# **Lab 2 - GitHub Essentials**

## **Introduction**

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere.

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.

To complete this tutorial, you need a [GitHub account](http://github.com/). You don't need to know how to code, use the command line, or install Git (the version control software that GitHub is built on). If you have a question about any of the expressions used in this guide, head on over to the [glossary](https://docs.github.com/en/get-started/quickstart/github-glossary) to find out more about our terminology.

## **(Extra Information) Adding a file to a repository on GitHub.**

## Files that you add to a repository via a browser are limited to 25 MB per file. You can add larger files, up to 100 MB each, via the command line. For more information, see "[Adding a file to a repository using the command line](https://docs.github.com/en/repositories/working-with-files/managing-files/adding-a-file-to-a-repository#adding-a-file-to-a-repository-using-the-command-line)." To add files larger than 100 MB, you must use Git Large File Storage. For more information, see "[About large files on GitHub](https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-large-files-on-github)."

## You can use GitHub Desktop to move your changes to a new branch and commit them. For more information, see "[Committing and reviewing changes to your project](https://docs.github.com/en/desktop/contributing-to-projects/committing-and-reviewing-changes-to-your-project)."

## On GitHub.com, navigate to the main page of the repository.

## Above the list of files, using the Add file drop-down, click Upload files.

## **"Upload files" in the "Add file" dropdown**

## Drag and drop the file or folder you'd like to upload to your repository onto the file tree.

## **Drag and drop area**

## At the bottom of the page, type a short, meaningful commit message that describes the change you made to the file. You can attribute the commit to more than one author in the commit message. For more information, see "[Creating a commit with multiple co-authors](https://docs.github.com/en/articles/creating-a-commit-with-multiple-authors)."

## **Commit message for your change**

## Below the commit message fields, decide whether to add your commit to the current branch or to a new branch. If your current branch is the default branch, you should choose to create a new branch for your commit and then create a pull request. For more information, see "[Creating a new pull request](https://docs.github.com/en/articles/creating-a-pull-request)."

## **Commit branch options**

## Click Commit changes.

## **Commit changes button**

## Adding a file to a repository using the command line

## You can upload an existing file to a repository on GitHub.com using the command line.

## Tip: You can also [add an existing file to a repository from the GitHub website](https://docs.github.com/en/articles/adding-a-file-to-a-repository).

## This procedure assumes you've already:

## [Created a repository on GitHub](https://docs.github.com/en/articles/creating-a-new-repository), or have an existing repository owned by someone else you'd like to contribute to

## [Cloned the repository locally on your computer](https://docs.github.com/en/articles/cloning-a-repository)

## **Warning:** Never **git add, commit**, or **push** sensitive information to a remote repository. Sensitive information can include, but is not limited to:

## **Passwords**

## **SSH keys**

## [**AWS access keys**](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSGettingStartedGuide/AWSCredentials.html)

## **API keys**

## **Credit card numbers**

## **PIN numbers**

## For more information, see "[Removing sensitive data from a repository](https://docs.github.com/en/articles/removing-sensitive-data-from-a-repository)."

## On your computer, move the file you'd like to upload to GitHub into the local directory that was created when you cloned the repository.

## Open Terminal.

## Change the current working directory to your local repository.

## Stage the file for commit to your local repository. **$ git add .**

## **# Adds the file to your local repository and stages it for commit. To unstage a file, use 'git reset HEAD *YOUR-FILE*'.**

## **Commit the file that you've staged in your local repository. $ git commit -m "Add existing file"**

## **# Commits the tracked changes and prepares them to be pushed to a remote repository. To remove this commit and modify the file, use 'git reset --soft HEAD~1' and commit and add the file again.**

## [Push the changes](https://docs.github.com/en/github/getting-started-with-github/pushing-commits-to-a-remote-repository) in your local repository to GitHub.com. $ git push origin YOUR\_BRANCH

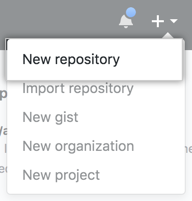
## **# Pushes the changes in your local repository up to the remote repository you specified as the origin**

