1.Dir : to see all the directory

2.Code .: to start VS code

3.rmdir /s <dir-name> : to dlt non-empty file

4.rmdir <dir-name> : to dlt empty file

5. rm <file-name>

Initializing git in a local repo

1. Initialize git in a folder to push file from : git init

To create PR

1.Clone Repo

2.Navigate to Dir

3. **create new Branch for your changes :**git checkout -b <new-branch-name>

4.Make the necessary changes to your files

5. commit your changes git add . and git commit -m "Description of your changes"

6. Push the new branch: git push origin <new-branch-name>

7.you can create a pull request using GitHub's web interface.

- Go to your repository on GitHub.
- You should see a prompt to create a pull request for the newly pushed branch. If not, navigate to the "Pull requests" tab and click on "New pull request".
- Select the base branch you want to merge into (usually `main` or `master`) and the branch you just pushed.
- Add a title and description for your pull request, then click "Create pull request".

create a PR from the terminal:

gh pr create --base <base-branch> --head <new-branch-name> --title "Title of your PR" --body "Description of your PR"

Replace `<base-branch>` with the branch you want to merge into (e.g., `main`), and `<new-branchname>` with the name of your new branch.

How To push

1.list branches : git branch

2.switch to the desired branch : git checkout <branch-name>

3.Add the changes you want to push to the staging area: git add .

Note: add specific files by replacing (`.->current file)` with the file names

4.Commit Changes: git commit -m "Your commit message"

5.Push the changes : git push origin <branch-name>

How to Delete a file

1. Navigate to Your Repository
2. Switch to your branch from which you want to delete a file : git checkout <branch-name>
3. Delete the File : rm <file-name>
4. Add the deletion to the staging area: git add <file-name>
5. Commit the Changes : git commit -m "Delete <file-path>"
6. Push the Changes : git push origin <branch-name>

#Additional

1.Merger branch: git merge <branch-name>

2. delete a branch: git branch -d <branch-name>

3. Force delete a branch: git branch -D <branch-name>

4. View Commits History : git log

5. View a Specific Commit: git show <commit-hash>

6. View Differences between commits : git diff and git diff <branch-name>

7. Revert a Commit: git revert <commit-hash>

8. Reset to a Previous Commit: git reset --hard <commit-hash> and git reset --soft <commit-hash>

Stash

9. when you need to change your branch to work on something you can stash all the untracked file of current branch

9.1 Stash your changes: git stash

9.2 Switch Branch : git checkout <your-branch-name>

9.3 switch back to your prev branch: git checkout <branch-name>

9.4 Reapply your stashed changes and continue working: git stash apply

10. To include untracked files in the stash : git stash -u

11 List all the stashes you have saved: git stash list

12. Apply the most recent stash to your working directory : git stash apply

13. To apply a specific stash: git stash apply stash@{n} where n is stash number from the list.

14. Apply the most recent stash and remove it from the stash list: git stash pop

15. Remove a specific stash from the list without applying it: git stash drop stash@{n}

16. To drop the most recent stash : git stash drop

17. Remove all stashes : git stash clear

Add tag to commits

1.Lightweight Tags(They do not have any extra information like who created the tag or when it was created) : git tag <tag-name>

2. Annotated Tags (tags contain metadata, including the tagger’s name, email, date, and a tagging message) : git tag -a <tag-name> -m "Tag message"

3. Creating a Tag: git tag <tag-name> for Lightweight tag
git tag -a <tag-name> -m "Tag message" for Annotated tag
4. Listing Tags: git tag
5. Showing Tag Details:git show <tag-name>
6. Pushing Tags: git push origin <tag-name>
7. To Push all tags : git push --tags
7.Deleting a Tag: git tag -d <tag-name>
git push origin :refs/tags/<tag-name> for Delete tag from remote
8.Checkout a tag: git checkout <tag-name>

Check Remote Configuration

1. check if you have a remote repository named 'origin' configured: git remote -v
2. If 'origin' is not listed when you run `git remote -v`, you can add it : git remote add origin <repository-url>
3. If you already have 'origin' configured but the URL is incorrect, you can update it : git remote set-url origin <new-repository-url>
4. Once 'origin' is correctly configured, you can push your changes : git push -u origin main

PULL

1. Navigate to Your Repository
2. **Fetch**: Downloads the latest changes (commits, files) from the remote repository to your local repository.
3. **Merge**: If there are no conflicts between your local changes and the changes fetched from the remote, Git will automatically merge the changes into your current branch. If there are conflicts, Git will prompt you to resolve them manually.
4. **Rebase Instead of Merge**: You can use `git pull --rebase` to rebase your local changes on top of the changes fetched from the remote repository instead of merging them.
5. To fetch and merge changes from the remote repository into your current branch (typically `main` or `master`), use: git pull origin <branch-name>

To transfer all commits/changes from one branch to another

1. switch to destination branch : git checkout <destinataion branch>

2.merge source branch to destination : Merge `<source branch>` into `<destination branch>`

3. add the changes: git add

4. commit the merge : `git commit` -m “message”

5. Once the merge is complete and conflicts are resolved, push the changes: git push origin <destination branch>

Cherry Picking

allows you to choose specific commits from one branch and apply them onto another branch. This can be useful when you want to transfer specific changes without merging entire branches

1. Identify Commits in your source branch to Cherry-Pick : git log in your source branch

2. Switch to the Target Branch: git checkout target-branch
3. Cherry-pick each commit from the `source-branch` : git cherry-pick <commit-id>
4. Open the conflicted files, resolve the conflicts,
5. Then stage the resolved files: git add <resolved-file(s)>
6. continue the cherry-pick process :git cherry-pick –continue
7. cherry-picked all desired commits, commit the changes : git commit
8. Push Changes in target branch : git push origin target-branch