GIT & GITHUB

“From Zero to Real Project Deployment”



INTRODUCTION

Why GitHub Matters:

- **Version Control:** Track changes in your code and manage project history
- **Collaboration:** Work with team members on the same project
- **Portfolio:** Showcase your projects to recruiters and companies

Before Learning GitHub

We need to understand the difference between

Git and GitHub



What is Git?

Git is a tool used to track changes in files, especially in coding projects. It keeps a record of every change so you can see what was modified and when. Git allows you to go back to previous versions if something goes wrong. It also supports collaboration by allowing multiple people to work on the same project without overwriting each other's work.

- Developed by Linus Torvalds
- First released in 2005

Example:

Imagine you are writing notes in Google Docs with your friends:

•You make changes → edits are saved automatically

•You can see previous versions → version history

•Many people can edit at the same time → teamwork

Connect to Git:

•Git saves every change like Google Docs history

•You can go back to old versions anytime

•Multiple people can work together without losing data



KEY FEATURES OF GIT

- **Version Tracking:**

Records every change made to your files so you can review or restore any previous version.

- **Distributed System:**

Each user has a complete copy of the project, allowing work even without an internet connection.

- **Branching & Merging:**

You can work on new ideas separately (branch) and later combine them into the main project safely

- **Collaboration:**

Multiple developers can work on the same project without overwriting each other’s work.

What is GITHUB?

GitHub is a cloud-based platform used to store, manage, and share code online. It works with Git to help developers collaborate on projects, track changes, and work together from anywhere. GitHub also allows you to showcase your projects and build your portfolio.

- Founded by Tom Preston-Werner, Chris Wanstrath, PJ Hyett, and Scott Chacon
- Launched in 2008
- Acquired by Microsoft in 2018



Why GitHub is Used

- **Store Code Online:**

  Keeps your projects safe and accessible from anywhere

- **Collaboration:**

  Multiple people can work on the same project easily

- **Track Changes:**

  See what changes were made and when

- **Backup & Security:**

  Your code is saved online, so it won’t be lost

- **Portfolio:**

  Show your projects to recruiters and companies



Key Features of GitHub

- **Repositories:**

  A folder that contains your project files and code.

- **Commit:**

  A save point of your work (like saving a file).

- **Branch:**

  A separate copy to work on new changes without affecting the main project.

- **Pull Request:**

  A request to add your changes to the main project.

- **GitHub Pages:**

  Used to create and host a website from your project.

Git vs GitHub

Feature	Git	GitHub
What it is	Tool	Platform (Website)
Meaning	Tracks changes in code	Stores and shares code online
Works on	Your computer (local)	Internet (cloud)
Internet needed	❌ Not required	✅ Required
Main use	Version control	Collaboration & backup
Example	Saving versions of files	Sharing project with others

Steps to Push Project to GitHub



Installation & Setup

Step 1: Install Git (Windows)

Go to the official website: https://git-scm.com/install/windows

Click Download for Windows

Open the downloaded file and start installation

Installation Steps (Keep Default)

- Click Next for most options
- Use default settings (recommended for beginners)
- Click Install
- Click Finish after installation



About

Learn

Tools

Reference

Install

Community

Install

Latest version: 2.54.0 (Release Notes)

Windows | macOS | Linux | Build from Source

Click here to download the latest (**2.54.0**) **x64** version of **Git for Windows**. This is the most recent maintained build. It was released **14 days ago**, on 2026-04-20.

Other Git for Windows downloads

Standalone Installer

Git for Windows/x64 Setup.



Git for Windows/ARM64 Setup.

Portable ("thumbdrive edition")

Git for Windows/x64 Portable.

Git for Windows/ARM64 Portable.

