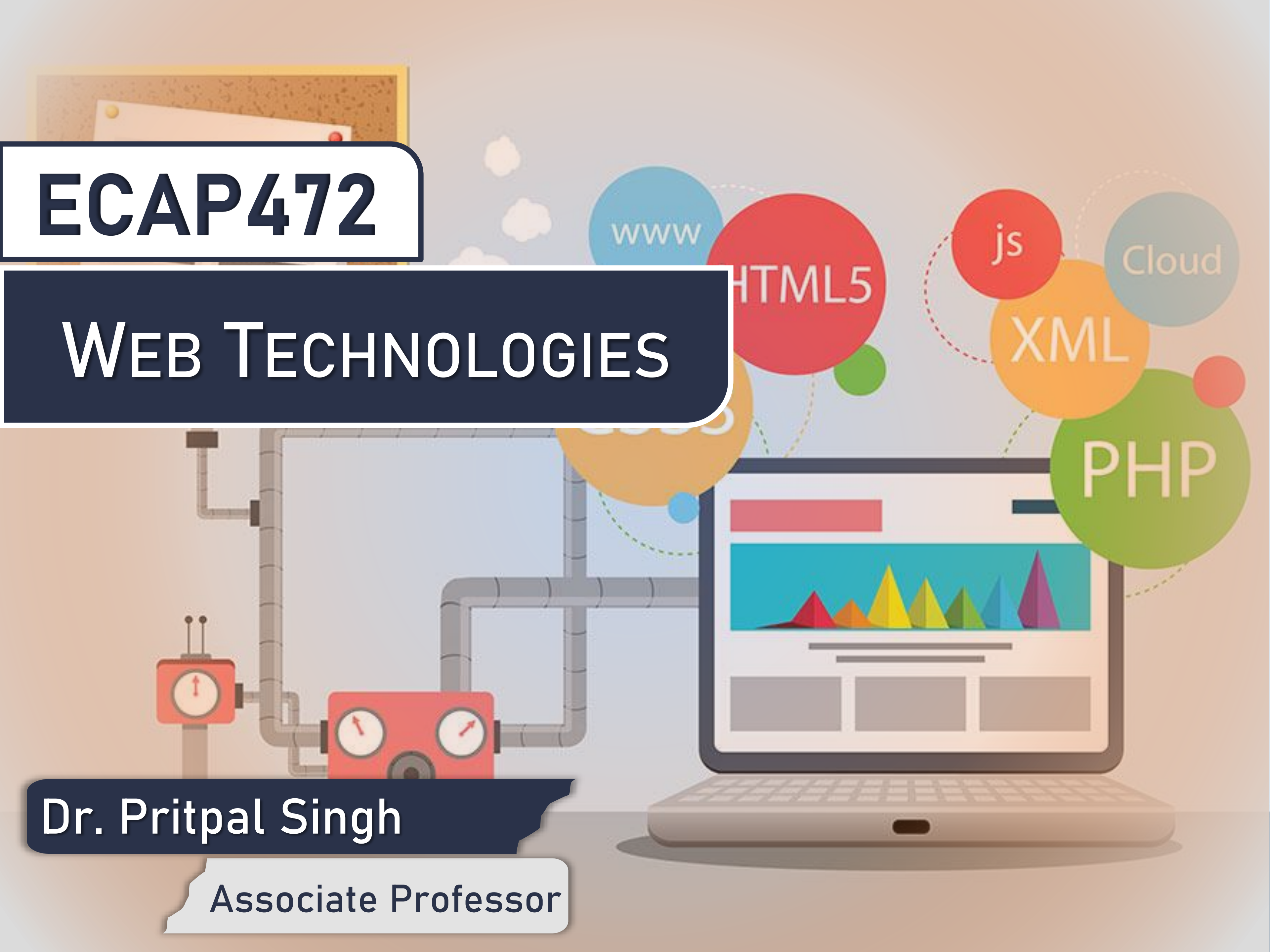
ECAP472

Web Technologies



Dr. Pritpal Singh

Associate Professor

Learning Outcomes

After this lecture, you will be able to

- walkthrough the basics of Git and GitHub concepts.

What is Version Control System (VCS)?

Version Control Systems are the software tools for tracking/managing all the changes made to the source code during the project development. It keeps a record of every single change made to the code.

It also allows us to turn back to the previous version of the code if any mistake is made in the current version. Without a VCS in place, it would not be possible to monitor the development of the project.

Types of VCS

The three types of VCS are:

