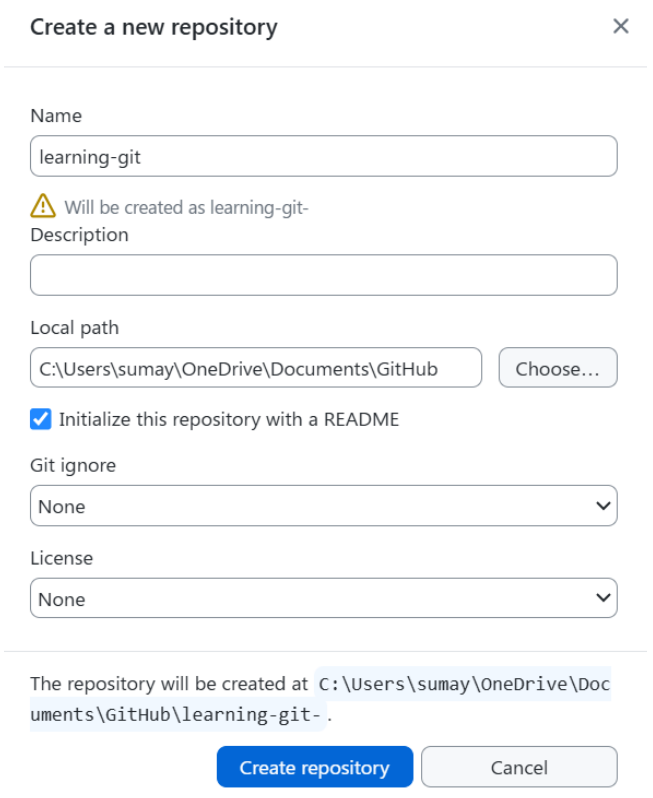
1. In GitHub Desktop, click **Create a New Repository on your Local Drive**.



2. Name the repository learning-git.
3. Select **Initialize this repository with a README** to create a blank README.md file automatically.

> **Tip**
> It's standard practice to include a README.md file, also known as a README, in your projects. READMEs typically contain information that helps others understand, set up, and run your project.

4. Click **Create repository**.


Create a new repository

Name: learning-git

Will be created as learning-git-

Description:

Local path: C:\Users\sumay\OneDrive\Documents\GitHub

☑ Initialize this repository with a README

Git ignore: None

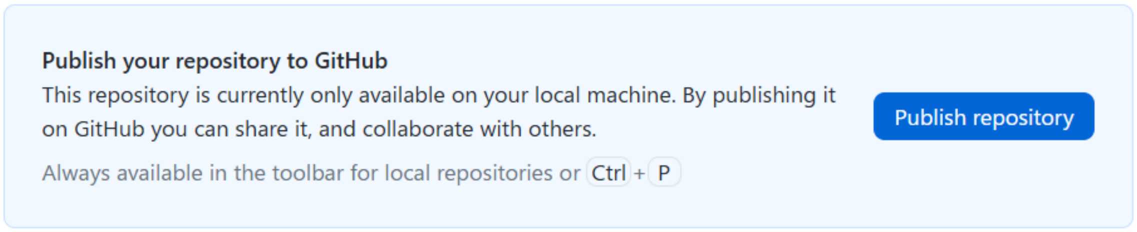
License: None

The repository will be created at `C:\Users\sumay\OneDrive\Documents\GitHub\learning-git-`.

Creating a remote repository

The local repository you just created lives on your computer. Now, let's create a **remote** repository for the same project, which will be hosted on GitHub. Linking a remote repository makes it easier to collaborate on and back up your work.

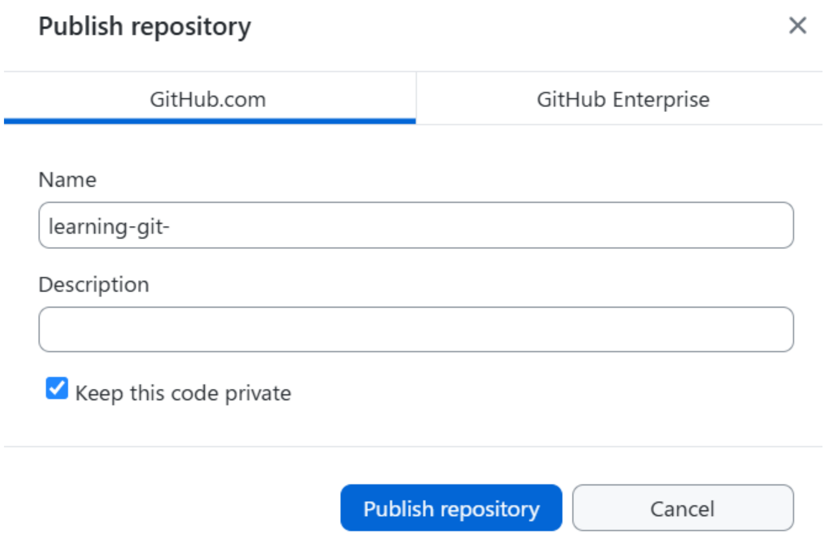
1. In GitHub Desktop, click **Publish repository**.


Publish your repository to GitHub

This repository is currently only available on your local machine. By publishing it on GitHub you can share it, and collaborate with others.

Always available in the toolbar for local repositories or Ctrl + P

2. In the pop up that appears, click **Publish repository** one more time.

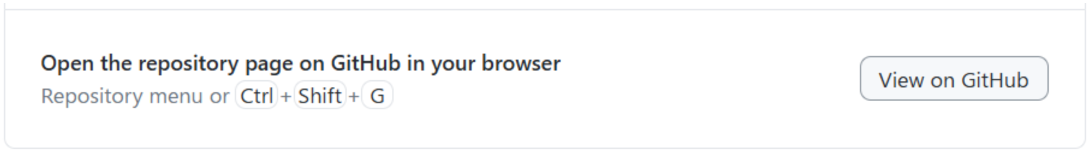

Publish repository

Name: learning-git-

Description:

