🔧 Git and GitHub

Course by @Fireship (YouTube)

📘 Chapter 1: What is Git and GitHub?

🔷 Git

- A **version control system** for managing code changes.
- Saves **snapshots** of your code over time.
- Works like a **timeline** with commits as checkpoints.
- Organized like a **tree** — with one main branch and other feature branches.
- Alternate branches let you work independently without disturbing the `main`.

🔷 GitHub

- A **web-based platform** for hosting Git repositories.
- Uses Git to manage code, but everything is stored in the **cloud**.
- Allows collaboration, code review, and project management.

⚙️ Chapter 2: Install Git

🪟 Windows

- Use the **Git Installer** from the official Git website.

🍎 Mac

- Use **Homebrew** (recommended):

```
brew install git
```

🛠️ Configure Git

```
git config --list                    # View current configuration
git config --global user.name "Your Name"
git config --global user.email "your@email.com"
```

> If stuck in terminal: use `:q` or `ctrl + c` to quit.

🗂️ Chapter 3: `git init`

- Initializes a new Git repository in the current folder:

```
git init
```

- Create a README file:

```
echo "README File Content" >> README.md
```

- In VS Code, `.git` folder might be hidden. To show:

  1. Go to **Settings** > search `files.exclude`.
  2. Locate `.git`, click **Edit** and remove it.

- To remove Git from a project, delete the `.git` folder.

➕ Chapter 4: `git add`

```
git add .               # Adds all changes
git add filename        # Adds specific file
git reset .             # Unstages all changes
```

- `A` icon indicates **added file** in VS Code.

📌 Chapter 5: `git status`

- Shows the state of the working directory and staging area.

> Green `U` = **Untracked**
> Yellow `M` = **Modified**

- Ignore files using `.gitignore`:

```
echo "filename" >> .gitignore
```

📝 Chapter 6: `git commit`

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

- Save snapshot with a message.

- To skip `git add`:

```
git commit -am "Message"
```

- Use `git log` to view commit history:
    - Shows commit ID, branch, author, time, and message.

💡 Chapter 7: VS Code Tips

- Use the **Source Control Panel** to:
    - View staged and unstaged changes.
    - Execute Git commands via UI.
    - See diffs between current and previous versions.
- Install **GitLens** for better visualization in collaborative projects.

🌐 Chapter 8: `git remote`

- Link your local repo to GitHub:

```
git remote add origin <GitHub-Repo-Link>
```

- View remotes:

```
git remote
git remote show origin
```

🚀 Chapter 9: `git push`

- Push local commits to GitHub:

```
git push -u origin master
```

> `-u` sets the upstream branch.

🔀 Chapter 10: `git merge`

- Sync local with remote:

```
git fetch
git merge origin/master
```

📥 Chapter 11: `git pull`

- Fetch + Merge in one:

```
git pull
```

> Not always recommended for resolving conflicts.

📦 Chapter 12: `git clone`

- Clone a GitHub repo:

```
git clone <repo-link> [custom-folder-name]
```

- Use `git log` to view commit history.

💻 Chapter 13: GitHub Codespaces

- Press `.` on any GitHub repo page to open in **Codespaces** (VS Code online).

> Note: This is a **paid service**.

🌿 Chapter 14: `git branch`

- View branches:

```
git branch
```

- Create and delete branches:

```
git branch <branch-name>
git branch -d <branch-name>     # Only deletes if fully merged
git branch -D <branch-name>     # Force deletes
```

- Rename current branch to `main`:

```
git branch -M main
```

🔄 Chapter 15: `git checkout`

- Switch branches:

```
git checkout <branch-name>
git checkout -b <branch-name>     # Create and switch
git checkout -                    # Switch to previous branch
```

⚔️ Chapter 16: Merge Conflicts

- If conflicts arise during merging, Git will notify.
- To abort merge:

```
git merge --abort
```

- You can resolve manually by selecting:
  - **Current** (main)
  - **Incoming** (feature)
  - **Both**

🍴 Chapter 17: `git fork`

- Fork = Copy someone's repo to your GitHub account.
- You can make changes and submit a **Pull Request** for review.

📤 Chapter 18: Pull Requests

1. Fork and clone the repo.
2. Create a new branch.
3. Push your changes.
4. Submit a pull request on GitHub.

To sync with original repo:

```
git remote add upstream <original-repo-link>
git fetch upstream
git rebase upstream/master
```

⏪ Chapter 19: `git reset`

- Undo commits:

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

> `--hard` removes all commits **after** the given ID.

↩️ Chapter 20: `git revert`

- Undo a commit (safely):

```
git revert <commit-id>
```

> Creates a new commit that reverses the changes.

🛠️ Chapter 21: `git commit --amend`

- Change last commit message or add files:

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

📥 Chapter 22: `git stash`

- Temporarily save changes:

```
git stash                   # Stash current changes
git stash save "stashname" # Optional naming
git stash list              # View stashes
git stash apply [index]    # Apply specific stash
git stash pop              # Apply and remove
```

🧬 Chapter 23: `git rebase`

- Re-apply commits on top of another branch:

```
git rebase master
```

> Used to maintain a clean history.

🗃️ Chapter 24: Git Squash

- Combine multiple commits into one:

```
git rebase -i master
```

- Change `pick` to `squash` in the interactive prompt.
- Save and confirm commit message.

> ✅ That's a wrap!
> You've now covered all the core commands to work efficiently with Git and GitHub 🚀