**Placement Empowerment Program**

***Cloud Computing and DevOps Centre***

**Create a simple Backup script** :

which backup your git repository to a local website

Name: **Priyadharshini S** Department: **ADS**



**Introduction:**

I developed a backup system for a Git repository using a script that automatically detects and pushes new changes from the local repository to a backup repository on GitHub. If no modifications are found, the script displays "up to date." However, when changes are made, it commits and pushes them to ensure the backup repository stays synchronized. This setup guarantees that the project is consistently backed up without requiring manual effort

**Overview**

Here’s what we will cover in this setup:

1. **Automated Change Detection** – The script checks for any uncommitted or un pushed changes in the local repository.
2. **Commit & Push Updates** – If changes are detected, the script stages, commits, and pushes them to the backup repository.
3. **Status Check** – When no modifications are found, it displays an "up to date" message, avoiding unnecessary operations.
4. **Scheduled Execution** – Can be run manually or automated using a cron job to ensure continuous backups.
5. **Minimal Manual Effort** – Once configured, the script runs independently, keeping the backup repository synchronized.

**Objectives**

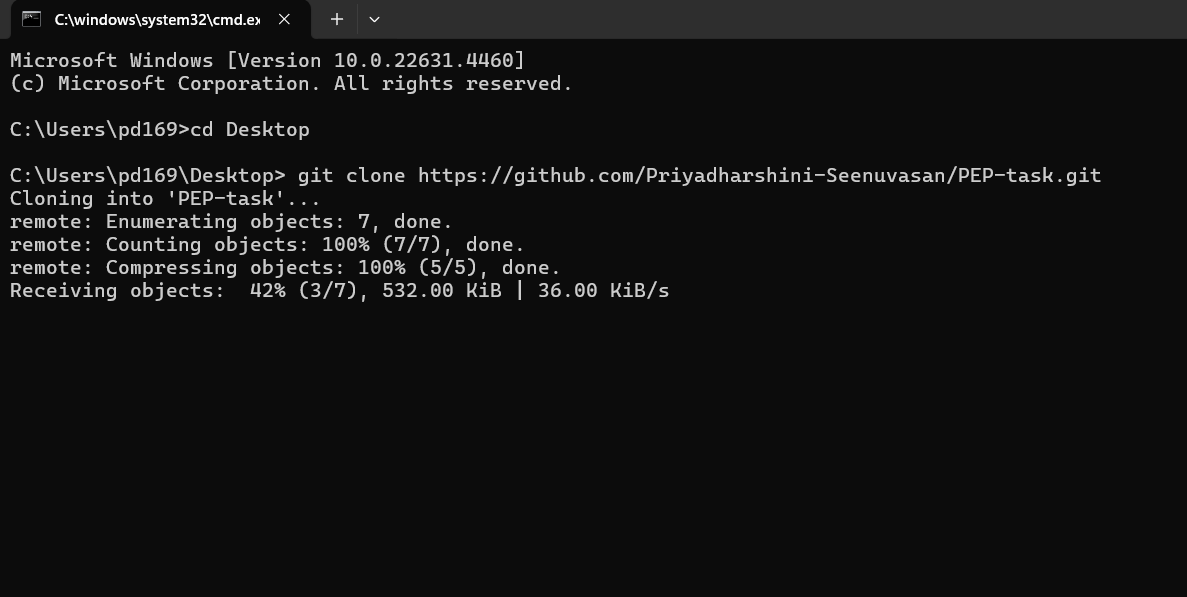
By the end of this POC, you will:

1. **Ensure Data Safety** – Prevent accidental data loss by maintaining an up-to-date backup repository.
2. **Automate Backup Process** – Eliminate the need for manual commits and pushes by automating the backup workflow.
3. **Keep Repository Synchronized** – Ensure that the backup repository always reflects the latest changes from the local repository.
4. **Enhance Efficiency** – Reduce the time and effort required for manual backups by implementing a hands-free solution.
5. **Provide Continuous Monitoring** – Regularly check for changes in the local repository and commit updates only when necessary.