✓ Keep this code private

3. To see your remote repository, click **View on GitHub**.

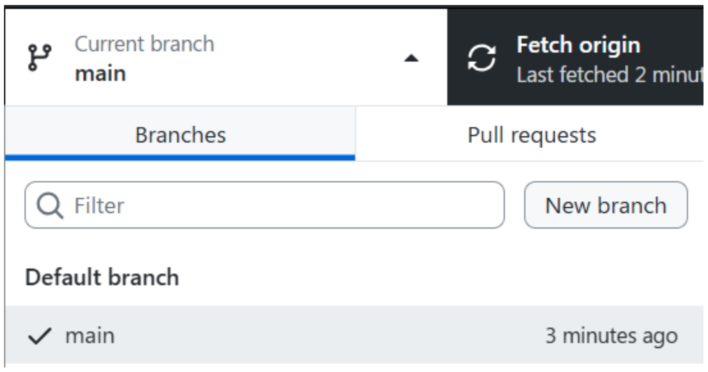

Open the repository page on GitHub in your browser

Repository menu or Ctrl + Shift + G

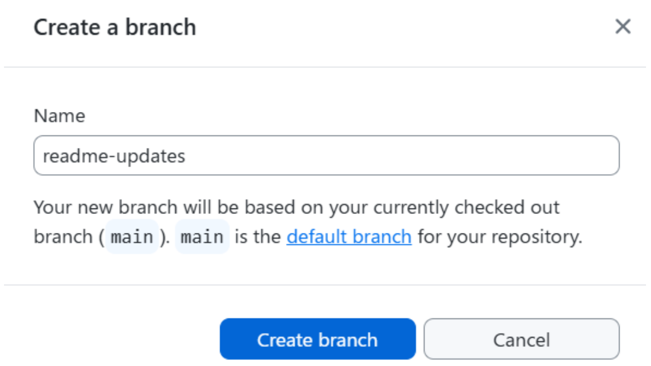
Setting up a space to make changes

Let's create a branch to work on changes in our repository:

1. In GitHub Desktop, select the **Current Branch** dropdown menu, then click **New Branch**.



2. Name your new branch readme-updates, then click **Create Branch**.



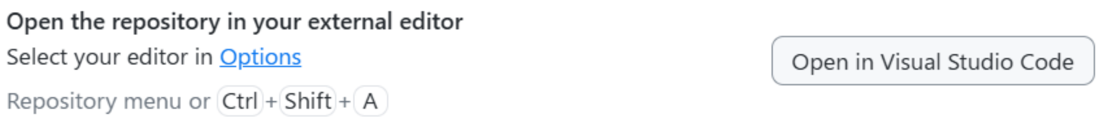
Your new branch will be based on your currently checked out branch (`main`). `main` is the default branch for your repository.

Saving snapshots of your project

To save your progress to your branch, you make a **commit**. You've actually already made your first commit: when you initialized your project with a README, GitHub Desktop automatically created an initial commit to add the README.md file.

Whenever you complete a chunk of work that you want to save, you should make a commit. After you do, you can always go back to that point in time, no matter how many changes you make in the future.

1. In GitHub Desktop, click **Open in Visual Studio Code**.



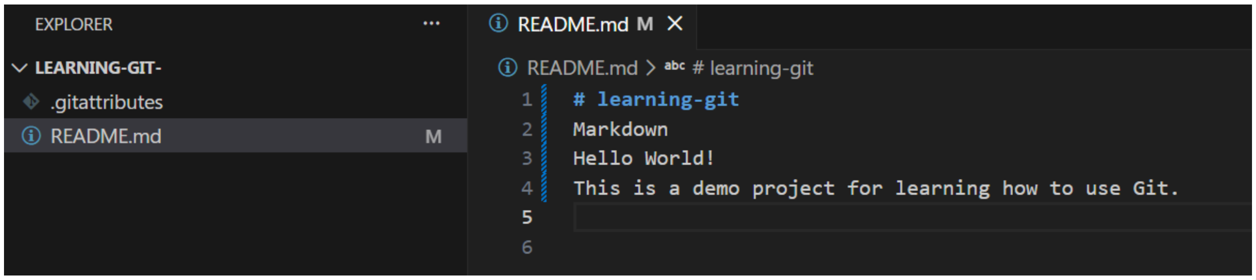
Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl + Shift + A

2. In VS Code, paste the following text into README.md and save your changes:

Markdown

Hello, World!

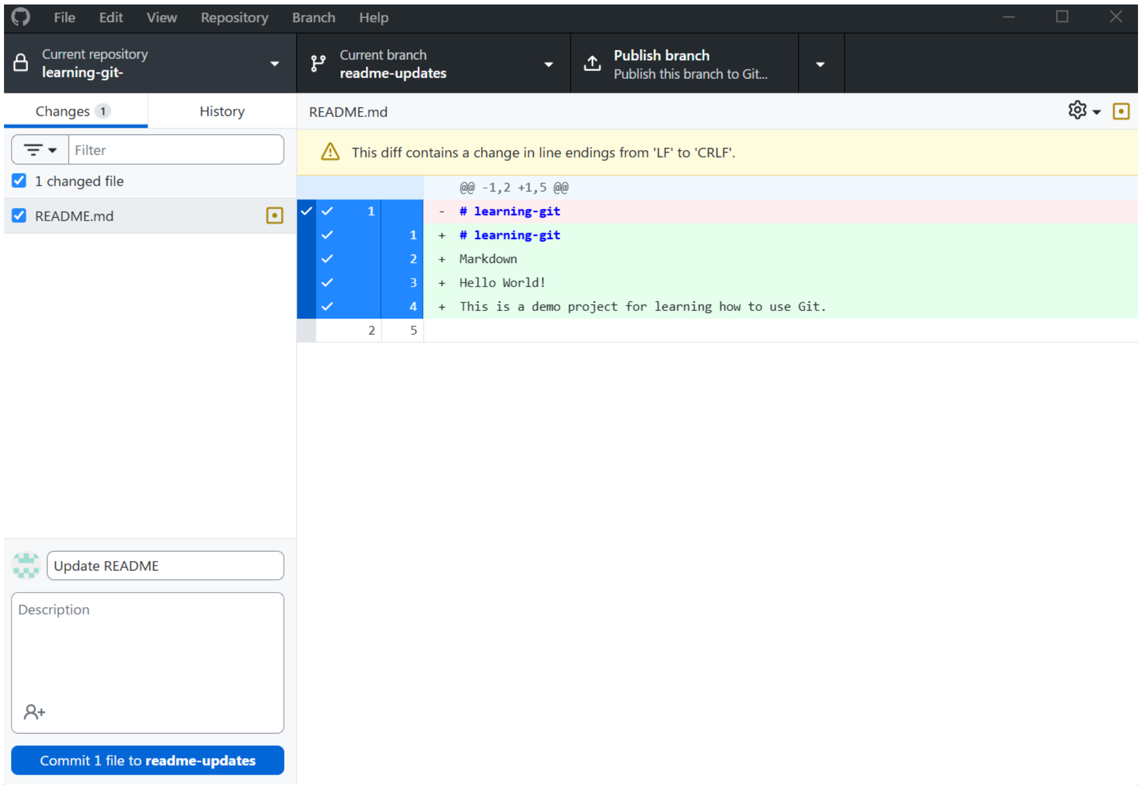
This is a demo project for learning how to use Git.



```
# learning-git
Markdown
Hello World!
This is a demo project for learning how to use Git.
```

