git config --global -e

.gitconfig global configuration

# This is Git's per-user configuration file.

[user]

name = Full Name

email = email.address

[core]

editor = code --wait

autocrlf = input

[alias]

st = status

ci = commit -m

br = branch

co = checkout

lg = log --oneline --graph

ph = push origin

[diff]

tools = vscode

[difftool "vscode"]

cmd = "code --wait --diff $LOCAL $REMOTE"

Git is a powerful version control system with a wide variety of commands for managing repositories, branches, commits, and remote repositories. Below is an overview of some of the most commonly used Git commands, categorized by their functionality.

### Basic Git Commands

1. git init
   * Initialize a new Git repository.

git init

1. git clone [url]
   * Clone a remote repository to your local machine.

git clone https://github.com/username/repository.git

1. git status
   * Show the status of the working directory (which files are staged, modified, untracked, etc.).

git status

1. git add [file]
   * Stage a specific file to be committed.

git add filename

1. git add .
   * Stage all changes (new, modified, and deleted files) in the working directory.

git add .

1. git commit -m "message"
   * Commit staged changes with a message.

git commit -m "Your commit message"

1. git log
   * View the commit history.

git log

1. git diff
   * Show the differences between working directory and the index (what’s staged).

git diff

1. git diff --cached
   * Show the differences between the staged changes and the last commit.

git diff --cached

1. git push [remote] [branch]
   * Push the committed changes to the remote repository.

git push origin main

1. git pull [remote] [branch]
   * Fetch the latest changes from the remote repository and merge them into your local branch.

git pull origin main

1. git fetch
   * Fetch changes from the remote repository without merging them into your local branch.

git fetch origin

1. **git restore**
   * Restore a specific file to its state in a previous commit, you can also use the --source option.

git restore <file-path>

git restore --source=<commit-hash> <file-path>

### Branching and Merging

1. git branch
   * List all branches or create a new branch.

git branch # List branches

git branch new-branch # Create a new branch  
git branch -M main # Rename the current branch to main or any

1. git branch -d [branch-name]
   * Delete a local branch.

git branch -d feature-branch

1. git checkout [branch]
   * Switch to another branch.

git checkout main

1. git checkout -b [branch-name]
   * Create and switch to a new branch in one step.

git checkout -b new-branch

1. git merge [branch]
   * Merge changes from one branch into the current branch.

git merge feature-branch

1. git rebase [branch]
   * Rebase the current branch onto another branch (reapply commits on top of another base branch).

git rebase main

1. git reset [file]
   * Unstage a file that has been added to the staging area.

git reset filename

1. git reset --hard
   * Reset the working directory and staging area to the last commit (warning: this will discard changes).

git reset --hard

1. git reset --soft [commit]
   * Reset to a specific commit but keep changes in the staging area.

git reset --soft HEAD~1 # Goes back 1 commit but leaves staged changes

1. git cherry-pick [commit]
   * Apply the changes from a specific commit onto your current branch.

git cherry-pick <commit-hash>

### Remote Repositories

1. git remote add [name] [url]
   * Add a remote repository.

git remote add origin https://github.com/username/repository.git

1. git remote -v
   * View the remotes associated with your local repository.

git remote -v

1. git push --set-upstream [remote] [branch]
   * Push a branch and set the upstream (tracking) branch.

git push --set-upstream origin main

1. git push --force
   * Force push a branch, overwriting changes on the remote (use with caution).

git push --force origin main

1. git pull --rebase
   * Pull changes from the remote and rebase the current branch on top of them.

git pull --rebase origin main

### Tagging

1. git tag [tag-name]
   * Create a new tag at the current commit.

git tag v1.0

1. git tag -a [tag-name] -m "message"
   * Create an annotated tag with a message.

git tag -a v1.0 -m "Release version 1.0"

1. git push origin [tag-name]
   * Push a tag to the remote repository.

git push origin v1.0

1. git push --tags
   * Push all tags to the remote repository.

git push --tags

### Git Configuration

1. git config --global user.name "[name]"
   * Set the global username for Git commits.

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

1. git config --global user.email "[email]"
   * Set the global email for Git commits.

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

1. git config --list
   * List all Git configuration settings.

git config --list

### Git Stash

1. git stash
   * Save changes that are not yet committed so you can work on something else. This stores your working directory and staging area.

git stash

1. git stash pop
   * Apply the most recent stashed changes and remove it from the stash list.

git stash pop

1. git stash list
   * List all stashes.

git stash list

1. git stash apply
   * Apply the most recent stash without removing it from the stash list.

git stash apply

1. git stash drop
   * Remove a specific stash from the stash list.

git stash drop stash@{0}

### Git Help

1. git help
   * Display a help message with information about Git commands.

git help

1. git [command] --help
   * Get detailed help for a specific Git command.

git commit --help

### Other Useful Git Commands

1. git show [commit]
   * Show details about a specific commit.

git show <commit-hash>

1. git clean -fd
   * Remove untracked files and directories.

git clean -fd

1. git reflog
   * View the history of your HEAD, which allows you to recover lost commits.

git reflog

1. git bisect
   * Use binary search to find the commit that introduced a bug.

git bisect start