# Lab Session 7: Developers Collaboration Using GitHub

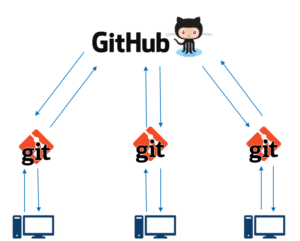
## **What is GitHub?**

GitHub is a web-based platform and a highly used software that is typically used for version control. It is helpful when more than just one person is working on a project. Say for example, a software developer team wants to build a website and everyone has to update their codes simultaneously while working on the project. In this case, GitHub helps them to build a centralized repository where everyone can upload, edit, and manage the code files.

### Why is GitHub so popular?

GitHub has various advantages but many people often have a doubt as to why not use drop box or any cloud-based system? Let me take the same example forward to answer this question. Say more than two software developers are working on the same file and they want to update it simultaneously. Unfortunately, the person who save the file first will get precedence over the others. While in GitHub, this is not the case. GitHub document the changes and reflect them in an organized manner to avoid any chaos between any of the files uploaded.

Therefore, using GitHub centralized repository, it avoids all the confusion and working on the same code becomes very easy.



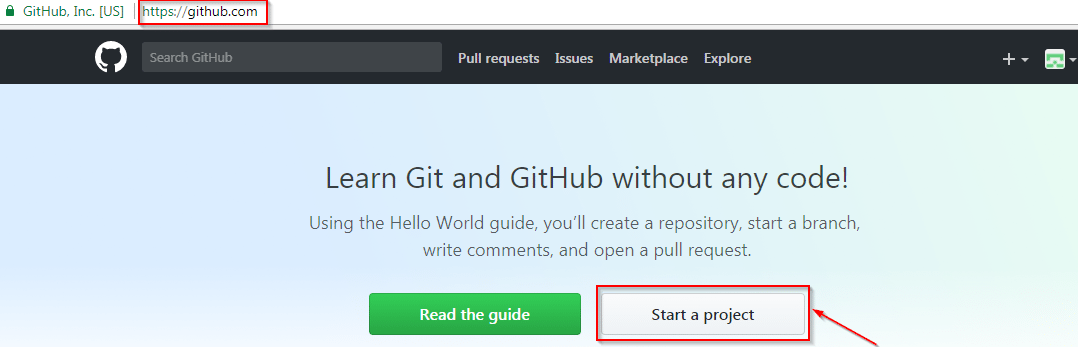
If you look at the image above, GitHub is a central repository and Git is a tool which allows you to create a local repository. Now people usually get confused between git and GitHub but it’s actually very different. Git is a version control tool that will allow you to perform all kinds of operations to fetch data from the central server or push data to it whereas GitHub is a core hosting platform for version control collaboration. GitHub is a company that allows you to host a central repository in a remote server.

## **How to create a GitHub Repository?**

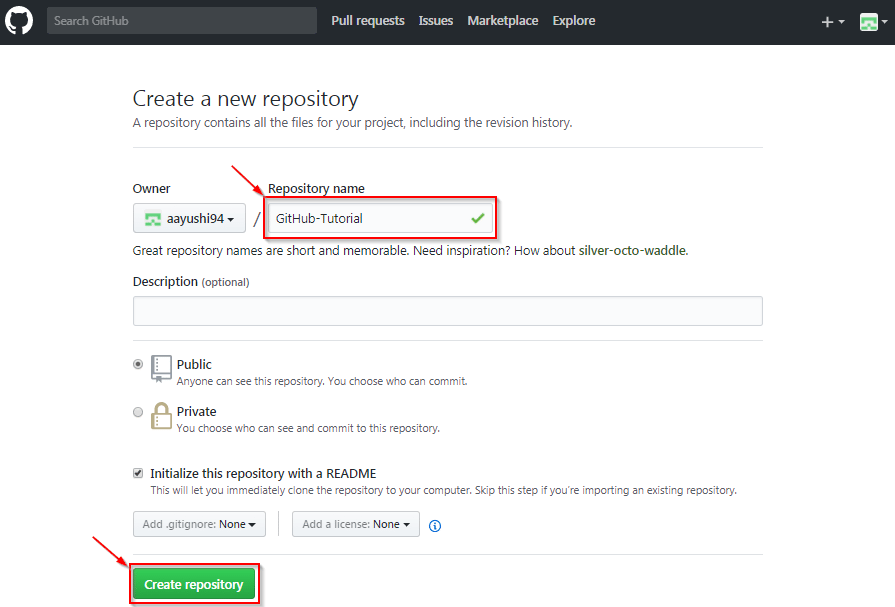
A repository is a storage space where your project lives. It can be local to a folder on your computer, or it can be a storage space on GitHub or another online host. You can keep code files, text files, images or any kind of a file in a repository. You need a GitHub repository when you have done some changes and are ready to be uploaded. This GitHub repository acts as your remote repository. So let make your task easy, just follow these simple steps to create a GitHub repository:

1. Go to the link: <https://github.com/> . Fill the sign up form and click on “Sign up for Github”.
2. Click on “Start a new project”.

Refer to the below screenshot to get a better understanding.

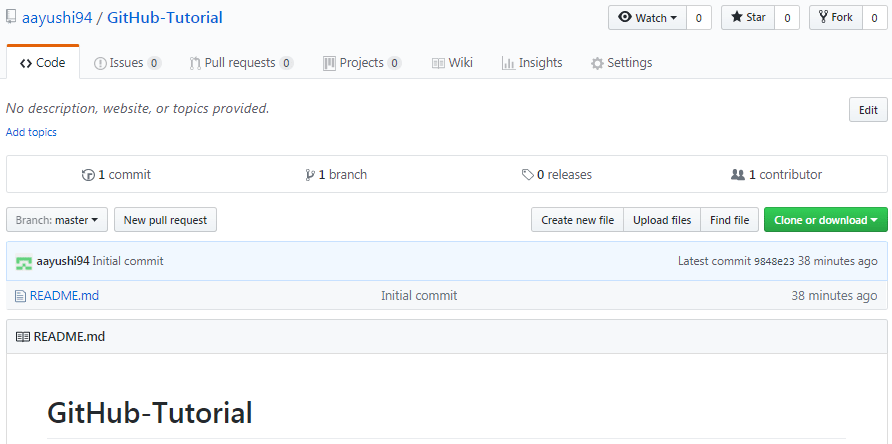


1. Enter any repository name and click on “Create Repository”. You can also give a description to your repository (optional).