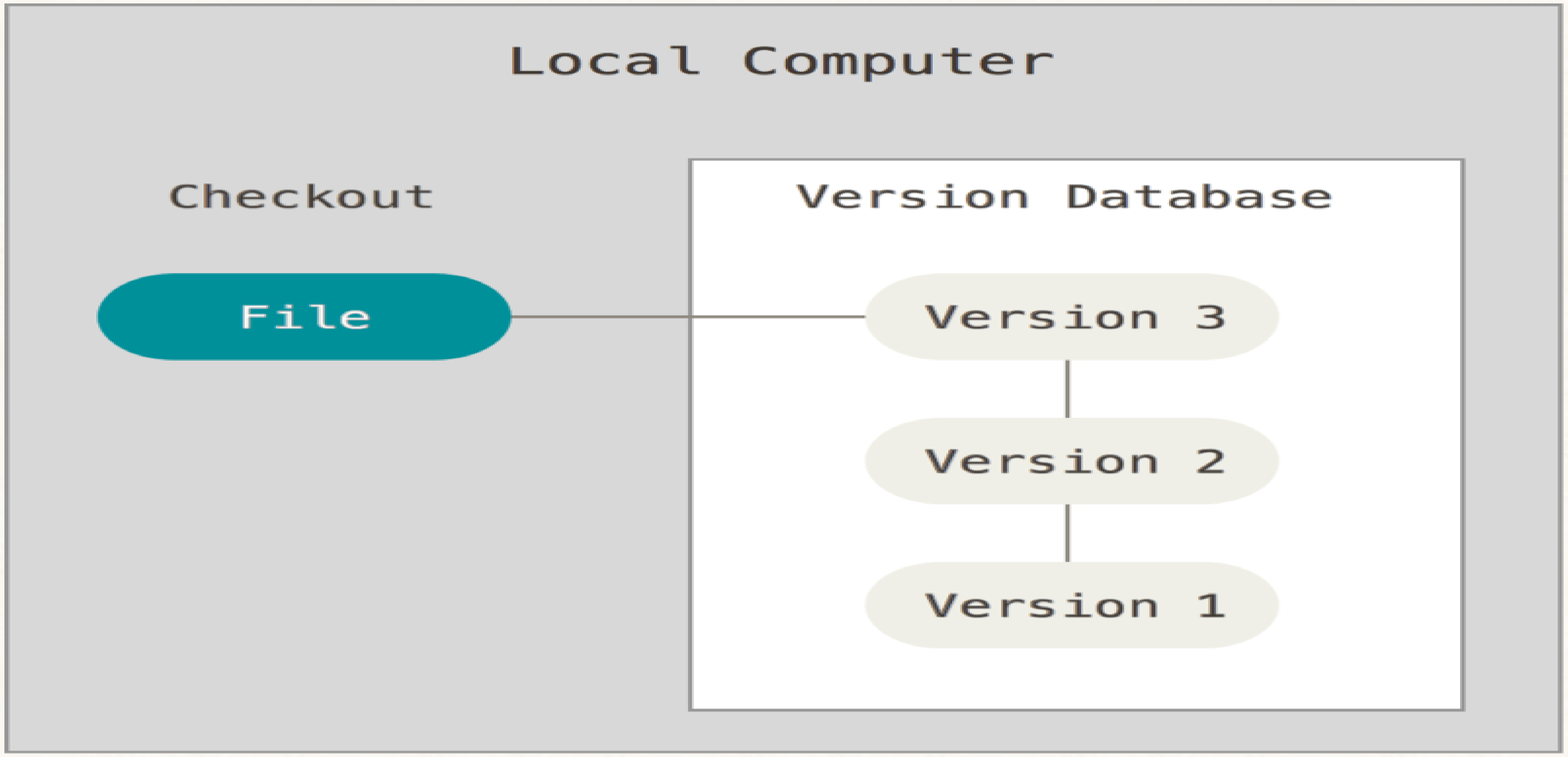
1. Local Version Control System
2. Centralized Version Control System
3. Distributed Version Control System

Local Version Control System

- Local Version Control System is located in your local machine. If the local machine crashes, it would not be possible to retrieve the files, and all the information will be lost. If anything happens to a single version, all the versions made after that will be lost.

Local Version Control System

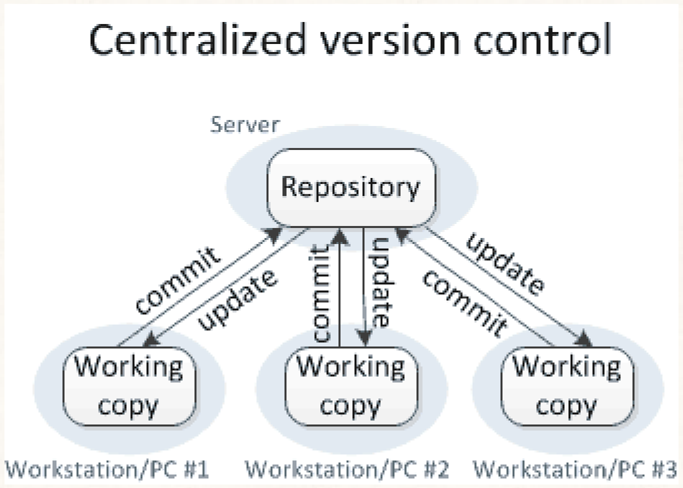
With the Local Version Control System, it is not possible to collaborate with other collaborators.



Centralized Version Control System

- To collaborate with other developers on other systems, Centralized Version Control Systems are developed.
- In the Centralized Version Control Systems, there will be a single central server that contains all the files related to the project, and many collaborators checkout files from this single server (you will only have a working copy).
- The problem with the Centralized Version Control Systems is if the central server crashes, almost everything related to the project will be lost.

