# GitHub

## Overview of Git/GitHub

In this video, you’ll get an overview of Git and GitHub, which are popular environments among developers and data scientists for performing version control of source code files and projects and collaborating with others. You can’t talk about Git and GitHub without a basic understanding of what version control is.

Play video starting at ::30 and follow transcript0:30

A version control system allows you to keep track of changes to your documents. This makes it easy for you to recover older versions of your document if you make a mistake, and it makes collaboration with others much easier. Here is an example to illustrate how version control works. Let’s say you’ve got a shopping list and you want your roommates to confirm the things you need and add additional items. Without version control, you’ve got a big mess to clean up before you can go shopping. With version control, you know EXACTLY what you need after everyone has contributed their ideas.

Play video starting at :1:9 and follow transcript1:09

Git is free and open source software distributed under the GNU General Public License. Git is a distributed version control system, which means that users anywhere in the world can have a copy of your project on their own computer; when they’ve made changes, they can sync their version to a remote server to share it with you. Git isn’t the only version control system out there, but the distributed aspect is one of the main reasons it’s become one of the most common version control systems available. Version control systems are widely used for things involving code, but you can also version control images, documents, and any number of file types. You can use Git without a web interface by using your command line interface, but GitHub is one of the most popular web-hosted services for Git repositories. Others include GitLab, BitBucket, and Beanstalk. There are a few basic terms that you will need to know before you can get started. The SSH protocol is a method for secure remote login from one computer to another. A repository contains your project folders that are set up for version control. A fork is a copy of a repository. A pull request is the way you request that someone reviews and approves your changes before they become final. A working directory contains the files and subdirectories on your computer that are associated with a Git repository. There are a few basic Git commands that you will always use. When starting out with a new repository, you only need create it once: either locally, and then push to GitHub, or by cloning an existing repository by using the command "git init".

Play video starting at :3:3 and follow transcript3:03

"git add" moves changes from the working directory to the staging area. "git status" allows you to see the state of your working directory and the staged snapshot of your changes. "git commit" takes your staged snapshot of changes and commits them to the project. "git reset" undoes changes that you’ve made to the files in your working directory. "git log" enables you to browse previous changes to a project. "git branch" lets you create an isolated environment within your repository to make changes. "git checkout" lets you see and change existing branches. "git merge" lets you put everything back together again. To learn how to use Git effectively and begin collaborating with data scientists around the world, you will need to learn the essential commands. Luckily for us, GitHub has amazing resources available to help you get started. Go to try.github.io to download the cheat sheets and run through the tutorials. In the following modules, we'll give you a crash course on setting up your local environment and getting started on a project.

## GitHub – Getting Started

In the previous video, you learned about Git and GitHub. Before you continue with this video, register for a GitHub account and log in. Let’s start by creating a new repository. Click + then click New Repository. To create a new repository, you need to provide these details: give your new repository a name; optionally, add a description of your repository; choose the repository visibility - whether you want it to be public or private; and choose the option to Initialize this repository with readme file. Then click Create Repository.

Play video starting at ::54 and follow transcript0:54

You will now be redirected to the repository you have created. The root folder of your repository is listed by default and it has just one file ReadMe.md.

Play video starting at :1:8 and follow transcript1:08

Now, it’s time to edit the readme. You can do this in your browser. Just click the pencil to open the online editor and you can change the text of the readme. To save your changes to the repository, you must commit them. After you have made your changes, scroll down to the Commit changes section. Add a commit message and optionally add a description, then click Commit changes. The "commit changes" is used to save your changes to the repository. Go back to the home screen by clicking the repository name link. Note that the readme file is updated and verify your changes.

Play video starting at :1:49 and follow transcript1:49

Let’s learn how to create a new file using the built-in web editor provided by GitHub which runs in the browser. Click Add File, then click Create New File to create the new file.

Play video starting at :2:4 and follow transcript2:04

To create a python file called firstpython.py. First, provide the file name. Next, add a comment that describes your code, then add the code.

Play video starting at :2:17 and follow transcript2:17

Once finished, commit the change to the repository. You can see that your file is now added to the repository and the repository listing shows when the file was added or changed. When you need to change the file, you can edit it again. Click the file name, and then click the pencil icon, make your edits and commit the changes.

Play video starting at :2:40 and follow transcript2:40

You can also upload a file from your local system into the repository. From the home screen of the repository, click Add File and choose the Upload files option.

Play video starting at :2:53 and follow transcript2:53

Click Choose Your Files and select the files you want to upload from your local system.

Play video starting at :2:59 and follow transcript2:59

The file upload process may take a short time, depending on what you are uploading. Once the files finish uploading, click Commit Changes. The repository now reflects the files that were uploaded. In this video, you learned how to create a repository, edit files, and commit changes using the web interface.