Now, if you noticed by default a GitHub repository is public which means that anyone can view the contents of this repository whereas in a private repository, you can choose who can view the content. Also, private repository is a paid version. Also, if you refer the above screenshot, initialize the repository with a README file. This file contains the description of the file and once you check this box, this will be the first file inside your repository.

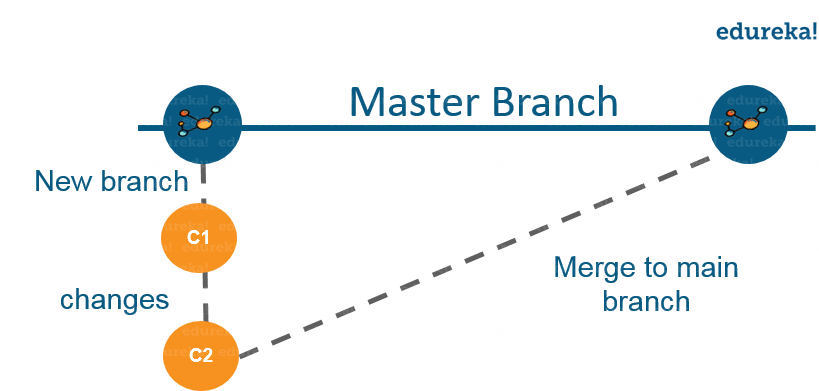
Congratulations, your repository is successfully created! It will look like the below screenshot:



So now central repository has been successfully created! Once this is done, you are ready to commit, pull, push and perform all the other operations. Now let’s move forward and understand branching in GitHub.

## **Create Branches and Perform Operations**

**Branching:**Branches help you to work on different versions of a repository at one time. Let’s say you want to add a new feature (which is in the development phase), and you are afraid at the same time whether to make changes to your main project or not. This is where git branching comes to rescue. Branches allow you to move back and forth between the different states/versions of a project. In the above scenario, you can create a new branch and test the new feature without affecting the main branch. Once you are done with it, you can merge the changes from new branch to the main branch. Here the main branch is the master branch, which is there in your repository by default. Refer to the below image for better understanding:

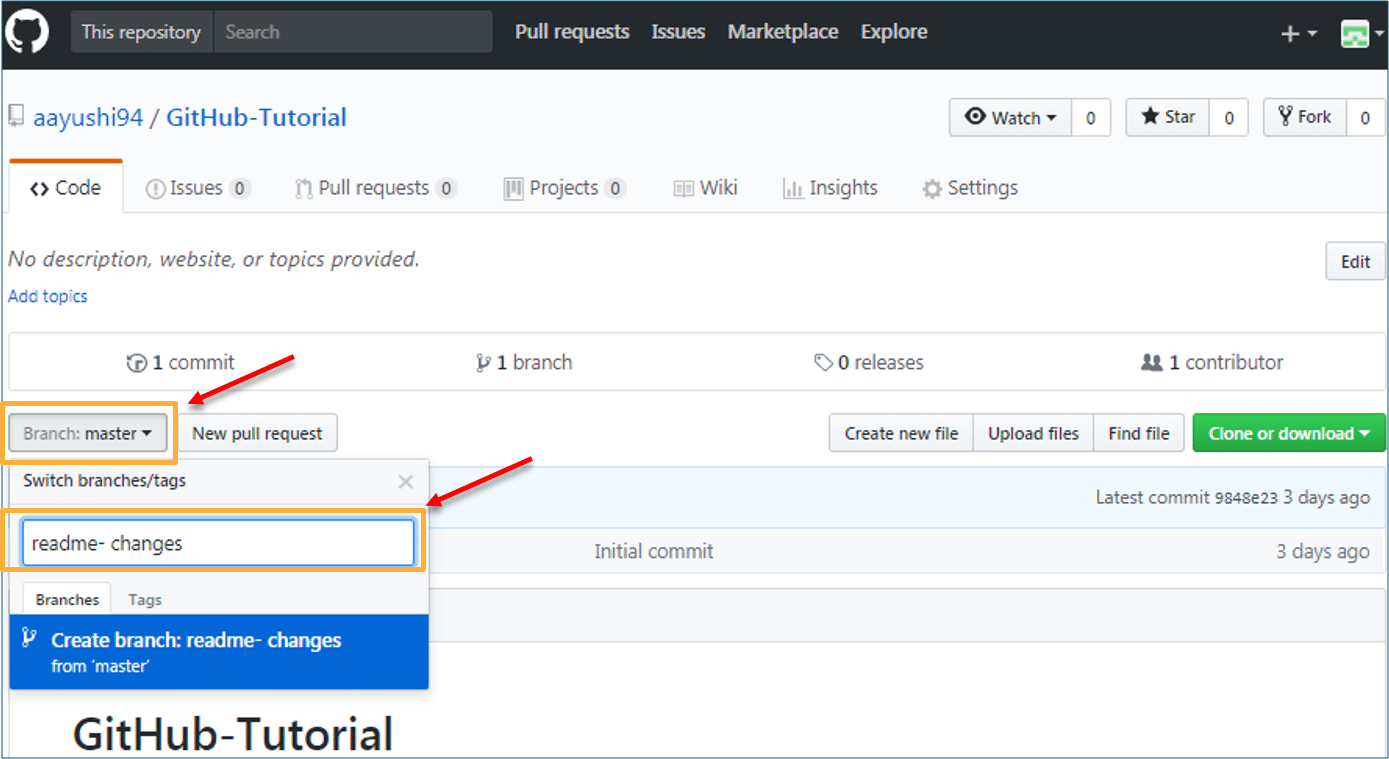


As depicted in the above image, there is a master/ production branch which has a new branch for testing. Under this branch, two set of changes are done and once it completed, it is merged back to the master branch. So, this is how branching works!

