**Placement Empowerment Program**

***Cloud Computing and DevOps Centre***

Create a Simple Backup Script: Create a script that backs up your entire Git repository to a local folder daily.

Name: Kirubalini S I Department : IT



**Introduction**

This project demonstrates how to create a simple yet effective backup script for your Git repository. Using a scripting language like Bash, we'll develop a script that automatically copies the entire contents of your Git repository to a local backup folder. This script will be designed to run daily, ensuring you have a recent local copy of your project, safeguarding your work against potential data loss or issues with your remote Git hosting service. This is a fundamental practice for version control and data protection.

**Overview**

This project focuses on building a practical backup solution for a Git repository by creating a script that automates the backup process. The core of the project is a shell script (e.g., Bash) designed to copy all files and folders within a given Git repository to a separate local directory. This script will be configured to run daily, providing a consistent and up-to-date backup. The overview includes setting up the script, defining the source Git repository and the destination backup directory, and scheduling the script's execution (e.g., using cron on Linux/macOS). This automated backup strategy protects your project's history and files, ensuring data integrity and recoverability.

**Objective**

The objective of this project is to create a robust and automated backup solution for a Git repository. This involves developing a script that can be scheduled to run daily, copying all files and directories within the repository to a designated local backup location. The goals of this "Create a Simple Backup Script" project are to:

* **Automate Git repository backups:** Develop a script that automatically backs up a Git repository, eliminating the need for manual intervention.
* **Ensure daily backups:** Schedule the script to run daily, providing consistent and up-to-date backups of the repository.
* **Create local backups:** Store the backups in a designated local folder, providing a readily accessible copy of the project data.
* **Protect against data loss:** Safeguard the project against accidental deletions, remote repository issues, or other unforeseen events.
* **Maintain data integrity:** Ensure the backup process accurately copies all files, directories, and Git history of the repository.
* **Implement a simple and robust solution:** Create a script that is easy to understand, maintain, and reliable in its backup operations.
* **Provide a practical learning experience:** Offer a hands-on project for learning about shell scripting, automation, and version control best practices

.

**Importance of Local Hosting**

Creating a simple backup script for your Git repository is important for several reasons:

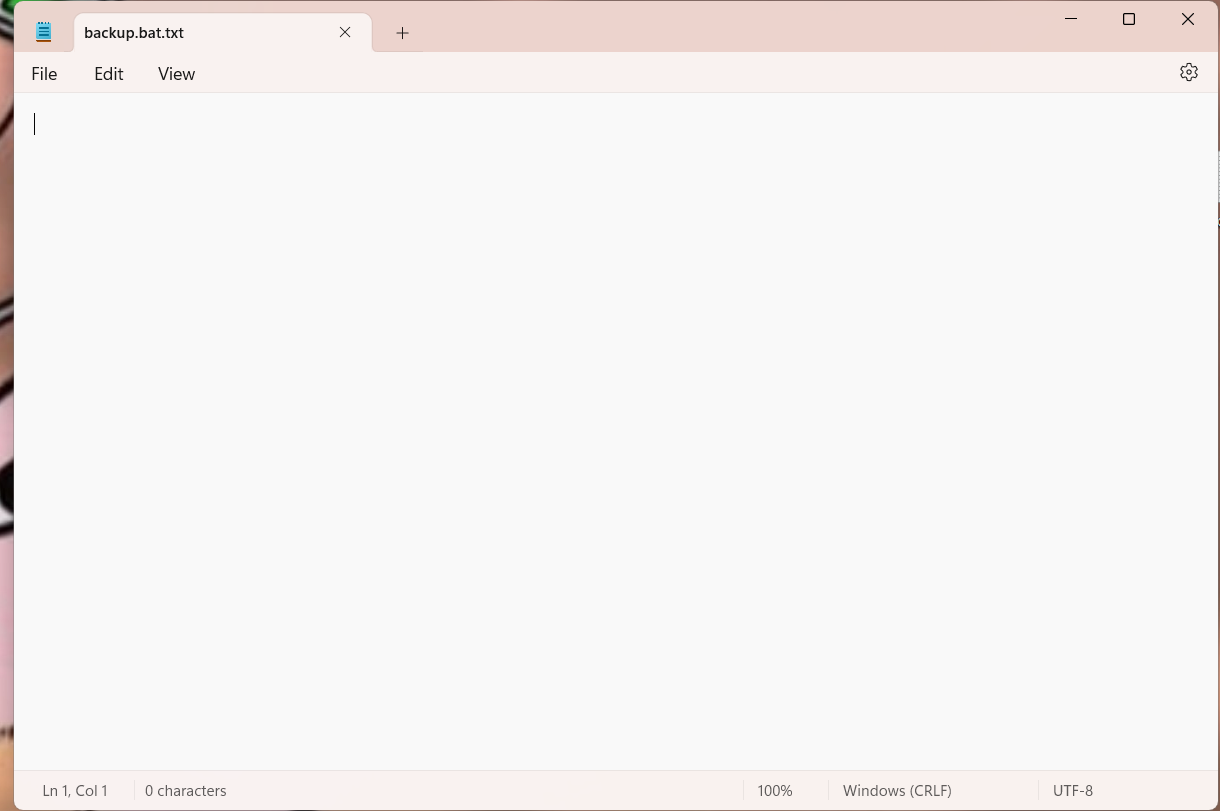
* **Data Loss Prevention:** Accidental deletion, hard drive failure, or issues with your remote Git provider (e.g., server outage, account compromise) can lead to the loss of your valuable code and project history. A local backup provides a safety net against these scenarios.
* **Version Control Reliability:** While Git itself is a version control system, relying solely on a single remote repository can be risky. Having a local backup ensures that you have a separate copy of your repository, protecting you from potential issues with the remote server.
* **Offline Access:** A local backup allows you to work on your project even when you don't have an internet connection. This can be crucial for productivity when traveling or experiencing network outages.
* **Faster Restores:** Restoring from a local backup is generally faster and easier than cloning a large repository from a remote server, especially if you have a slow internet connection.
* **Compliance and Archiving:** Some projects or organizations may have requirements for maintaining local copies of code repositories for compliance or archival purposes.
* **Peace of Mind:** Knowing that you have a recent backup of your work provides peace of mind and allows you to focus on development without worrying about data loss.
* **Learning Experience:** Creating a backup script is a valuable learning experience in shell scripting, automation, and version control best practices. It helps you understand how Git works and how to manage your repositories effectively.

**Step-by-Step Overview**

Step1:

