Git and GitHub Essentials for Developers:

Git:

- Git is a Version Control System, a tool to track changes in code, and it's one of the most popular, free, and open-source options available. It's known for being fast and scalable. Git operates on the principle of snapshots, with each commit representing a snapshot of your project at a specific point in time.

GitHub:

- GitHub is a website that allows developers to store and manage their code using Git. The folders where code is stored are called repositories. GitHub offers additional features beyond basic version control, such as issue tracking, project boards, and wiki pages, enhancing project management and collaboration within development teams.

Setting up Git:

- Install Git software, **Gitbash.**
- Verify installation: **git --version.**
- Configure Git globally:
- **git config --global user.name "<Your Name>"**
- **git config --global user.email "<Your Email>"**

Basic Operations to know:

- Git operates with two main aspects: **Remote** (GitHub) and **Local** (your laptop or PC).
- Common terminal commands include:

1. to change directories: **cd <directory_name>**
2. to move to previous directory: **cd ..**
3. to list all files: **ls**
4. to list all files including hidden: **ls -a**
5. to clear the terminal: **clear**

Initialization:

- Initialize a Git repository in a directory: **git init**

Undo-initialization:

- Undo git init: **rm -rf .git**

Status:

- Check the status of your working directory and staging area: git status
- FOUR TYPES OF STATUS

1. **untracked:** new file that git doesn't tracked or unknown about
2. **modified:** file is updated and modified yet commitment is left
3. **staged:** file is ready to commit
4. **unmodified:** unchanged

Adding Changes:

- Add changes from your working directory to the staging area: **git add <file_name>** or **git add .** (to add all changes)

Committing Changes:

- Record changes in the repository: **git commit -m "Your commit message"**
- Directly commit without going to staging area: **git commit -a**

Remote Repository Interaction:

- Connect a local repository to a remote repository: **git remote add origin <remote_repository_URL>**
- To verify remote: **git remote -v**
- To remove remote: **git remote remove origin**
- Push changes from your local repository to a remote repository: **git push origin <branch_name>**
- Pull changes from a remote repository to your local repository: **git pull origin <branch_name>**

Branching:

- Create a new branch: **git branch <branch_name>**
- List all branches in the repository: **git branch**
- Switch to a different branch: **git checkout <branch_name>**
- Delete a branch: **git branch -d <branch_name>**
- Rename a branch: **git branch -M <new_name>**
- To compare commits, branch, file and more in branches: **git diff <branch name>**
- Merge changes from one branch into another: **git merge <branch_name>**
- **git fetch:** The git fetch command is used to download commits, files, branches, and tags from a remote repository. It does not affect your local repository's working state, making it a safe way to review remote changes before integrating them with your local repository.

  The general syntax for the command is: **git fetch origin <branch_name>**

Undoing Changes:

- Undo changes in the working directory: **git checkout -- <file>**
- Case1: Undo changes in the staging area: **git reset <file_name>**
- Case2: Undo commits (for one commit): **git reset HEAD~1**
- Case3: Undo commits (for many commits): **git reset --hard <commit_hash>**
- **git revert <commit_hash>** (keeps track of the undoing change in git log)

--soft: uncommit changes, changes are left staged (index).

--mixed (default): uncommit + unstage changes, changes are left in working tree.

--**hard:** uncommit + unstage + delete changes, nothing left.

Deleting directory:

- DELETE FROM BOTH LOCAL AND REMOTE

git rm -r <name_of_directory>

git commit . -m "Your commit message"

git push origin <your-git-branch> (typically main)

- DELETE FROM REMOTE NOT LOCAL

git rm -r --cached <name_of_directory>

Repository Management:

- Clone a repository from GitHub to your local machine: **git clone <repository_URL>**
- Create a new repository on GitHub: (done through the GitHub website)
- To see all commits, hash: **git log** (press q to exit)

Collaboration:

- **Fork** a repository on GitHub: (done through the GitHub website) a fork is a new repository that shares code and visibility settings with the original "upsteam" repository, Fork is a basically a rough copy.
- Select a option Fork on website it will generate a copy for you to use you can add changes and then start a pull request to show of your changes to the main original location
- **Pull request:** It lets you tell others about changes you've pushed to a branch in a repository on github.

---

by chhavirohilla