Let’s move ahead in ‘how to use GitHub’ blog, and learn how you can create a branch.

To create a branch in GitHub, follow the below steps:

1. Click on the dropdown “Branch: master”
2. As soon as you click on the branch, you can find an existing branch or you can create a new one. In my case, I am creating a new branch with a name “readme- changes”. Refer to the below screenshot for better understanding.



Once you have created a new branch, you have two branches in your repository now i.e., read-me (master branch) and readme- changes. The new branch is just the copy of master branch. So, let’s perform some changes in our new branch and make it look different from the master branch.

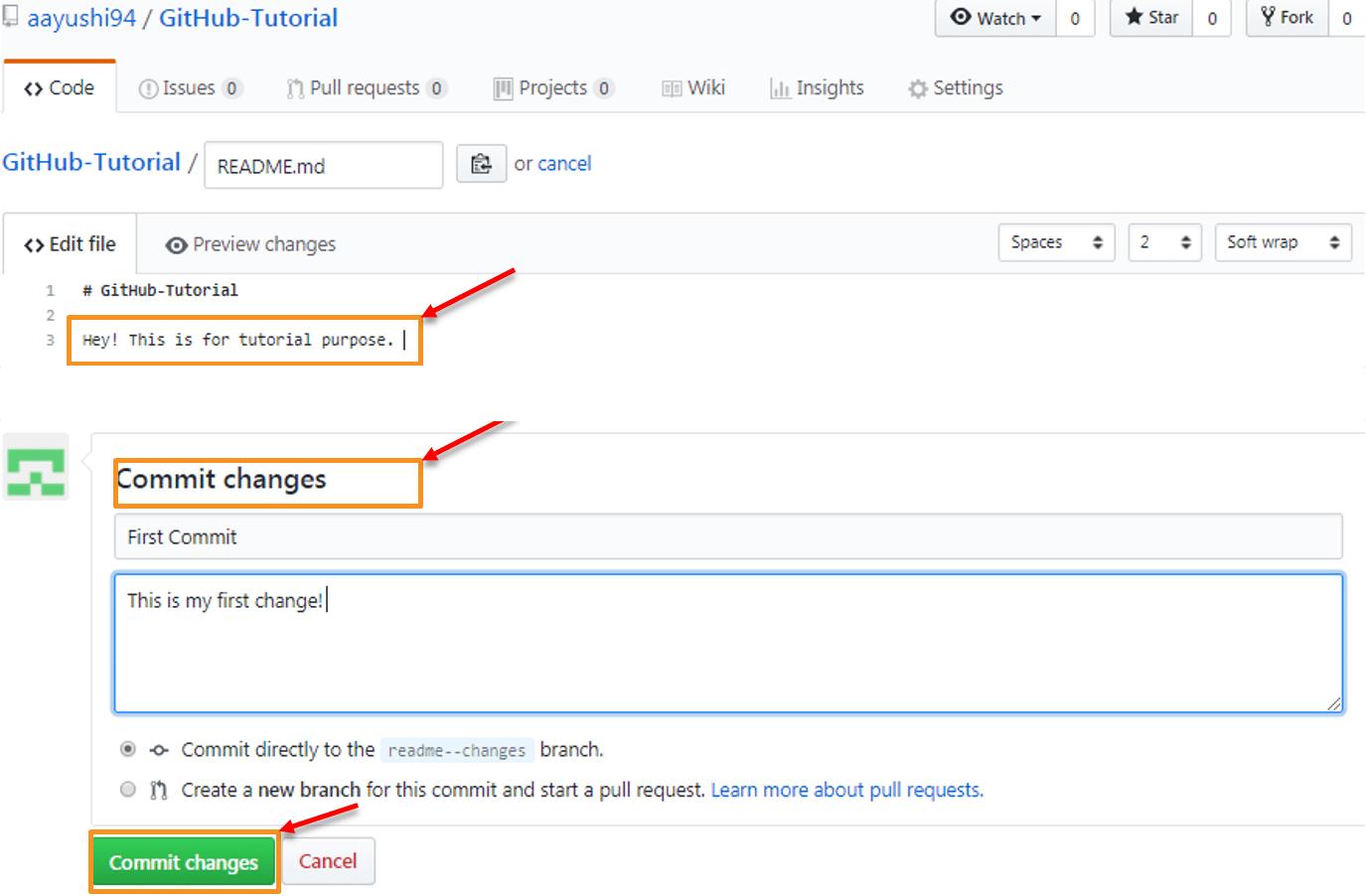
### GitHub Operations

**Commit Command**

This operation helps you to save the changes in your file. When you commit a file, you should always provide the message, just to keep in the mind the changes done by you. Though this message is not compulsory but it is always recommended so that it can differentiate the various versions or commits you have done so far to your repository. These commit messages maintain the history of changes which in turn help other contributors to understand the file better. Now let’s make our first commit, follow the below steps:

1. Click on “readme- changes” file which we have just created.
2. Click on the “edit” or a pencil icon in the rightmost corner of the file.
3. Once you click on that, an editor will open where you can type in the changes or anything.
4. Write a commit message which identifies your changes.
5. Click commit changes in the end.

