

Git

Version control systems, such as Git, play a crucial role in tracking file changes over time and facilitating collaboration among developers. These systems manage code modifications through specialized databases. Platforms like GitHub, Bitbucket, and GitLab provide remote repositories for storing and sharing Git projects, with options ranging from free to commercial.

Git, an open-source distributed version control system, is well-suited for projects of any size. It prioritizes speed and efficiency, fostering seamless collaboration among developers within the same workspace. While Git is the foundation for services like GitHub and GitLab, it can also be utilized independently, both privately and publicly. Originating in 2005 for the Linux Kernel by Linus Torvalds, Git boasts an easy learning curve and rapid performance, surpassing other SCM tools such as Subversion, CVS, Perforce, and ClearCase.

Advantages of Git include swift operation, offline capability, robust error reversal options, and effective change tracking through features like Diff, Log, and Status.

Git provides various tools to enhance functionality and user interface:

- **Git Bash:** Offers a Git command-line experience on Windows, providing a robust shell interface with essential commands.
- **Git GUI:** A graphical version of Git's command line, facilitating visual diff tools for easier navigation and interaction with Git functionalities.
- **Gitk:** A graphical history viewer acting as a shell over git log and git grep, aiding in visualizing project history and navigating past changes.

Git terminology encompasses terms like branch, checkout, cherry-picking, clone, fetch, HEAD, index, master, merge, origin, pull/pull request, push, rebase, remote, repository, stashing, tag, upstream/downstream, revert, reset, ignore, diff, cheat sheet, Git Flow, squash, rm, fork, and more.

Day 3 focuses on basic Git commands:

- **Git Config:** Configures user name and email globally.
- **Git Init:** Initializes a new Git repository.
- **Git Clone:** Creates a local copy of a remote repository.
- **Git Add:** Adds files to the staging area.
- **Git Commit:** Records changes in the repository.
- **Git Status:** Displays the status of changes.
- **Git Push:** Uploads local changes to a remote repository.
- **Git Pull:** Fetches and merges changes from a remote repository.
- **Git Branch:** Lists, creates, or deletes branches.
- **Git Merge:** Integrates changes from one branch into another.
- **Git Log:** Displays the commit history.

Staging and committing operations involve Git Add and Git Commit, each serving specific purposes.

Day 4 covers inspecting and undoing changes:

- **Git Log:** Displays commit history.
- **Git Log Stat:** Shows modified files, line additions/deletions, and a summary of changes.
- **Git Log Patch:** Reveals modified files and specific line changes.
- **Git Checkout:** Switches between different versions of a repository.

Day 4 also introduces Git Fetch, Git Pull/Pull Request, Git Push, Git Force Push, and deleting a remote branch, explaining their purposes and usage.