3. Back in GitHub Desktop, you'll see the updates you just made to your README. In the bottom left, next to your GitHub profile picture, type "Update README" in the text box. This is called a **commit message**, and it helps you keep track of the changes you make in each commit.

4. To make your commit, click **Commit to readme-updates**.



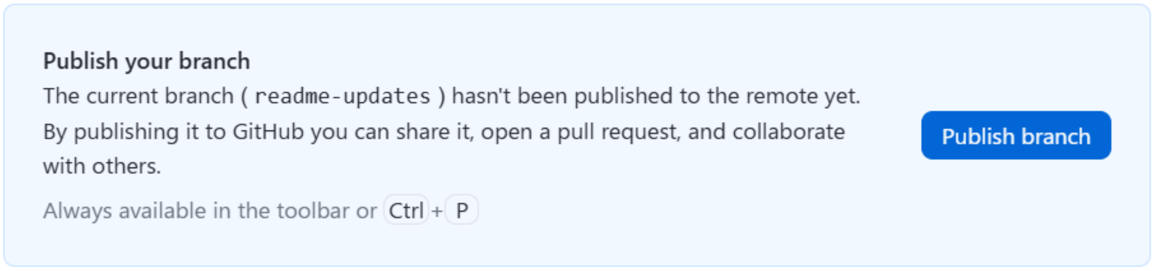
This diff contains a change in line endings from 'LF' to 'CRLF'.

```
@@ -1,2 +1,5 @@
- # learning-git
+ # learning-git
+ Markdown
+ Hello World!
+ This is a demo project for learning how to use Git.
```

Bringing your changes into your main branch

When you're happy with the changes you've made on a branch, you can publish your branch to the remote repository and create a **pull request**. Pull requests let you review a set of proposed changes, then merge them from one branch into another. In our case, we'll create a pull request that brings the changes we made in readme-updates into our original branch, main.

1. Click **Publish branch** to push the readme-updates branch with your changes to the remote repository.

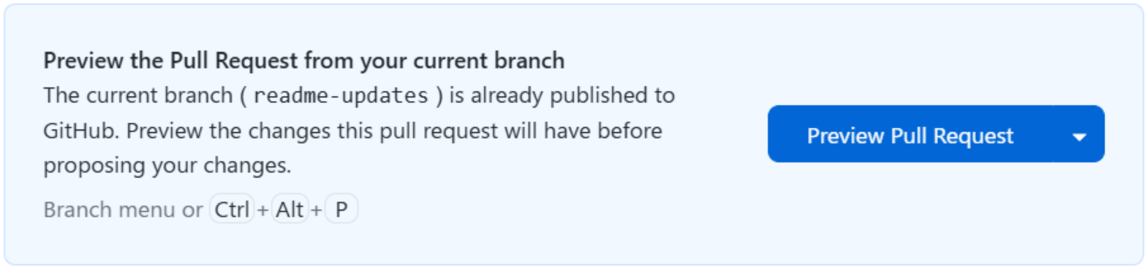


Publish your branch

The current branch (readme-updates) hasn't been published to the remote yet. By publishing it to GitHub you can share it, open a pull request, and collaborate with others.