Refer to the below screenshot for better understanding:



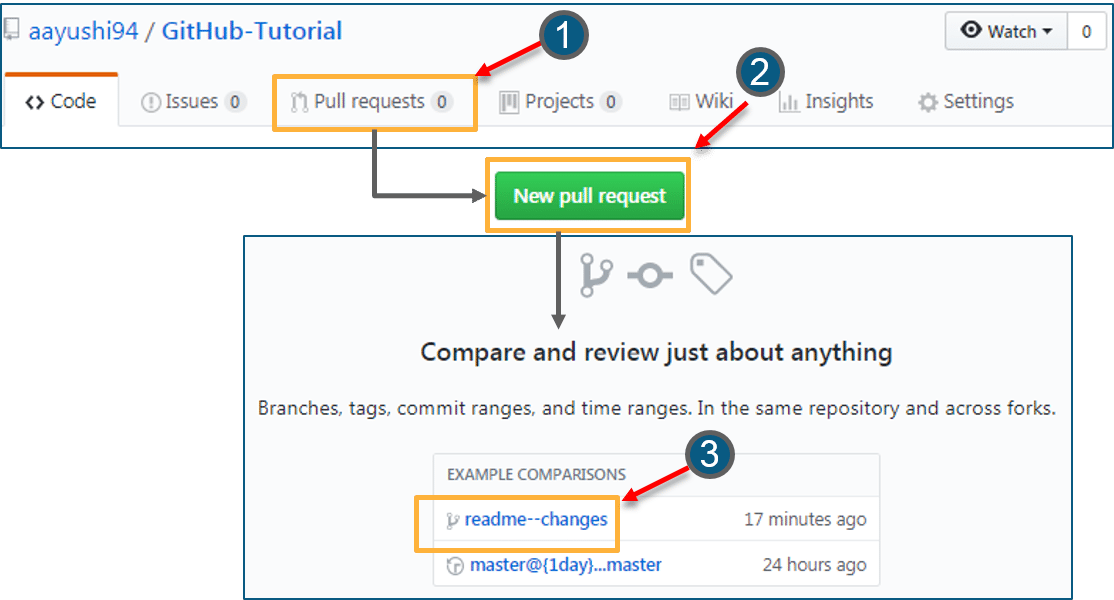
We have successfully made our first commit. Now this “readme- changes” file is different from the master branch. Next, let us see how can we open a pull request.

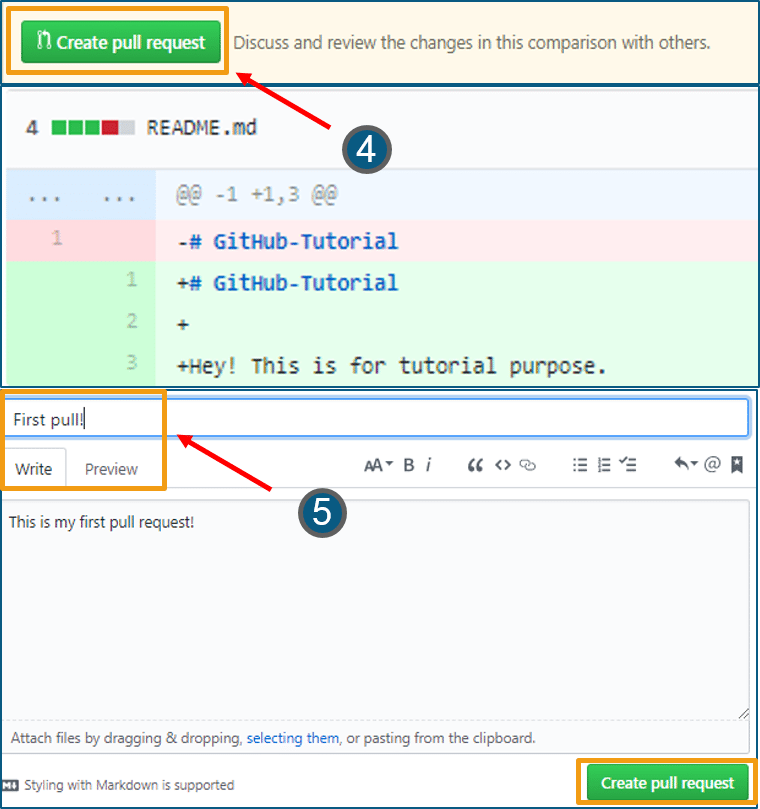
**Pull Command**

Pull command is the most important command in GitHub. It tells the changes done in the file and request other contributors to view it as well as merge it with the master branch. Once the commit is done, anyone can pull the file and can start a discussion over it. Once it’s all done, you can merge the file. Pull command compares the changes which are done in the file and if there are any conflicts, you can manually resolve it.

Now let us see different steps involved to pull request in GitHub.

1. Click the ‘Pull requests’ tab.
2. Click ‘New pull request’.
3. Once you click on pull request, select the branch and click ‘readme- changes’ file to view changes between the two files present in our repository.
4. Click “Create pull request”.
5. Enter any title, description to your changes and click on “Create pull request”. Refer to the below screenshots.