## **1) Creating a repository**

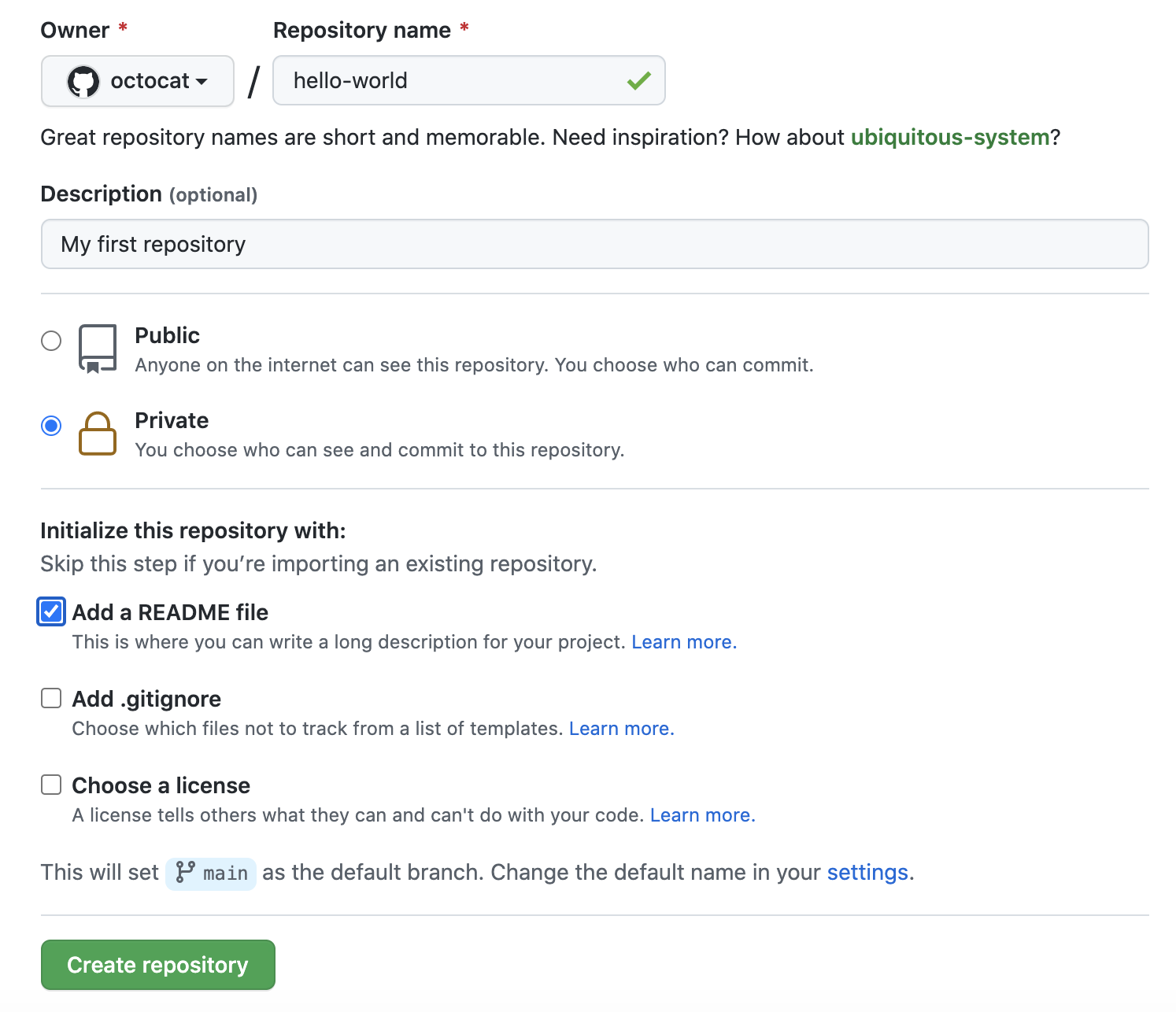
A repository is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets -- anything your project needs. Often, repositories include a *README* file, a file with information about your project. *README* files are written in the plain text Markdown language. You can use this [cheat sheet](https://www.markdownguide.org/cheat-sheet/)to get started with Markdown syntax. 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.

