**GitHub**

GitHub is a **web-based platform** used for **collaborative work and project management**. It allows users to **store, share, modify, and delete** project data seamlessly. Built on **Git**, GitHub helps developers **track changes, collaborate efficiently, and manage code effectively**.

**Features of GitHub:**

* **Open Source** – Supports open-source development and contributions.
* **Collaborative Tools** – Enables teamwork through issues, discussions, and pull requests.
* **Code Reviewing** – Allows reviewing and suggesting improvements before merging code.
* **Code Hosting and Sharing** – Stores and shares project files in repositories.
* **Code Management** – Tracks changes, maintains version history, and supports branching.

**If GitHub Didn’t Exist: Key Impacts**

* **Local Versioning** – Developers would rely on manually saving different file versions, making tracking changes inefficient.
* **Email & File Sharing** – Collaboration would be slower and chaotic, with no centralized version control.
* **Community Isolation** – Developers would struggle to find and contribute to open-source projects, reducing learning and networking opportunities.
* **Impact on Open Source** – Growth of projects like **Linux, TensorFlow, and React** would slow down, and innovation would be fragmented.

Without **GitHub**, software development would be **less efficient, less collaborative, and slower to innovate**.

**Git (Global Information Tracker):**

Git is a **distributed version control system (VCS)** that tracks changes in any set of computer files, usually used for **coordinating work among programmers** who are collaboratively developing source code.

Git acts as a **medium between the development environment and the storing platform**.

**Key Features of Git:**

* **Open Source and Community Support**
* **Branching and Merging**
* **Distributed Version Control**
* **Staging Area**
* **Pull Requests and Code Push**

**Three Components of Git**

Developer (Local Machine)

│

▼

+-------------------+

| Working Directory | (Modified Files)

+-------------------+

│

git add ▼

+-------------------+

| Staging Area | (Files Ready for Commit)

+-------------------+

│

git commit ▼

+-------------------+

| Local Repository | (Committed Changes)

+-------------------+

│

git push ▼

+-------------------+

| Remote Repository | (GitHub, GitLab, etc.)

+-------------------+

**Check Configuration in Git Bash**

To check the existing configuration, type:

git config --list

Example Output:

user.name = pavan kiran

user.email = pavankiran26082003@gmail.com

**If Git is Not Linked, Configure it Using the Following Commands:**

git config --global user.name "Your GitHub Name"

git config --global user.email "your@email.com"

**How to Upload Files to GitHub Using Git**

**Process 1 (Simplified Steps):**

1. Select a folder and open **Git Bash Terminal**.
2. Run the following command:
3. git init # (Optional if cloning)
4. Clone the repository:
5. git clone "URL"
6. Change the directory:
7. cd foldername # or close and reopen terminal inside folder
8. Copy files into the folder.
9. Check the status using:
10. git status
11. Add files:
12. git add filename # or use git add . to add all files
13. Commit changes with a meaningful message:
14. git commit -m "Your commit message"
15. Push the code to GitHub:
16. git push -u origin main

**Process 2 (Detailed Steps)**

**1. Install Git**

* Download and install Git from [git-scm.com](https://git-scm.com/).
* Verify installation by running:
* git --version

**2. Configure Git**

* Set your username and email:
* git config --global user.name "Your Name"
* git config --global user.email "your-email@example.com"

**3. Initialize a Git Repository**

* Navigate to your project folder and initialize Git:
* git init

This creates a hidden .git folder in your project directory.

**4. Add and Commit Files**

* Add files to the staging area:
* git add filename # Add a specific file
* git add . # Add all files
* Commit the changes with a message:
* git commit -m "Initial commit"

**5. Connect to a Remote Repository**

* Create a repository on **GitHub/GitLab/Bitbucket**.
* Link your local repository to the remote:
* git remote add origin https://github.com/your-username/repo-name.git
* Push your code to GitHub:
* git push -u origin main

**6. Clone a Repository**

If you want to copy an existing repository:

git clone https://github.com/username/repo-name.git

**7. Pull and Push Changes**

* To get the latest changes from the remote repository:
* git pull origin main
* To send your local changes to GitHub:
* git push origin main

**8. Check Git Status**

git status

**This shows the current state of your repository.**

**9. Create and Switch Branches**

* Create a new branch:
* git branch new-feature
* Switch to the new branch:
* git checkout new-feature
* Merge a branch into the main branch:
* git checkout main
* git merge new-feature

**10. Undo Changes**

* Undo changes before committing:
* git checkout -- filename
* Remove a file from the staging area:
* git reset filename
* Undo the last commit:
* git reset --soft HEAD~1

**Important Notes:**

* **When creating a new project**, use a **new repository**.
* **When working in a team**, **clone the repository**, make changes, and then **deploy**.

**Create a new repository on the command line**

echo "# git-blash" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/pavan26082003/git-blash.git

git push -u origin main

**…or push an existing repository from the command line**

git remote add origin https://github.com/pavan26082003/git-blash.git

git branch -M main

git push -u origin main