**Day 1: Creating Repository & Exploring Git Commands**

**What is Git?**

* Git is a **distributed version control system (VCS)**.
* It tracks changes in files and manages project history.

**What is Repository in Git?**

A repository is a storage space where your project's code and its entire history are saved using Git.

Types of Git Repositories:

* **Local Repository**
* Stored on your computer.
* Created using git init.
* Contains .git folder with all commit history, branches, and configuration.
* **Remote Repository**
* Stored on a server like GitHub
* Used for collaboration with others.
* You can **push** (upload) and **pull** (download) code to/from the remote repo.

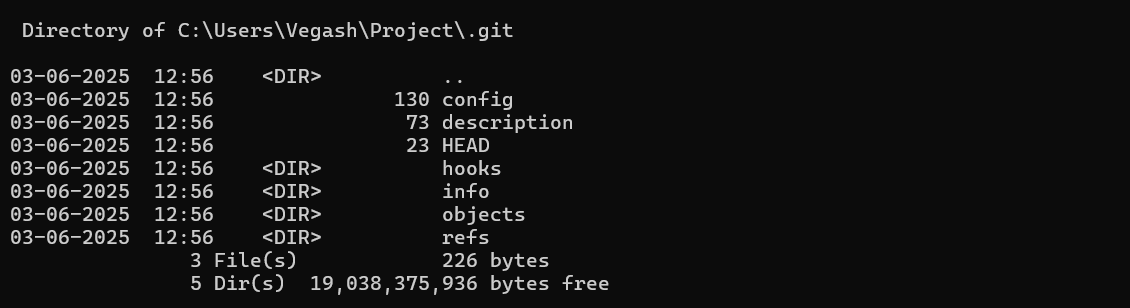
**Repository Setup**



* git init is the command used to **initialize a new Git repository** in your project folder.
* It sets up the directory so Git can start tracking changes to your files.

**What Happens When You Run git init?**

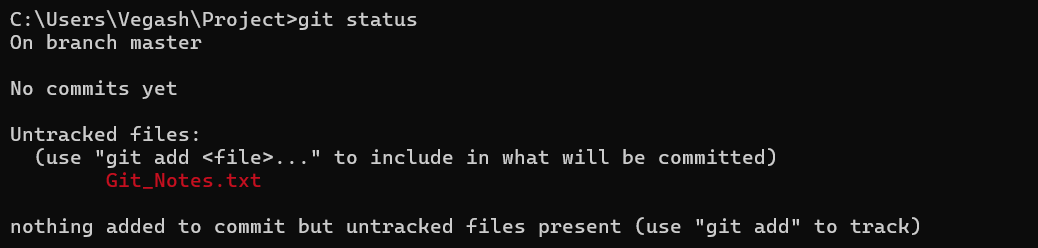
* A hidden folder named **.git** is created inside your project directory.
* This .git folder contains all the internal Git files like:
  + HEAD – Points to the current branch.
  + config – Stores configuration settings.
  + description – Description of the project.



**Git File Tracking Commands**

**git status**

* Check current state of files
* Shows modified, staged, untracked files
* Use before committing changes

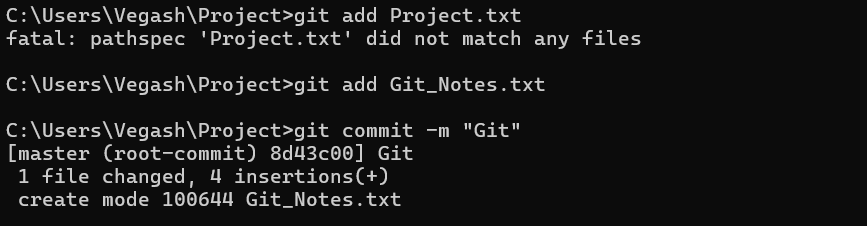


**git add**

* Stage files for commit
* Use git add <filename> or git add .
* Moves changes to staging area

**git commit**

* Save staged changes to repository
* Use git commit -m "message"
* Commits only staged files