Always available in the toolbar or Ctrl + P

2. To review your suggested changes, click **Preview Pull Request**.

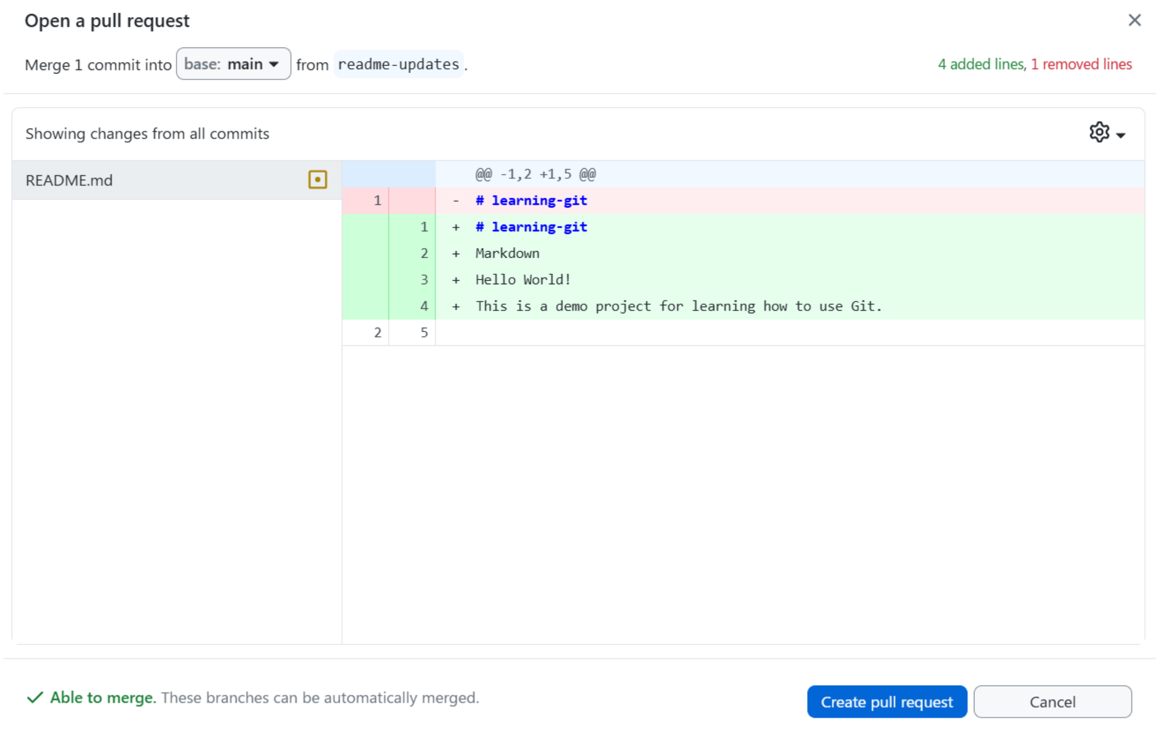


Preview the Pull Request from your current branch

The current branch (readme-updates) is already published to GitHub. Preview the changes this pull request will have before proposing your changes.

Branch menu or Ctrl + Alt + P

3. Click **Create Pull Request**.



Open a pull request

Merge 1 commit into base: main from readme-updates .

4 added lines, 1 removed lines

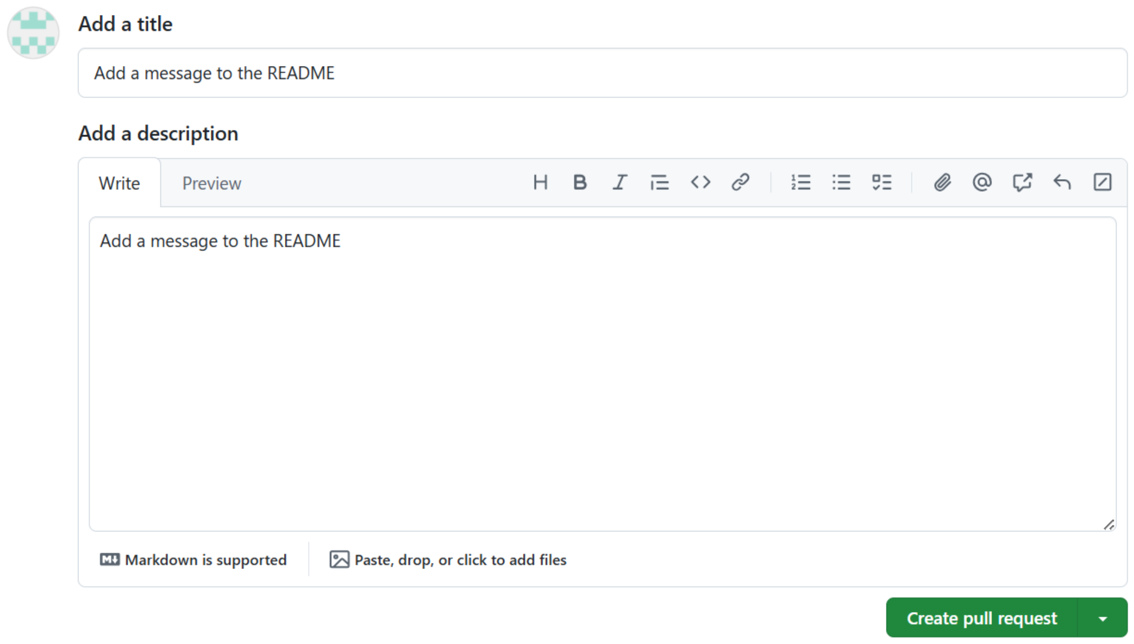
Showing changes from all commits

README.md

```
@@ -1,2 +1,5 @@
1     - # learning-git
   1  + # learning-git
   2  + Markdown
   3  + Hello World!
   4  + This is a demo project for learning how to use Git.
2  5
```

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

4. In the GitHub window that appears, change your pull request title to "Add a message to the README", then write a brief description of your changes in the comment box.
5. Click **Create pull request**.


Add a title
Add a message to the README
Add a description
Write
Preview
Add a message to the README
Markdown is supported
Paste, drop, or click to add files
Create pull request

6. To bring your changes into the main branch, at the bottom of the page, click **Merge pull request**.

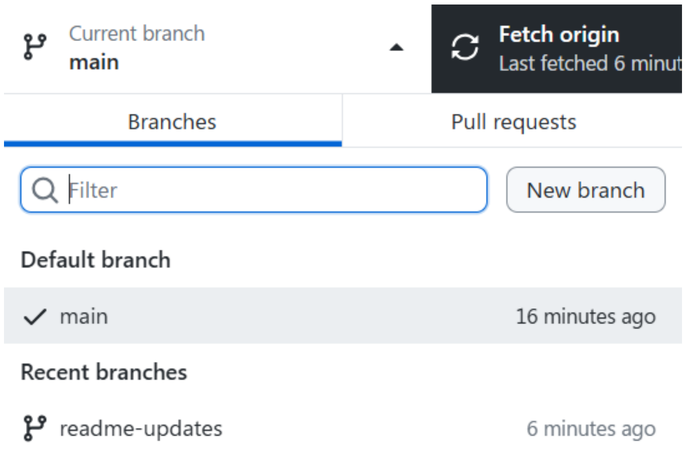
Note
When you're working on a project with other developers, it's standard practice for someone else to review your pull request before it's merged.

7. Near the bottom of the page, click **Delete branch**. Deleting branches that have been merged into main helps keep your repository clean and easy to navigate.

Preparing to make more changes

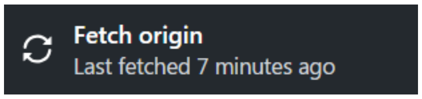
Now that you've successfully brought your changes into the main branch, there are a few steps you should take to get ready for your next round of changes:

1. In GitHub Desktop, if you aren't on the main branch, select the **Current Branch** dropdown menu, then click **main**.

