**Placement Empowerment Program**

***Cloud Computing and DevOps Centre***

Deploy Your Static Website Using Github Pages: Host your local Git repository’s static website directly using Github pages

Name: Indhumathi S Department: CSE



**Introduction**

GitHub Pages is a free and straightforward hosting service provided by GitHub for static websites. By leveraging this feature, you can host personal, project, or organization websites directly from a GitHub repository. This guide walks you through the process of deploying your static website using GitHub Pages.

**Objectives**

By the end of this POC, you will:

1.Understand the basics of GitHub Pages and how it hosts static websites.

2.Learn to configure a GitHub repository for website hosting.

3.Successfully deploy a static website using GitHub Pages.

**Importance of deploy your static website using Github pages**

**Free Hosting** – GitHub Pages provides a **cost-free** solution for hosting static websites, eliminating the need for paid hosting services.

**Easy Deployment** – Websites can be deployed **directly from a**

**GitHub repository** without needing complex configurations or additional tools.

**Automatic Updates** – Any changes pushed to the repository are **automatically updated** on the live website, ensuring seamless maintenance.

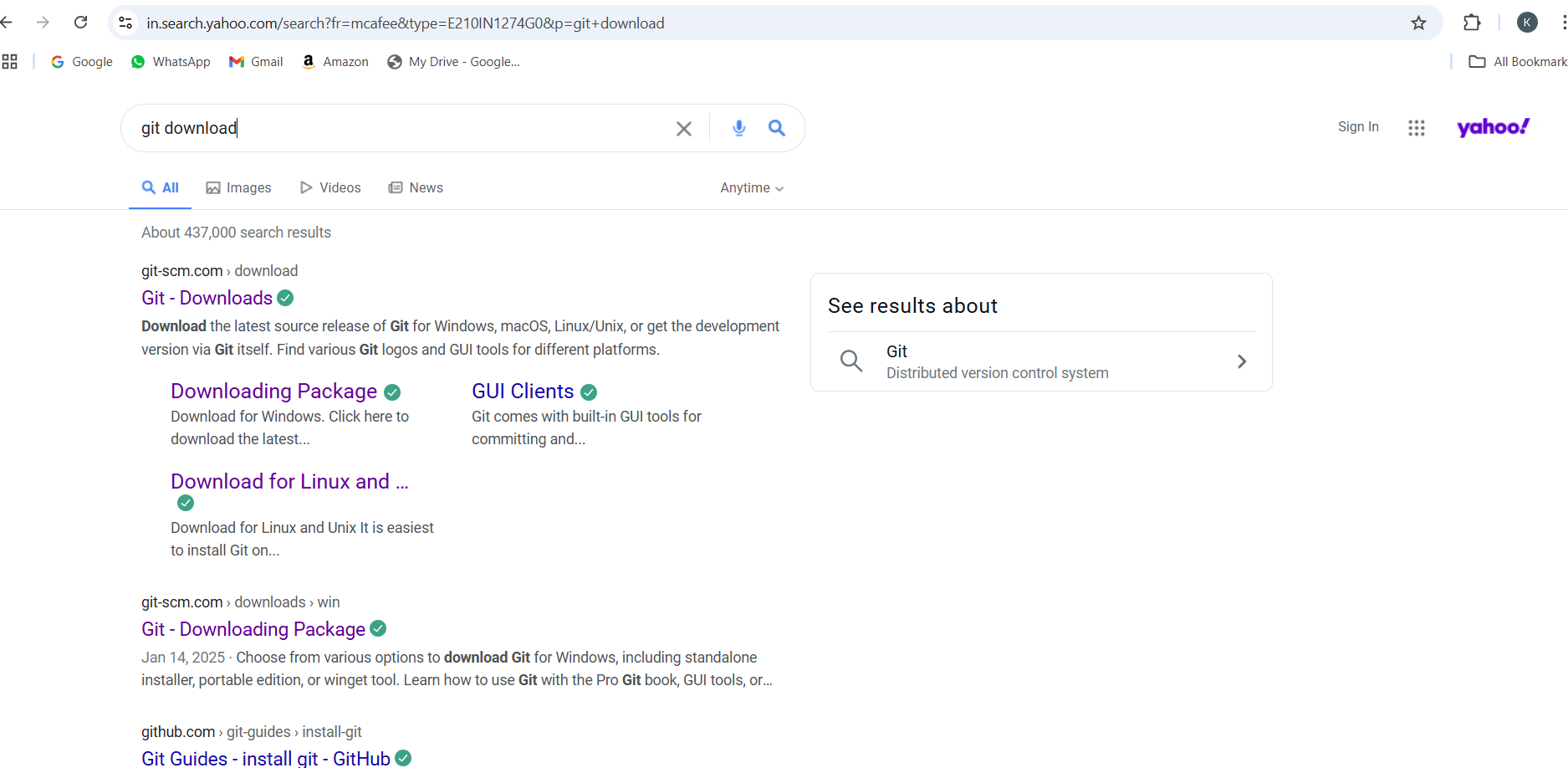
**Version Control** – Since the website is hosted on GitHub, you get **built-in version control**, allowing easy tracking of changes and rollbacks if needed.