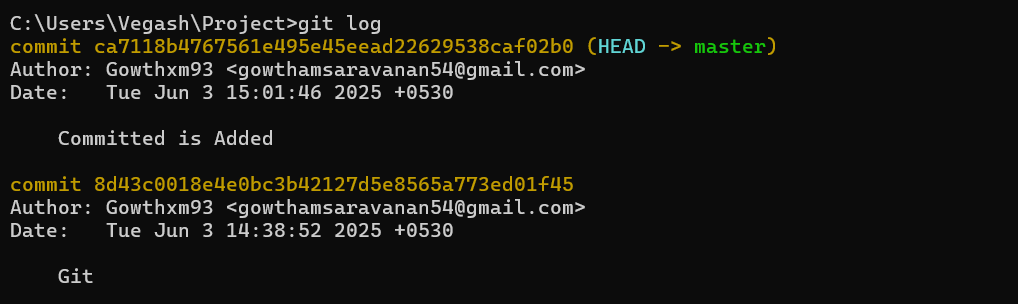


**git restore**

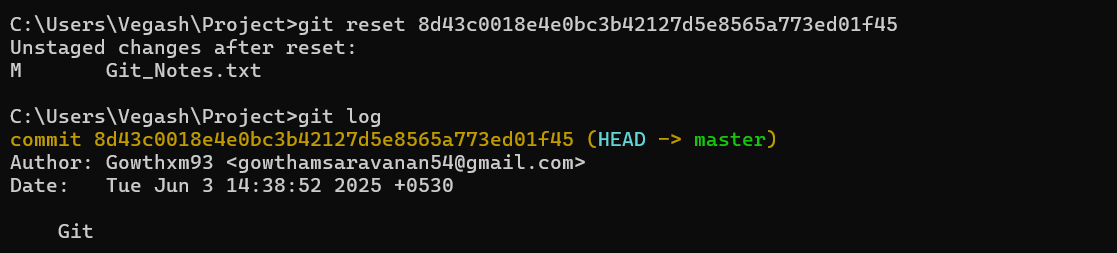
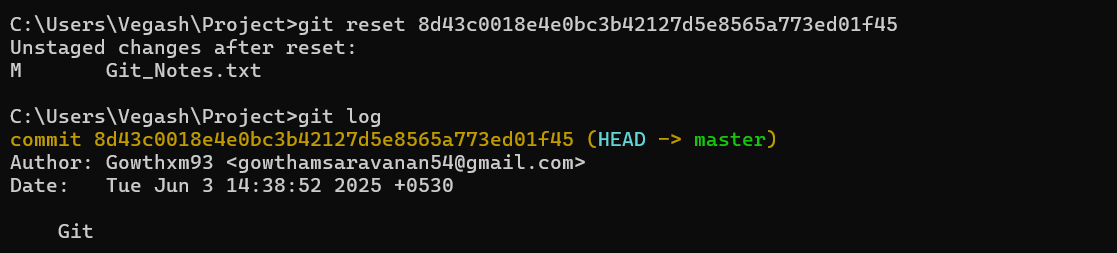
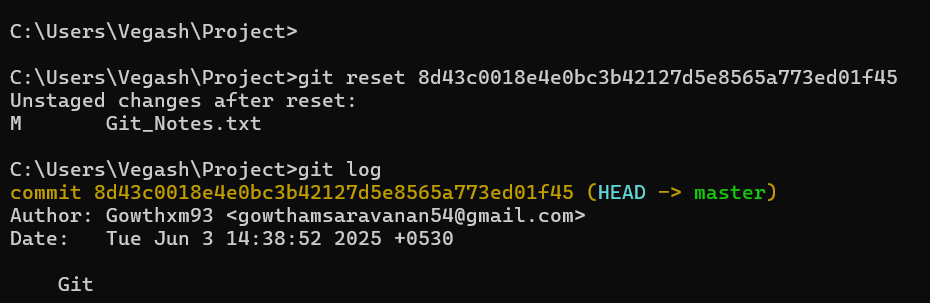
* Undo changes in working directory
* Use to discard uncommitted changes
* Use git restore <filename>