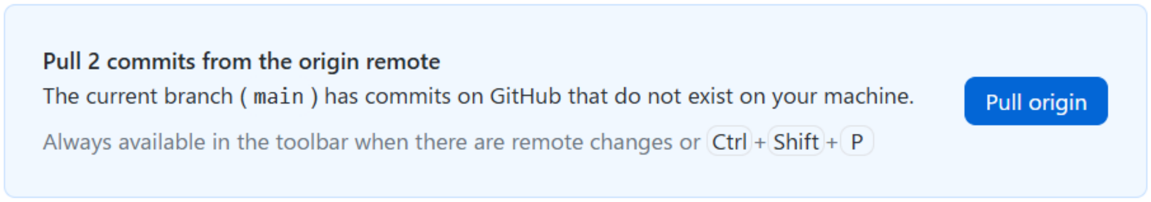


<u>You should almost always switch back to the main branch before creating a new branch,</u> since new branches are created as copies of the currently selected branch.

2. To check if any changes have been made to your remote main branch, click **Fetch origin**.



3. Finally, to update your local main branch with changes to the remote main branch, click **Pull origin**.



Pull 2 commits from the origin remote

The current branch (`main`) has commits on GitHub that do not exist on your machine.

Always available in the toolbar when there are remote changes or Ctrl + Shift + P

You now have all of the skills necessary for setting up and using Git on a project!

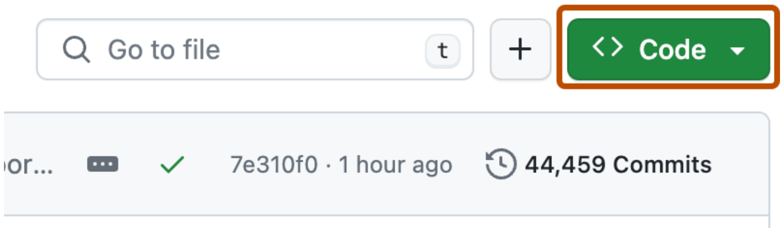
Cloning a repository from GitHub to GitHub Desktop

You can use GitHub to clone remote repositories to GitHub Desktop.

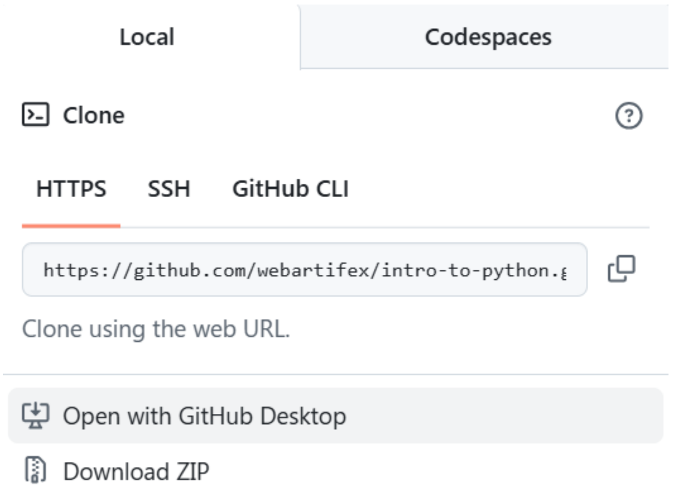
Tip

You also can use GitHub Desktop to clone repositories that exist on GitHub. For more information, see Cloning and forking repositories from GitHub Desktop.

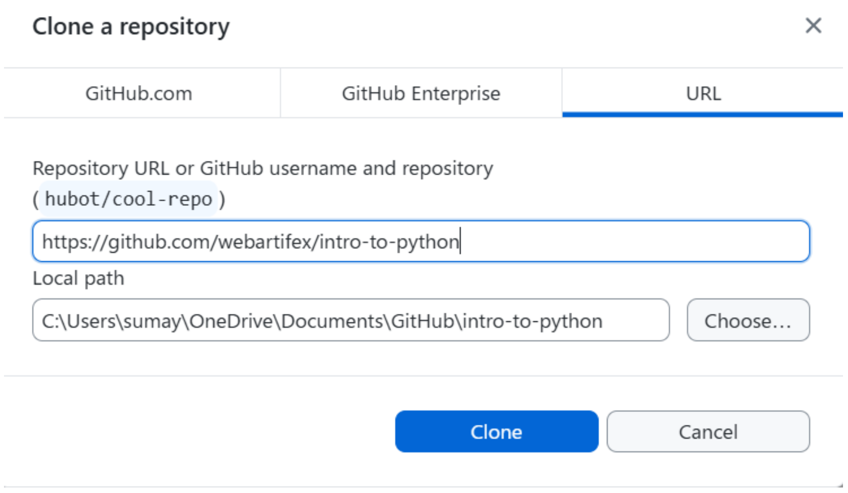
1. Sign in to GitHub and GitHub Desktop before you start to clone.
2. On GitHub, navigate to the main page of the repository. For example, navigate to this repository https://github.com/Python-Crash-Course/Python101
3. Above the list of files, click **Code**.



4. To clone and open the repository with GitHub Desktop, click **Open with GitHub Desktop**.



5. Click **Choose...** and navigate to a local directory where you want to clone the repository.


Clone a repository

Repository URL or GitHub username and repository
(hubot/cool-repo)

https://github.com/webartifex/intro-to-python

Local path

C:\Users\sumay\OneDrive\Documents\GitHub\intro-to-python

6. Click **Clone**.

Creating your first Notebook in Visual Studio Code

In this tutorial, you will learn how to create **Jupyter Notebook** files using **Visual Studio Code (VS Code)**. A Jupyter Notebook is an interactive document that combines **code** (such as Python, R, or Julia), **text** (formatted with Markdown), and **output** (including tables, charts, images, and interactive visualizations) within a single file.

Before starting this tutorial, please ensure that you have the following prerequisites installed.

Prerequisites

- Visual Studio Code
- GitHub Desktop
- Anaconda Distribution

Anaconda Distribution Installation

Step 1: Open the hello-world Repository in VS code

To create a Jupyter Notebook file, you first need a repository.