**Custom Domain Support** – GitHub Pages allows you to configure a **custom domain** for a professional web presence with proper branding.

**Step-by-Step Overview**

Step 1:

Search for "Git" in Chrome, download it, and click the "Downloads" option on the website.



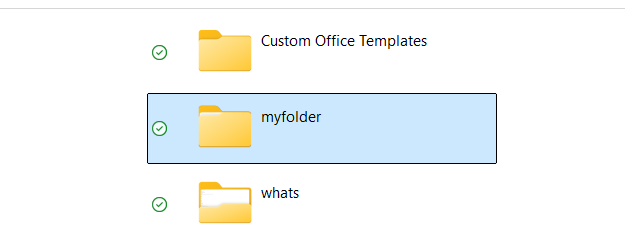


Step 2:

Click the **Windows** option on the download page and follow the installation wizard.

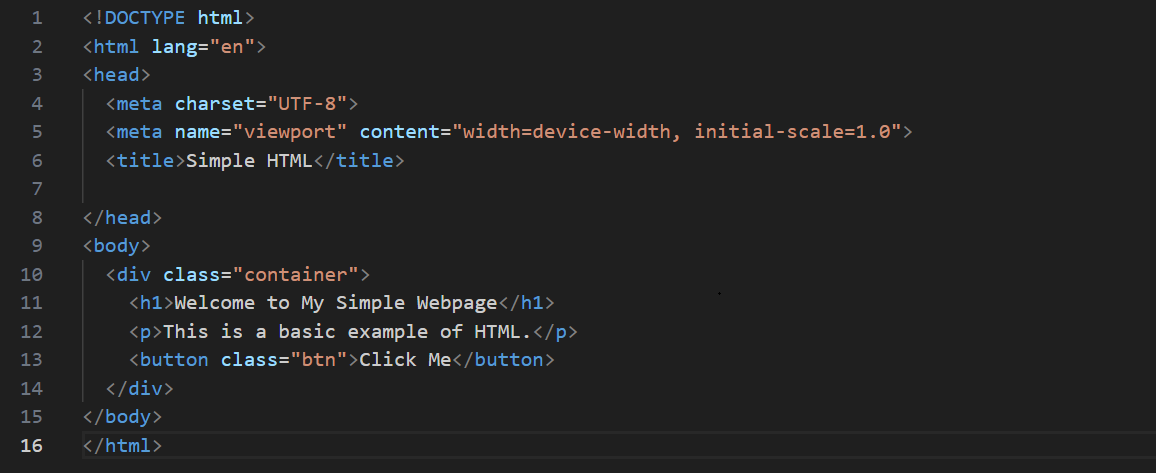
Step 3:

First, open your terminal or command prompt and create a new folder for your project. Let's name it myfolder



Step 4:

Open VS code, create a simple HTML file named **myweb.html**. You can write some basic HTML



Step 5:

Output for the HTML code



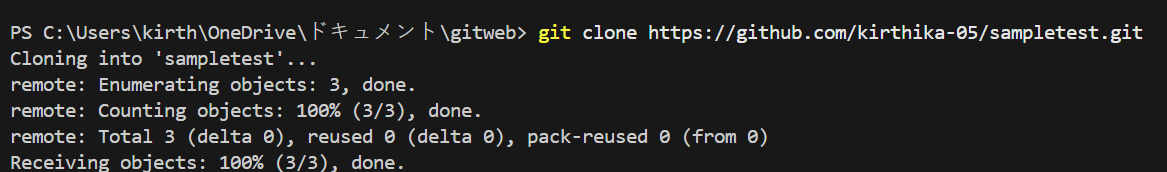
Step 6:

Open the **VS code Terminal** and set the path to the folder named that we created.

**Git clone**

**B**asic syntax for git clone is **git clone <repository\_url>**

 Find the repository URL on GitHub (or another Git hosting service).  
 Run the command:



Step 7:

The command **cd website/** changes the directory to the website folder, ensuring you are in the correct project location for deployment. This allows you to run Git commands like git init, git add ., and git push to deploy your static website using GitHub Pages.