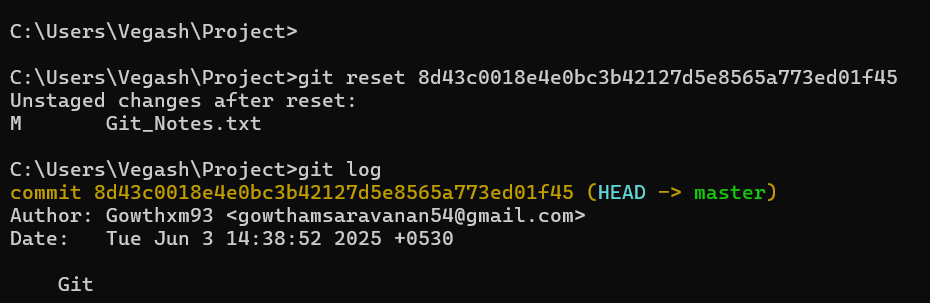
**git log**

* View commit history
* Shows commit IDs, messages, authors, dates



**git reset**

* Undo commits or unstage files
* Move HEAD to previous commit
* Use carefully to avoid losing work**Top of Form**

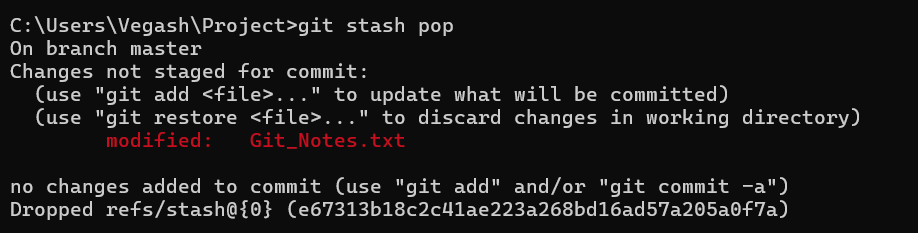


**git stash**

* Temporarily save uncommitted changes
* Useful when switching branches without committing
* Use git stash to store current changes safely

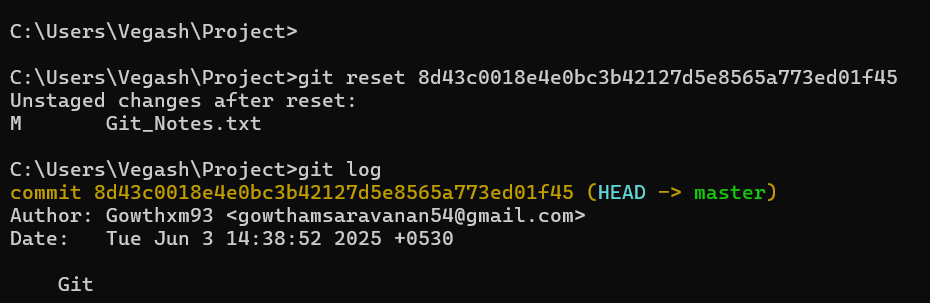
**git stash pop**

* Apply last stashed changes back to working directory
* Removes that stash from stash list

****

**git stash clear**

* Deletes all stashed changes permanently
* Use only when you're sure you no longer need stashed work