## GitHub Lab – getting started

In this lab, you will get started with GitHub by creating a GitHub account and project and adding a file to it using its Web interface.

Objectives

After completing this lab, you will be able to:

1. Describe GitHub
2. Create a GitHub account
3. Add a Project / Repo
4. Edit / Create a file
5. Upload a file & Commit

GitHub Overview

First, let us introduce to GitHub. GitHub in simple words is a collection of folders and files. It is a Git repository hosting service, but it adds many of its own features. While Git is a command-line tool and a server needs to be hosted and maintained via command line as well, GitHub provides this Git server for you and a Web-based graphical interface. It also provides access control and several collaboration features, such as wikis and basic task management tools for every project. GitHub provides cloud storage for source code, supports all popular programming languages, and streamlines the iteration process. GitHub includes a free plan for individual developers and for hosting open source projects.

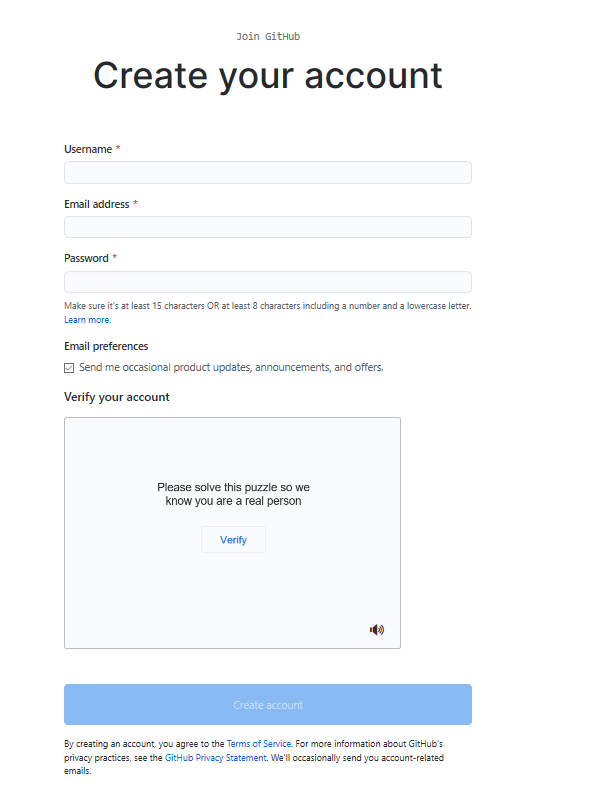
Exercise 1: Creating a GitHub Account

Please follow the steps given below to create an account in GitHub:

Step 1: Create an account: <https://github.com/join>

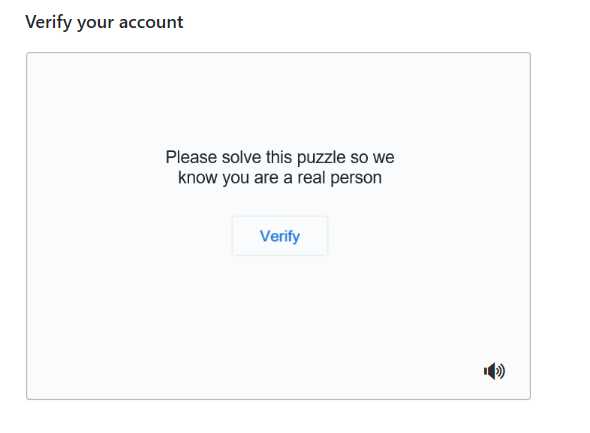
**NOTE:** If you already have a GitHub account, you can skip this step and simply login to your account.

Step 2: Provide the necessary details to create an account as shown below:

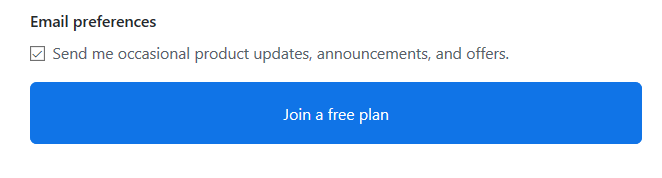


and click Create account.

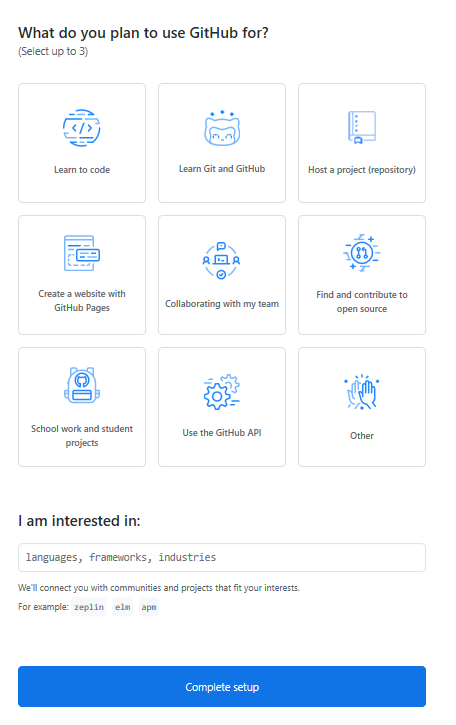
Step 3: Click Verify to verify the account and click Done