Centralized Version Control System



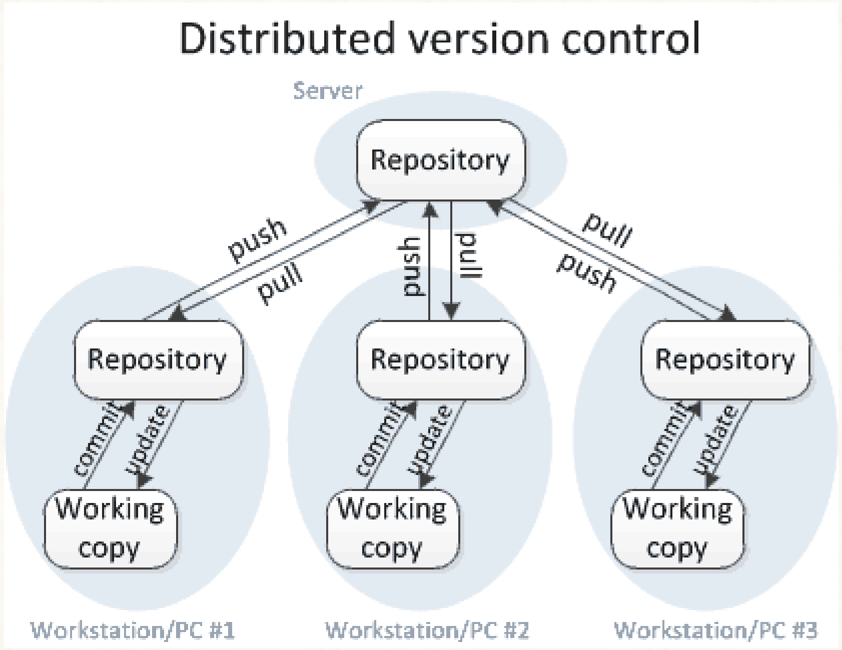
- To overcome all the above problems, Distributed Version Control Systems are developed.

Distributed Version Control System

- In a distributed version control system, there will be one or more servers and many collaborators similar to the centralized system.
- But the difference is, not only do they check out the latest version, but each collaborator will have an exact copy (mirroring) of the main repository(including its entire history) on their local machines.

Distributed Version Control System

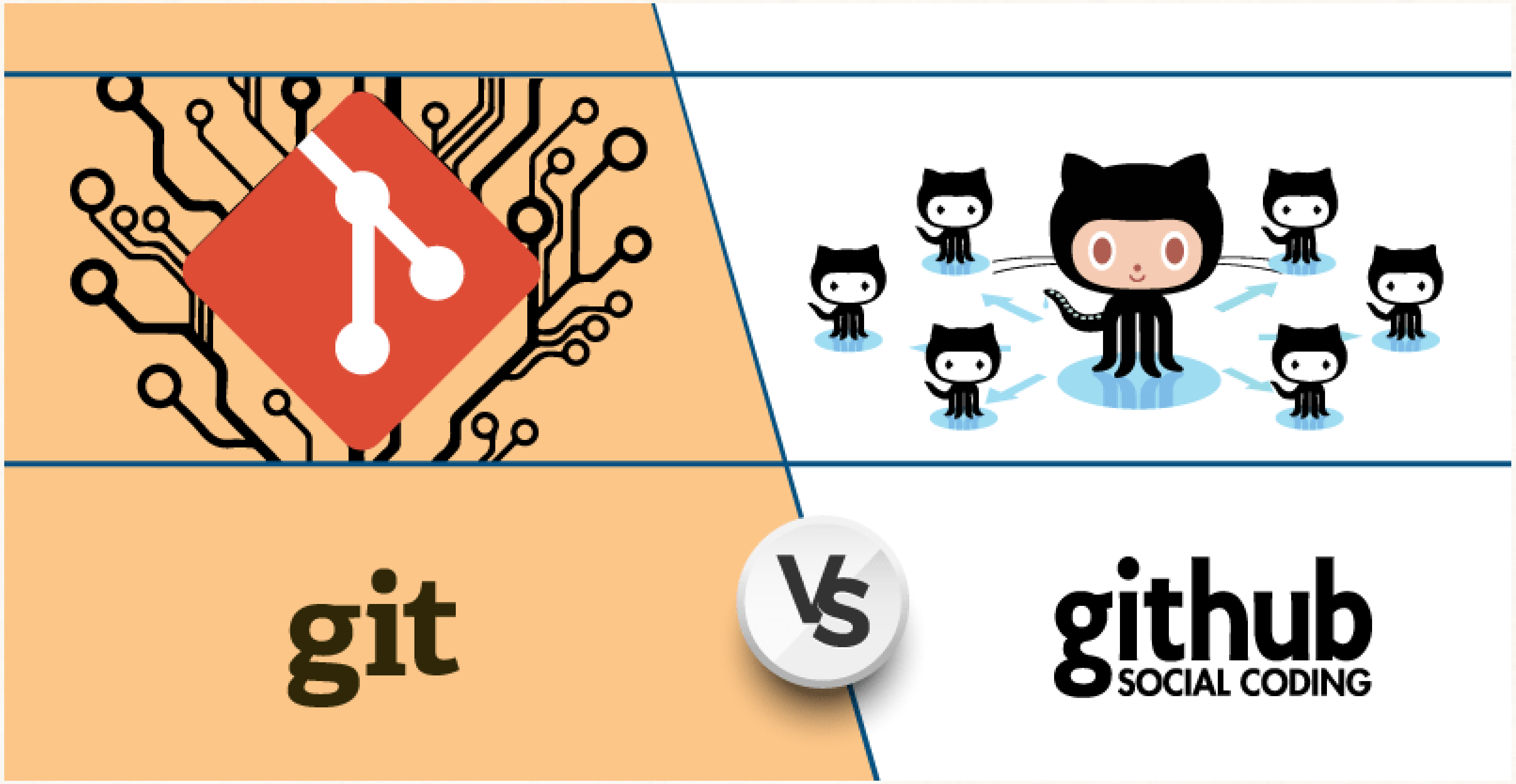
- Each user has their own repository and a working copy. This is very useful because even if the server crashes we would not lose everything as several copies are residing in several other computers.