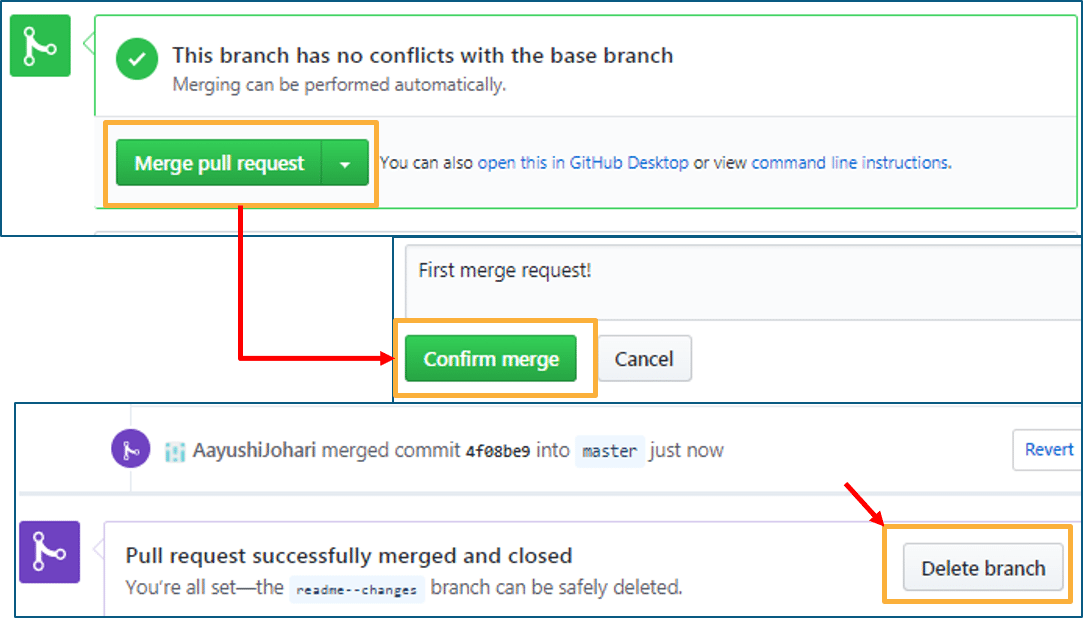
Next, let us move forward and see how can you merge your pull request.

**Merge Command**

Here comes the last command which merge the changes into the main master branch. We saw the changes in pink and green color, now let’s merge the “readme- changes” file with the master branch/ read-me. Go through the below steps to merge pull request.

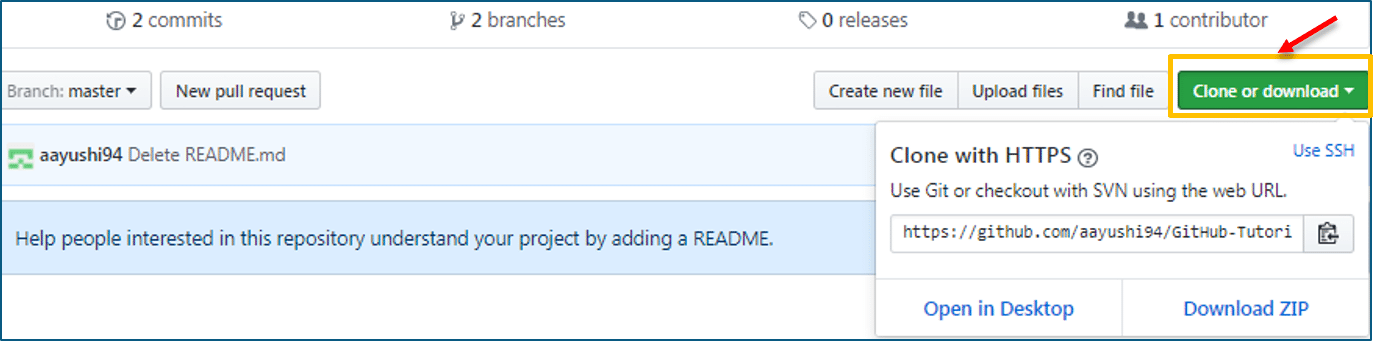
1. Click on “Merge pull request” to merge the changes into master branch.
2. Click “Confirm merge”.

You can delete the branch once all the changes have been incorporated and if there are no conflicts. Refer to the below screenshots.



## **Cloning and Forking GitHub Repository**

**Cloning:** First let us understand why do we need to clone a repository. The answer is simple! Suppose you want to use some code which is present in a public repository, you can directly copy the contents by cloning or downloading. Refer to the below screenshot for a better understanding.



Cloning is really simple! In case you are facing any challenges on how to use GitHub, please comment your problems in the section below. Moving forward, let’s see what forking is.

**Forking:** First, let us talk about why do we need forking. Suppose, you need some code which is present in a public repository, under your repository and GitHub account. For this, we need to fork a repository.

Before we get started with forking, there are some important points which you should always keep in mind.

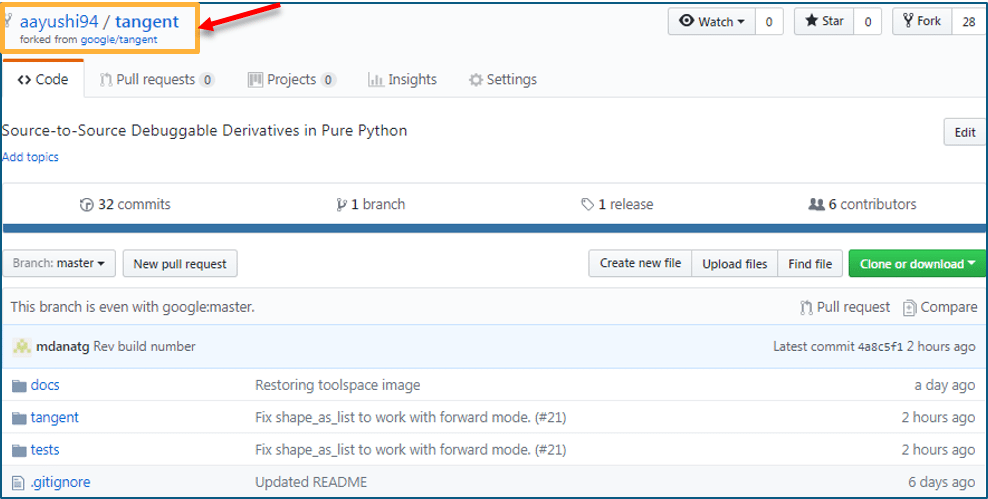
* Changes done to the original repository will be reflected back to the forked repository.
* If you make a change in forked repository, it will not be reflected to the original repository until and unless you have made a pull request.