Create a Batch Script

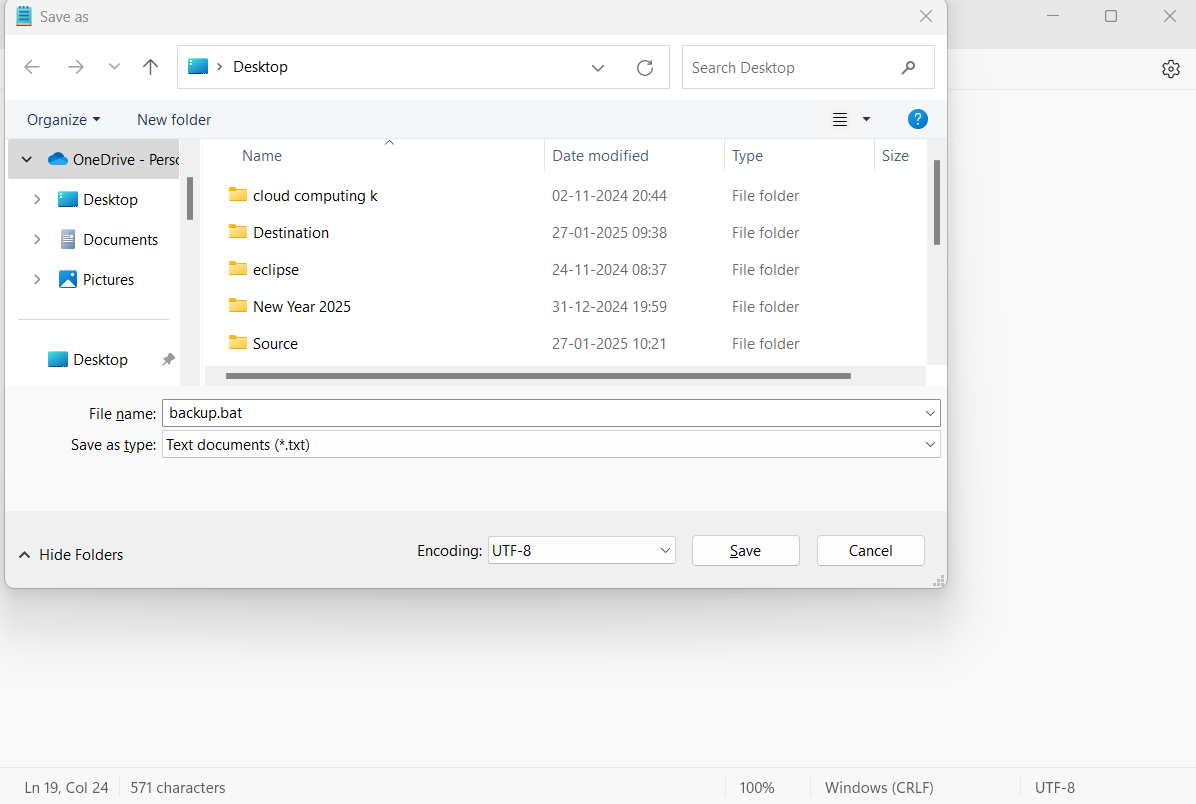
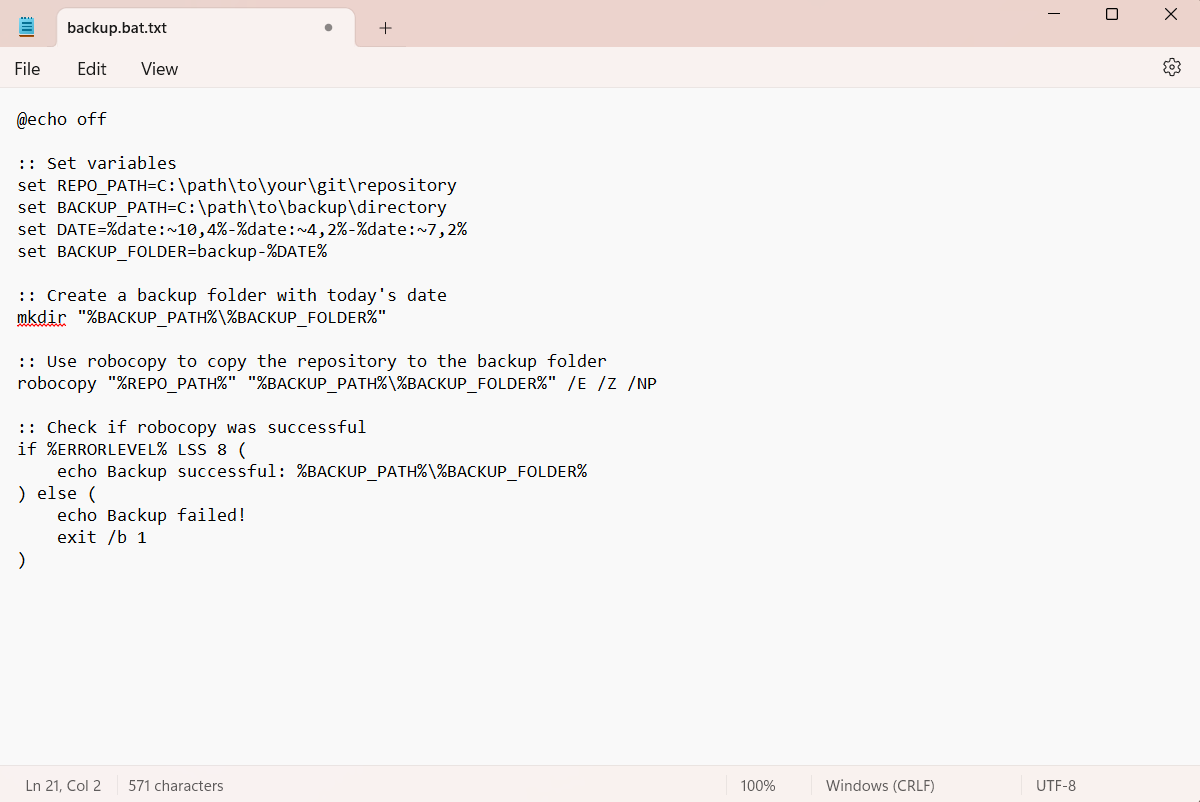
Open a text editor (like Notepad).