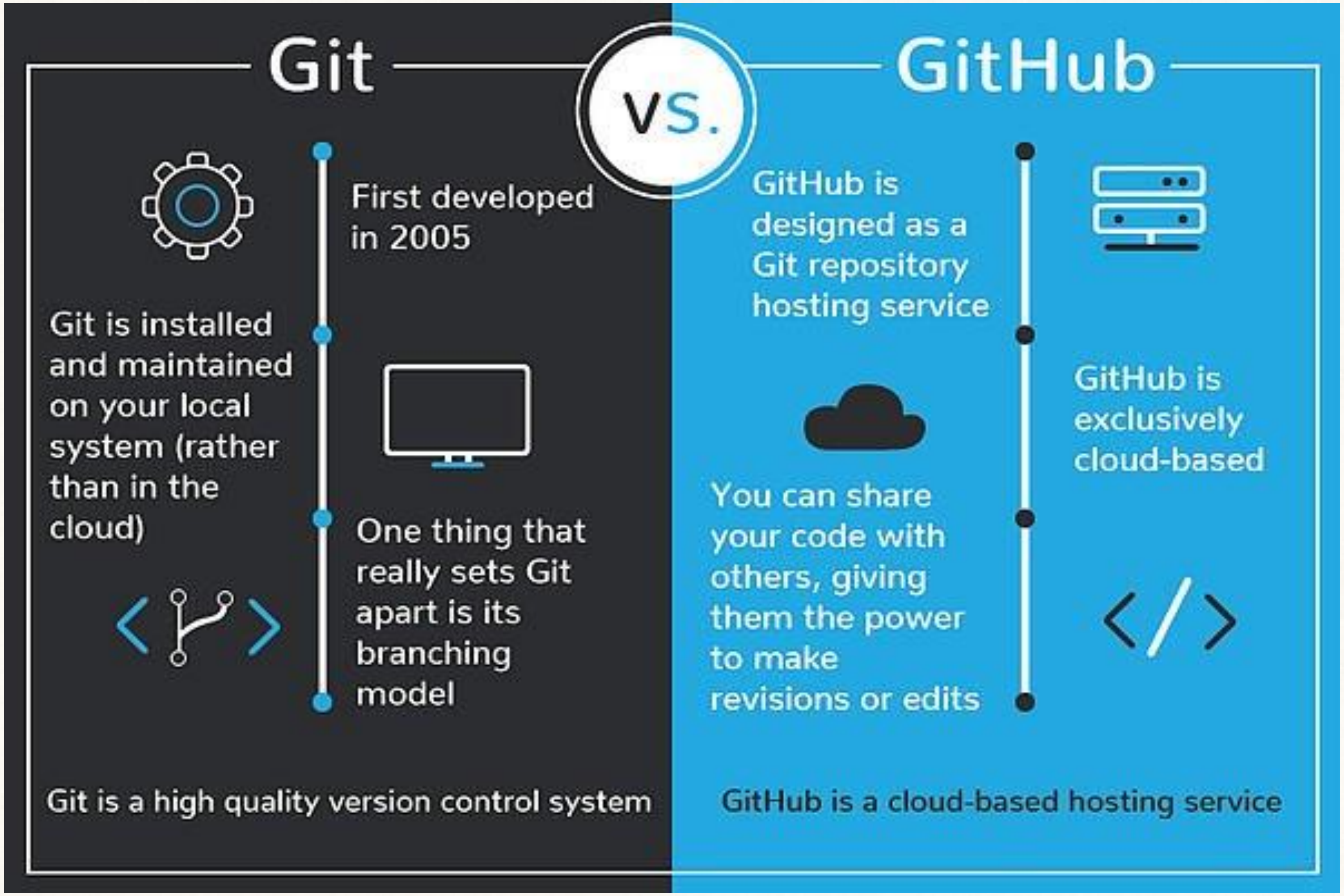
Difference between Git and GitHub

- Git is a version control tool (software) to track the changes in the source code.
- GitHub is a web-based cloud service to host your source code(Git repositories). It is a centralized system.
- Git doesn’t require GitHub but GitHub requires Git.

Difference between Git and GitHub



Difference between Git and GitHub



Git

vs.

GitHub

First developed in 2005

Git is installed and maintained on your local system (rather than in the cloud)

One thing that really sets Git apart is its branching model

Git is a high quality version control system

GitHub is designed as a Git repository hosting service

GitHub is exclusively cloud-based

You can share your code with others, giving them the power to make revisions or edits

GitHub is a cloud-based hosting service

What exactly is GitHub?

- To be very crisp about what exactly is GitHub, it is a file or code-sharing service to collaborate with different people.
- GitHub is 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 drobox or any cloud based system?

