Published in Knowledge Brewery

Archit Singh Follow

Jun 9, 2020 · 9 min read · Listen

Introduction to git and GitHub

A photo from FaisalWeb.com

Just started with web development and need to learn about git and GitHub? Working on a project and need to collaborate with your team members? Just realized that you pretty much need to know how to use git if you seriously want to dive deep into tech?

If yes, then this one is for you !!

Frankly speaking, getting familiar with git and GitHub is not that complicated as it may look. You don't need to be some kind of master coder or anything like that to start learning git.

Once you're done with the article, you'll get to know about stuff like creating your own repository, creating branches and how to make changes to them, creating push and pull requests and a lot more. You'll also get to know about the commands that are frequently used while working with git. in this article, we'll only be covering the basics of git. There's a lot of stuff to learn if you want to use Git and GitHub like a pro, of course. You can go way beyond this introductory information! I'm going to leave the next-level stuff for another time, though.

Easier said than done, lets get started !!

What is git?

A photo from cloudsavvyit

Git is a free and **open-source distributed version control system** for tracking changes in source code and is used for software development. It was created by **Linus Torvalds (creator and developer of Linux)** in 2005. It allows you to record changes in your file over time so that you can access specific versions of your file later on. Moreover, you can also coordinate with several people working on a single file or files with the help of git. So git is basically a content tracker.

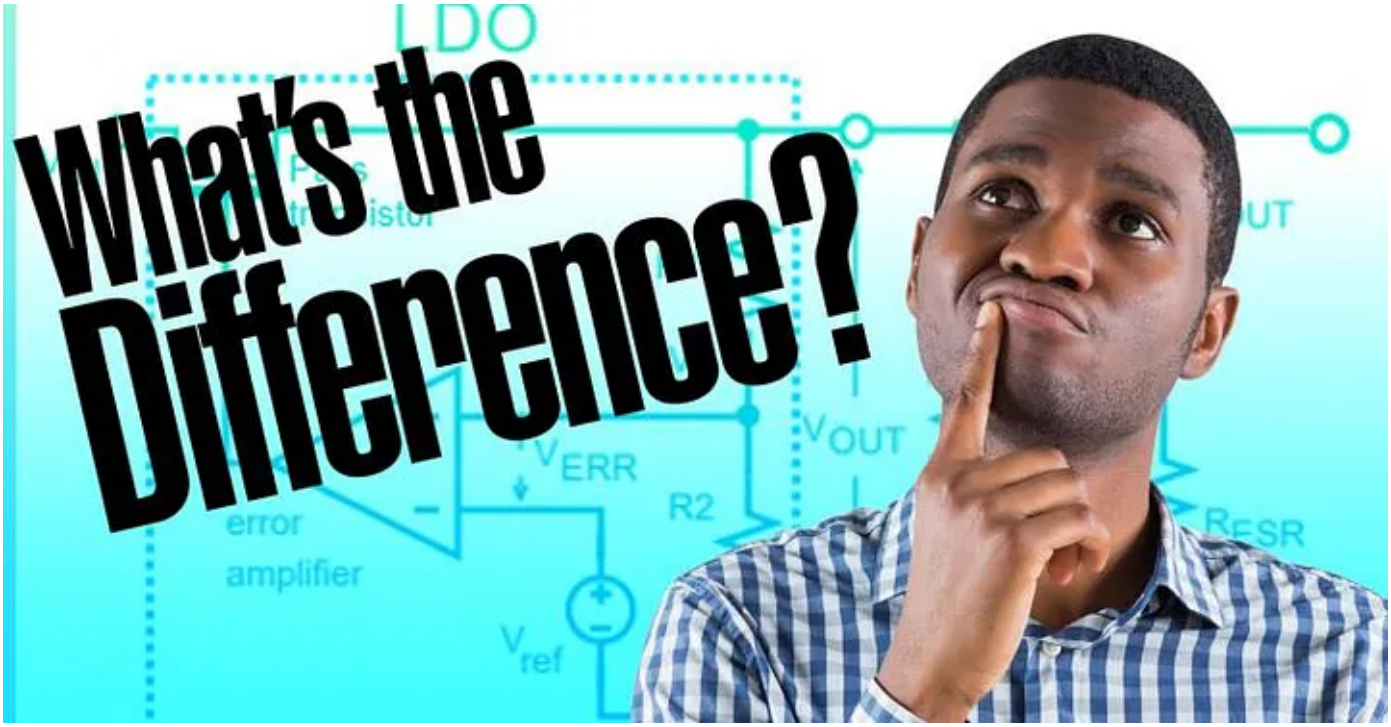
To put it all together, with git you can :

