Click Here

Given text to paraphrase here Here are fifty essential Git commands that developers should know, categorized for easy reference. **Getting Started** Initializing a new repository: `git init` Cloning an existing repository: `git clone [url]` Adding files to the staging area: `git add [file]` Committing changes with a message: `git commit -m "[message]"` Viewing the working directory status: `git status` Showing file differences: `git diff` **Branching and Merging** Listing local branches: `git branch` Creating a new branch: `git branch [branch-name]` Switching to a branch: `git checkout [branch-name]` Merging a branch into the current one: `git merge [branch-name]` Deleting a branch: `git branch -d [branch-name]` **Remote Repositories** Adding a remote repository: `git remote add origin [url]` Pushing changes to the remote: `git push origin [branch-name]` Pulling changes from the remote: `git pull origin [branch-name]` Fetching changes without merging: `git fetch` **Undoing Changes** Unstaging a file: `git reset [file]` Discarding file changes: `git checkout -- [file]` Creating a new commit to undo changes: `git revert [commit]` Optional: Average flags by command (2) Learning all available Git commands can be overwhelming, but using cheat sheets and resources like the "Using Git" cheat sheet and the Git and GitHub learning resources page can help. A handy guide to essential Git commands is also available for developers. starts from here - Initializing a Local Repo, Adding Files & Commits git init - starts a new local Git repo in your project root; replace filename_here with the name of the file for git add. You can add all files using . and every file will be added. For a specific pattern use fil*. git status shows repository status including staged, unstaged, and untracked files. git commit opens a text editor for a full commit message or you can specify a short summary with -m. Use -a and -m to add and commit tracked files at once. git log displays the commit history; git log -p adds change details. You can view a specific commit by replacing commit-id. git show shows statistics about changes in each commit. You can use -staged to see staged changes or -p for partial staging. Use git add -p for staging changes and git rm filename for deleting files. Create a .gitignore file, stage the changes, then commit it. Use checkout to unstage changes. You can reset changes with -p or amend the most recent commit. git commit --amend is used for fixing local commits but be careful when pushing them to shared repositories. Use git revert HEAD to revert the latest commit or use a specific commit id and open an editor for a message. The default branch is main, create a new one using git branch and switch manually with next command. git checkout branch_name lets you view all created branches with git branch command. it shows a list of all branches and marks current branch with an asterisk in green. git branch in one command creates and switches to new branch, use git checkout -b branch_name. after merging, delete the branch with git branch -d branch_name. merge current branch history with branch_name using git merge branch_name. for commit log as graph, use --graph, and limit to single line with --oneline, git log --graph --oneline. same but for all branches, git log --graph --oneline --all. if you want to throw away a merge and start over, run git merge --abort. add remote repository with git add remote . see all remote repositories with git remote -v. replace origin with name obtained by running git remote -v command, then use git remote show origin. push changes with git push when ready. retrieve latest changes from remote with git pull. show remote branches being tracked for current repository with git branch -r. download changes without merging with git fetch. commit log of remote repository is found using git log origin/main. merge remote changes with local with git merge origin/main. update remote without merging content into local branches with git remote update. push a branch to remote with git push -u origin branch_name. remove remote branch if no longer needed, use git push --delete origin branch_name_here. transfer completed work from one branch to another using git rebase. but use with caution and refer to official documentation before running git rebase. run git rebase interactively with -i flag, it opens editor for commands like p, pick; r, reword; e, edit; s, squash; f, fixup; x, exec; d, drop. this command forces a push request, usually used Using these Git commands for pull request branches can help maintain security by keeping the code private. Unlike public repositories, it's not recommended to use this method. To overwrite previous versions in a public repository, you would use git push -f. Having a cheat sheet like this one can save time and effort when working with Git. It's worth saving or bookmarking for future reference.

What are the git commands.