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Verify Installation

- Open Command Prompt or Git Bash
- Type:

```
git --version
```

- If installed correctly, it will show the Git version

Step 2: Create GitHub Account

Go to the official website : https://github.com

Sign Up Process

1.Click Sign up

2.Enter your Email address

3.Create a Password

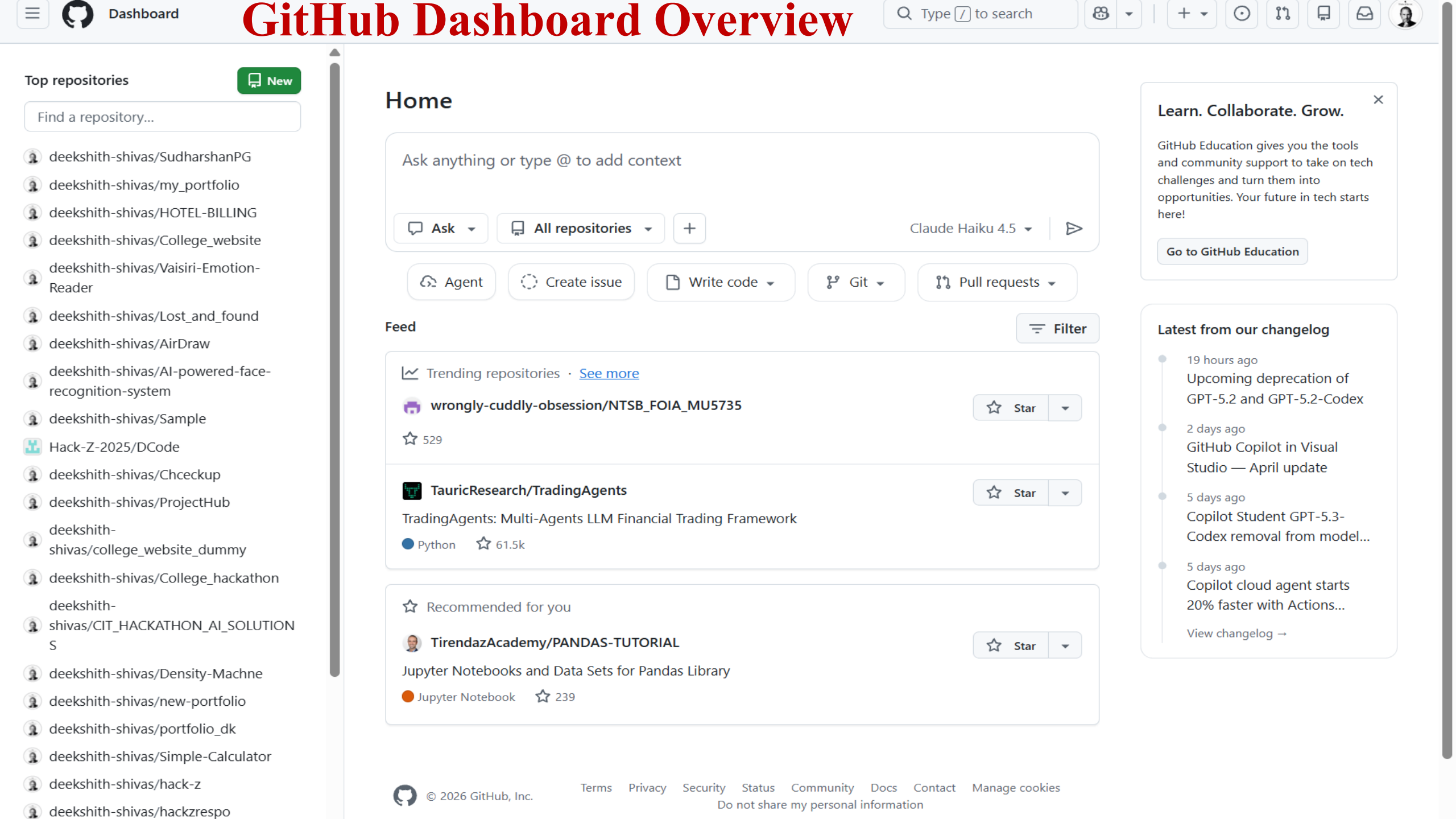
4.Choose a Username

5.Click Continue

Verification

Verify your email (OTP or code) or Complete simple verification (captcha)

GitHub Dashboard Overview



Step 3: Git Configuration (Meaning & Explanation)

[open Git Bash or VScode Terminal]

Git configuration is used to set your identity (name and email) so that Git can record who makes changes in a project.