Step 8:

Now, initialize Git by typing this command:

**git init**

This command will create a .git folder inside your project folder, which tells Git to start tracking your files.

Next, we need to tell Git to start tracking your website files.

To tell Git which files to track, use the git add command. If you want to track all the files in your folder, type

**git add .**

Basic syntax is **git add newrepo.html** This command adds all the files to Git’s tracking system.

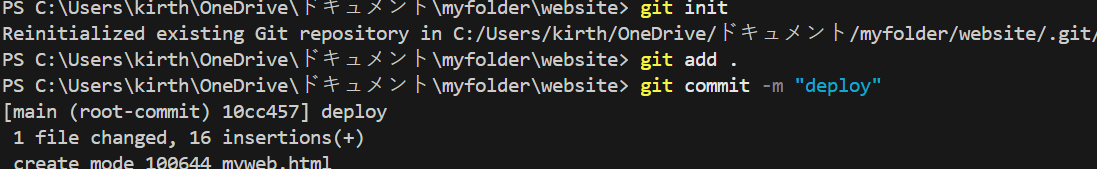
**git commit**

Basic syntax is **git commit -m "Your commit message"**

A **commit** is a snapshot of the project's current state. When you commit, Git records changes to tracked files and saves them in the repository history.

 **git commit**: Saves changes in the local repository.

**-m "message**": Adds a message describing what the commit does.



Step 9:

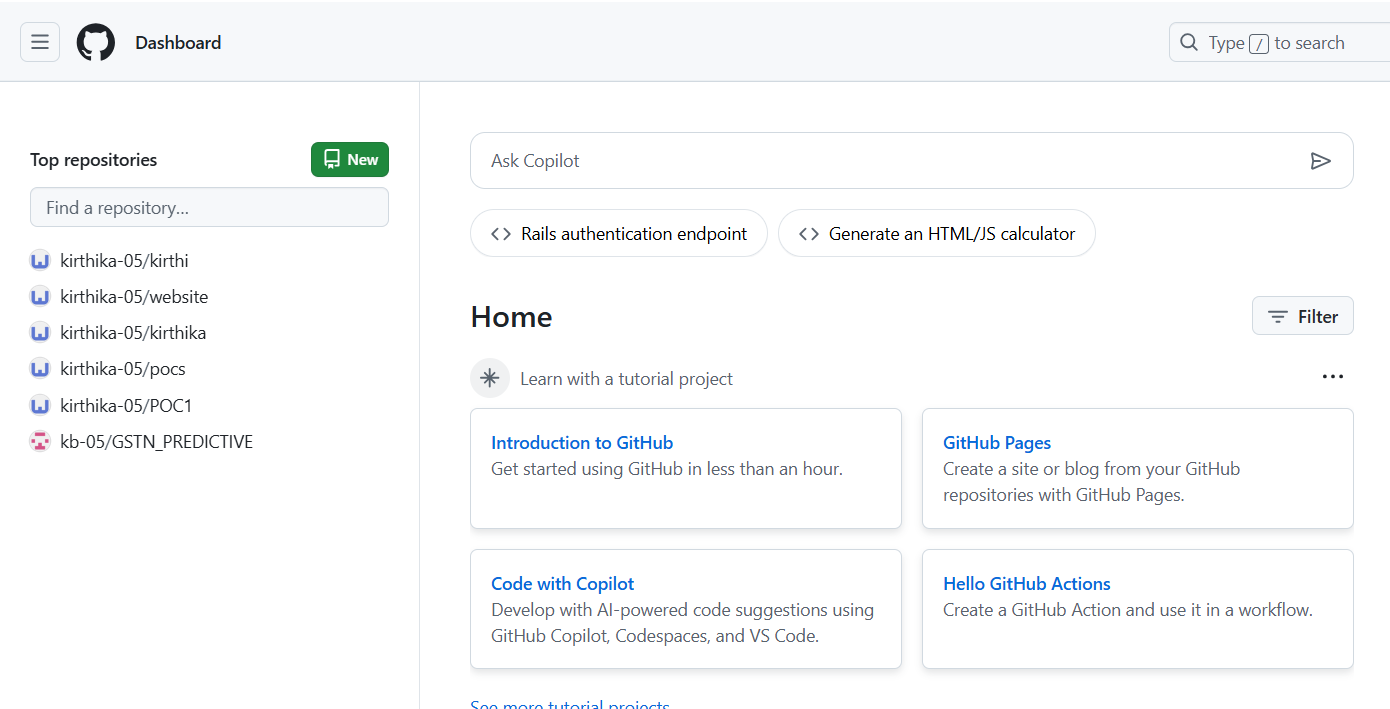
Set Up Your Name and Email Globally Git doesn’t know who is making the commit because you haven’t configured your name and email yet. Git uses this information to track who made the changes.



