Git-Github Notes

Git & GitHub – Full Notes (Basics to Advanced)

---

1. Introduction to Git & GitHub

- **Git** → Distributed version control system (DVCS) that tracks changes in code.
- **GitHub** → Cloud-based hosting service for Git repositories + collaboration tools.
- **Version Control** → Keeps history of changes, enables branching, collaboration, rollback.

Why use Git?

- Tracks changes.
- Enables teamwork.
- Maintains code history.
- Experiment without breaking main project.

---

2. Git Installation & Setup

Install

- **Windows** → git-scm.com
- **Linux/Mac** → Use package manager (apt, brew, etc.)

Configure for First Time

```
git config --global user.name "Your Name"
git config --global user.email "youremail@example.com"
git config --global core.editor "code"   # VS Code as default editor
git config --list    # Check configuration
```

---

3. Basic Git Workflow

Lifecycle:

Working Directory → Staging Area → Local Repository → Remote Repository

Common Commands

```
git init                # Initialize Git repo
git clone <url>           # Clone existing repo
git status               # Show changes
git add <file>            # Stage file
git add .                # Stage all files
git commit -m "message"    # Commit staged changes
git log                 # View commit history
git diff                # Show unstaged changes
```

---

4. Working with Remote Repositories (GitHub)

```
git remote add origin <url>    # Link local repo to GitHub
git remote -v               # View remotes
git push origin main          # Push commits
git pull origin main          # Pull changes
git fetch                   # Download latest changes without merging
```

---

5. Branching & Merging

Branch Basics

```
git branch               # List branches
git branch <name>          # Create branch
git checkout <name>        # Switch branch
git checkout -b <name>     # Create & switch
git merge <branch>         # Merge into current branch
```

Merge Conflicts

- Happen when same file lines are edited in different branches.
- Resolve manually → git add → git commit.

---

6. Undoing Changes

```
git restore <file>                # Discard changes in file
git reset HEAD <file>             # Unstage file
git reset --hard <commit-hash>    # Reset to specific commit (dangerous)
git revert <commit-hash>          # Create new commit to undo previous one
```

---

7. Git Log & History

```
git log --oneline                 # Short commit view
git log --graph --oneline --all   # Branch history visual
git show <commit-hash>            # See details of a commit
```

---

8. Stashing

- Temporarily save changes without committing.

```
git stash save "message"
git stash list
git stash apply stash@{0}
git stash drop stash@{0}
```

---

9. Tagging

- Mark specific points (e.g., releases).

```
git tag v1.0.0
git push origin v1.0.0
git tag -a v1.0.0 -m "Version 1.0"
```

---

10. Git Ignore

- .gitignore file → Ignore files/folders.
  Example:

```
node_modules/
.env
*.log
```

---

11. GitHub Features

- **Pull Requests (PR)** → Request to merge code.
- **Issues** → Track bugs/features.
- **Forking** → Copy repo to own account.
- **GitHub Actions** → CI/CD automation.
- **Projects** → Kanban boards.
- **Releases** → Publish versioned builds.

---

12. Collaboration Workflow

Common:

1. Fork repo
2. Clone locally
3. Create branch
4. Make changes → Commit → Push
5. Open Pull Request (PR) on GitHub

---

13. Git Advanced Commands

Rebase

```
git rebase main
```

- Moves commits on top of another branch.
- Used for cleaner history (avoid unnecessary merges).

Cherry Pick

```
git cherry-pick <commit-hash>
```

- Apply specific commit to current branch.

Amend Commit

```
git commit --amend -m "New message"
```

---

14. Working with Multiple Remotes

```
git remote add upstream <url>
git fetch upstream
git merge upstream/main
```

---

15. Git Hooks

- Scripts triggered before/after Git events.
- Example: pre-commit, post-merge.

---

16. GitHub Security

- Enable **2FA**.
- Use **SSH keys** for authentication.

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

---

17. Troubleshooting

- **Detached HEAD** → You're on a commit, not a branch. Fix:

```
git checkout main
```

- **Merge Conflicts** → Edit, stage, commit.
- **Push Rejected** → Pull first:

```
git pull --rebase origin main
```