**Step-by-Step Overview**

Step 1: **Clone Your Local Repository**

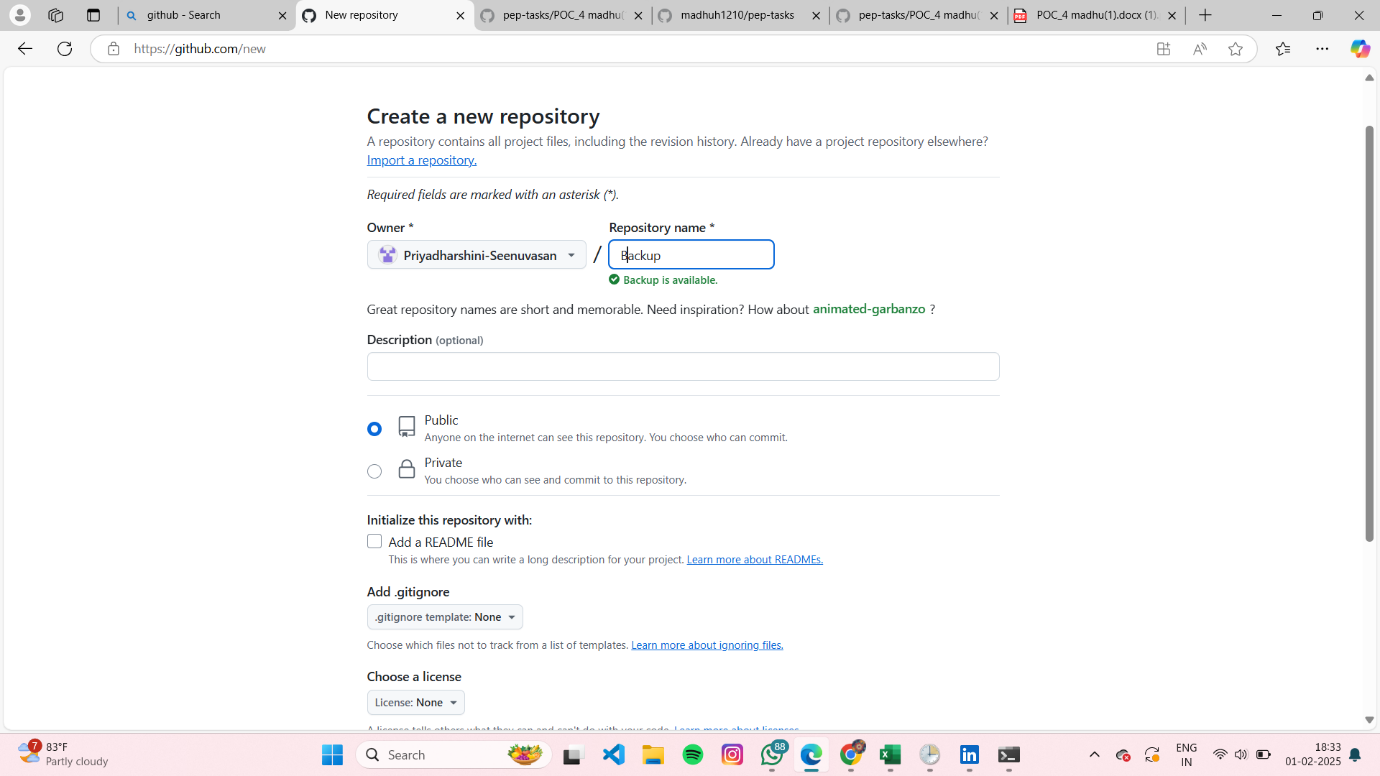
If your repository is not yet cloned, use the following command to download it to your local system