Step 4: After verification, click Join a Free Plan

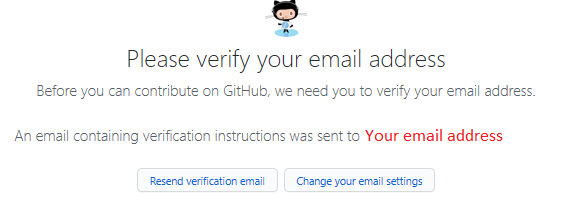


Step 5: Select the details as shown below and click Complete Setup

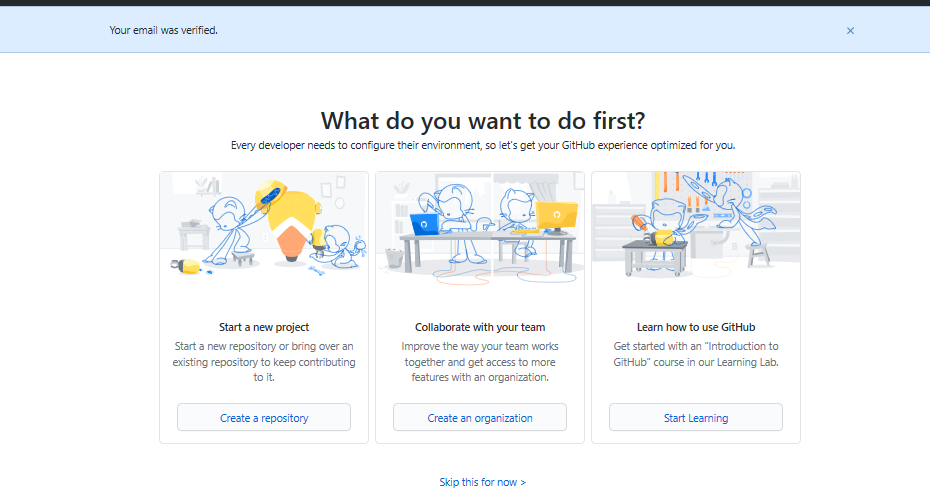


Step 6: Go to your email, find the verification email from GitHub, and click on the link/button in that email to verify your email.