Same example

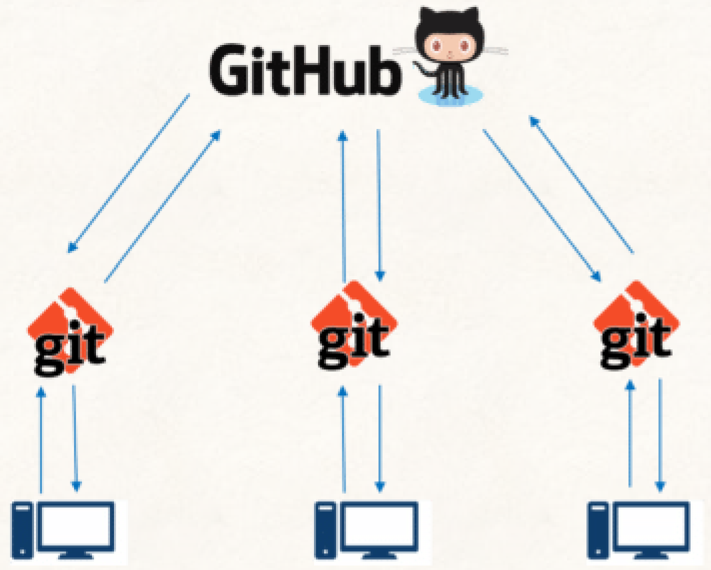
- Let me take the same example forward to answer this question
- Say more than two software developers are working on the same file and they want to update it simultaneously. Unfortunately, the person who save the file first will get precedence over the others.

GitHub

- 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.

GitHub



If you look at the image on the left, 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 its 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.

GitHub makes git simple

- GitHub provides you a beautiful visual interface which helps you to track or manage your version controlled projects locally.
- Once you register on GitHub, you can connect with social network and build a strong profile.

How to create a GitHub Repository?

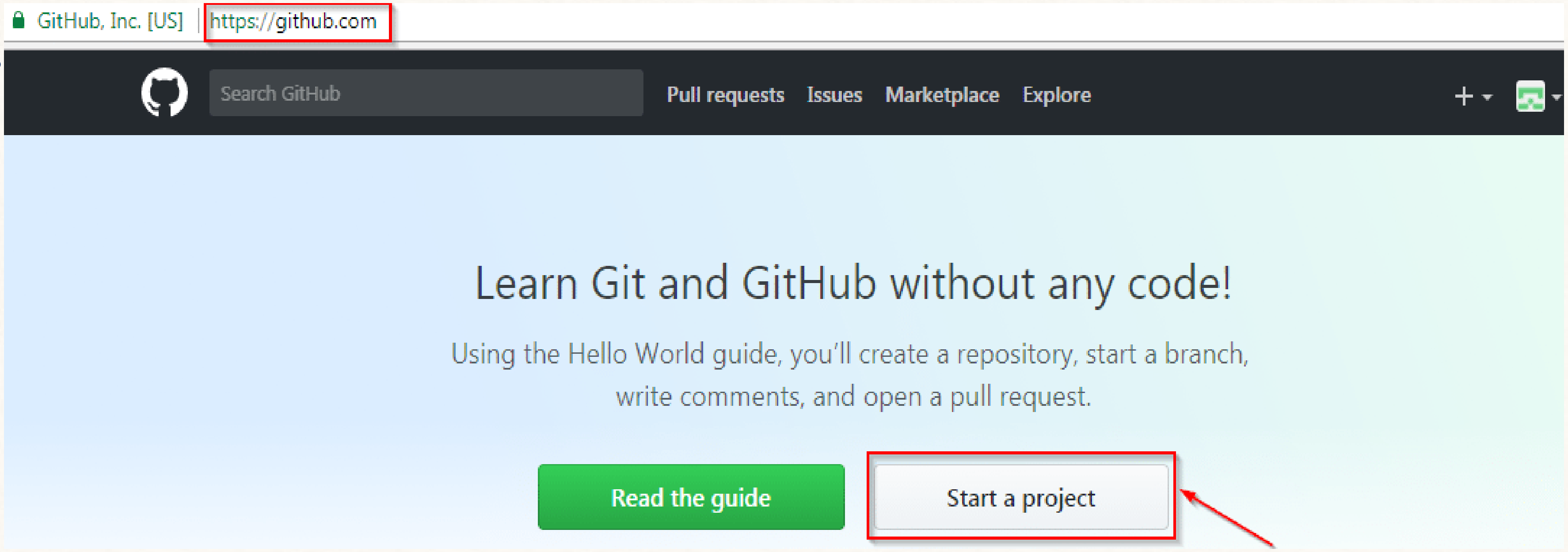
- A repository is a storage space where your project lives. It can be local to a folder on your computer, or it can be a storage space on GitHub or another online host. You can keep code files, text files, images or any kind of a file in a repository.
- You need a GitHub repository when you have done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

How to create a GitHub Repository?