> *Record changes in your project and its files*
>
> *Revert back to previous states at different points in time*
>
> *Collaborate with multiple people on one codebase*
>
> *See changes over time*
>
> *Develop multiple features at once*

But what's GitHub then? Do they mean the same?

A photo from skyose

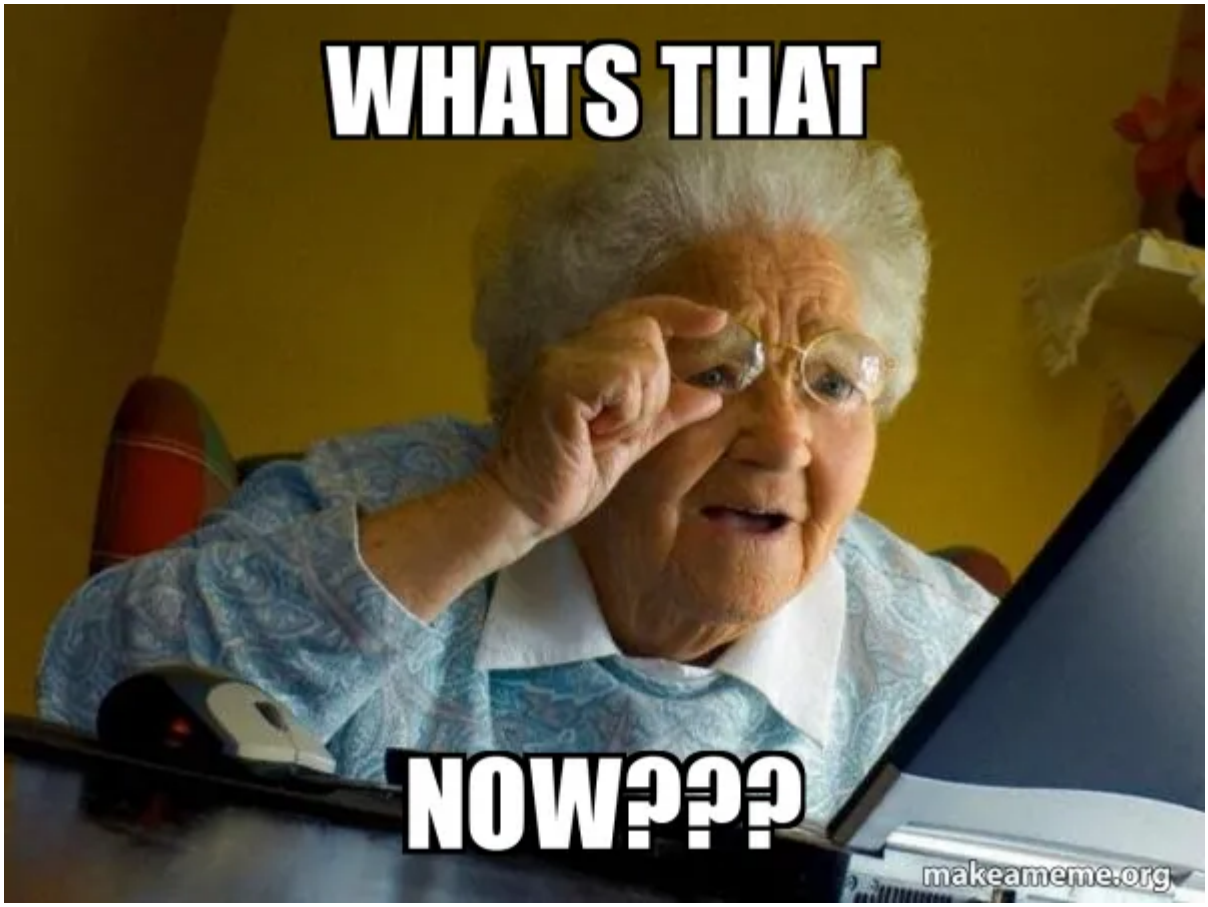
GitHub is a Git **repository** hosting service, but it adds many of its own features. While Git is a command-line tool, GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as wikis and basic task management tools for every project.