**NOTE:** If you do not receive verification email, click Resend verification email.

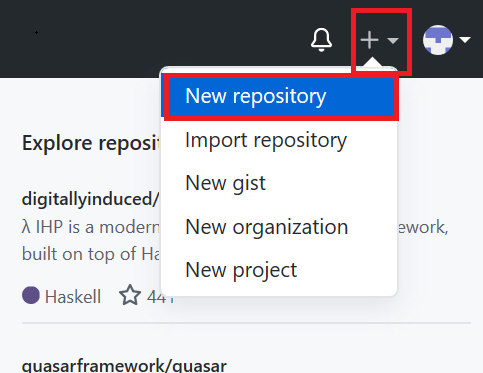


Email is verified

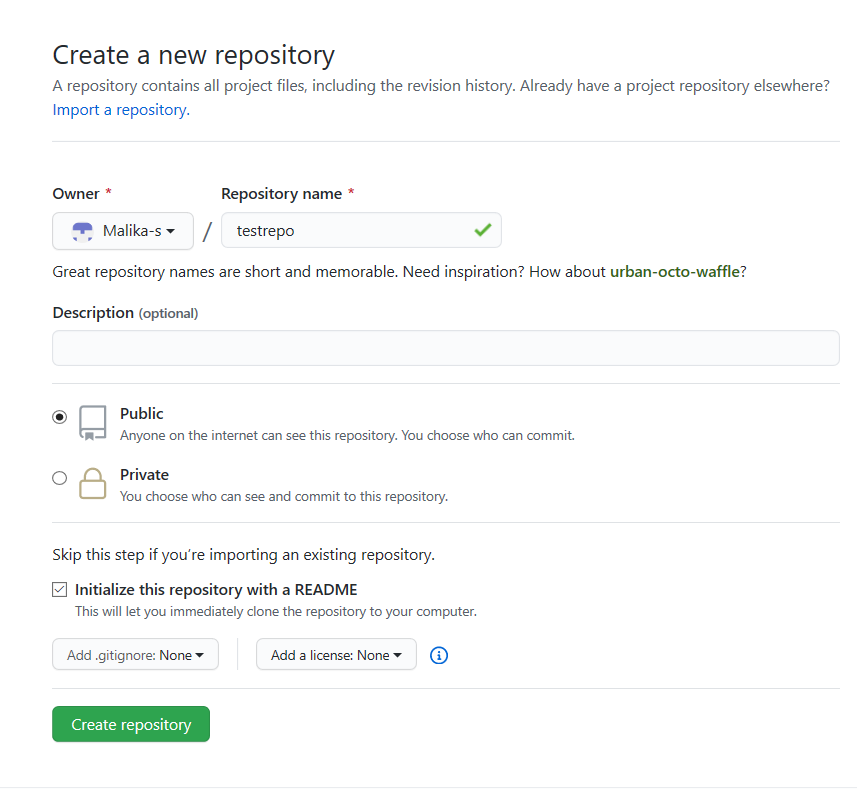


Exercise 2: Adding a Project / Repo

Step 1: Click on the + symbol and click New repository.



Step 2: Provide a repository a name and initialize with the empty README.md file.



and click Create repository.

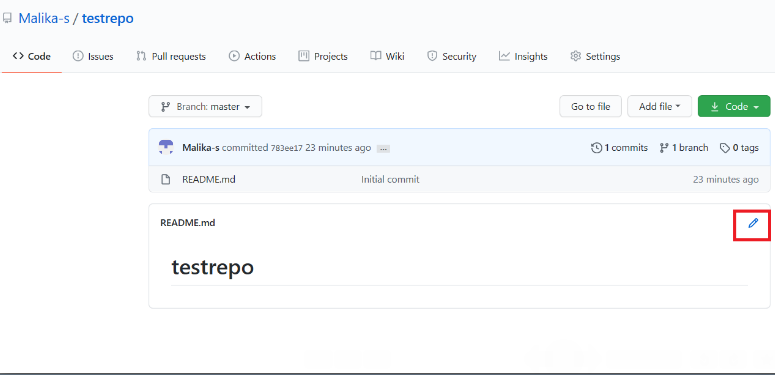
Now, you will be redirected to the repository you have created.

Let's start editing the repository.

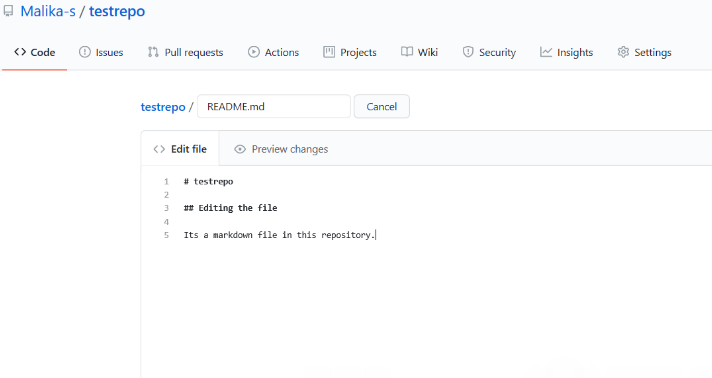
Exercise 3: Create / edit a file

Exercise 3a: Edit a file

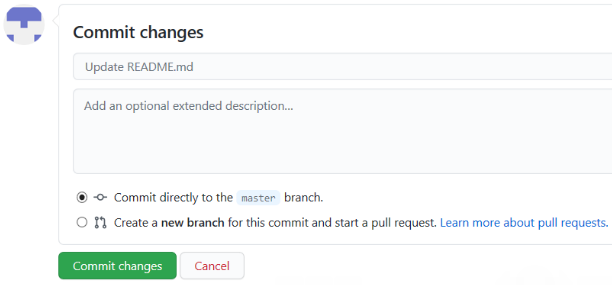
Step 1: Once the repository is created, the root folder of your repository is listed by default and it has just one file ReadMe.md. Click on the pencil icon to edit the file.



Step 2: Add text to file.



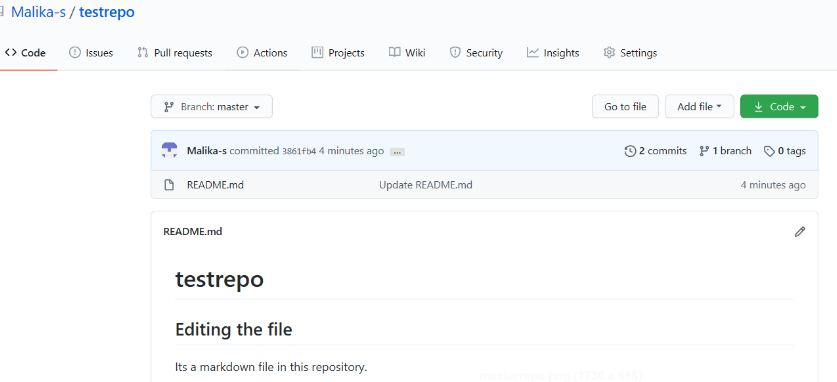
Step 3: Scroll down the page after adding the text and click Commit Changes.