Simple steps to create a GitHub repository:

Start a new project

- Go to the link: https://github.com/ . Fill the sign up form and click on "Sign up for Github".
- Click on "Start a new project".
- Refer to the below screenshot to get a better understanding.
-

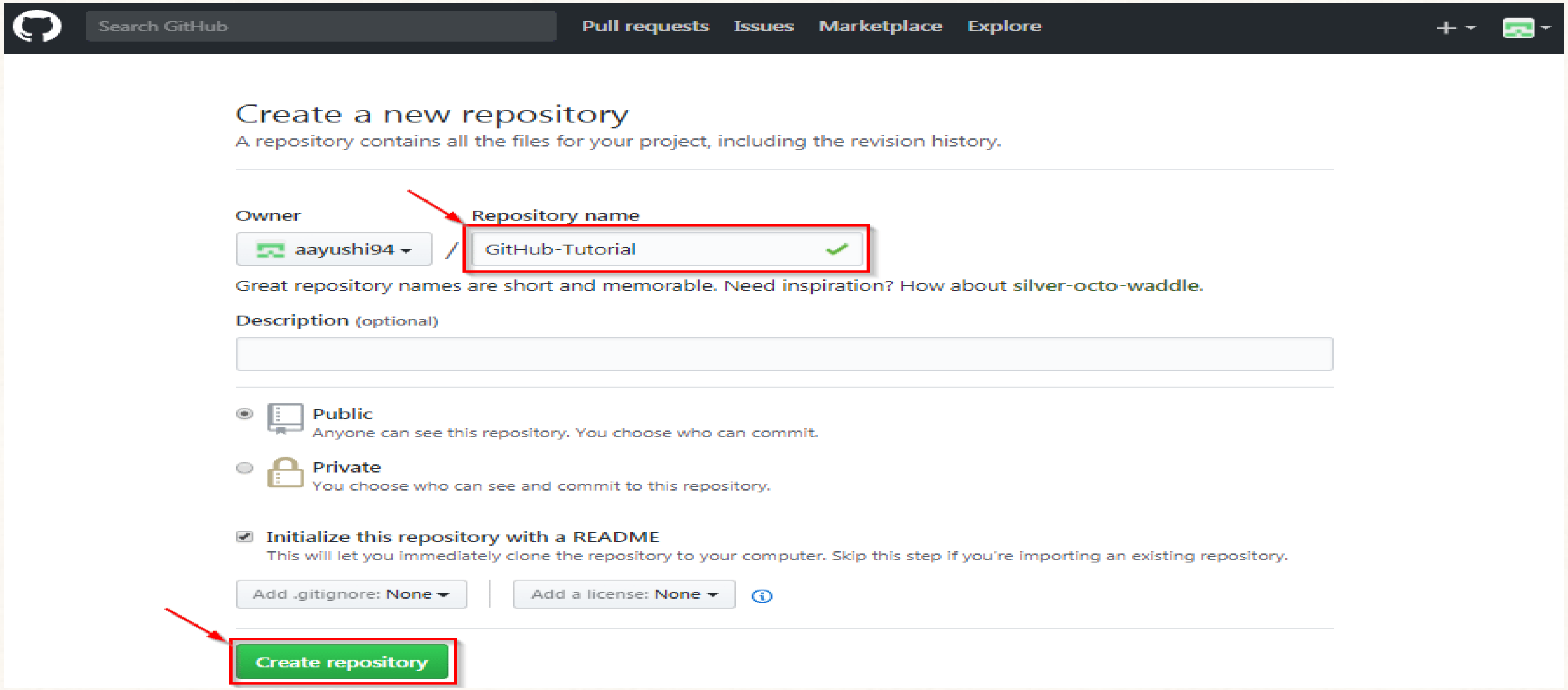


Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Repository name

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



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: aayushi94 / Repository name: GitHub-Tutorial

Great repository names are short and memorable. Need inspiration? How about silver-octo-waddle.

Description (optional)

● Public
Anyone can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

[✓] Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

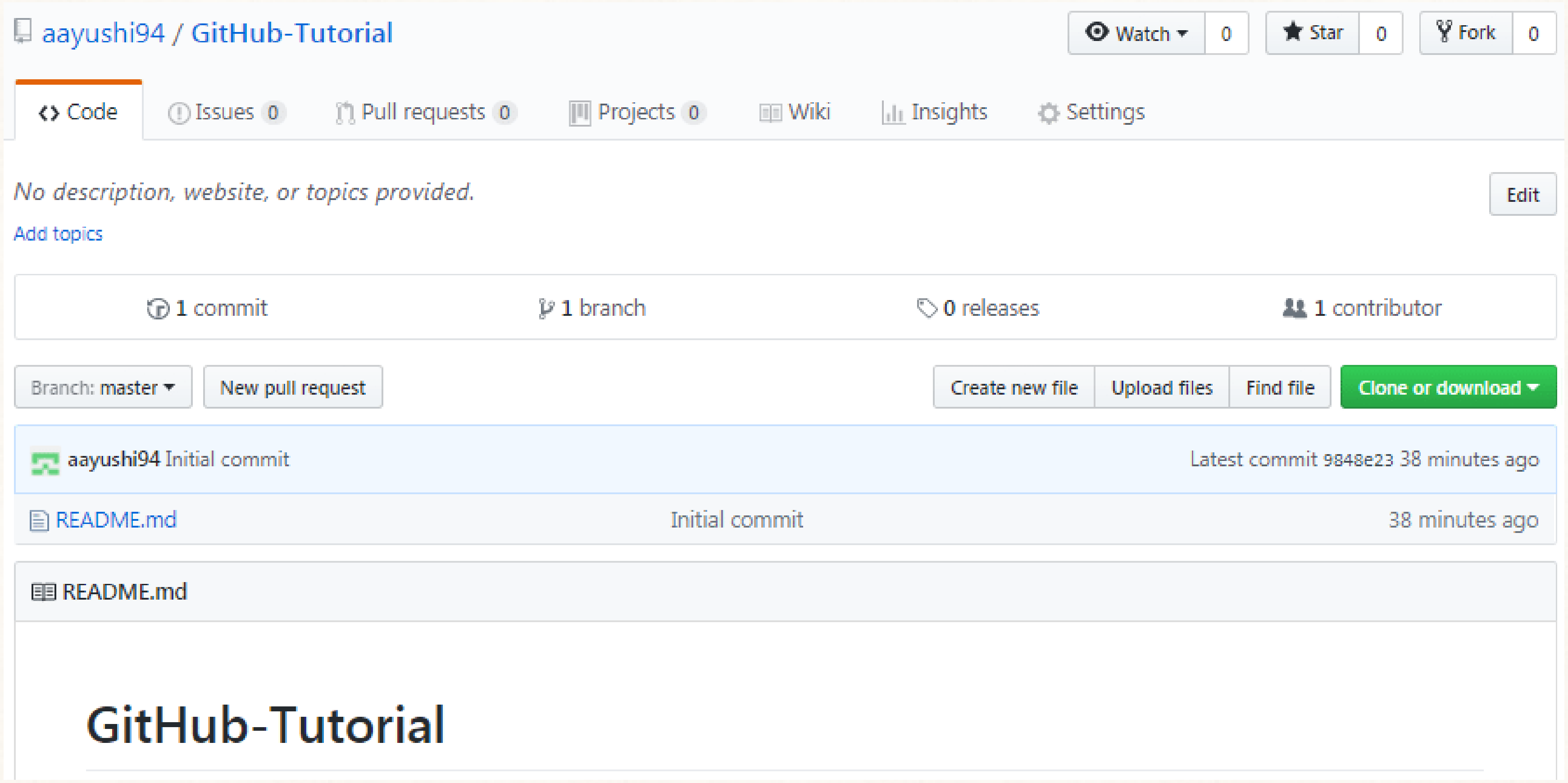
Add .gitignore: None | Add a license: None