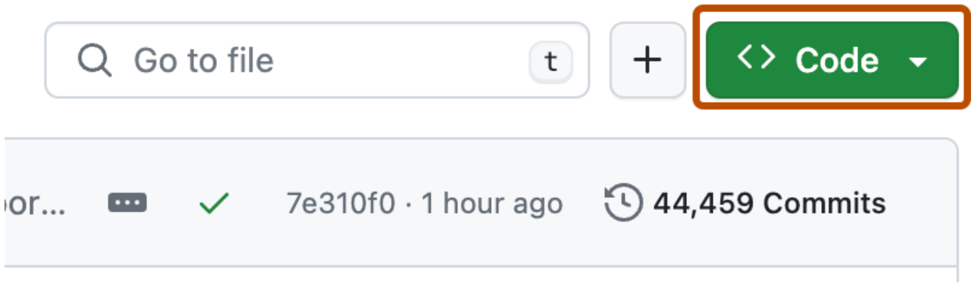
You may either:

- Create a new local repository using GitHub Desktop, then publish it to GitHub, or
- Clone the *hello-world* GitHub repository created in the first tutorial of this document.

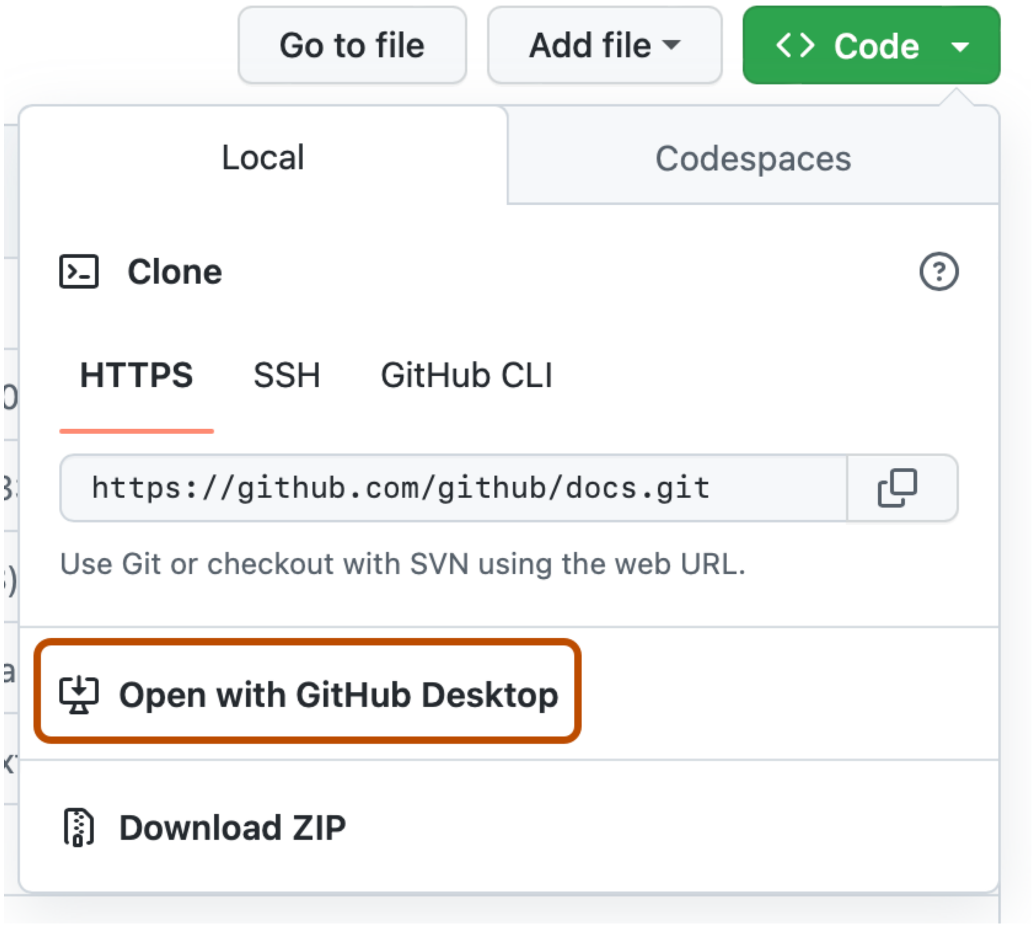
Note: When you create a repository on GitHub, it exists as a *remote* repository. Cloning the repository creates a *local* copy on your computer, allowing you to sync changes between the local and remote repositories.

Cloning the hello-world repository

1. On GitHub, navigate to the main page of the repository (*hello-world*).
2. Above the list of files, click **Code**.



3. To clone and open the repository with GitHub Desktop, click **Open with GitHub Desktop**.



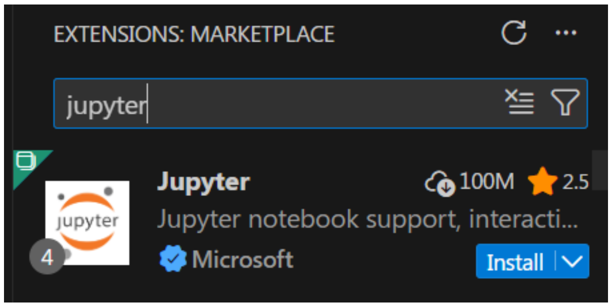
```
https://github.com/github/docs.git
```

Use Git or checkout with SVN using the web URL.

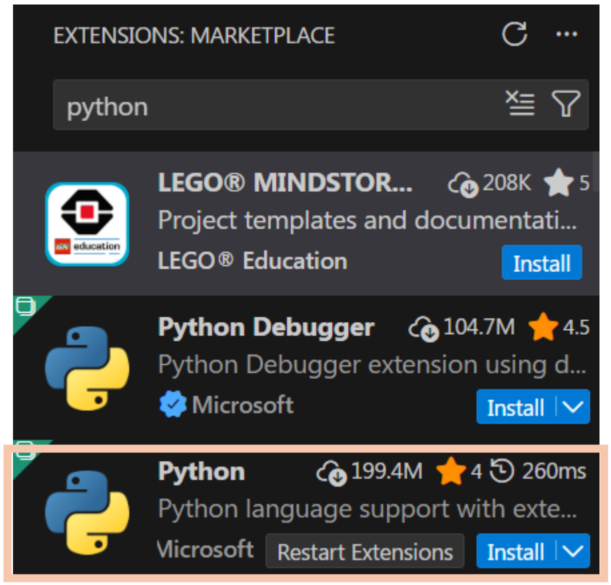
4. Follow the prompts in GitHub Desktop to complete the clone.
5. Click **Open in Visual Studio Code**.