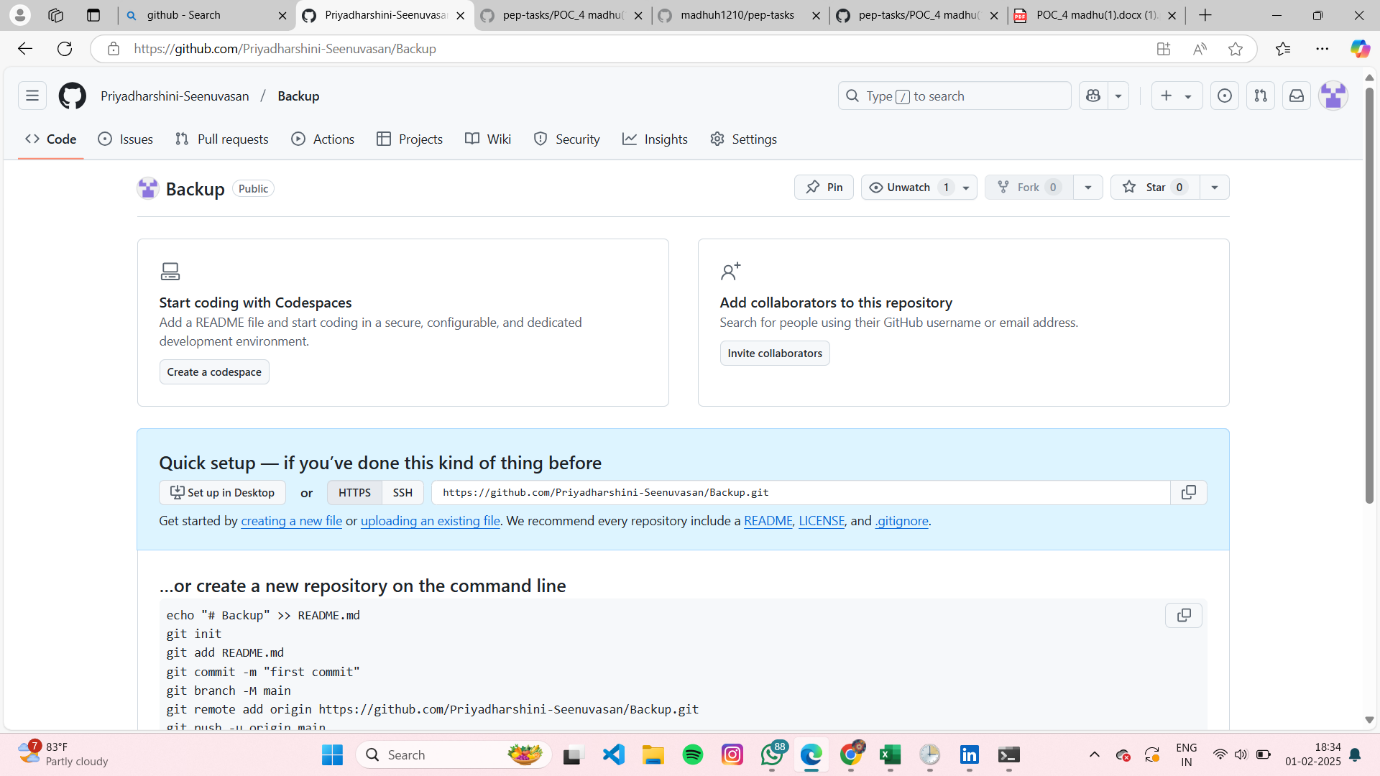
Step 2

**Create a Backup Repository on GitHub**

* Sign in to GitHub and create a new repository (e.g., backup-your-repository).
* Choose to make it **public or private**, based on your preference.
* Do **not** add any files like README.md during creation.
* Once the repository is created, note down its remote URL.



