**GITHUB**

GitHub is a platform for hosting and collaborating on software projects using the Git version control system. Git stores the complete history of your files for a project in a special directory (a.k.a. a folder) called arepository*,* or repo. This repo is usually in a hidden folder called “.git” sitting next to your files.

**Repositories**

The “.git” directory that we have mentioned is known as the *repository*, or repo for short. It contains all our history, including the actual data, the hashes of the data, dates and authors of changes, and tags and branches.

Once upon a time, before git was invented, repositories were always in a central location such as your department's main computing server. (The most popular version control system before git, Subversion, followed this model.) This made sharing of access unnecessarily cumbersome: you had to be very careful whom you allowed to commit changes to the repository, and you couldn't access the repository without an internet connection and sometimes a VPN.

In git, repositories can be replicated, either in full or in part, as many times as necessary. Git hashes enable this: since every version has a commit hash that is practically guaranteed to be unique, as well as one or more *parent* commits that are also hashed, it is easy to reason about duplicated versions of the history.

In addition to enabling backups, this also helps with collaboration, but adds an extra layer that you need to consider when using git: in addition to your repository in .git and your working copy, there are also "clone" repositories, also called "forks". These are useful for backups and to collaborate with others.

While we commit to a repo from a working copy, and we check out to a working copy from a repo, we push to a fork from our repo, or fetch from a fork to our repo.

**How GIT works?**

Git is a distributed version control system designed to track changes in source code during software development. It's decentralized, meaning every Git directory on every computer is a full-fledged repository with complete history and version-tracking abilities. Here's a simplified overview of how Git works:

1. **Initialization**
2. **Create Repository**: To start using Git, you initialize a new repository in your project directory.

**git init**

1. **Clone Repository**: Alternatively, you can clone an existing repository from a remote server to your local machine.

**git clone <repository\_url>**

1. **Staging and Committing changes:**
2. **Add Changes**: You modify files in your project directory. To include these changes in the next commit, you stage them using **git add**.

**git add <file\_name>**

1. **Commit Changes**: Once changes are staged, you commit them to the repository with a descriptive message using **git commit**.

**git commit -m "Commit message"**

1. **Branching and Merging:**
2. **Create Branches**: Git allows you to create branches to work on features or fixes independently. This keeps your changes isolated from the main codebase.

**git branch <branch\_name>**

1. **Switch Branches**: You can switch between branches using **git checkout**.

**git checkout <branch\_name>**

1. **Merge Changes**: After completing work on a branch, you merge it back into the main branch (e.g., **master**) using **git merge**.

**git merge <branch\_name>**

1. **Collaboration**:
2. **Push Changes**: If you're working in a collaborative environment, you can share your changes with others by pushing them to a remote repository.

**git push origin <branch\_name>**

1. **Pull Changes**: Conversely, you can pull changes made by others from the remote repository to your local repository using **git pull**.

**git pull origin <branch\_name>**

1. **History and Inspection:**
2. **View Log**: You can view the commit history of a repository using **git log**.

**git log**

1. **Inspect Changes**: You can view the changes made to a file using **git diff**.

**git diff <file\_name>**

1. **Remote Repositories:**
2. **Add Remote**: You can add a remote repository, typically hosted on services like GitHub or GitLab, using **git remote add**.

**git remote add origin <repository\_url>**

1. **Fetch and Pull**: Fetch retrieves changes from the remote repository without merging them into your local repository. Pull combines fetch and merge.

**git fetch origin**

**git pull origin <branch\_name>**

This is a simplified overview of how Git works. Git offers a wide range of commands and features to manage and track changes in software projects efficiently. It's highly flexible and can accommodate various workflows and collaboration scenarios.