Commands:

```
git config --global user.name "Your Name"
git config --global user.email your@email.com
```

- These commands tell Git who you are.
- Whenever you commit, Git records your **name and email** with that change.
- You need to do this **only once** (first-time setup)
- No need to repeat for every project

Basic Git Commands

git init

- Initializes a new Git repository (starts tracking your project) by creating a hidden **.git** folder inside your project directory
- This folder stores all **history, versions, and tracking data**

git add .

- Adds all files and changes to the **staging area** (a place where changes are prepared before saving (commit).
- “.” means all files in the folder
- It collects all the changes and prepares them to be saved.”

git commit -m "message"

- Saves the changes with a message (creates a version)
- Saves them permanently in Git
- **commit** = save your work
- **-m "message"** = short description of what you changed

git status

- Shows the current status of files and changes
- Tells which files are changed
- Shows which files are added (staged)
- hows which files are not yet saved (unstaged)



Step 4: Initialize Git Repository

Command:

```
git init
```

Meaning :

- Initializes a new Git repository in your project folder
- Creates a hidden **.git** folder
- Starts tracking all changes in the project

Option 1: Using VS Code (Direct Method)

Steps:

1. Open your project folder in **VS Code**
2. Open Terminal → **Terminal → New Terminal**
3. Run:

```
git init
```

- VS Code terminal automatically opens in your project folder
- No need to use cd
- Git starts tracking immediately

Option 2: Using Git Bash

Steps:

1.Open **Git Bash**

2.Navigate to your project folder:

```
cd "your-folder-path"
```

Example:

```
cd "C:\Users\YourName\Documents\my-project"
```

3.Then run: `git init`

- Git Bash opens in a default location
- You must first go to your project using cd
- Then initialize Git

Step 5: Add Files to Staging Area

Command :

```
git add .
```

Meaning :

- Adds all files and changes to the staging area
- Prepares files to be saved (commit)
- “.” means all files in the current folder



Step 6: Commit Changes

Command:

git commit -m “initial commit”

Meaning :

- Saves the staged changes as a new version
- **- m** → adds a message describing the changes
- Creates a snapshot of your project

This step saves your work permanently in Git

Step 7: Create a Repository on GitHub

Go to GitHub

- Open GitHub
- Login to your account

Create New Repository

1. Click “+” (top right corner)
2. Select “New repository”

Fill Repository Details

- Repository Name: (e.g., my-project)
- Description: (optional)
- Select Public (for sharing) or Private

Public Repository

A public repository is a project that is accessible and visible to everyone on GitHub.

Private Repository

A private repository is a project that is accessible only to the owner and selected users.

Final Step

- Click Create Repository

✓ Your repository is now ready

⚠️ Important Note

Do NOT add a README file while creating the repository

Step 8: Connect Local Repository to GitHub

Where to Find the Command

- Go to your repository on GitHub
- Click **“Code”**
- Copy from **“Push an existing repository from the command line”**

Push Existing Project

```
git remote add origin https://github.com/username/my-project.git
git branch -M main
git push -u origin main
```

git remote add origin

- Connects your local project to GitHub
- **origin** = name given to your GitHub repository
- Stores the repository link for future use

git branch -M main

- Sets your branch name to **main**
- Ensures your local branch matches GitHub's default branch

git push -u origin main

- Uploads your project to GitHub
- **-u** → sets default upstream (so next time you can just use git push)
- Sends code from local → GitHub

PROJECT SUCCESSFULLY UPLOADED

- Initialized Git in our project
- Added and committed files
- Created a repository on GitHub
- Connected local project to GitHub
- Pushed code successfully
- Your project is now live on GitHub
- Accessible from anywhere
- Ready to share with others

What is README.md?