GitHub repository

- 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

Repository is successfully created

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



aayushi94 / GitHub-Tutorial

No description, website, or topics provided.

1 commit | 1 branch | 0 releases | 1 contributor

aayushi94 Initial commit — Latest commit 9848e23 38 minutes ago

README.md — Initial commit — 38 minutes ago

README.md

GitHub-Tutorial

Repository is successfully created

- So now central repository has been sucessfully 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

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.

Pull command

- Pull command is the most important command in GitHub. It tell 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.

Merge Command

- Here comes the last command which merge the changes into the main master branch.

Installation of Git

- 1. **Install Git for using WSL** (Windows Subsystem for Linux).
- 2. **Install Git software for windows.**

Git operations and commands

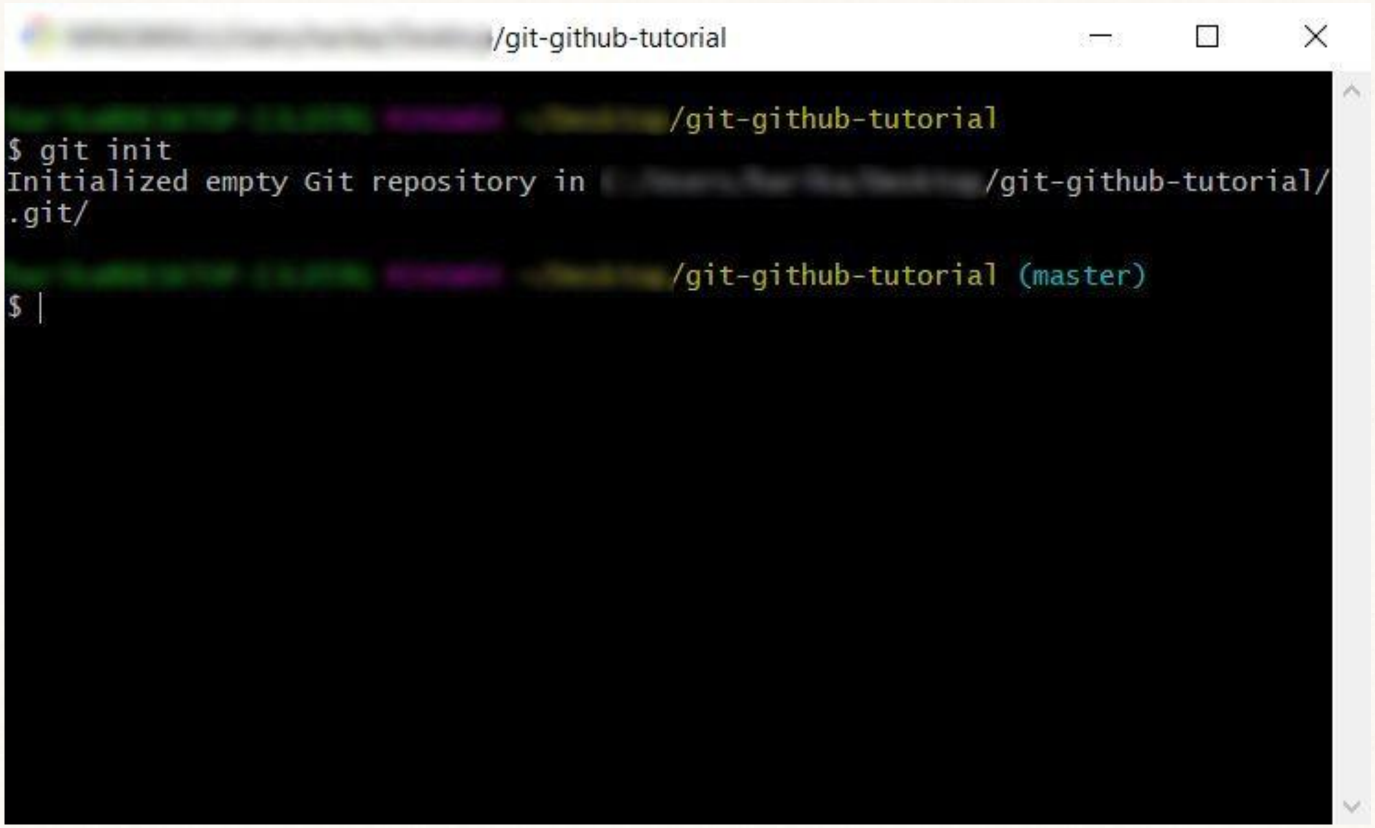
Before deep-diving into Git operations and commands, create an account for yourself on GitHub if you don’t have it already.