Step 10:

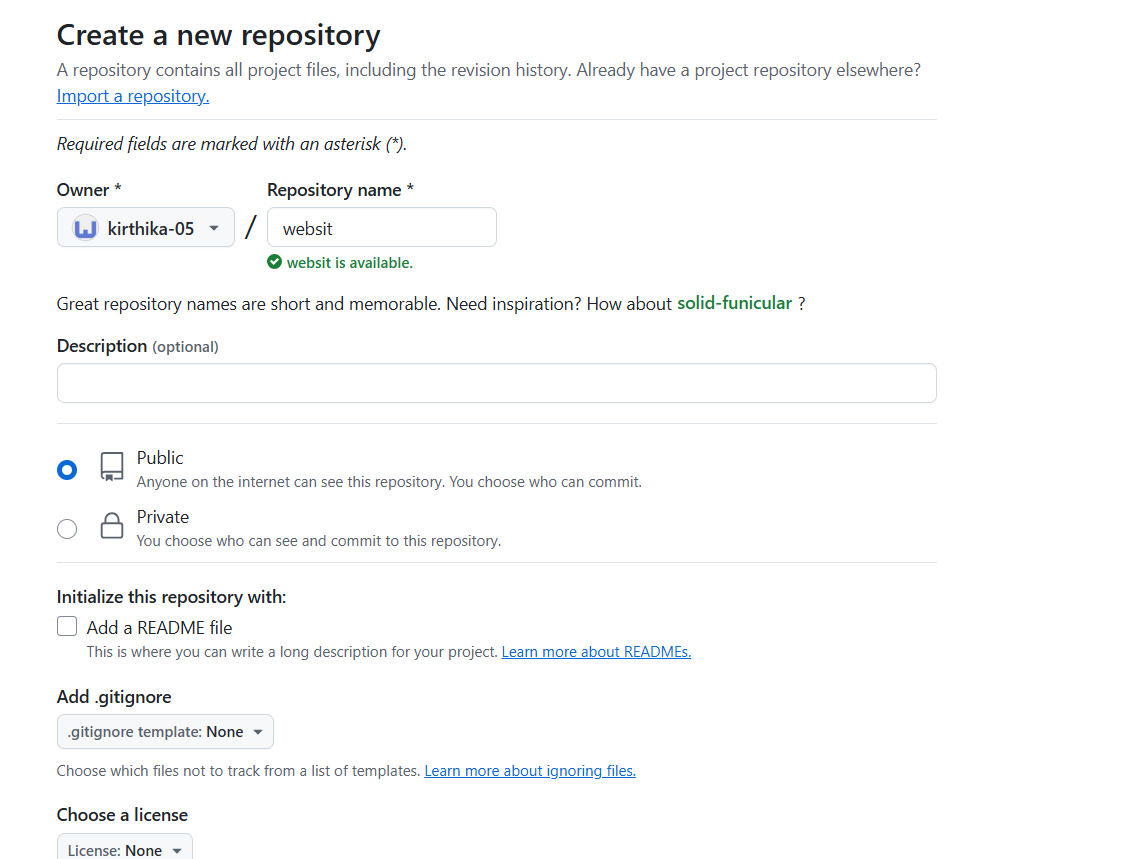
**Create a New Repository**:

Once you're logged in, click the green **"New"** button on the top- right of your GitHub homepage to create a new repository.



Step 11:

Give your repository a name, for example, test and click **"Create repository"**.



Step 12:

**Add the Remote Repository URL to Your Local Repository**:

Go back to your Command Line and type the following:

**git remote add origin https://github.com/yourusername/my-website.git**

Replace your username with your GitHub username and with the name of your GitHub repository

Step 13:

**git branch -V** & **git branch -M**

The main command is used to **rename the current branch** to main. Here's what it does:

**-M**: This flag forces the renaming, even if a branch named main already exists. It will overwrite the existing main branch.

