What’s a version control system?

A version control system, or VCS, tracks the history of changes as people and teams collaborate on projects together. As the project evolves, teams can run tests, fix bugs, and contribute new code with the confidence that any version can be recovered at any time. Developers can review project history to find out:

- Which changes were made?
- Who made the changes?
- When were the changes made?
- Why were changes needed?

What's a distributed version control system?

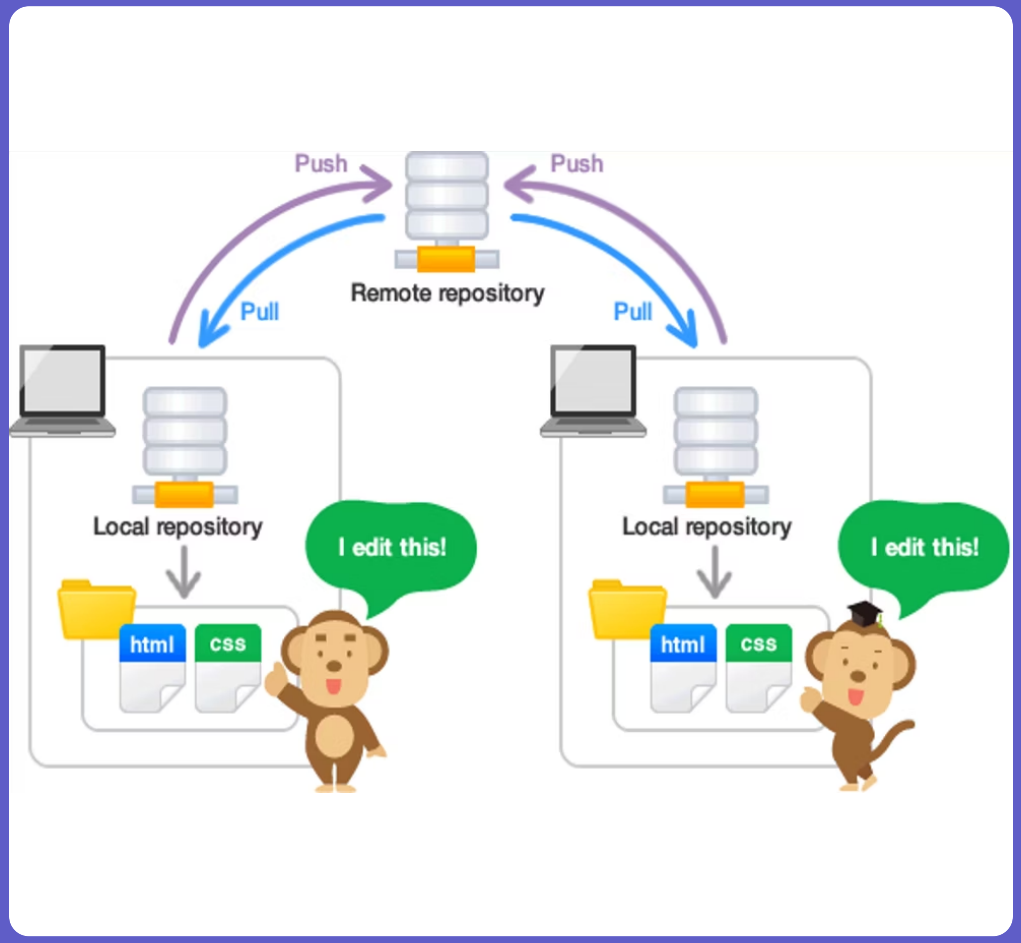
A distributed version control system (DVCS) brings a local copy of the complete repository to every team member's computer, so they can commit, branch, and merge locally. The server doesn't have to store a physical file for each branch — it just needs the differences between each commit.

Distributed source code management systems, such as Git, Mercurial, and Bazaar, mirror the repository and its entire history as a local copy on individual hard drives.

Distributed version control systems help software development teams create strong workflows and hierarchies, with each developer pushing code changes to their own repository and maintainers setting a code review process to ensure only quality code merges into the main repository.

What’s a repository?

A repository, or Git project, encompasses the entire collection of files and folders associated with a project, along with each file’s revision history.



Basic Git commands

To use Git, developers use specific commands to copy, create, change, and combine code. These commands can be executed directly from the command line or by using an application like GitHub Desktop or Git Kraken.

`git init` initializes a brand new Git repository and begins tracking an existing directory. It adds a hidden subfolder within the existing directory that houses the internal data structure required for version control.

`git clone` creates a local copy of a project that already exists remotely. The clone includes all the project's files, history, and branches.

`git add` stages a change. Git tracks changes to a developer's codebase, but it's necessary to stage and take a snapshot of the changes to include them in the project's history.

`git checkout` Switch branches or restore working tree files

`git reset` Reset current HEAD to the specified state

`git commit` saves the snapshot to the project history and completes the change-tracking process. In short, a commit functions like taking a photo. Anything that's been staged with git add will become a part of the snapshot with git commit.

`git-log` Show commit logs

`git status` shows the status of changes as untracked, modified, or staged.

`git branch` shows the branches being worked on locally.

`git rm` Remove files from the working tree and from the index

`git-mv` - Move or rename a file, a directory, or a symlink

`git pull` updates the local line of development with updates from its remote counterpart. Developers use this command if a teammate has made commits to a branch on a remote, and they would like to reflect those changes in their local environment.

`git push` updates the remote repository with any commits made locally to a branch.