SCRIPT MODULE 3

STEP 1:

1. Let’s begin by connecting our local Git project with GitHub.  
   GitHub acts like a **remote drive** for your code, just like Google Drive or Dropbox.  
   The git remote command is how we tell Git where to send or fetch code from.

**Push**: Sending your local changes to GitHub (like uploading a file to Drive)

**Pull**: Getting the latest changes from GitHub to your local system

**Clone**: Making a fresh copy of a GitHub project onto your local system

We’ll now create a GitHub repository and push our local Markdown resume project into it.

**Step 1: Create a New Repo on GitHub**

* Go to GitHub → Click **"New repository"**
* Name it, e.g., git-trial
* ✅ Don’t initialize with a README (since we’ll push local code)

Step 2: On Your Computer (Git Bash)

mkdir git-trial

cd git-trial

git init

Initializes a new **Git repository** locally (creates a hidden .git folder)

echo "# My First Git Repo" > README.md

Creates a file called README.md with a simple heading.

git add .

git commit -m "Initial commit"

Step 3: Link with GitHub & Push

git remote add origin <https://github.com/SoniaRajput/git-trial.git>

Adds a remote connection named origin pointing to your GitHub repo.

git push -u origin main

Pushes your local main branch to GitHub.  
-u sets origin/main as the default upstream branch.

1. git remote, push, pull, clone Explained

git remote -v

Lists current remotes. Useful to check where your local repo is connected.

git push origin main

Pushes your committed changes to the GitHub repo's main branch.

git pull origin main

Pulls the latest changes from GitHub (in case someone else pushed something).

git clone <https://github.com/SoniaRajput/git-trial.git>

Clones a **copy of the repo** from GitHub to your local machine.  
It will create a folder named git-trial.

STEP 2: GITHUB UI FEATURES

Once we upload our project, we’ll explore GitHub’s interface.  
GitHub is not just a storage place — it helps us understand the **history of changes**, **who made them**, and **why**.

* **Code** tab shows the project files
* **Raw** lets you view plain file content
* **Blame** shows who made each change and when
* **History** shows the list of all commits
* **Star** is like bookmarking your favorite project
* **Fork** creates your personal copy of someone else’s project, so you can experiment or improve it

**3. GitHub UI Features: Visual Tour**

1. **Code**
   * Click this tab to copy the repo's clone URL (HTTPS or SSH)
   * Useful for git clone
2. **Raw**
   * When viewing a file, click **Raw** to see it as plain text — helpful for downloading or viewing code without formatting.
3. **Blame**
   * View who wrote each line of code, when, and why (great for debugging).
4. **History**
   * Shows commit history for a file.
5. **Star**
   * Like "bookmarking" a repo — saves it to your Starred list.
6. **Fork**
   * Creates a personal copy of someone else's repo to your account.
   * Used to suggest changes without affecting the original repo.

STEP 3: Pull vs Clone vs Fork

* **Clone** is like copying your friend’s entire notebook when you join the class late.
* **Pull** is like asking your friend to send you new pages whenever they update their notes.
* **Fork** is like taking a public recipe and then modifying it with your own twist.

These concepts help us interact and contribute to each other’s projects while keeping control over our own versions.

| **Scenario** | **Action** |
| --- | --- |
| You want to start working on your own GitHub repo | git clone |
| You want to update your local repo | git pull |
| You want to contribute to someone else's project | **Fork** it → clone it → make changes → PR |