Save the following as backup.bat

Step 2 :

Upload the code in “ Backup.bat ” text document and save it in the Desktop



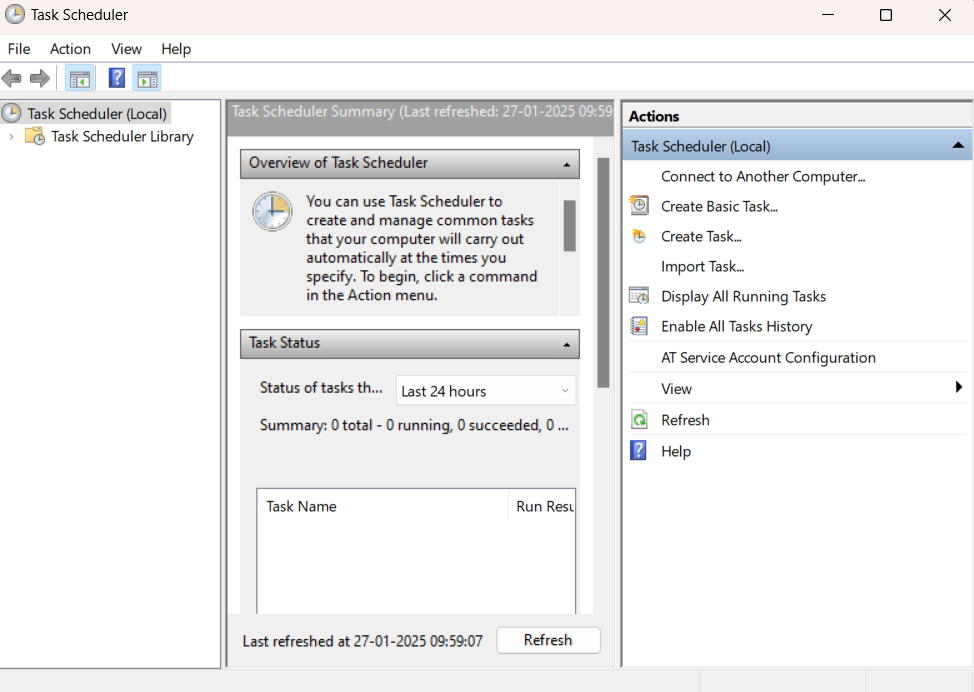
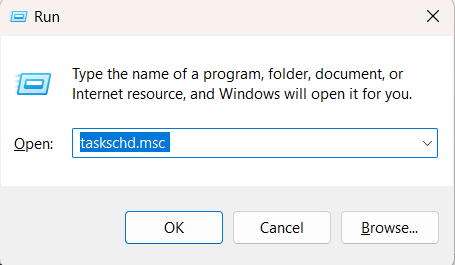
Step 3 :

Automating with Task Scheduler:

1. Create the Task:

a. Open Task Scheduler (search "Task Scheduler" in the Start menu).

b. Click Create Task (not "Basic Task").



Step 4 :

Open Command Prompt as Administrator (Windows + R, type cmd, right-click and select 'Run as Administrator') and use the command cd C:\path\to\apache\bin to set the path to the Apache bin folder.

Screenshot 2025-01-22 155734.png

Step 5 :

Then Run the installation command :

**httpd.exe -k install**



Step 6 :

Navigate to the Apache folder you downloaded, go to the **conf** folder, and right-click on the httpd.conf file; select 'Edit with Notepad'

(Apache/conf/httpd.conf)



Step 7 :

Inside the **httpd.conf** file, replace the content with the provided configuration. Ensure you update the SRVROOT directive with your Apache installation path. This configuration defines the server’s root directory, listening port, modules, document root for serving web files, logging paths, and basic permissions, ensuring Apache serves content correctly from the specified htdocs directory.





Step 8 :

Open Command Prompt and type the command **httpd.exe -t** to test the configuration file. If the configuration is correct, you should see **'Syntax OK**'.



Step 9 :

Run the command **httpd.exe -k** start to start the Apache server.