**Exploring GitHub and Connecting to local Project**

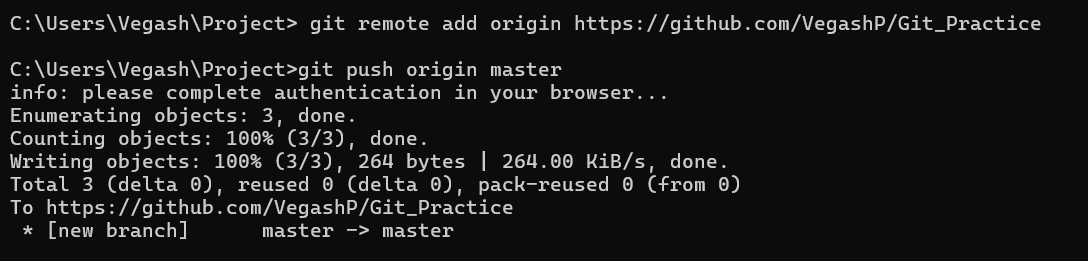
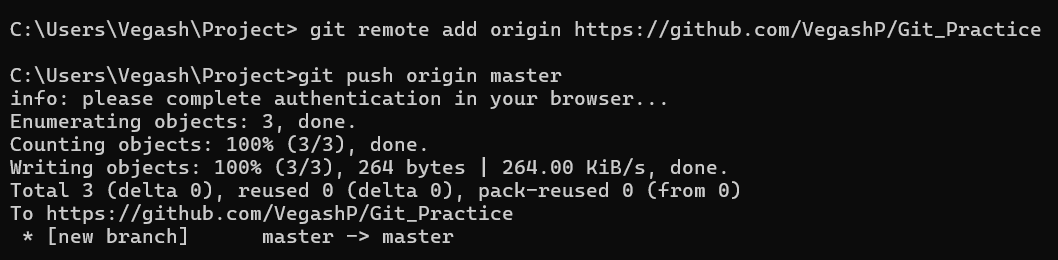
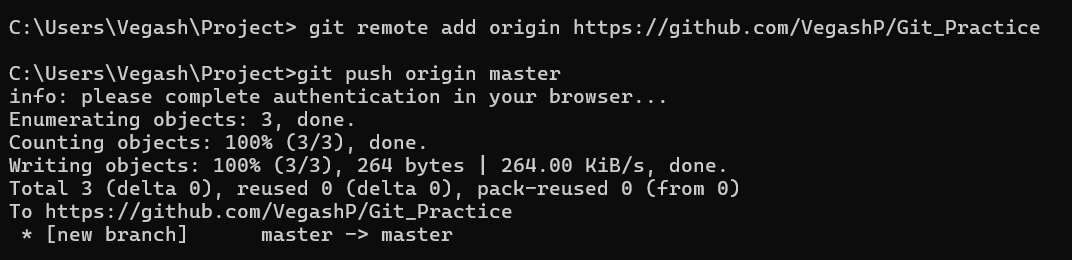
**After creating a GitHub repo:**

1. **Copy the repo HTTPS URL (e.g., https://github.com/username/myproject.git)**
2. **Go to your local project folder in CMD**
3. **Run this command to link the GitHub repo:**

**git remote add origin https://github.com/username/myproject.git**

1. **Push your code to GitHub:**

**git push origin master**

****

**Branching**

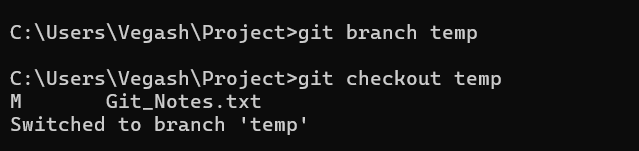
**What is Branching?**

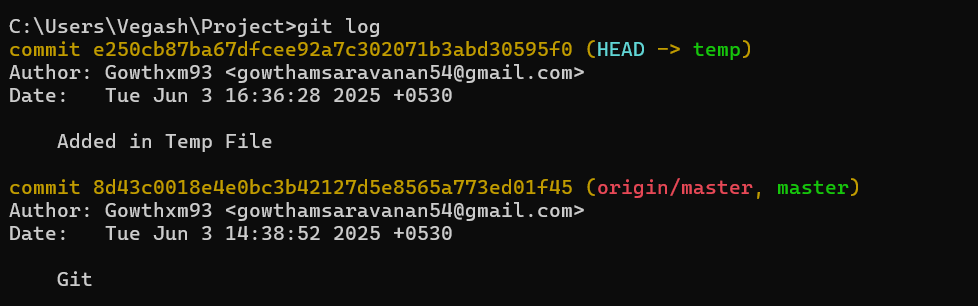
* A branch is like a separate line of development
* Lets you work on new features without affecting the main code
* Helps in testing, development, and team collaboration

**Branch Commands in Git**

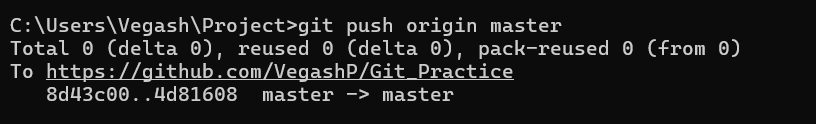
git branch (name) - Creates a new branch

git checkout (name) - Switches to the given branch





git push origin (name) - Pushes a local branch to the remote GitHub repository



git branch -d (name) - **Deletes** the branch **locally** & Only works if the branch is already **merged.**

git merge (name) - Merges the specified branch into the current branch

**Bottom of Form**