**Day – 01 : GIT AND GITHUB**

1. **Understanding what a Git repository**

**What is a Git Repository?**

A **Git repository** is a storage space where your project’s files and the entire history of all changes made to them are tracked.

Type of Repository:

1. Local Repository (A Git repository stored on your own machine for offline development and version control.)
2. Remote Repository (A Git repository hosted on a server (like GitHub or GitLab) used for sharing and collaborating with others.)

**Initializing a New Repository using git init**

* Creates .git folder in the current directory
* Converts the directory into a Git repository
* The .git folder is a hidden directory created when you run git init in a project. It is the heart of your Git repository**.**

****

**Why .git folder is used:**

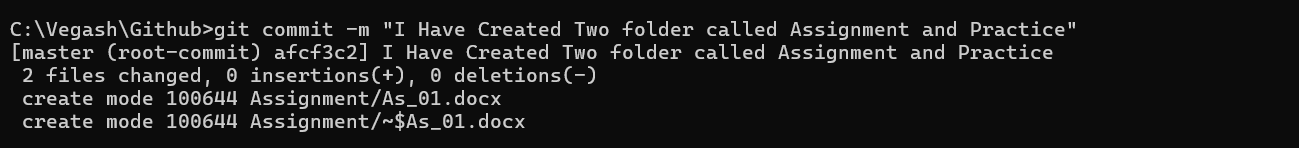
* Stores full version history of your project.
* Tracks commits, branches, and tags.
* Maintains configuration settings (like your username/email).
* Manages staging area, logs, and references.

**Key contents inside .git folder:**

|  |  |
| --- | --- |
| **File/Folder** | **Purpose** |
| HEAD | Points to the current branch. |
| config | Contains Git config (user, remote info). |
| refs/ | Stores branches and tags. |
| objects/ | Stores all commits, files, and trees. |
| index | Tracks staged changes (staging area). |
| logs/ | Keeps logs of changes to refs (like HEAD). |

1. **Committing Changes:** (Git add, commit, and commit messages)

* git add stages changes (files or updates) for commit.
* git commit saves the staged changes with a message.



* Once committed using git commit, the changes are stored in the local repository's history.

**Basic Git Commands:**

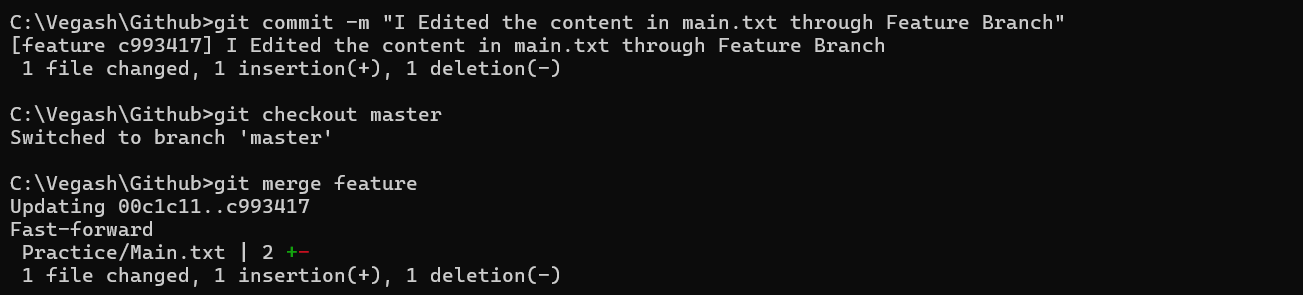
|  |  |
| --- | --- |
| **Command** | **Description** |
| **git init** | **Initializes a new Git repository** |
| **git status** | **Shows current changes and staging info** |
| **git add <filename>** | **Stages a specific file for commit** |
| **git add -A or git add .** | **Stages all changes (added, modified, deleted files)** |
| **git log** | **Displays the commit history** |
| **git config --global user.Name** | **Sets your Git username globally** |
| **git config --global user. Email** | **Sets your Git email globally** |

1. **Branching and Merging**

* **Branching** means making a copy of your main code to work on something new.
* You can add features or fix bugs **without touching the main project**.
* **Merging** means putting your changes from that copy **back into the main code**.

|  |  |
| --- | --- |
| **Command** | **Description** |
| git branch | Lists all branches in the repository |
| git branch <branch-name> | Creates a new branch |
| git checkout <branch-name> | Switches to an existing branch |
| git checkout -b <branch-name> | Creates and switches to a new branch in one step |
| git merge <branch-name> | Merges the given branch into the current branch |
| git branch -d <branch-name> | Deletes a branch (after it's merged) |





* I created a new branch called feature using git branch feature.
* I switched to the feature branch using git checkout feature.
* In this branch, I added the line **"Modern cars come equipped with advanced features"** to the file.
* These changes did **not affect the master branch**, since I was working in a separate branch.
* The feature branch works like a **copy of the project**, which is useful for safe editing or testing.
* After confirming the changes, I switched back to the master branch using git checkout master.
* Then I merged the changes from feature into master using git merge feature