1. In the upper-right corner of any page, use the drop-down menu, and select **New repository**.



1. In the **Repository name** box, enter hello-world.
2. In the **Description** box, write a short description.
3. Select **Add a README file**.
4. Select whether your repository will be **Public** or **Private**.
5. Click **Create repository**.



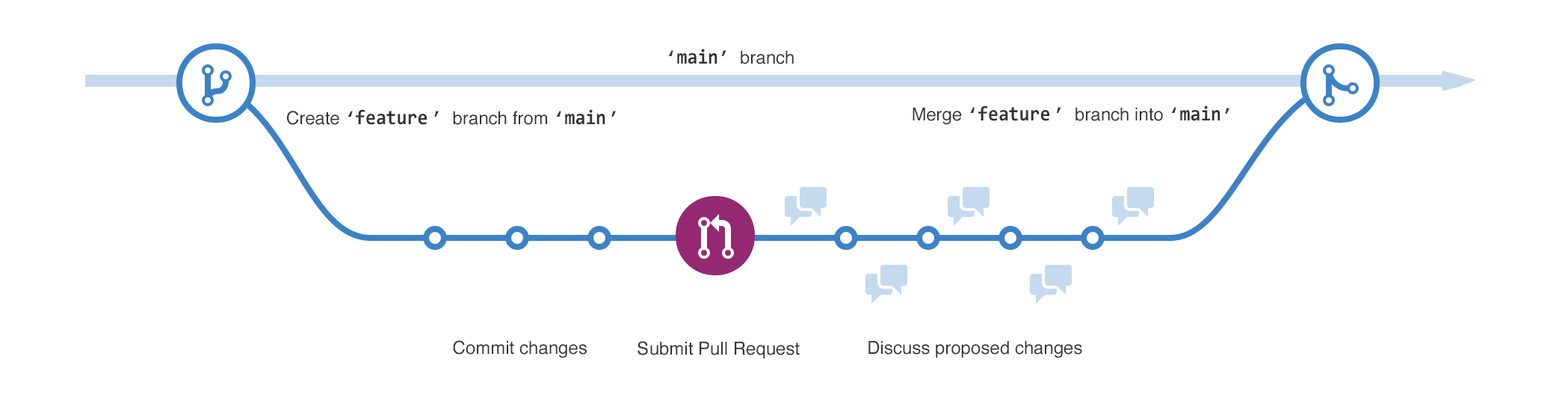
## **2) Creating a branch**

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. You can use branches to have different versions of a project at one time. This 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 before it's merged into main



Have you ever saved different versions of a file? Something like:

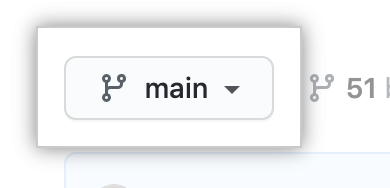
* story.txt
* story-edit.txt
* story-edit-reviewed.txt

Branches accomplish similar goals in GitHub repositories.

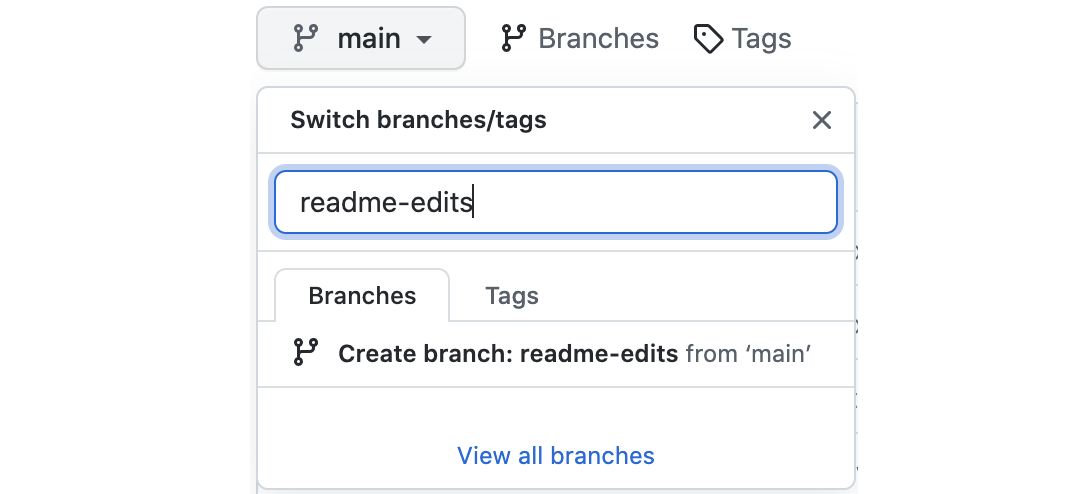
Here at GitHub, our developers, writers, and designers use branches for keeping bug fixes and feature work separate from our main (production) branch. When a change is ready, they merge their branch into main.