Now let’s see how can you want to fork a repository. For that, follow the below steps:

1. Go to Explore and search for public repositories.
2. Click “fork”. Note that this “tangent” repository is already forked 27 times and it is under “google” account. Refer the below image for better understanding.



As soon as you click on “Fork”, it will take some time to fork the repository. Once done you will notice that the repository name is under your account. For reference, you can have a look at the below screenshot.



## **Working with Files**

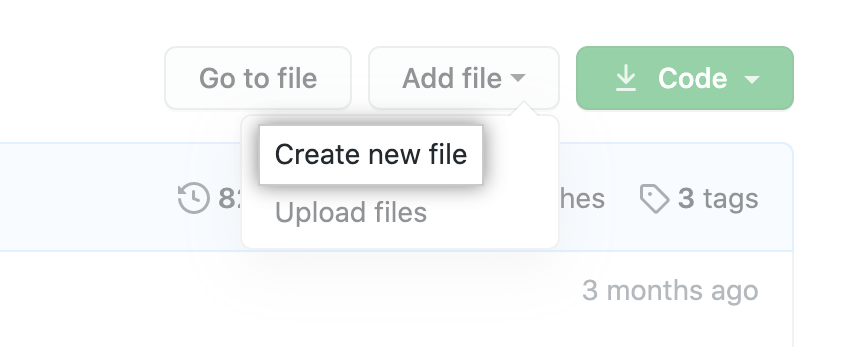
### Creating new files

You can create new files directly on GitHub in any repository you have write access to. When creating a file on GitHub, consider the following:

If you try to create a new file in a repository that you don’t have access to, we will fork the project to your user account and help you send [a pull request](https://docs.github.com/en/articles/about-pull-requests) to the original repository after you commit your change.

File names created via the web interface can only contain alphanumeric characters and hyphens (-).

1. On GitHub, navigate to the main page of the repository.
2. In your repository, browse to the folder where you want to create a file.
3. Above the list of files, using the Add file drop-down, click Create new file.



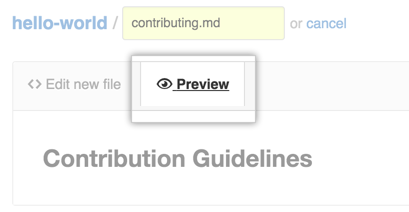
1. In the file name field, type the name and extension for the file. To create subdirectories, type the / directory separator.



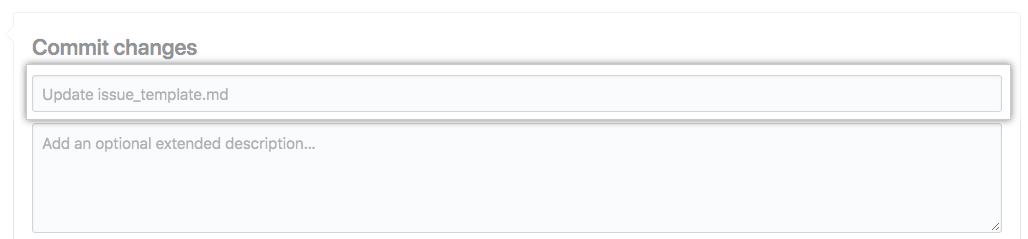
1. On the **Edit new file** tab, add content to the file.



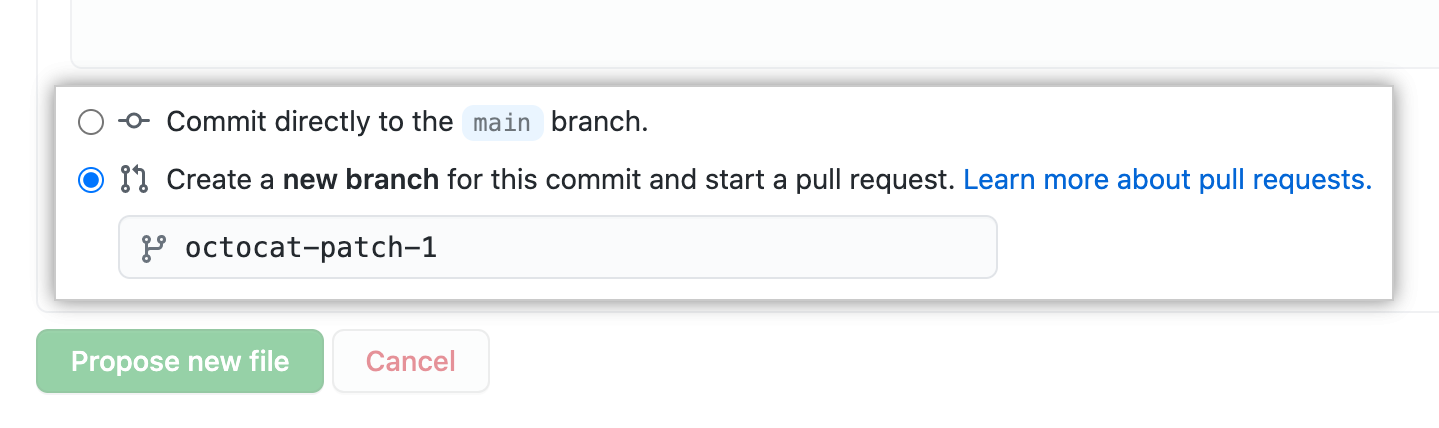
1. To review the new content, click **Preview**.



