Introduction to Git: Understanding “origin” and Basic Commands

Ever wondered what is the meaning of “origin” in the command `git pull origin master`?

If so, you’re about to embark on an insightful journey into the world of Git, a powerful and widely-used version control system that has become an integral part of modern software development.

Git helps teams collaborate on projects by tracking changes in source code during software development. Understanding its terminology and commands is crucial for effective team collaboration and maintaining a smooth workflow.

What is “origin” in Git?

In Git, “origin” is not a command or a branch, but a reference to the default remote repository from which you have cloned your project. When you clone a repository using `git clone`, Git automatically names this remote repository as ”origin”. It’s a shorthand alias for the repository’s URL, making it easier to perform remote operations like fetching, pulling, and pushing changes without typing the full URL every time.

Basic Git Commands

As you start with Git, here are some essential commands you should know:

- **git init**: Initializes a new Git repository.
  - Usage: `git init`
- **git clone [url]**: Clones a repository into a new directory.
  - Usage: `git clone https://github.com/user/repo.git`
- **git status**: Displays the state of the working directory and staging area.
  - Usage: `git status`
- **git add [file]**: Adds a file to the staging area.
  - Usage: `git add example.txt`
- **git commit -m ”[commit message]”**: Records file snapshots in the version history.
  - Usage: `git commit -m "Initial commit"`
- **git push [remote] [branch]**: Pushes the branch to the remote repository.
  - Usage: `git push origin master`
- **git pull [remote] [branch]**: Fetches and merges changes from the remote.
  - Usage: `git pull origin master`
- **git branch**: Lists, creates, or deletes branches.
  - Usage: `git branch` to list, `git branch [branch-name]` to create

- **git checkout** [**branch**]: Switches to a specified branch.
  - Usage: `git checkout feature-branch`
- **git merge** [**branch**]: Merges a branch into the current branch.
  - Usage: `git merge feature-branch`

Bonus Point: Advantage of `git fetch` and `merge` Over `git pull`

A significant advantage of using `git fetch` followed by `git merge`, instead of `git pull`, is the control it offers over integrating changes from a remote repository. In essence, the command *git pull* in Git is a combination of two other commands: *git fetch* followed by *git merge*. When you execute *git pull*, it first performs git fetch, which downloads content from the specified remote repository. Then, immediately after, *git pull* executes *git merge* to merge the fetched changes into your current branch. This process automates the sequence of fetching and merging, making it a convenient one-step action for updating your local repository with changes from a remote source.