Step 2: Install Required VS Code Extensions

1. In VS Code, click the Extensions tab in the left sidebar .
2. Search for Jupyter and install the Jupyter extension.



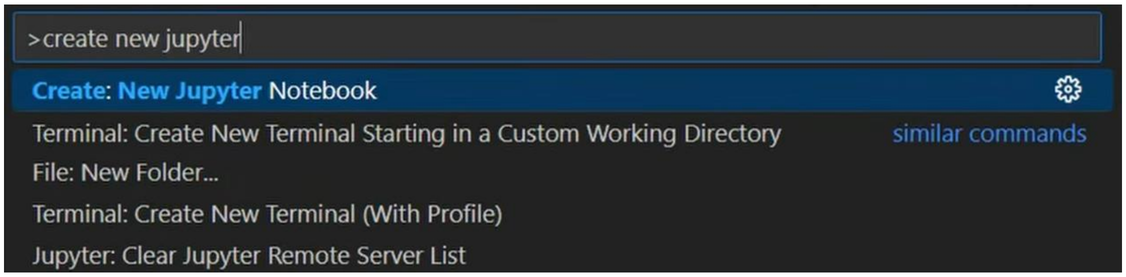
3. Search for Python and install the Python extension.



Step 3: Create a Jupyter Notebook File

You can create a Jupyter Notebook in either of the following ways:

- Run the Create: **New Jupyter Notebook** command from the Command Palette (Ctrl + Shift + P), or
- Create a new file in your workspace and name it with the .ipynb extension.

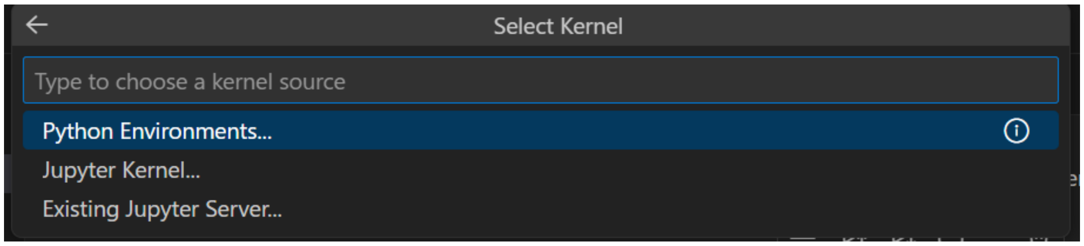

>create new jupyter
Create: New Jupyter Notebook
Terminal: Create New Terminal Starting in a Custom Working Directory
similar commands
File: New Folder...
Terminal: Create New Terminal (With Profile)
Jupyter: Clear Jupyter Remote Server List

This will open a new blank Jupyter Notebook. Save the file and name it hello-world.ipynb.

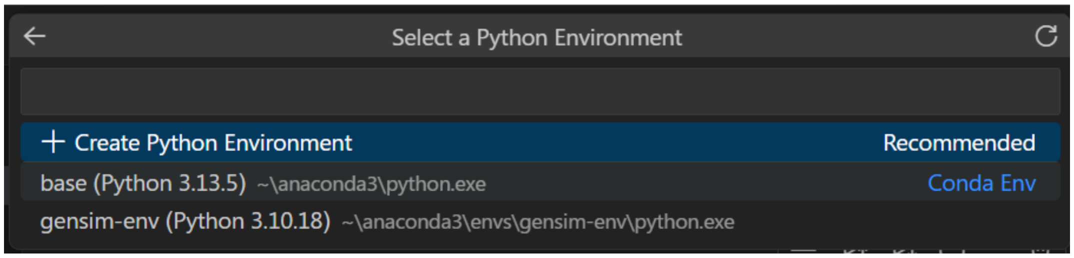
Step 4: Connect to a Kernel

Before running code, you must connect the notebook to a **kernel**, which is the computing engine that executes the code.

1. In the top-right corner of the notebook, click **Select Kernel** .
2. Choose **Python Environments...**



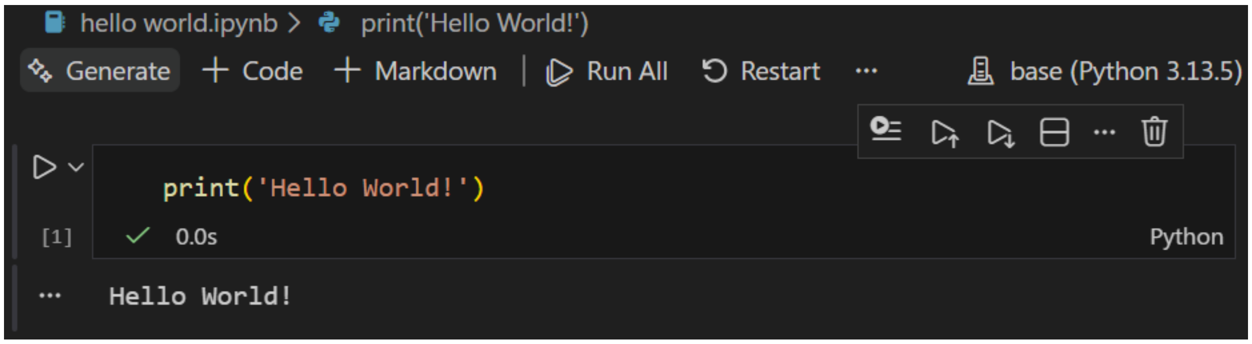
3. Select the desired environment from the list (for this demo, select **base**).