Simply put, Git is a version control system that lets you manage and keep track of your source code history. GitHub is a cloud-based hosting service that lets you manage Git repositories. If you have open-source projects that use Git, then GitHub is designed to help you better manage them.

Creating a repository

To start working with git, you need to have a **repository**.



According to the Oxford dictionary :

> *repository : a place where something is stored in large quantities. a person or book that is full of information.*

Your **repository** is where you'll organize your project. You can keep folders, files, images, videos, spreadsheets, Jupyter notebooks, data sets, and anything else your project needs. Before you can work with Git, you have to initialize a repository for your project and set it up so that Git will manage it.

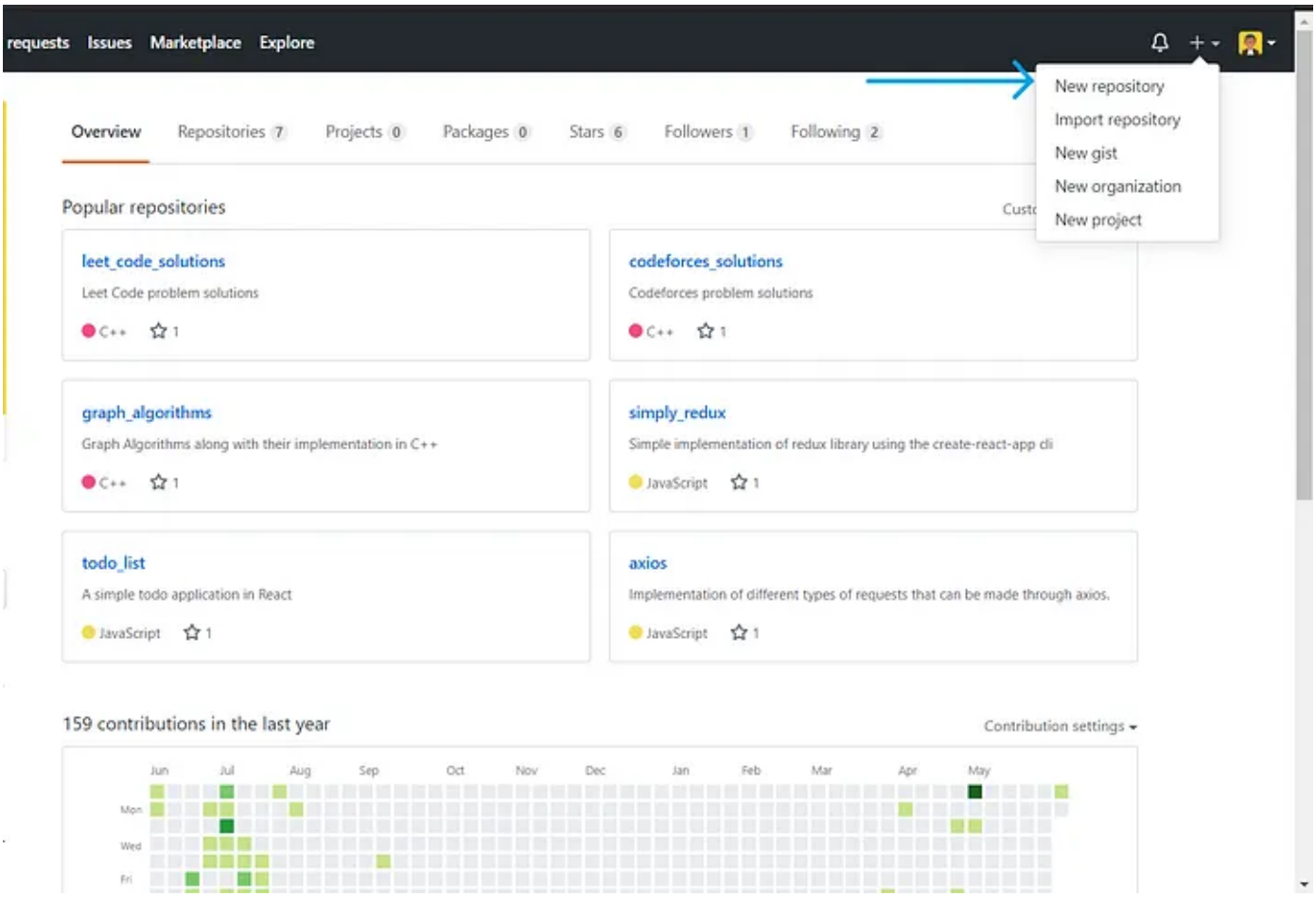
Now in terms of git, a repository is the **.git/** folder inside a project. This repository tracks all changes made to files in your project, building history over time. Meaning, if you delete the .git/ folder, then you delete your project's history.