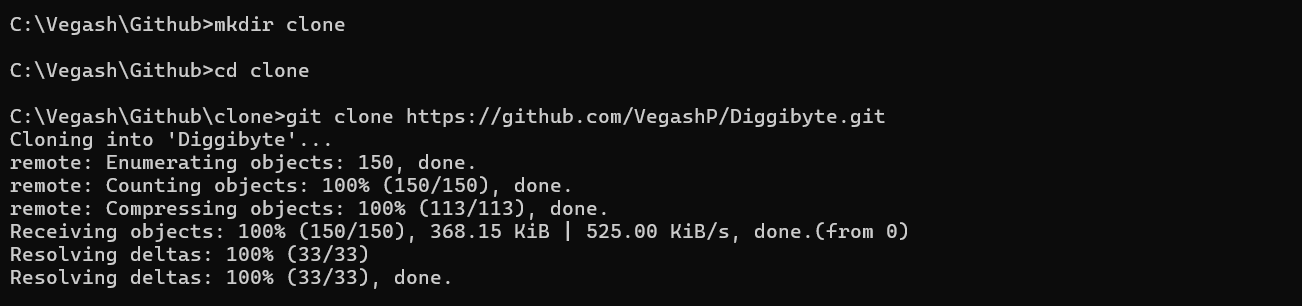


* Finally, I deleted the feature branch using git branch -d feature to clean up.

1. **Remote Repositories:**

**Cloning in Git means creating a local copy of a remote repository.**

* Cloning creates a local copy of a remote repository.
* It includes all project files and complete Git history.
* Used to start working on someone else’s project locally.
* Command: git clone https://github.com/username/diggibyte.git
* A new folder named diggibyte is created in your system.
* The remote URL (origin) is set automatically for future pushes or pulls.



* 1. **What is Pushing Changes to Remote Repositories:**

Pushing means uploading your local commits (changes saved with git commit) to a remote repository like GitHub.

* **Syncs local work with remote repository** (e.g., GitHub).
* **Backs up your code** on a remote server.
* **Enables collaboration** with team members — others can pull your latest changes.
* **Provides visibility** to others viewing the project (e.g., on GitHub).
* **Helps in code review processes** when working in teams.

NOTE: In the previous operation, I had cloned the project into my current local repository. But when I tried to commit, it showed some warnings. So, I went back to a previous commit — essentially to the state before cloning — and now I’m pushing the changes to the remote repository.

1. **Collaborative Work:**

**Clone Repo**: git clone <url> – copy project to your local.

**Remote Setup**: git remote add origin <url> – link to remote.

**Fetch**: git fetch – download changes without merging.

**Pull**: git pull – fetch + merge changes into your branch.

**Push**: git push origin <branch> – upload your changes.

**Branching**: git branch, git switch – work separately without affecting main.

**Pull Request**: Submit changes for review (on GitHub).

**Best Practices**:

* Commit often
* Pull before push
* Use clear commit messages
* Don’t push directly to main

**Day 02: (19-06-2025)**

**5. Collaborative Work:**

5.1 Working with others on the same project.

* I created a remote GitHub repository for the project.
* I added Mrudula (mrudula-diggibyte) as a collaborator to the repo.
* She accepted the invitation via email and was granted access.

**We both contributed to the project using proper Git collaboration practices such as:**

* Working on separate branches.
* Making commits and pushes to the remote repo.
* Communicating and reviewing each other’s changes.

**A screenshot of a computer

AI-generated content may be incorrect.**

**5.2 Handling merge conflicts.**

* I created a remote GitHub repository with one Python file.
* I cloned the repository into my local system.
* I then made a change directly in the remote repository (on GitHub) — edited one line in the Python file and committed it.
* Without pulling the recent remote change, I edited the same line in the local Python file.
* I committed the local changes.
* After that, I ran git pull to fetch and merge remote changes into my local branch.
* Since both remote and local changes modified the same line, Git raised a merge conflict.
* I manually resolved the conflict by choosing or combining the correct version of the line.
* Then I added and committed the resolved file locally.
* Finally, I pushed the resolved changes to the remote repository successfully.

**A screenshot of a computer

AI-generated content may be incorrect.**

**6. Git Ignore:**

* Ignoring Files and Folders Using .gitignore
* .gitignore is a special file used to tell Git not to track certain files or folders.
* You can use it to ignore things like:
  + Temporary files
  + Log files
  + IDE or editor files
* This helps keep the repo clean and avoids pushing unnecessary or sensitive files.

**7. Creating and Managing Tags in Git**

**What is Tagging in Git?**

Tagging is a way to **mark specific points in history** as important — commonly used for **releases (v1.0, v2.1, etc.)**. Tags are like **bookmarks** to a specific commit.

|  |  |  |
| --- | --- | --- |
| **Type** | **Command Example** | **Description** |
| **Lightweight Tag** | git tag \_Tag\_Name\_ | A simple pointer to a commit (like a branch but doesn’t move). |
| **Annotated Tag** | git tag -a \_Tag\_Name\_ -m "Message" | Contains some Extra Details like Date Congif. |

|  |  |  |
| --- | --- | --- |
| **Feature** | **Tag** | **Branch** |
| Moves | Fixed to a commit | Moves with new commits |
| Editable | Not usually modified | Can be edited and merged |
| Purpose | Release version marking | Ongoing development |
| Checkout | Can checkout, but can’t commit to it | Can checkout and continue work |

**A computer screen shot of a black screen

AI-generated content may be incorrect.**