### **2a) Create a branch**

1. Click the **Code** tab of your hello-world repository.
2. Click the drop down at the top of the file list that says **main**.



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



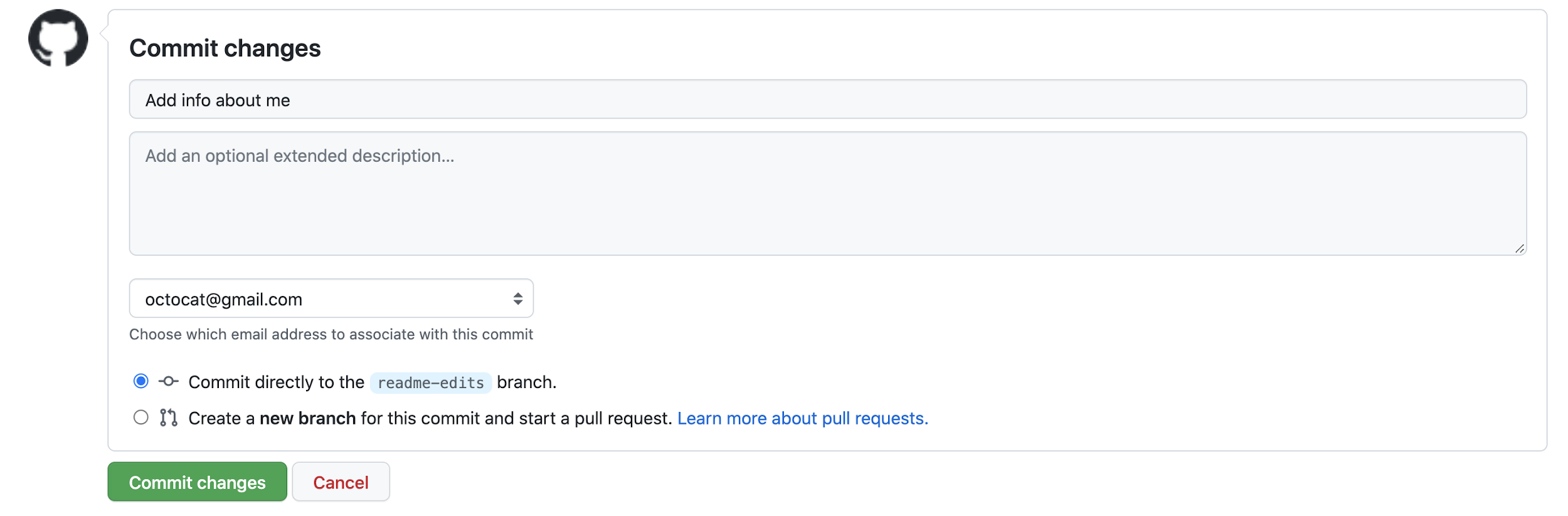
Now you have two branches, main and readme-edits. Right now, they look exactly the same. Next you'll add changes to the new branch.

## **Making and committing 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. Click to edit the file.
3. In the editor, write a bit about yourself. Try using different Markdown elements.
4. In the **Commit changes** box, write a commit message that describes your changes.
5. 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.