How can we create one?

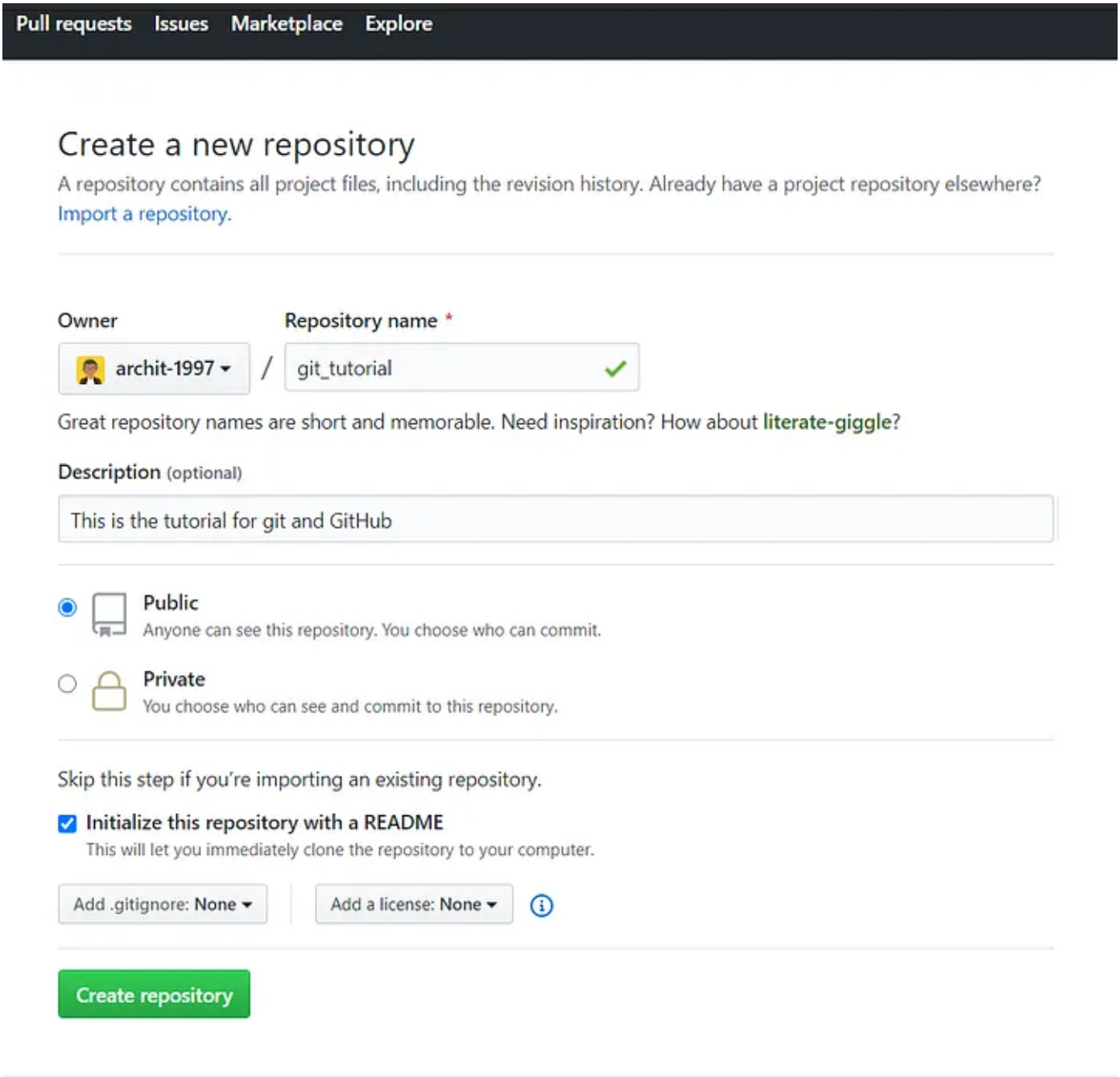
Creating a repository in GitHub is pretty simple, whereas in git you'll follow a sequence of commands to do that.