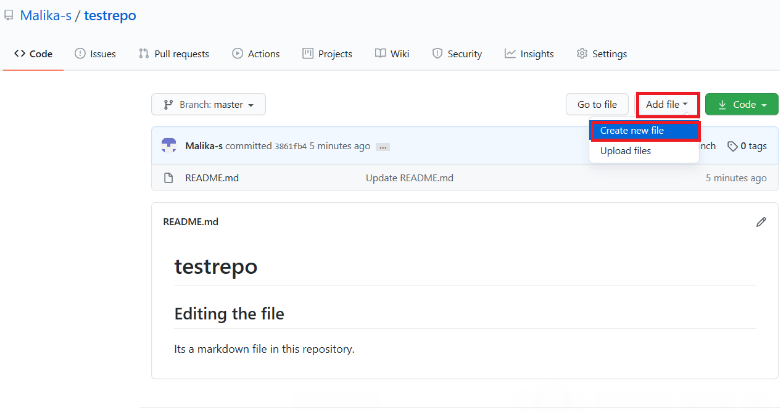
Now, check your file is edited with the new text.

Exercise 3b: Create a new file

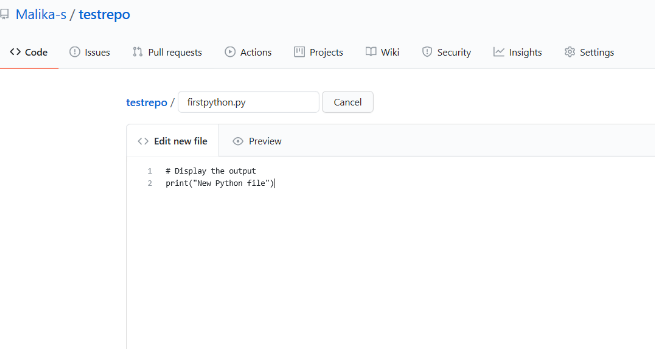
Step 1: Click on the repository name to go back to the master branch like in this testrepo.



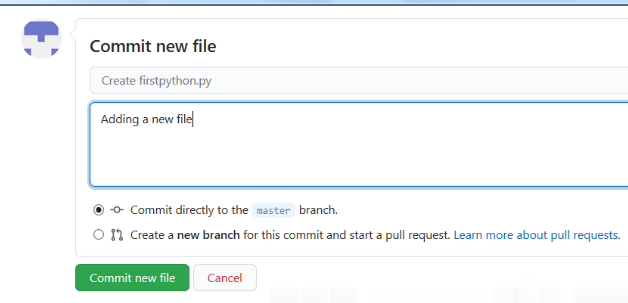
Step 2: Click Add file and select Create New file to create a file in the repository.



Step 3: Provide the file name and the extension of the file. For example, firstpython.py and add the lines.



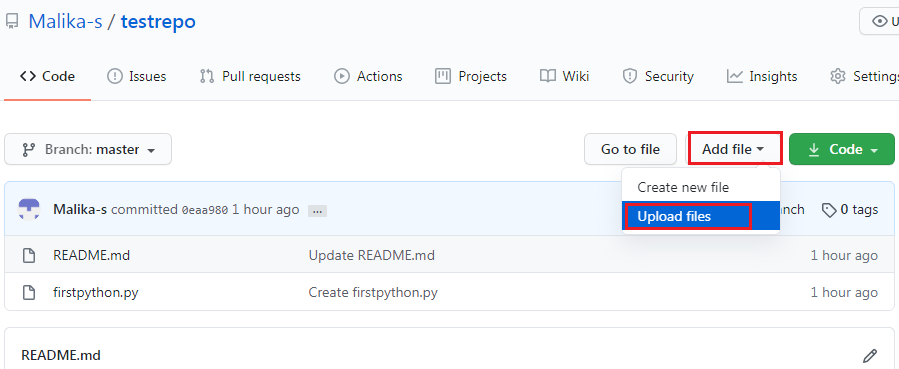
Step 4: Scroll down the page after adding the text. Add description of the file (optional) and click Commit new file.



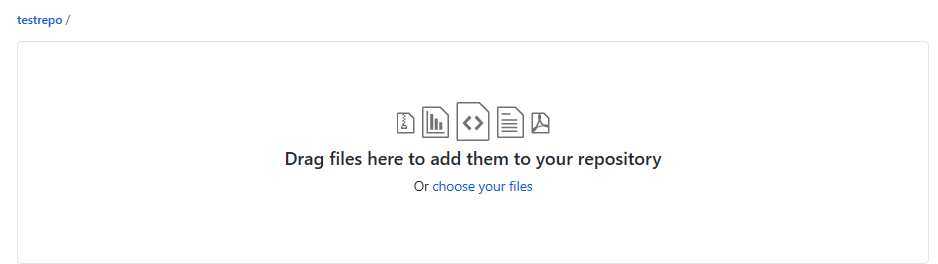
Step 5: Your file is now added to your repository and the repository listing shows when the file was added/changed.

