

ChatGPT - Git Command Overview

all git command

Here's a comprehensive list of common **Git commands** grouped by their purpose:

Basic Commands

- `git init` : Initialize a new Git repository.
 - `git clone <url>` : Clone a repository from a remote location.
 - `git status` : Check the status of the working directory and staging area.
 - `git add <file>` : Add a file to the staging area.
 - `git add .` : Add all changes in the current directory to the staging area.
 - `git commit -m "<message>"` : Commit changes with a message.
 - `git log` : Show the commit history.
 - `git show <commit>` : Show changes in a specific commit.
-

Branching

- `git branch` : List all branches.
 - `git branch <branch-name>` : Create a new branch.
 - `git checkout <branch-name>` : Switch to a branch.
 - `git switch <branch-name>` : Switch to a branch (modern alternative to `checkout`).
 - `git switch -c <branch-name>` : Create and switch to a new branch.
 - `git branch -d <branch-name>` : Delete a branch.
 - `git branch -D <branch-name>` : Force delete a branch.
-

Merging & Rebasing

- `git merge <branch-name>` : Merge a branch into the current branch.
 - `git rebase <branch-name>` : Reapply commits on top of another branch.
 - `git merge --abort` : Abort a merge in progress.
 - `git rebase --abort` : Abort a rebase in progress.
-

Remote Repositories

- `git remote add <name> <url>` : Add a new remote repository.
 - `git remote -v` : List remote repositories.
 - `git fetch <remote>` : Fetch changes from the remote repository.
 - `git pull` : Fetch and merge changes from the remote repository.
 - `git push <remote> <branch>` : Push changes to a remote branch.
 - `git push -u <remote> <branch>` : Set an upstream branch and push.
-

Stashing

- `git stash` : Save uncommitted changes for later.
 - `git stash list` : List stashed changes.
 - `git stash apply <stash>` : Reapply a stashed change.
 - `git stash drop <stash>` : Delete a stash.
-

Undoing Changes

- `git checkout -- <file>` : Discard changes in the working directory.
- `git reset <file>` : Unstage a file.

- `git reset --soft <commit>` : Undo commits but keep changes staged.
 - `git reset --mixed <commit>` : Undo commits and unstage changes.
 - `git reset --hard <commit>` : Undo commits and discard changes.
 - `git revert <commit>` : Create a new commit that undoes a specific commit.
-

Viewing History

- `git log --oneline` : View a compact commit history.
 - `git diff` : Show changes not yet staged.
 - `git diff --staged` : Show staged changes.
 - `git blame <file>` : Show who modified each line of a file.
-

Tagging

- `git tag <tag-name>` : Create a tag.
 - `git tag -a <tag-name> -m "<message>"` : Create an annotated tag.
 - `git tag` : List tags.
 - `git push origin <tag-name>` : Push a tag to the remote repository.
-

Collaboration

- `git pull --rebase` : Rebase instead of merging when pulling.
 - `git cherry-pick <commit>` : Apply changes from a specific commit.
 - `git fetch --prune` : Remove stale branches from the local repository.
 - `git push origin --delete <branch>` : Delete a remote branch.
-

Miscellaneous

- `git config --global user.name "<name>"` : Set global username.
- `git config --global user.email "<email>"` : Set global email.
- `git config --list` : List configuration settings.
- `git archive` : Create a tarball of the repository.
- `git gc` : Cleanup unnecessary files and optimize the repository.

Let me know if you want a specific example or workflow!

and all shortcut git command

Here's a list of commonly used **Git shortcuts** or single-line commands that combine multiple operations to save time:

Shortcuts for Everyday Tasks

1. Stage all changes and commit:

```
bash

git commit -am "<message>"
```

- Combines `git add .` and `git commit -m "<message>"` (only works for changes to tracked files).

2. Create and switch to a new branch:

```
bash

git checkout -b <branch-name>
```

- Combines `git branch <branch-name>` and `git checkout <branch-name>`.

3. Delete a branch locally:

```
bash
```

```
git branch -d <branch-name>
```

- Shortcut to delete a branch.

4. Force delete a branch locally:

```
bash
```

```
git branch -D <branch-name>
```

5. Pull and rebase in one step:

```
bash
```

```
git pull --rebase
```

- Combines fetching and rebasing.

6. Push and set upstream:

```
bash
```

```
git push -u origin <branch-name>
```

- Pushes a branch and sets the upstream branch for future pulls.

Remote Shortcuts

7. Add a remote and fetch:

```
bash
```

```
git remote add origin <url> && git fetch origin
```

8. Fetch and prune stale branches:

```
bash
```

```
git fetch --prune
```

9. Delete a remote branch:

```
bash

git push origin --delete <branch-name>
```

Log and History Shortcuts

10. One-line commit log:

```
bash

git log --oneline
```

11. Graph view of commits:

```
bash

git log --oneline --graph --all
```

12. Show only recent commits:

```
bash

git log -n <number>
```

- Example: `git log -n 5` (shows the last 5 commits).

13. Compact blame view:

```
bash

git blame -c <file>
```

Diff and Compare

14. Show staged changes:

```
bash
```

```
git diff --staged
```

15. Compare branches:

```
bash
```

```
git diff <branch1>..<branch2>
```

16. Compare a file with a previous commit:

```
bash
```

```
git diff <commit> <file>
```

Undo Shortcuts

17. Unstage a file:

```
bash
```

```
git reset <file>
```

18. Undo the last commit but keep changes:

```
bash
```

```
git reset --soft HEAD~1
```

19. Undo the last commit and discard changes:

```
bash
```

```
git reset --hard HEAD~1
```

Tagging Shortcuts

20. Create a lightweight tag:

```
bash

git tag <tag-name>
```

21. Push all tags:

```
bash

git push --tags
```

22. Delete a tag locally:

```
bash

git tag -d <tag-name>
```

23. Delete a tag remotely:

```
bash

git push origin --delete <tag-name>
```

Configuration Shortcuts

24. Set username and email globally:

```
bash

git config --global user.name "<name>"
git config --global user.email "<email>"
```

25. Set a global default editor:

```
bash

git config --global core.editor <editor>
```


- Example: `git config --global core.editor "code --wait"`
-

Custom Aliases

You can create your own shortcuts using **Git aliases**:

```
bash

git config --global alias.co checkout
git config --global alias.br branch
git config --global alias.cm commit
git config --global alias.st status
```

Now you can use:

- `git co <branch>` instead of `git checkout <branch>`
 - `git br` instead of `git branch`
 - `git cm -m "message"` instead of `git commit -m "message"`
-

Would you like help setting up shortcuts or aliases?