Follow the steps to create a repository on GitHub

- GitHub website, look in the upper right corner, and click the + sign and then click "New repository.



- Name the repository, and add a quick description.
- Decide whether you want this to be a public or a private repository
- Select "Initialize this repository with a README" option if you want to include the **README** file. (I definitely recommend doing this! It's the first thing people are going to look at when they check out your repository. It's also a great place to put information that you need to have in order to understand or run the project



- And there you go, you've created a new repository.

Follow the steps to create a repository using git CLI

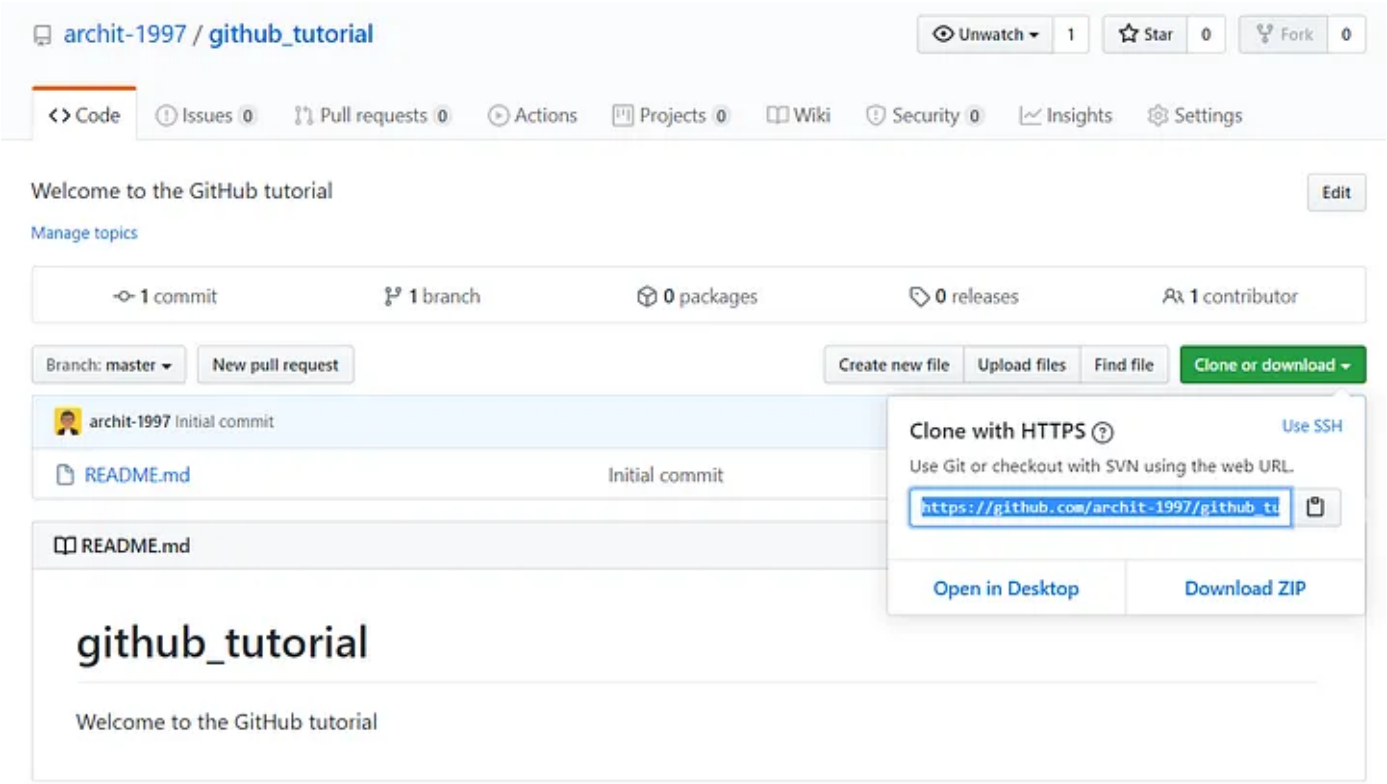
- Head over to the folder where you want to create a repository and create a folder with the respective name.
- Go into the directory
- Type git init
- This was easy and simple though !! :-)

Now, since we had skipped the cloning part of the repository, let's learn about that also.