Exercise 4: Upload a file & Commit

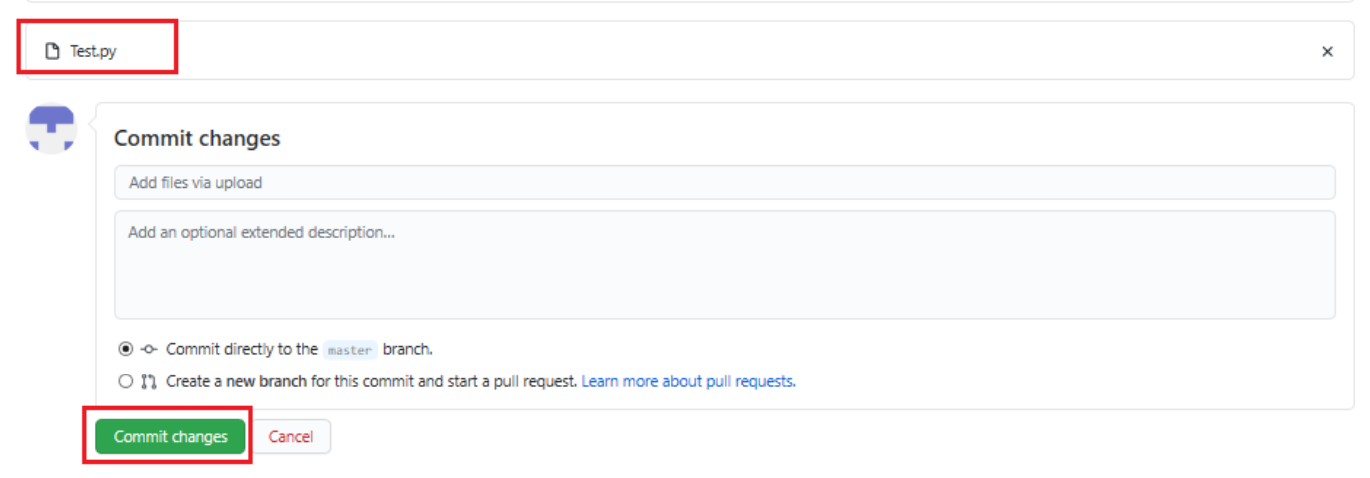
Step 1: Click Add file and select Upload files to upload a file (Upload any .txt,.ipynb, .png file) in the repository from the local computer.



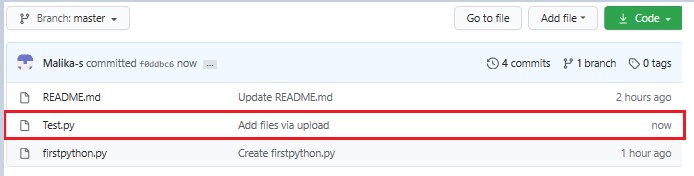
Step 2: Click on choose your files and choose any files from your computer.



Step 3: Once the file finishes uploading, click on Commit Changes



Step 4: Now, your file is uploaded in the repository.



## **Hands-on Lab: Creating and merging branches in your GitHub repo**

**Effort:** 20 min

## **Objectives**

After completing this lab you will be able to:

1. Create a branch
2. Commit changes in the child branch
3. Open a pull request (PR)
4. Merge the PR into the master branch

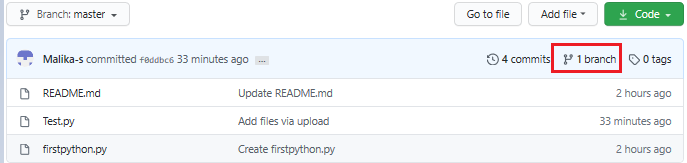
## **Pre-requisites**

This hands-on lab requires you to have already created a GitHub account, and added a project to it, as illustrated in the [previous lab](https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0105EN-SkillsNetwork/labs/Module2/GitHub1_Getting_Started.md.html)

## **Exercise 1: Creating a branch**

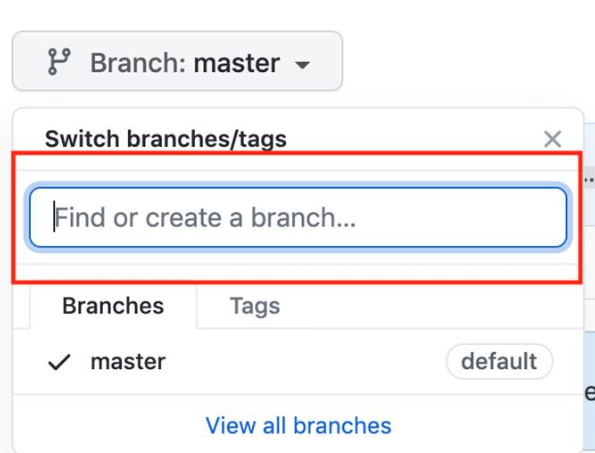
You can create or delete branches directly on GitHub.

