Git & GitHub Notes

These notes are structured for **GitHub README / personal notes**, with **definition → syntax → example** format for each topic.

---

1. Difference Between Git and GitHub

Git

Definition:
Git is a **distributed version control system** used to track changes in source code locally.

Key Points: - Works offline - Installed on local system - Tracks file changes

GitHub

Definition:
GitHub is a **cloud-based hosting platform** for Git repositories that enables collaboration.

Key Points: - Works online - Stores Git repositories - Supports collaboration, issues, pull requests

Git	GitHub
Tool	Platform
Local	Cloud-based
Tracks versions	Hosts repositories

---

2. GitHub Repository Workflow

Steps: 1. Create repository on GitHub 2. Clone it to local system 3. Make changes 4. Add & commit changes 5. Push changes to GitHub

---

3. Clone Repository

Definition:
Creates a local copy of a remote GitHub repository.

Syntax:

```
git clone <repository-link>
```

Example:

```
git clone https://github.com/username/repo-name.git
```

---

4. Basic Terminal Commands

cd (Change Directory)

Definition: Navigate between folders.

Syntax:

```
cd <folder-name>
```

Example:

```
cd projects
```

---

pwd (Present Working Directory)

Definition: Shows current directory path.

Syntax:

```
pwd
```

---

ls

Definition: Lists files and folders.

Syntax:

```
ls
```

---

ls -a

Definition: Lists all files including hidden files.

Syntax:

```
ls -a
```

---

clr / clear

Definition: Clears terminal screen.

Syntax:

```
clear
```

---

5. git init

Definition: Initializes a new Git repository locally.

Syntax:

```
git init
```

---

6. git status

Definition: Shows the current state of the working directory.

Syntax:

```
git status
```

---

7. GitHub File Status

- **Untracked:** New file, not tracked by Git
- **Modified:** File changed but not staged
- **Staged:** File added to staging area
- **Unstaged:** Modified but not staged
- **Unmodified:** No changes

---

8. git add

Add Single File

Syntax:

```
git add <filename>
```

Add All Files

Syntax:

```
git add .
```

---

9. git commit

Definition: Saves staged changes to local repository.

Syntax:

```
git commit -m "message"
```

Example:

```
git commit -m "Initial commit"
```

---

10. git remote add origin

Definition: Connects local repo to remote GitHub repo.

Syntax:

```
git remote add origin <repo-link>
```

---

11. git remote -v

Definition: Shows connected remote repositories.

Syntax:

```
git remote -v
```

---

12. git push

Push to Main Branch

Syntax:

```
git push origin main
```

First Time Push

Syntax:

```
git push -u origin main
```

---

13. git branch

Definition: Lists or manages branches.

Syntax:

```
git branch
```

---

14. Rename Branch to Main

Syntax:

```
git branch -M main
```

---

15. git checkout

Switch Branch

Syntax:

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

Create & Switch Branch

Syntax:

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

---

16. Delete Branch

Syntax:

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

---

17. Merging Code

Method 1: Using Git (Command Line)

a. git diff

Definition: Shows differences between branches.

Syntax:

```
git diff <branch-name>
```

b. git merge

Definition: Merges another branch into current branch.

Syntax:

```
git merge <branch-name>
```

Method 2: Using GitHub

Steps: 1. Push branch to GitHub 2. Create Pull Request 3. Review code 4. Merge PR

---

18. git pull

Definition: Fetches and merges remote changes.

Syntax:

```
git pull origin main
```

---

19. git diff main

Definition: Shows difference between current branch and main.

Syntax:

```
git diff main
```

---

20. Merge Conflict

Definition: Occurs when Git cannot automatically merge changes.

Steps to Resolve: 1. Open conflicted file 2. Resolve conflict manually 3. Add file 4. Commit changes

---

21. Undoing Changes

Case 1: Staged Changes

```
git restore --staged <file>
```

Case 2: Undo Last Commit

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

Case 3: Undo Multiple Commits

```
git reset --hard <commit-hash>
```

⚠️ Hard reset deletes changes permanently.

---

22. Fork

Definition: Creates a personal copy of another user's repository.

Use Case: - Open source contribution - Experimenting without affecting original repo

---

23. Complete Git Workflow (Summary)

```
GitHub Repo → Clone → Changes → Add → Commit → Push
```