Cloning a repository

If a project has already been set up in a central repository, the **git clone** command is the most common way for users to obtain a development copy. Like git init, cloning is generally a one-time operation. Once a developer has obtained a working copy, all version control operations and collaborations are managed through their local repository. It is primarily used to point to an existing *repo (repository)* and make a clone or copy of that repo at in a new directory, at another location. The original repository can be located on the local files system or on remote machine accessible supported protocols. The git clone command copies an existing Git repository.

To clone a repository means that you're taking a repository that's on the server and cloning it to your computer — just like downloading it. On the repository page, you need to get the "HTTPS" address.



Copy the address and use the following command on your terminal :

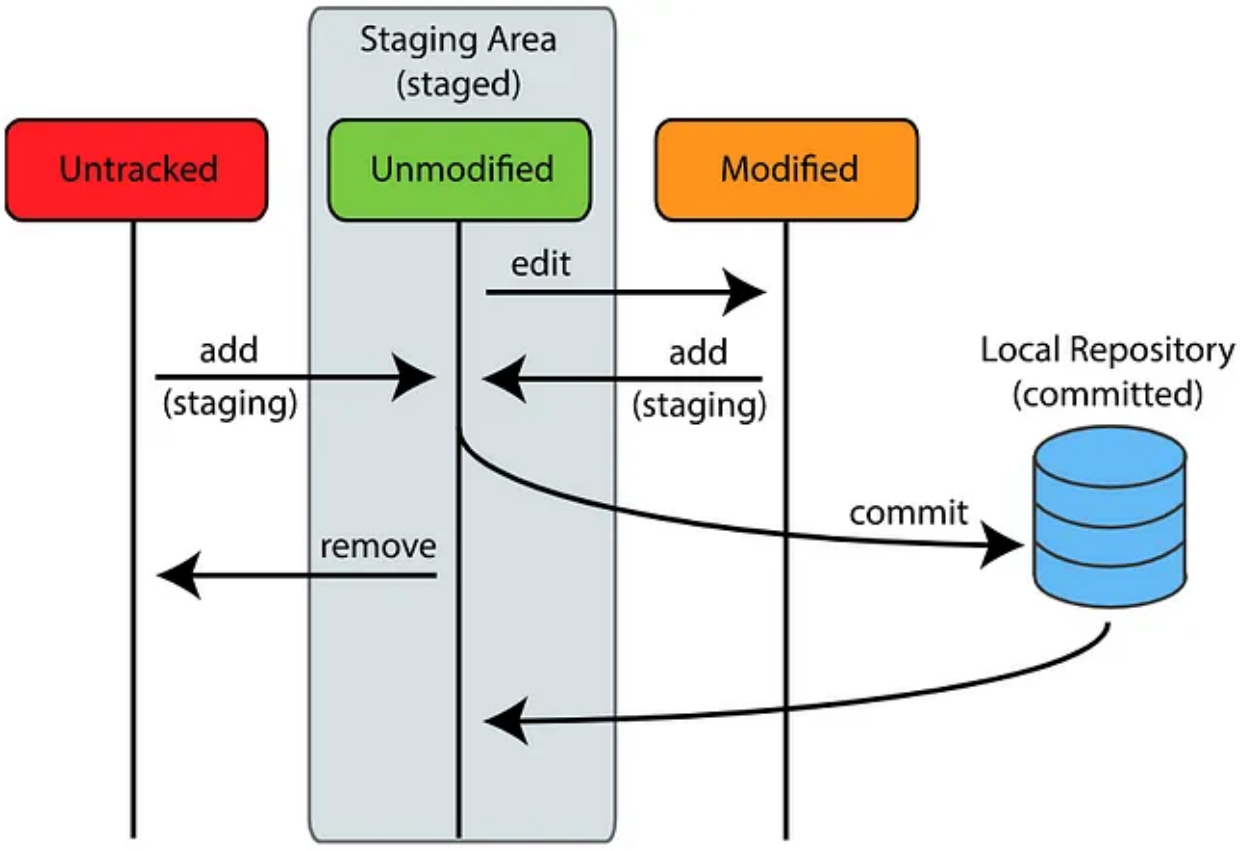
```
git clone "http-address that you have just copied"
```

