Git tutorial

-Difference between Git and GitHub:

Feature	Git	GitHub
Definition	A version control system (VCS) that helps track changes in code.	A cloud-based platform for hosting and collaborating on Git repositories.
Purpose	Manages code versions locally on a computer.	Allows teams to store, share, and collaborate on Git repositories online.
Usage	Used via command line (git CLI) or GUI tools.	Accessed via website, CLI, or GitHub Desktop.
Storage	Stores code locally on your machine.	Stores repositories remotely (cloud-based).
Collaboration	Not built for direct collaboration, but supports merging and branching.	Enables teams to work on code together with pull requests, forks, and issues.
Authentication	No login required, runs on a local machine.	Requires an account to use features like repositories, pull requests, and issue tracking.
Example Command	git commit -m "Update" (commits changes locally)	git push origin main (uploads changes to GitHub)

- Git is a tool for tracking code changes.
- GitHub is a platform for hosting and sharing Git repositories.
- Git can exist without GitHub, but GitHub requires Git for version control.

-Git Commands:

- **`git init`** → Sets up a new Git repository in a folder. Think of it as turning on version control for a project.

- **`git clone <repo_url>`** → Copies an existing remote repository to your computer. It's like downloading a project while keeping all its history.

- **`git add <file>`** → Stages changes before saving them. Imagine highlighting parts of your work that you want to include in the next save.
- **`git commit -m "message"`** → Saves a snapshot of staged changes. It's like taking a photo of your project at a certain point.
- **`git status`** → Shows the current state of your files—whether they are untracked, modified, or staged. It's like checking what's ready to be saved.
- **`git branch`** → Lists all branches in your project. Think of branches as different workspaces within the same project.
- **`git merge <branch>`** → Combines changes from one branch into another. Like merging two versions of a document into one.
- **`git pull`** → Fetches and applies the latest changes from the remote repository to your local project. Useful when working with a team.
- **`git push`** → Sends your committed changes from your local repository to the remote one. Like uploading your latest work for others to see.

-Contribute to an Existing Repository

1. Clone the repository:

```
Unset
git clone https://github.com/owner/repo.git

cd repo
```

2. Create and switch to a new branch:

```
Unset
git branch my-branch

git checkout my-branch
```

3. Make changes, stage, and commit:

```
Unset
git add file1.md file2.md

git commit -m "my snapshot"
```

4. Push changes to GitHub:

```
Unset
git push --set-upstream origin my-branch
```

-Start a New Repository and Publish to GitHub

1. Initialize a new Git repository:

```
Unset
git init my-repo

cd my-repo

touch README.md

git add README.md

git commit -m "Initial commit"
```

2. Connect to GitHub and push:

```
Unset
git remote add origin
https://github.com/YOUR-USERNAME/YOUR-REPOSITORY-NAME.git

git push --set-upstream origin main
```

-Contribute to an Existing Branch

1. Navigate to the repository and update it:

```
Unset
cd repo

git pull
```

2. Switch to the existing branch:

```
Unset
git checkout feature-a
```

3. Make changes, stage, commit, and push:

```
Unset
git add file1.md

git commit -m "edit file1"

git push
```

-Collaboration Models on GitHub

- **Shared Repository** → Teams work on a single repository with defined permissions.
- **Fork & Pull** → Contributors fork a repository, make changes, and submit a pull request for review.