 

**Placement Empowerment Program**

***Cloud Computing and DevOps Centre***

Set Up a Local Git Repository: Initialize a Git repository locally and version control your static website

Name: Nanthini R Department: ADS



**Introduction**

Version control is an essential part of modern software development. It allows developers to manage changes to their code over time, collaborate with others, and easily revert to previous versions when needed. **Git** is one of the most popular version control systems due to its speed, flexibility, and distributed nature.

In this Proof of Concept (PoC), we will set up a **local Git repository** to version control your static website. This setup will enable you to:

* Track changes to your website’s files
* Experiment with new features in a controlled way
* Easily revert back to previous versions if something goes wrong
* Share and collaborate on your project with others

Setting up Git for version control is a crucial skill for developers, whether you are working alone or as part of a team, and helps to ensure an organized development process.

**Overview**

Here’s what we will cover in this setup:

**1. Installing Git**: Ensure Git is installed on your system and properly configured.

**2. Creating a Local Repository**: Initialize a Git repository in the root folder of your static website

**3. Staging and Committing Files**: Add your project files to the staging area and commit them to the repository to save a snapshot of your work.

**4. Reviewing the** evolves. **Repository State**: Use Git commands to check the status of your repository and verify that everything is tracked properly.

**Objectives**

By the end of this POC, you will:

**1. Understand the Basics of Version Control**: Gain insight into the importance of Git for managing and tracking changes in your projects.

**2. Set Up a Git Repository**: Learn how to initialize a Git repository to version control your static website locally.

**3. Track Changes Effectively**: Understand how to stage and commit files to ensure every change is logged.

4. **Organize Your Project**: Maintain a clean and structured workflow for your static website, with the ability to roll back changes when needed.

5. **Prepare for Collaboration**: Lay the groundwork to share your repository and collaborate with others using Git when required

**Importance of Setting Up a Local Git Repository**

**Track Changes**: Git records all modifications, ensuring a clear history of your project.

**Rollback**: Easily revert to previous versions to recover from mistakes.

**Collaboration**: Prepares your project for team work, enabling smooth integration of changes.

**Step-by-Step Overview**

**Step 1: Install Git**

Before using Git, it’s essential to install it on your system. Follow these steps:

1. Open your browser and search for "Git."
2. On the official Git website, click on the **Downloads** link.
3. Choose the correct version for your operating system (Windows, macOS, or Linux) and download the installer.

After the download is complete, run the installer and follow the prompts. For most users, the default settings are perfectly fine.

To check if Git installed correctly, open your terminal or Command Prompt and run:

bash

Copy

git --version

If you see a version number displayed, then Git is installed successfully.

**Step 2: Install Git on Windows**

If you are on Windows, after downloading the installer:

1. Run the setup file and follow the installation wizard.
2. For simplicity, leave most of the settings as the default options. You can choose additional features if needed, but the defaults should work for most cases.
3. After installation is finished, open **Command Prompt** and type:

bash

Copy

git --version

This should confirm that Git was installed correctly by displaying the version number.

**Step 3: Create Your Project Folder**

Let’s set up your project folder to store your website files and initialize Git. Here’s what to do:

1. Create a new folder on your **Desktop** or in a location of your choice. Name it **website**.
2. Inside the **website** folder, create a simple **HTML file**. You can name the file index.html and write some basic HTML code inside it:

html

Copy

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>My Static Website</title>

</head>

<body>

<h1>Welcome to My Static Website</h1>

<p>This is a simple static website that will be version-controlled using Git!</p>

</body>

</html>

This will be your starting point—Git will track changes to this file as we progress.

**Step 4: Open Command Prompt/Terminal and Navigate to Your Folder**

Now, we need to tell Git where to find your project. Open your **Command Prompt** (on Windows) or **Terminal** (on macOS/Linux), and use the cd (change directory) command to navigate to the **website** folder you just created:

bash

Copy

cd path/to/your/website

Replace path/to/your/website with the actual path to the **website** folder.

**Step 5: Initialize a Git Repository**

To begin using Git, you need to initialize a Git repository inside the **website** folder. Run the following command:

bash

Copy

git init

This command creates a hidden .git folder inside your project. This folder will store all the version history and tracking information.

**Step 6: Set Up Your Identity in Git**

Before making your first commit, you need to configure your Git identity—Git needs to know who is making the changes. Set your **name** and **email address** by running the following commands:

bash

Copy

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

git config --global user.email "youremail@example.com"

These settings will be used for every commit you make on this machine. If you only want to change the identity for a specific repository, you can omit the --global flag.

**Step 7: Add Files to Git**

Now, let’s tell Git to start tracking the files in your project folder. To add all the files in the folder to Git’s staging area, run:

bash

Copy

git add .

The . refers to "all files in the current directory". If you only want to add specific files, replace the . with the filename, like so:

bash

Copy

git add index.html

The staging area is like a holding area where files are prepared before they are committed.

**Step 8: Commit Your Changes**

Once your files are staged, it's time to commit them. A commit in Git is like taking a snapshot of your project at a certain point in time. Run the following command to commit your changes:

bash

Copy

git commit -m "Initial commit of my static website"

The -m flag is followed by a short message describing the changes you made. In this case, "Initial commit of my static website" tells Git that this is your first commit for the website project.

**Step 9: Create a Remote Repository on GitHub**

You can now store your project online using GitHub, making it easier to share and collaborate. Here’s how:

1. Go to [GitHub](https://github.com) and log in or create an account.
2. Click the **New** button in the top-right corner to create a new repository.
3. Name your repository (e.g., my-website), leave the default options, and click **Create repository**.

**Step 10: Link Your Local Repository to GitHub**

Now, we need to connect your local Git repository to the GitHub repository. Copy the repository URL from GitHub, and run the following command in your terminal:

bash

Copy

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

Make sure to replace yourusername with your actual GitHub username, and my-website with the name of your repository.

**Step 11: Push Your Changes to GitHub**

To upload your local repository to GitHub, use the following command:

bash

Copy

git push -u origin main

This pushes your changes to GitHub and sets the main branch as the default for future pushes.

**Step 12: Verify Files on GitHub**

Visit your GitHub repository URL and you should see the index.html file that you just pushed

**Expected Outcome**

By following these steps, you will:

1. **Initialize a Git repository** in your local project folder.
2. **Track and commit changes** to your website files.
3. **Push your local repository** to GitHub to store your project online.
4. **Gain hands-on experience with Git** commands like git init, git add, git commit, and git push.