**Complete Git Guide (Zero to Pro)**

**Introduction to Git**

Git is a distributed version control system used to track changes in source code during software development. It allows multiple developers to work on the same project efficiently.

**Why Use Git?**

* Tracks changes and maintains history.
* Enables collaboration in teams.
* Supports branching and merging.
* Works with remote repositories like GitHub, GitLab, and Bitbucket.

**Git Installation**

To install Git, visit: <https://git-scm.com/downloads>

Verify installation:

git --version

**Basic Git Commands**

**1. Configuration Commands**

git config --global user.name "Your Name"

git config --global user.email "your.email@example.com"

git config --global core.editor "code --wait" # Set default editor

Check current configuration:

git config --list

**2. Initialize a Git Repository**

git init

This initializes a new Git repository in the current directory.

**3. Clone a Repository**

git clone <repository-url>

Example:

git clone https://github.com/user/repo.git

**4. Check Repository Status**

git status

Shows the status of the working directory and staged changes.

**5. Staging and Committing Changes**

git add <file>

Stage a specific file.

git add .

Stage all changes.

git commit -m "Commit message"

Commit staged changes with a message.

**6. Viewing Commit History**

git log

Shows commit history.

git log --oneline --graph

Displays a compact commit history with a graphical representation.

**7. Branching in Git**

git branch

List all local branches.

git branch <new-branch>

Create a new branch.

git checkout <branch-name>

Switch to another branch.

git checkout -b <new-branch>

Create and switch to a new branch.

git branch -r

List all remote branches.

git branch -a

List all local and remote branches.

git branch --show-current

Show the current branch name.

**8. Merging Branches**

git merge <branch-name>

Merges the specified branch into the current branch.

**9. Rebasing a Branch**

git rebase <branch-name>

Reapply commits on top of another base branch.

git rebase -i HEAD~3

Squash the last three commits into one.

**10. Deleting a Branch**

git branch -d <branch-name>

Delete a local branch.

git push origin --delete <branch-name>

Delete a remote branch.

**11. Fetching and Pulling Changes**

git fetch

Fetches changes from the remote repository but does not merge them.

git pull origin <branch-name>

Fetches and merges changes from the remote branch.

**12. Pushing Changes to Remote Repository**

git push origin <branch-name>

Push local changes to the remote repository.

git push -u origin <branch-name>

Push a new branch to the remote repository and set it to track.

**13. Resetting and Reverting Changes**

git reset --soft HEAD~1

Undo the last commit but keep changes staged.

git reset --hard HEAD~1

Undo the last commit and discard all changes.

git revert <commit-hash>

Revert a specific commit without deleting history.

**14. Stashing Changes**

git stash

Saves current changes temporarily without committing.

git stash pop

Applies the latest stashed changes and removes them from the stash list.

git stash list

Displays all stashed changes.

**15. Finding a Commit that Introduced a Bug**

git bisect start

git bisect bad

git bisect good <commit-hash>

Helps identify the commit that introduced an issue.

**16. Renaming a Branch**

git branch -m old-branch-name new-branch-name

Renames a local branch.

**17. Checking Differences Between Commits**

git diff

Shows differences between working directory and staged files.

git diff <commit-hash> <commit-hash>

Shows differences between two commits.

**18. Restoring a Deleted File**

git checkout -- <file>

Restores a deleted file if it was previously committed.

**19. Git Hooks**

cd .git/hooks

Example: Creating a pre-commit hook to check for syntax errors.

echo "#!/bin/sh

eslint ." > .git/hooks/pre-commit

chmod +x .git/hooks/pre-commit

**Conclusion**

Git is an essential tool for version control, enabling efficient collaboration and project management. Mastering these commands will help streamline your development workflow.

This guide provides all the necessary Git commands and explanations to help you become proficient in Git. 🚀