## **3) Opening a pull request**

*Some pull requests may not have something to compare. That is fine as long as you demonstrate that you have done or in the process that is good!*

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

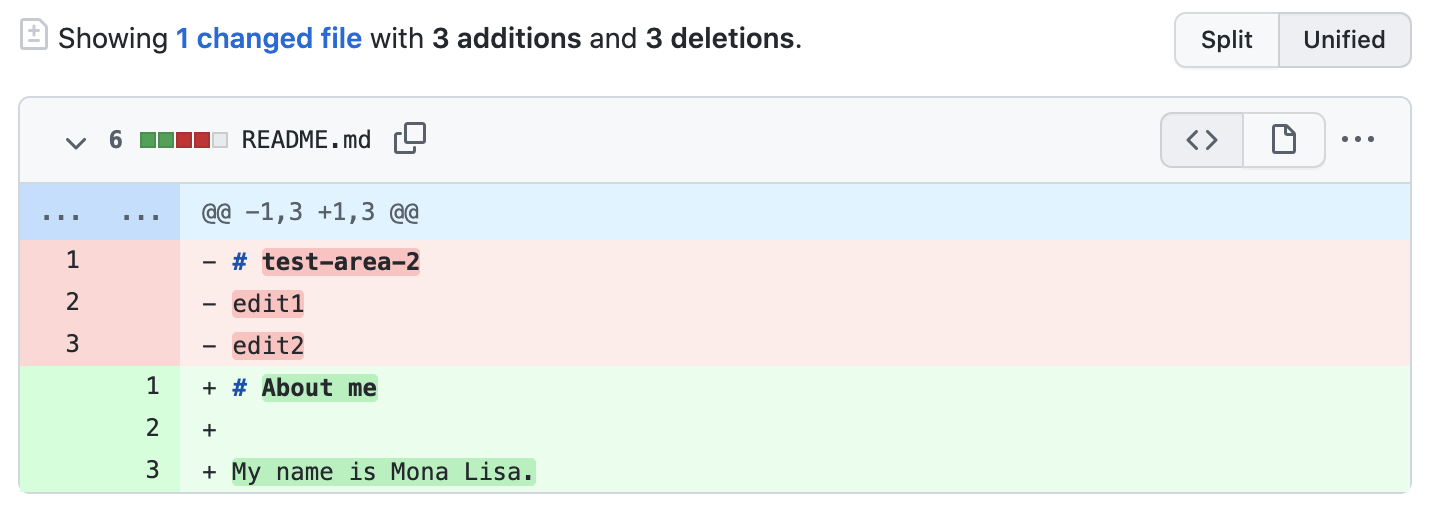
Pull requests are the heart of collaboration on GitHub. When you open a pull request, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show diffs, or differences, of the content from both branches. The changes,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

By using GitHub's @mention feature in your pull request message, you can ask for feedback from specific people or teams, whether they're down the hall or 10 time zones away.

You can even open pull requests in your own repository and merge them yourself. It's a great way to learn 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.



1. Click **Create pull request**.
2. 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.
3. Optionally, to the right of your title and description, click the next to **Reviewers**. **Assignees**, **Labels**, **Projects**, or **Milestone** to add any of these options to your pull request. You do not need to add any yet, but these options offer different ways to collaborate using pull requests. For more information, see "[About pull requests](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests)."
4. Click **Create pull request**.

Your collaborators can now review your edits and make suggestions.

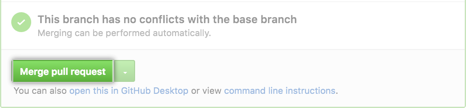
## **4) Merging your pull request**

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. Click **Merge pull request** to merge the changes into main.



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

## **Next steps**

By completing this tutorial, you've learned to create a project and make a pull request on GitHub.

Here's what you accomplished in this tutorial:

* Created an open source repository
* Started and managed a new branch
* Changed a file and committed those changes to GitHub
* Opened and merged a pull request

Take a look at your GitHub profile and you'll see your work reflected on your contribution graph.

**Submission**

* To submit your assignment, take a screenshot of your contribution graph and submit that via Blackboard.

**Reference**

* *Let's build from here*. GitHub. (n.d.). Retrieved February 1, 2023, from https://github.com/