U will get this page once u finish the above one,

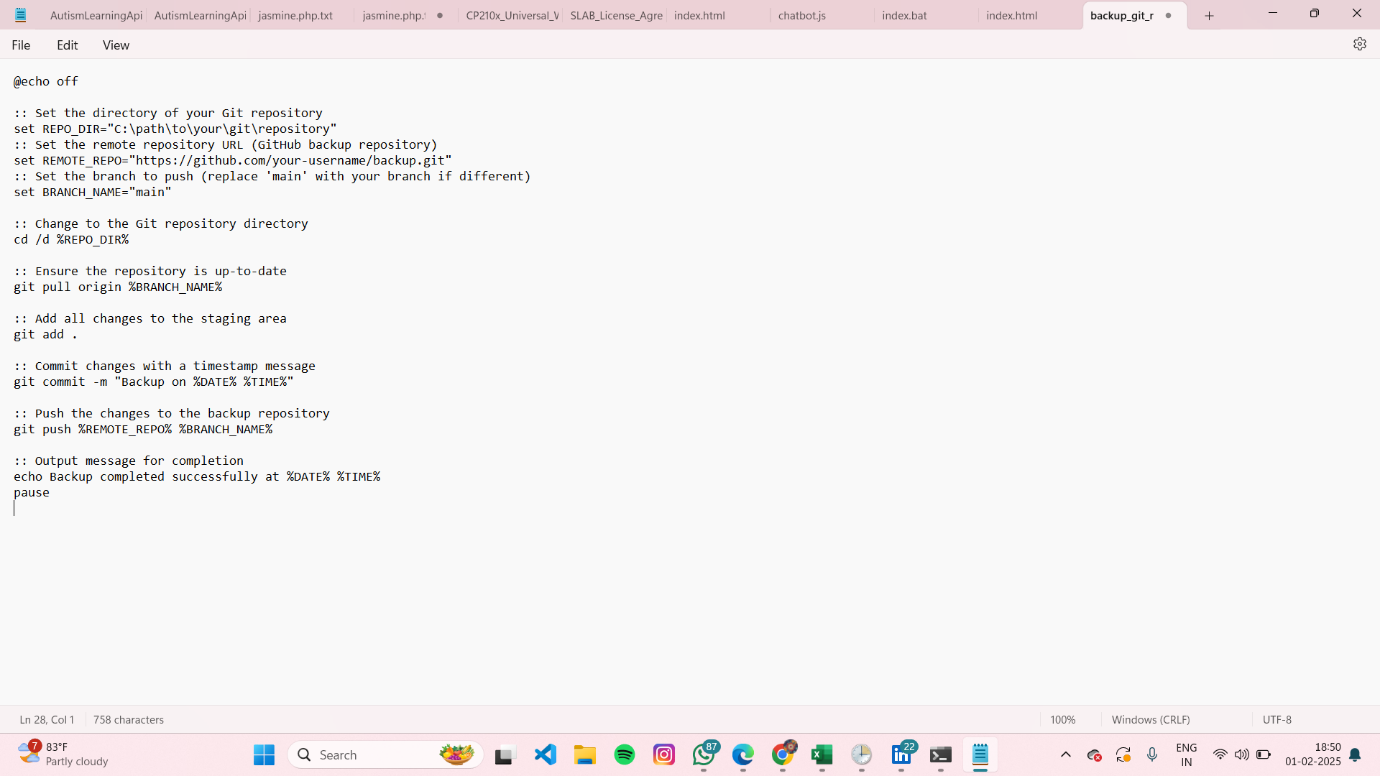


Step 3

**Create the Backup Script**

Open a text editor (Notepad for Windows, or a terminal-based editor for Linux/macOS) and create a script named backup\_git\_repo.bat (for Windows) or backup\_git\_repo.sh (for Linux/macOS).

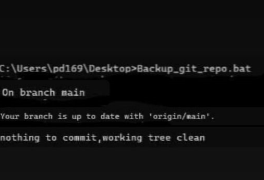
**For Windows (backup\_git\_repo.bat)**



Step 4

**Execute the Backup Script**

* Save the script in your local repository folder.
* **For Windows**, double-click backup\_git\_repo.bat to run it.
* **For Linux/macOS**, open the terminal, navigate to the script's directory, and execute



**Step 5: Verify the Backup**

* Once executed, the script will confirm that all changes have been backed up.
* Open GitHub and check the backup repository to ensure the latest commits have been pushed successfully.



**Expected Outcome**

1. **Changes Are Detected and Committed** – Modified files are staged, committed, and pushed to the backup repository.
2. **No Changes Detected** – If the repository is up to date, Git displays a message: *"nothing to commit, working tree clean."*
3. **Backup Repository Gets Updated** – New commits appear in the backup repository with a timestamped message.
4. **Script Executes Without Errors** – Runs smoothly if the repository path and Git credentials are correctly configured.
5. **Automated & Regular Backups** – When scheduled, the backup process occurs at set intervals without manual intervention.