A **README.md** file is a document that explains your project.

It provides important information like purpose, usage, and instructions.

Why README is Important:

- Explains what your project does
- Helps others understand your code
- Provides setup and usage instructions
- Improves your project presentation

What to Include in README:

- Project Title
- Description
- Features
- Installation Steps
- Usage Instructions



Deploy Your Project on GitHub



Deploy Your Project on GitHub

[Using GitHub Pages]

Steps:

1. Go to your repository on GitHub
2. Click **Settings**
3. Go to **Pages** (left side menu)
4. Under **Source**, select:
    1. Branch → main
    2. Folder → / (root)
5. Click **Save**

Result: Your website will be live at: https://username.github.io/repository-name

Getting Code from GitHub



Option 1: Clone Repository

Command :

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

What Happens Step-by-Step:

- You run the command
- Git connects to GitHub
- Downloads the repository
- Creates a folder in your system
- Project is ready to use



Example:

Command:

git clone https://github.com/user/my-project.git

Result:

- Folder created → my-project/
- Inside → all files + history

Next:

- Open the folder
- Start editing or running the project



Option 2: Download ZIP

Steps:

1. Go to the repository on GitHub
2. Click Code button
3. Select Download ZIP
4. Extract the ZIP file on your computer

- Downloads the project as a ZIP file
- Contains only project files (HTML, CSS, JS, etc.)
- Does NOT include Git history or tracking
- No .git folder is included

Next :

1. Files are downloaded as compressed ZIP
2. You extract it
3. A normal project folder is created
4. You can open and use the files

Example:

- Download project → project.zip
- Extract → project/ folder
- Open in VS Code → Start working



Handling File Changes in Git



❖ If You Modify (Edit) Files

File is changed

Commands:

git add .

git commit -m “updated files”

git status

git push

git status : checks the current state of your project and shows what changes have been made.

It displays modified, new, and deleted files, and indicates what is ready to commit.

❖ If You Add New Files

A new file is created (not tracked by Git)

Check Status:

```
git status                    // Shows the file as untracked
```

Add & Commit:

```
git add .

git commit -m "added new file“

                              //Git now starts tracking and saves the file
```

Push to GitHub:

```
git push origin main          //Uploads the new file to GitHub
```

❖ If You Delete Files

A file is removed from your project folder

Check Status:

```
git status                    //Shows the file as deleted
```

Add & Commit:

```
git add .

git commit -m "deleted file"
                              //Git stages the deletion and saves it as a new version
```

Push to GitHub:

```
git push origin main          //The file is also removed from GitHub
```

Git Workflow (How Git Works)

Main Flow:

Working Directory → **Staging Area** → **Repository**

1 Working Directory

- This is your project folder.
- You create or edit files here.

Example:

You change code in index.html.



This is the place where you work on your files.

2 Staging Area

- You select changes using:

```
git add .
```

- It prepares files to be saved.

Think:

"Which changes do I want to save?"



Changes are ready to be saved.

You decide which changes to include in the next save.

3 Repository

- You save changes using:

```
git commit -m "message"
```

- Git stores it as a version (snapshot).

Now your changes are permanently saved.



A new version (snapshot) is created and stored.

Simple Understanding:

Edit → **Add** → Save

Work → Prepare → Store

Real Example:

1. Edit file → Working Directory
2. Run **`git add .`** → Staging Area
3. Run **`git commit -m "message"`** → Repository

→ Version created successfully!

WHAT WE COVERED

- Introduction to Git and GitHub
- Difference between Git and GitHub
- Installation and setup
- Basic Git commands (init, add, commit, status)
- Creating and pushing a project to GitHub
- Cloning and downloading projects
- Handling file changes (add, edit, delete)
- Deploying project using GitHub Pages



Get This PPT / PDF

🔗 **Download**

https://deeks-github.vercel.app/

💜 Thank You!