Step 1: Currently, there is one branch as shown below:

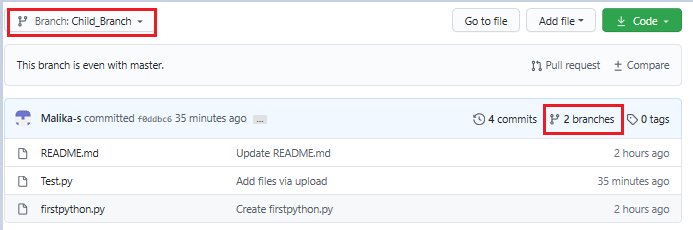


Step 2: On GitHub, navigate to the main page of the repository.

Step 3: Click the Branch selector menu. Enter the name of the branch you want to create and press Enter.



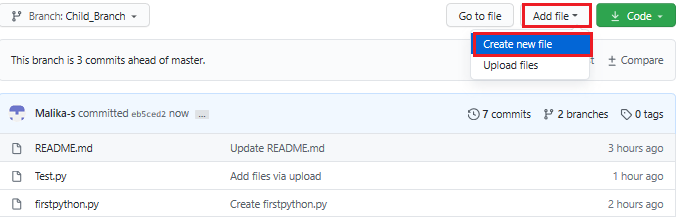
Step 4: Observe that your repository has two branches Master and Child\_Branch(check using arrow).



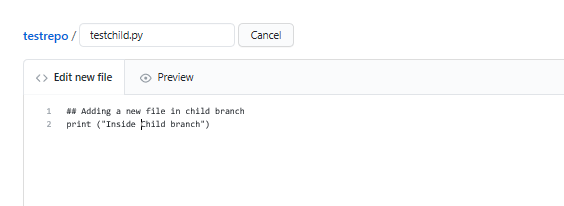
Whatever, is in the master file is copied to the child branch. But when we add a file or edit any file in child branch that will not reflect in the 'Master' branch.

## **Exercise 2: Adding a file in the Child Branch**

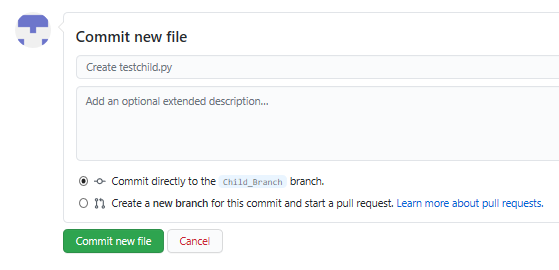
Step 1: Click Add file and select Create New file to create a file in the repository.



Step 2: Provide the file name and the extension of the file. For example, testchild.py and add the lines.



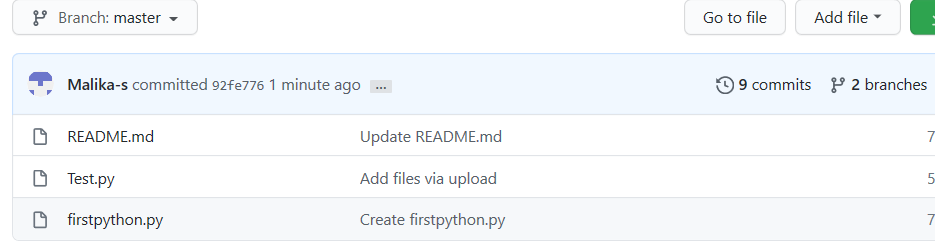
Step 3: Scroll down the page after adding the text. Add a description of the file (optional) and click Commit new file.



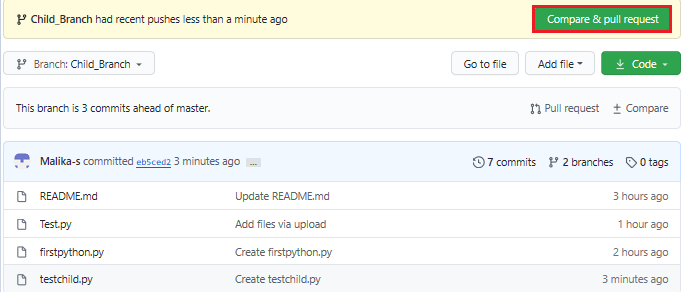
The file has added to the child branch.