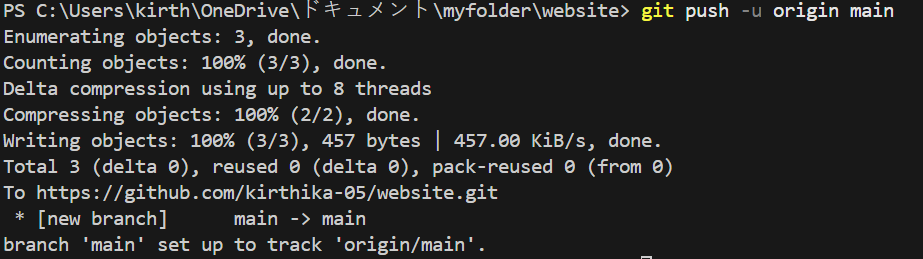
**main**: This is the new name for the current branch.



Step 14:

**git push -u origin main**

The command is used to **push your local main branch to the remote repository (origin)** and set it as the upstream branch

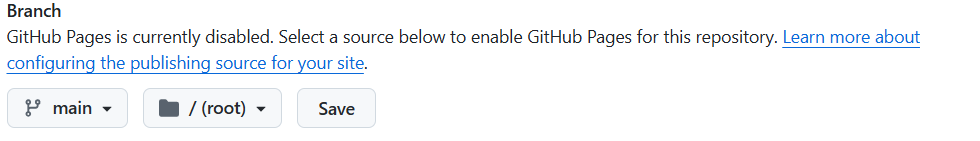


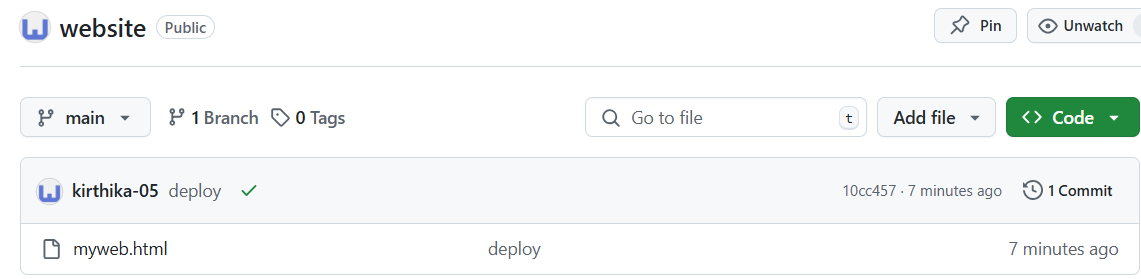
Step 15:

Verify Your Files on GitHub

Go to your GitHub Repository:

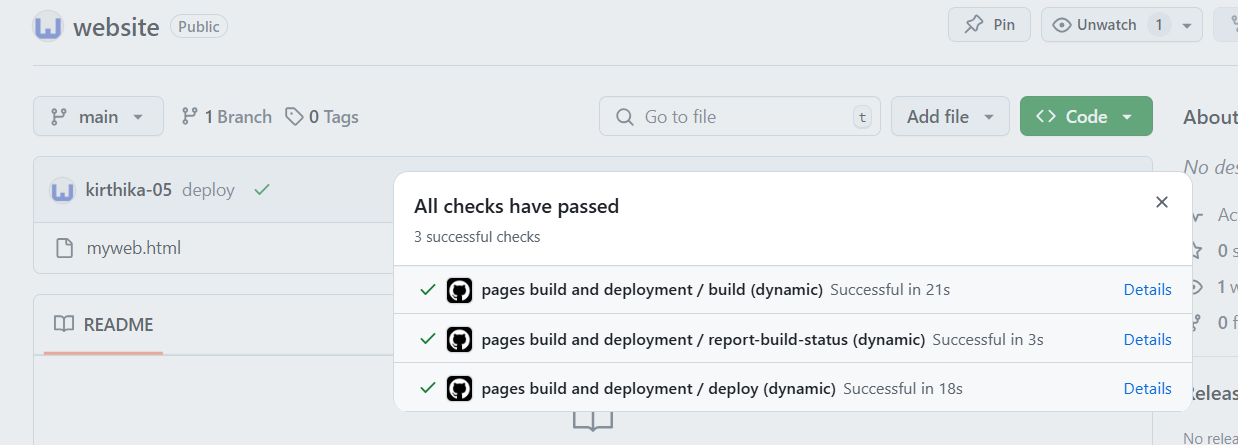
Open your web browser and navigate to your GitHub repository (e.g., <https://github.com/yourusername/my-website>).

****



Step 16:

With GitHub Pages, updates are automatic whenever you push changes to the repository, the site is instantly redeployed, ensuring seamless version control. Your Website is **deployed**



**Outcome**

By completing this PoC, you will have:

Deploying a static website using **GitHub Pages** is an efficient, cost-free, and beginner-friendly method for hosting web projects. It simplifies website management, enables **automatic updates**, and enhances collaboration while providing secure and reliable hosting.