5 Git operations and commands

Create repositories

- Create a remote central repository on GitHub.
- Open your file explorer, navigate to the working directory, right-click and select “Git Bash Here”. This opens the Git terminal. To create a new local repository use the command git init and it creates a folder .git.
- git init to create a new Git repository

git init



[redacted]/git-github-tutorial

```
[redacted]/git-github-tutorial
$ git init
Initialized empty Git repository in [redacted]/git-github-tutorial/
.git/

[redacted]/git-github-tutorial (master)
$ |
```

(master) is the default branch of the local repository.
Next, we need to sync the local and the central repositories.

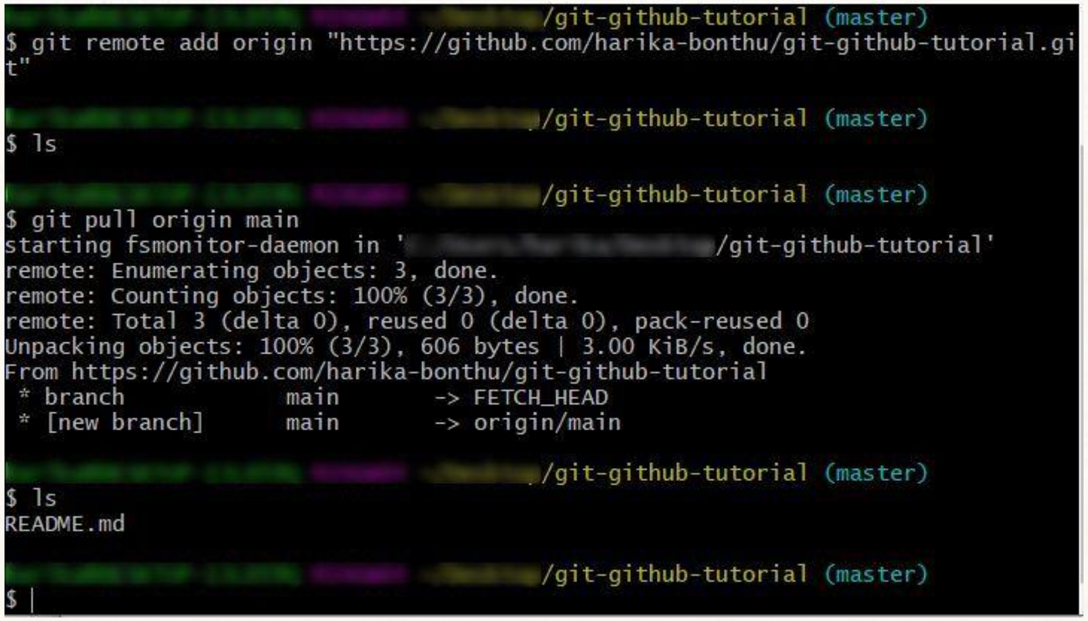
Git remote add to add a new remote repository.

- To get the URL of the central repo, open your repository in GitHub and copy the link.
- Run the below command,
- $ git remote add origin "https://github.com/harika-bonthu/git-github-tutorial.git"
- Generally, Origin is the shorthand name of the remote repository that we are cloning.

Git pull to download all the content from the remote repo

- **$ git pull origin main**
- (main is the branch in our central/remote repository.

  Kindly check the branch name before pull request)



```
[redacted]/git-github-tutorial (master)
$ git remote add origin "https://github.com/harika-bonthu/git-github-tutorial.gi
t"

[redacted]/git-github-tutorial (master)
$ ls

[redacted]/git-github-tutorial (master)
$ git pull origin main
starting fsmonitor-daemon in '[redacted]/git-github-tutorial'
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 606 bytes | 3.00 KiB/s, done.
From https://github.com/harika-bonthu/git-github-tutorial
 * branch            main       -> FETCH_HEAD
 * [new branch]      main       -> origin/main

[redacted]/git-github-tutorial (master)
$ ls
README.md

[redacted]/git-github-tutorial (master)
$ |
```

Git status to check the status of the working directory and the staging area.

- Working directory – It is the place where we make changes to the existing files or create new files.
- Staging area – It is the place where the files are ready to be committed.

Git add to add files to the index or the staging area.

- After making changes in the working directory, once again check the status using the command **git status.**