## **Exercise 3: Open a Pull Request**

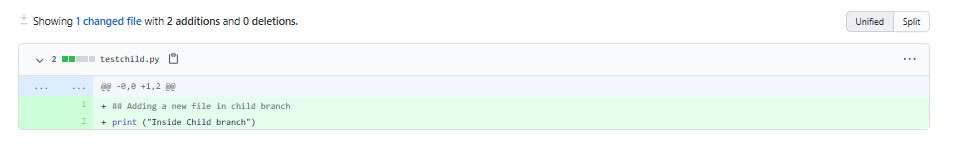
You can check the master branch now there is no testchild.py file by selecting the Branch selector menu.



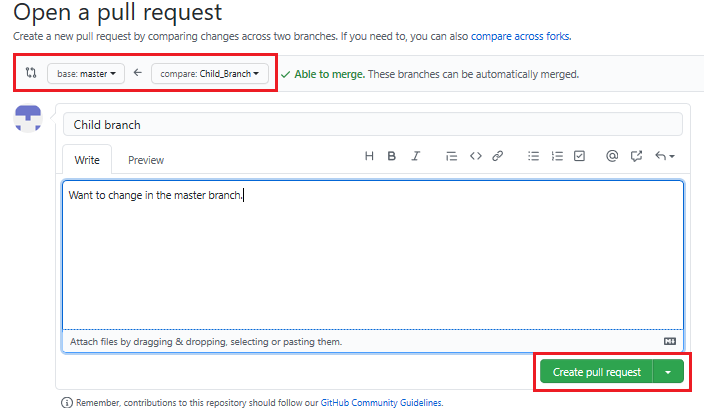
Or you can also compare the file as shown below with the option given Compare and pull request.



Step 1: Scroll down the page, you will get 1 file changed

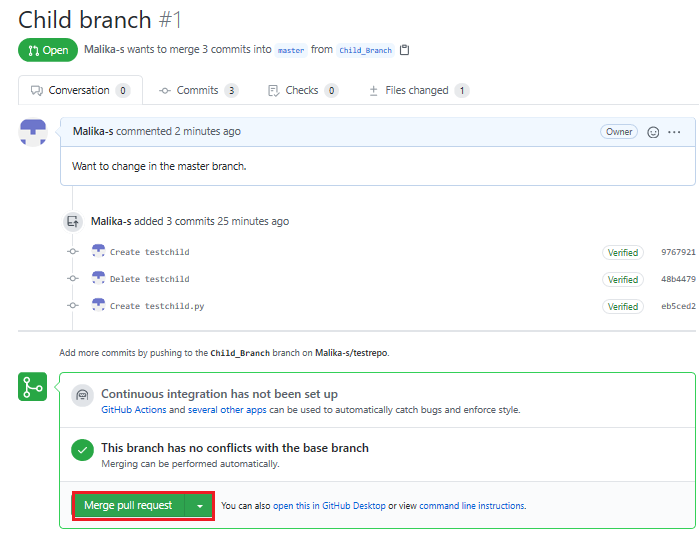


Step 2: Scroll up and create a pull request using the option Create Pull request. In the highlight, you can see the arrow which means that you are comparing and creating a pull request. Add the comments (optional) to create a request.

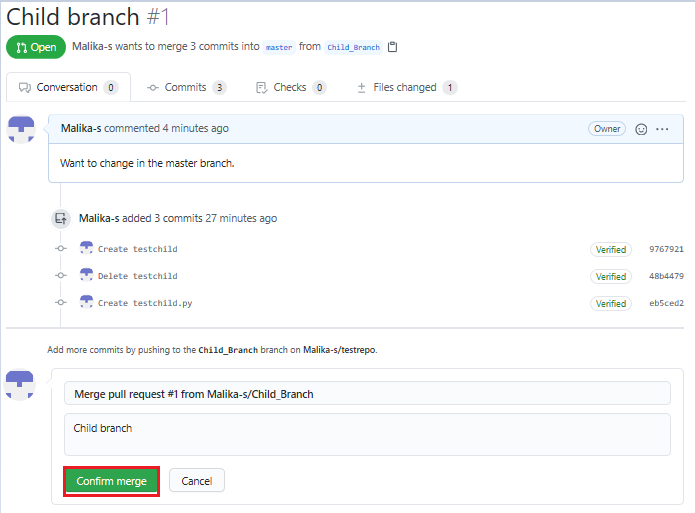


## **Exercise 4: Merge the Pull Request**

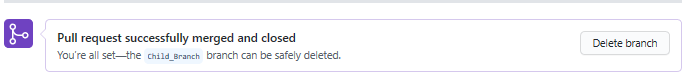
To accept the pull request, click the Pull Requests tab to see a summary of pending pull requests. If you are happy with the changes, click Merge Pull request to accept the pull request and perform the merge. You can add a comment if you want.



* Once you click Merge Pull request, you will see a button Confirm merge.



Your request has now merged successfully.



Now, the child branch has completely merged with the Master branch. You can check the Master branch is having the testchild.py file.