Now your repository is on your system and you can move in with the following command:

```
cd "repository-name"
```

Now let's have a look at the states and areas of git

The three states and areas of git



States and stages of a file in git

A file in git can only have three states, namely :

- **committed**

- **modified/ untracked**
- **staged**

Modified/Untracked and your working directory

Git views untracked and modified files similarly. Untracked means that the file is new to your Git project. Modified means that the file has been seen before, but has been changed, so is not ready to be snapshotted by Git. Modification of a file occurs in your working directory.

Staged and staging area

When a file becomes staged, it's taken into the staging area. This is where Git is able to take a snapshot of it and store its current state to your local repository. This area is also known as the Index

Committed and the git directory (aka your local repository)

Committed means that Git has officially taken a snapshot of the files in the staging area, and stored a unique index in the Git directory. The terms snapshotted and committed are very similar. The significance of being committed is that you can now revert back to this project's current state at any time in the future.

The term for the very last snapshot you've made for commitment is known as the **HEAD**.

Basic Version Control

Navigate to the directory in which you had cloned the repository "github_tutorial" (you can use any other name as well) using the standard "cd" command. Now you can initialize a git repository with the following command:

```
git init
```

This creates a new subdirectory named **.git** that contains all of your necessary repository files — a Git repository skeleton. At this point, nothing in your project is tracked yet.

To start versioning new or any existing files, you should start by tracking those files and doing an initial commit. To accomplish that, you start by adding the files to git that you would like to be attached to your git project.

As of now, you won't be having any file in this repository because you haven't created any. So, create a sample text file in the very same directory and name it **test.txt**

Now to add this file, enter the following command:

```
git add "file-name"
```

The **git add** command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit.

If you want to add multiple files with a single command, use the git add command followed by a list of space-separated file names.

```
git add "file-name-1" "file-name-2" "file-name-3"
```

To add the files in your working directory, you can use the following command :

```
git add --all
```

You can see check the current state of the working directory and the staging area using the following command:

```
git status
```

This lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git. The status output does *not* show you any information regarding

the committed project history. For this, you need to use **git log (We'll talk about this later on)**

How to make your first commit in git?

```
git commit -m "commit-text"
```

If you want to commit all the files to the local repository with the same commit text, then you can use the following command :

```
git commit -am "commit-text"
```

So now, the changes that you had made in your working directory have been updated in your local repository as well.

Pushing updates to the local repository

Now you have to update the remote repository also with the respective changes. For this purpose, we use the git push command. The **git push** command is used to upload

local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo. After a local repository has been modified a push is executed to share the modifications with remote team members.

```
git push origin master
```

git push origin master will push your changes to the remote server. “master” refers to the master branch in your repository.If you want to push your changes to any other branch (say test-branch), you can do it by:

```
git push origin test-branch
```

This will push your code to the origin of test-branch in your repository.

So now you have a GitHub repository and you know how to add files and changes to it!

So now, you know how to use git and GitHub, how to create a repository and how to make commits to your local repository.

In the next post, we'll be talking about **The main game of git and GitHub: collaboration.** That means that we'll learn how to create branches, checking out of them and then finally merging them into the main branch or any other branch that we want. In addition to this, we'll also cover some of the advanced commands that we can use while working with git to make it much easier. There's a lot of stuff that you can do with git, it's just that you haven't explored it all.

Resources

- Official documentation: git
- Getting git right
- getting-started-with-git-and-github : towards data science

I'd love to hear about your suggestions if any in the comments section.

Thanks for sticking by. There a lot more to come. Thank you and have a nice day !!

Git Github Web Development Computer Science Programming
