# History of Git and Github

## What is Version Control System:

Version Control Systems (VCS) are essential tools for software developers, allowing them to track changes made to files over time. They provide a way to collaborate effectively, revert to previous versions, and manage different branches of development.

### Types of VCS:

Centralized Version Control Systems:

CVS maintained a central repository where all changes were stored. It was simple to use but had limitations in terms of performance and scalability. Developers work with a local copy of the repository, but all changes must be pushed to the central server. They were simple and easy to manage and used for smaller teams and projects. But they had drawbacks like single point of failure, slow working and performance for large repositories, and limited offline capabilities. Examples of CVCS are: CVS (Concurrent Versions System), and SVN (Subversion).

Distributed Version Control Systems:

In DVCS every developer has a complete copy of the repository, including its history. This makes it more resilient to failures, provide better performance for larger projects, support distributed workflow, great for geographically distributed teams and enables offline work. Examples of DVCS are: BitKeeper, Git, and Mercurial.

Local Version Control Systems:

An LVCS is a type of version control system that is designed to track changes to files on a single computer. It doesn't involve any centralized repository or network connectivity. This means that all changes are stored and managed locally on the developer's machine. Examples of LVCS are: RCS (Revision Control Systems), and SCCS (Source Code Control System).

### What VCS Were Used Before Git and What Let To The Creation of Git:

Before Git, the most widely used version control system was Subversion (SVN) which was a Centralized Version Control System. While SVN was a significant improvement over its predecessors like CVS, it still had limitations such as its centralized architecture and slower performance for large repositories.

Another VCS used before Git was BitKeeper, commercial DVCS that was briefly used by the Linux kernel development team before Git. The experience of using BitKeeper and its licensing restrictions influenced Linus Torvalds, the creator of Git, to develop a new VCS that addressed those shortcomings.

Git was inspired by BitKeeper but incorporated significant improvements. It adopted a distributed architecture, efficient branching and merging, performance optimizations, robust conflict resolution, and an open-source model. These advancements addressed the limitations of BitKeeper and established Git as a leading version control system.

## A Brief History of Git:

Git was created in 2005 by Linus Torvalds, the creator of the Linux kernel. The impelling force for Git's development was the revocation of the free license for BitKeeper, a proprietary source control system used by the Linux kernel development team. Torvalds, seeking a free and distributed alternative, crafted Git in just over a week.

Key characteristics of Git:

- **Distributed Version Control System:** Every developer has a complete copy of the repository, making it resilient to failures.
- **Fast Performance:** Git is optimized for speed, especially when dealing with large projects.
- **Branching and Merging:** Git's branching model is lightweight, making it easy to create and experiment with different versions of code.

# A Brief History of Github:

GitHub was launched in 2008 as a web-based platform for hosting Git repositories. It quickly became a popular tool for developers to collaborate on open-source projects and share code. GitHub's features include:

- **Repository Hosting:** Developers can store and manage their Git repositories on GitHub.
- **Issue Tracking:** Projects can use GitHub to track and manage bugs, feature requests, and other tasks.
- **Pull Requests:** A mechanism for proposing changes to a repository, facilitating collaboration and code review.
- **Social Features:** GitHub fosters a community of developers through features like profiles, following, and commenting.

# Git Commands:

1. Set Up:

   These commands are used to configure user information across all repositories. The Commands are as follows:

   - git config - -global user.name “username” -> used to change or update the username.
   - git config - -global user.name “username” -> used to change or update the email of git profile.
   - git config list -> used to see the configuration details.
2. Clone & Status:
   - Cloning the repo on your local machine. -> git clone <some link>
   - Display the status of the files either changed or not staged. -> git status

3. Add, Commit & Push:
    - Adds the untracked (new) or changed (modified) files in your working directory to the git staging area. -> git add <file name> or git add . (for adding changes to all the files in cd).
    - It is the record of changes made often coupled with a message.
      -> git commit -m “some message”
    - It is used to upload local repo to the remote repo.
      -> git push origin <branch name>
    - To get the commit history. -> git log
4. Git Branches:
    - To check the branches -> git branch
    - To rename a branch -> git branch -M <new name>
    - To navigate to another branch -> git checkout <branch name>
    - To create new branch -> git checkout -b <branch name>
    - To delete a branch -> git checkout -d <branch name>
5. INIT:
    - Initialize an existing directory as a Git Repo -> git init
6. Merge:
    - Merging two branches -> git merge <branch name>