GIT AND GITHUB

GitHub is a web-based platform for version control and collaboration that allows developers to manage and share their code repositories. It is built on top of **Git**, which is a distributed version control system that helps track changes in source code during software development. GitHub extends Git by providing a range of additional features, making it easier for teams to work together, manage projects, and maintain code integrity.

GitHub (Git) commands and their functions:

1. `git init`

- **Function**: Initializes a new Git repository in the current directory. This creates a `.git` folder, which tracks all changes in the directory.

Usage:
```
git init
```

2. `git clone`

- **Function**: Clones an existing repository (from GitHub or other sources) to your local machine.

Usage:
```
git clone <repository_url>
```

3. `git status`

- **Function**: Shows the current status of the repository, including staged changes, untracked files, and changes yet to be staged.

Usage:
```
git status
```

4. `git add`

- **Function**: Adds changes (from working directory) to the staging area, preparing them for a commit.

Usage:
```
git add <file_name>     # Add specific file
git add .               # Add all changes
```

5. `git commit`

- **Function**: Records or snapshots changes to the repository. A message is typically required to describe the changes made.

Usage:
```
git commit -m "Your commit message"
```

6. `git push`

- **Function**: Pushes the committed changes from the local repository to the remote repository (e.g., GitHub).

Usage:
```
git push origin <branch_name>
```

7. `git pull`

- **Function**: Fetches changes from a remote repository and merges them into your current branch.

Usage:
```
git pull origin <branch_name>
```

8. `git branch`

- **Function**: Lists all the branches in your repository or creates a new branch.

Usage:
```
git branch            # List all branches
git branch <branch_name>  # Create a new branch
```

9. `git checkout`

- **Function**: Switches to a different branch or commit.

Usage:
```
git checkout <branch_name>   # Switch to a branch
git checkout <commit_hash>   # Switch to a specific commit
```

10. `git merge`

- **Function**: Merges another branch into your current branch.

Usage:
```
git merge <branch_name>
```

11. `git log`

- **Function**: Shows the commit history for the repository.

Usage:
```
git log
```

12. `git remote`

- **Function**: Manages connections to remote repositories (like GitHub).

Usage:
```
git remote add origin <remote_repo_url>   # Add a remote repository
git remote -v   # List the remote connections
```

13. `git diff`

- **Function**: Shows the differences between files in the working directory and the last commit.

Usage:
```
git diff
```

14. `git reset`

- **Function**: Unstages files that have been added to the staging area or resets the commit history.

Usage:
```
git reset <file_name>   # Unstage a file
git reset --hard        # Reset to the last commit, discarding changes
```

15. `git fetch`

- **Function**: Downloads changes from the remote repository without merging them.

Usage:
```
git fetch origin
```

16. `git rebase`

- **Function**: Reapplies commits on top of another base commit, used to keep a clean commit history.

Usage:
```
git rebase <branch_name>
```

17. `git stash`

- **Function**: Temporarily saves changes that are not ready to be committed, allowing you to work on something else.

Usage:
```
git stash      # Stashes changes
git stash pop  # Reapplies stashed changes
```

18. `git tag`

- **Function**: Adds tags to specific commits (often used for versioning).

Usage:
```
git tag <tag_name>
```

19. Delete Files

To delete files from your repository, use the following command:

- **`git rm`**: Removes files from both the working directory and the staging area.

Usage:

```
git rm <file_name>       # Delete a specific file
git rm -r <directory_name>  # Delete a directory recursively
```

If you want to delete the file from the repository but keep it locally on your machine, use:

```
git rm --cached <file_name>  # Unstage and remove file from Git, but
keep it locally
```

20. Commit Changes (Including File Deletion)

After deleting the file, you need to commit the change to the repository:

Usage:

```
git commit -m "Deleted <file_name> from the repository"
```

21. Push Changes to Remote

Once you've deleted the file and committed the changes, push the changes to the remote repository:

Usage:

```
git push origin <branch_name>
```

Example Workflow for Deleting Files:

```
git rm unwanted_file.txt          # Stage file for deletion
git commit -m "Removed unwanted_file.txt"  # Commit file deletion
git push origin main              # Push changes to remote repository
```