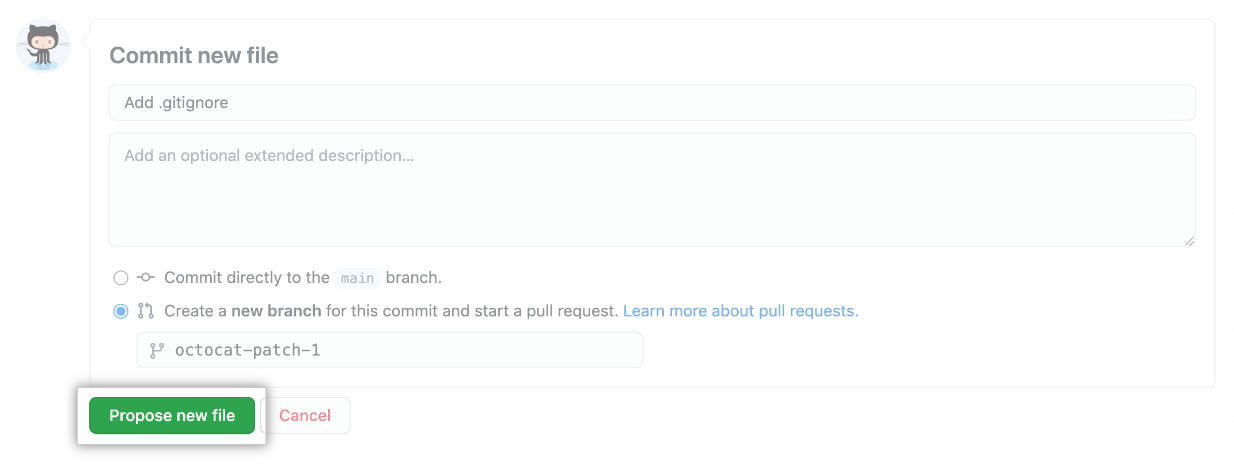
1. At the bottom of the page, type a short, meaningful commit message that describes the change you made to the file.



1. 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.



1. Click **Propose new file.**

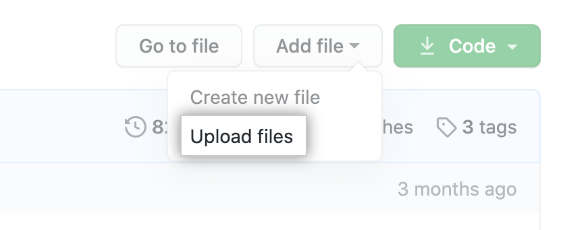


### Adding a file to a repository

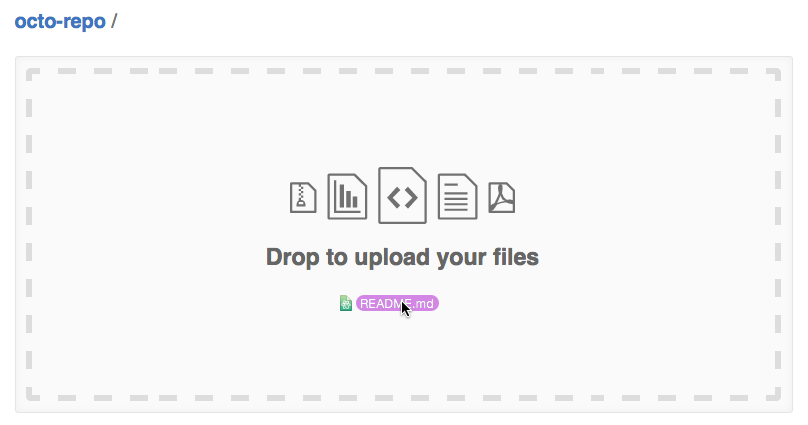
You can upload and commit an existing file to a repository on GitHub or by using the command line.

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.

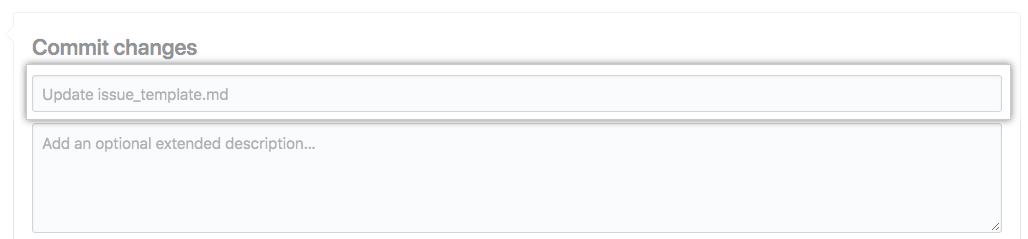
1. On GitHub, navigate to the main page of the repository.
2. Above the list of files, using the **Add file** drop-down, click **Upload files**.



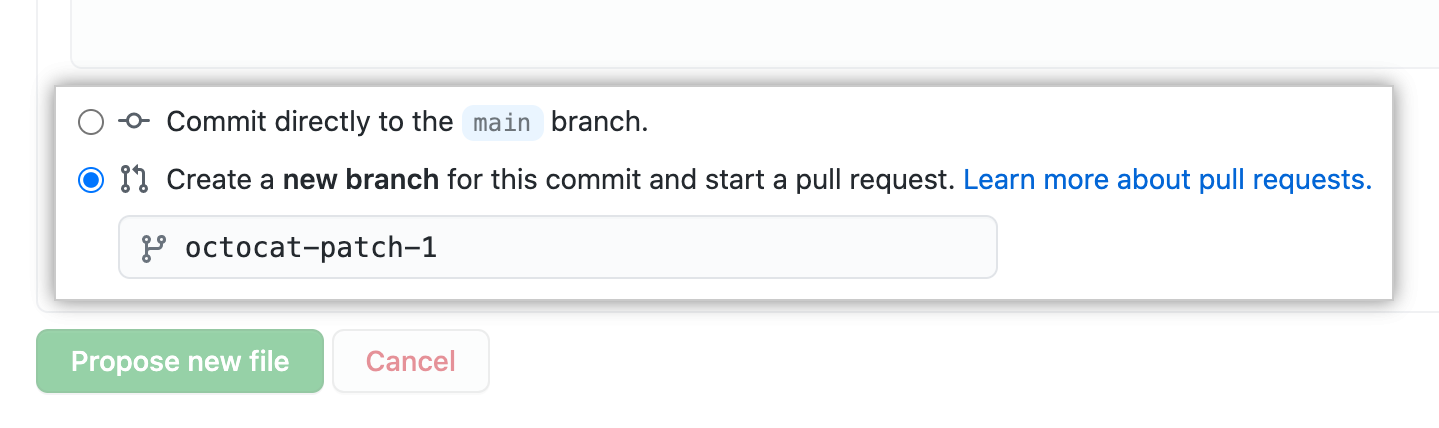
1. Drag and drop the file or folder you'd like to upload to your repository onto the file tree.