Select a Python Environment
+ Create Python Environment
Recommended
base (Python 3.13.5) ~\anaconda3\python.exe
Conda Env
gensim-env (Python 3.10.18) ~\anaconda3\envs\gensim-env\python.exe

Step 5: Run Your First Code Cell

1. Click inside a code cell.
2. Write your first Python statement.
3. Click Execute Cell .

Congratulations! You have successfully created and run your first Jupyter Notebook in VS Code.



hello world.ipynb > print('Hello World!')

```
print('Hello World!')
```

[1] ✓ 0.0s

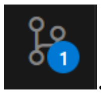
```
Hello World!
```

Step 6: Commit Your Notebook to GitHub

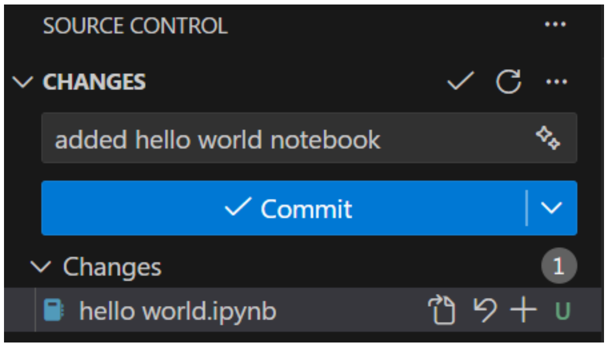
The final step is to commit your work to the GitHub remote repository.

You may commit using GitHub Desktop as explained earlier, or directly within VS Code by following these steps:

1. Save your notebook.

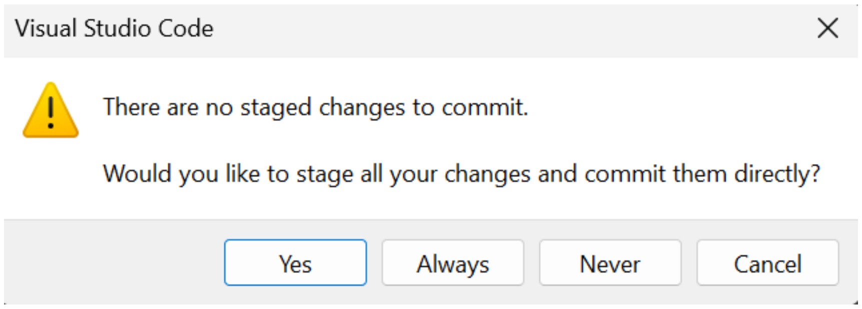
   In the left sidebar, click the Source Control icon . You'll notice that you have one pending change that needs to be committed.

2. Enter a commit message and then click Commit.



added hello world notebook

3. When prompted, click Yes.

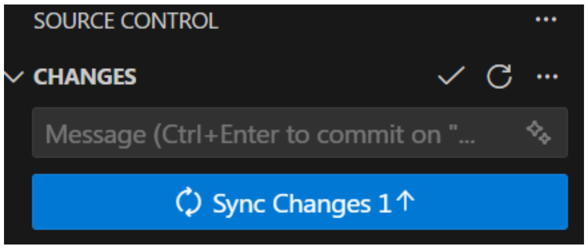


Visual Studio Code

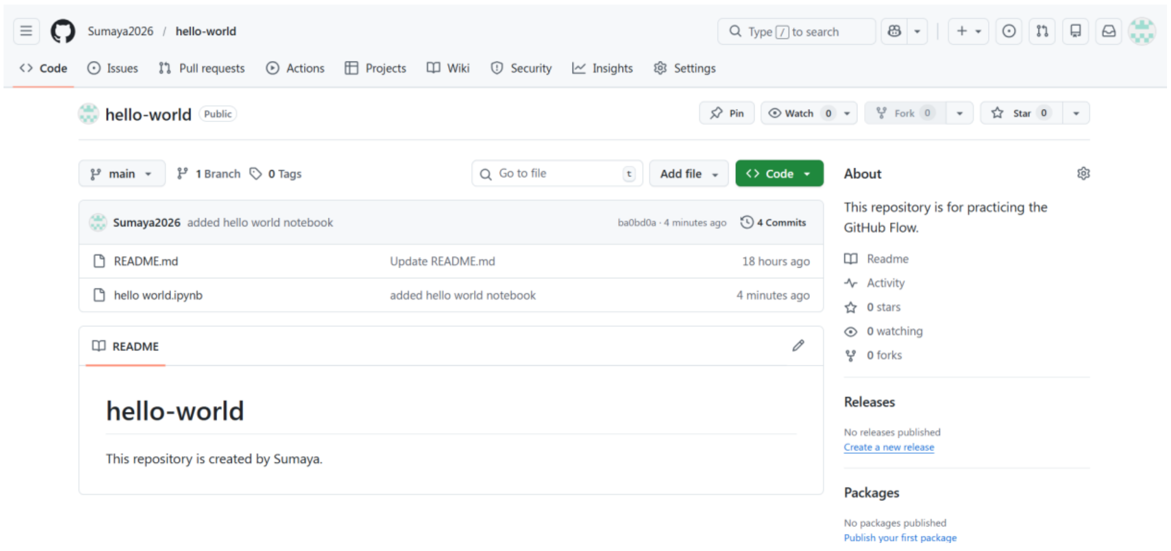
There are no staged changes to commit.

Would you like to stage all your changes and commit them directly?

4. Click Sync Changes to push your commit to the main branch.



5. If prompted, authenticate using your GitHub credentials.
6. Navigate to the *hello-world* repository on GitHub. You should see the newly created Jupyter Notebook file listed among the repository files.



Sumaya2026 added hello world notebook · ba0bd0a · 4 minutes ago · 4 Commits

README.md	Update README.md	18 hours ago
hello world.ipynb	added hello world notebook	4 minutes ago

This repository is for practicing the GitHub Flow.

hello-world

This repository is created by Sumaya.

References

1. GitHub, Inc. (n.d.). *GitHub Docs*. GitHub. Retrieved January 14, 2026, from https://docs.github.com/en
2. Microsoft. (n.d.). *Source control overview — Visual Studio Code documentation*. Visual Studio Code. Retrieved January 14, 2026, from https://code.visualstudio.com/docs/sourcecontrol/overview
3. Microsoft. (n.d.). *Jupyter notebooks — Visual Studio Code documentation*. Visual Studio Code. Retrieved January 14, 2026, from https://code.visualstudio.com/docs/datascience/jupyter-notebooks