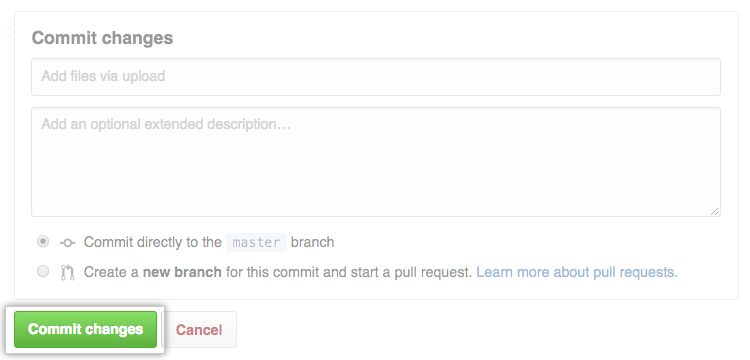
1. At the bottom of the page, type a short, meaningful commit message that describes the change you made to the file. You can attribute the commit to more than one author in the commit message.



1. 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.



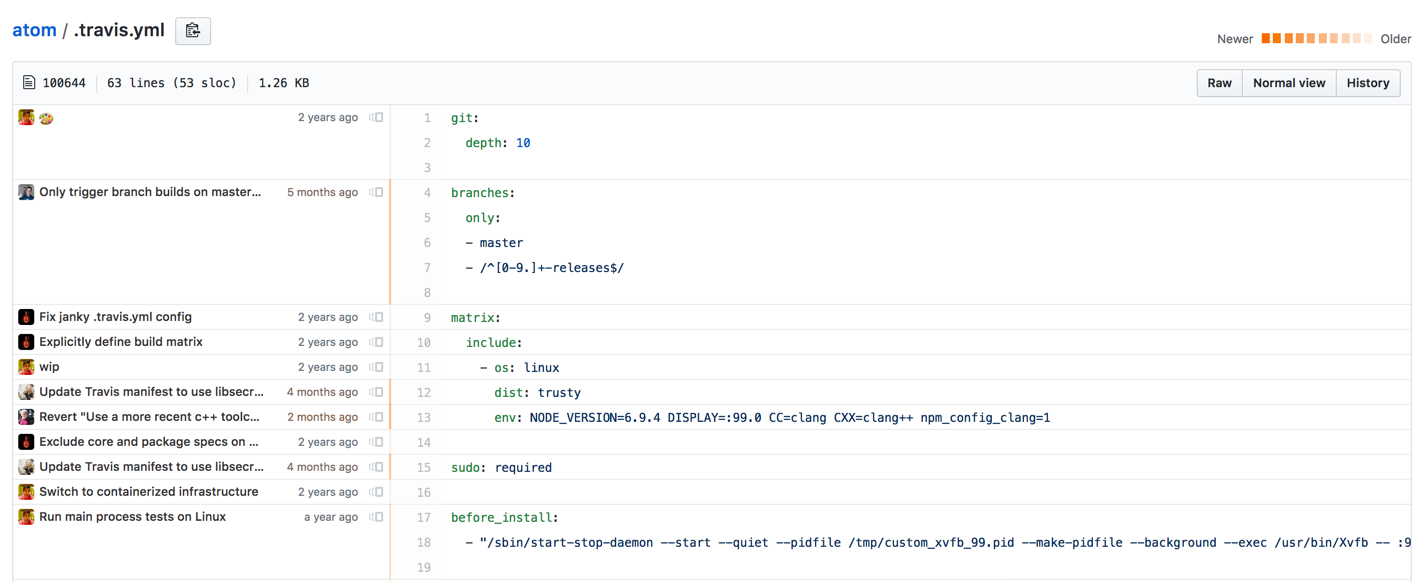
1. Click **Commit changes**.



### Tracking changes in a file

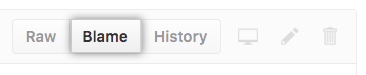
You can trace changes to lines in a file and discover how parts of the file evolved over time.

With the blame view, you can view the line-by-line revision history for an entire file, or view the revision history of a single line within a file by clicking. Each time you click, you'll see the previous revision information for that line, including who committed the change and when.



In a file or pull request, you can also use the menu to view Git blame for a selected line or range of lines.

1. On GitHub, navigate to the main page of the repository.
2. Click to open the file whose line history you want to view.
3. In the upper-right corner of the file view, click Blame to open the blame view.



1. To see earlier revisions of a specific line, or reblame, click until you've found the changes you're interested in viewing.



## **Exercise**

1. Log in to your GitHub account at <https://github.com/>.
2. Create a new repository either by clicking the green **New** button or from the dropdown menu on the top right.
3. Name the repository **Exercise1**.
4. Select the checkbox to Add a README file, then click Create repository.
5. Edit the README.md document
6. Add a line with some extra information.
7. Commit the changes.
8. Fork an existing repository (try this: https://github.com/seankross) and try to identify something valuable you could contribute. Make changes or additions to that repository, then open a pull request.