Step10:

Go to the Apache folder, navigate to the **htdocs** folder, and find the **index.html** file. Right-click on it and select 'Edit with Notepad'.





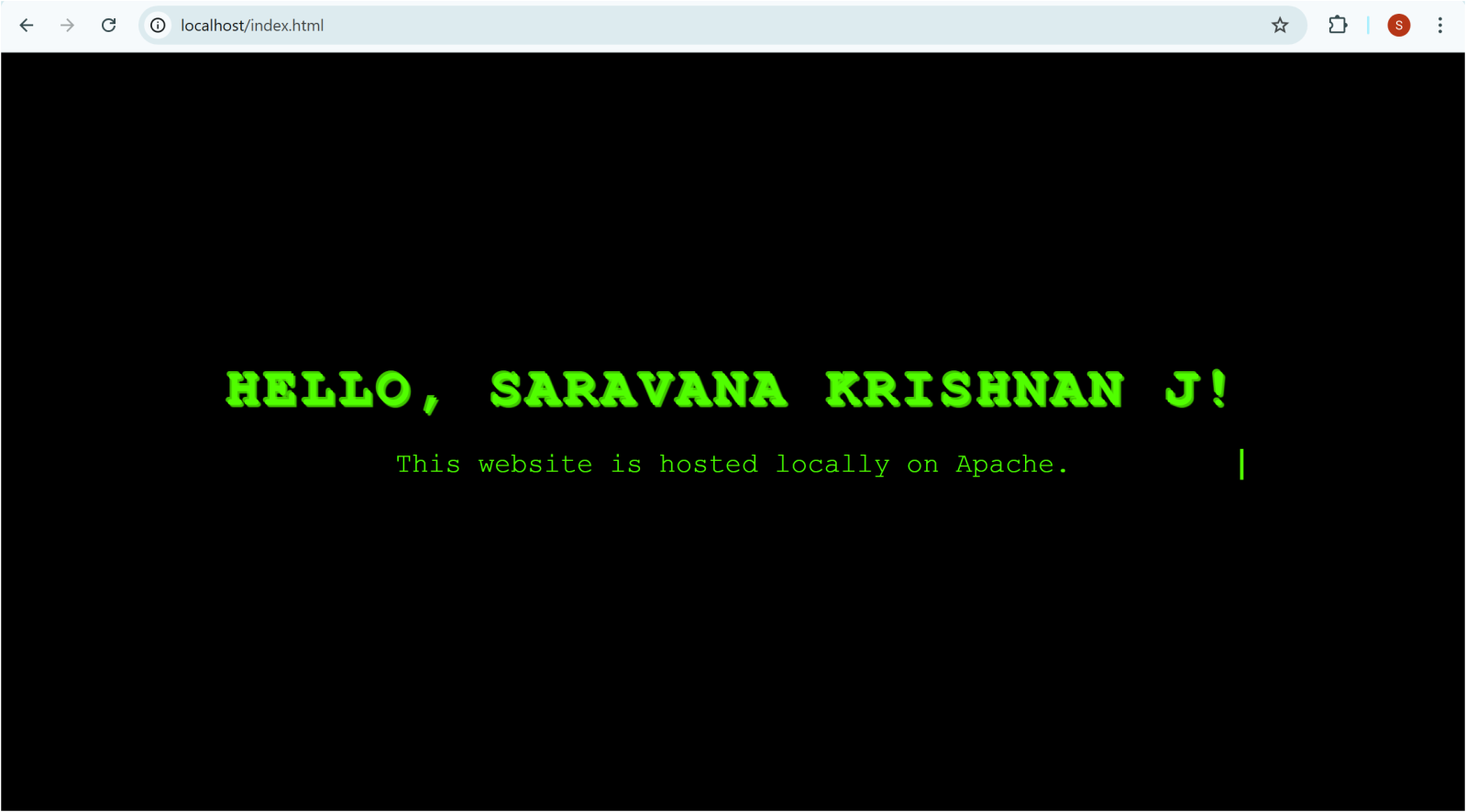
Step 11 :

Create a simple model to display your name in HTML (you may optionally add CSS for styling).



Step 12 :

Open the Chrome browser and type **localhost/index.html** in the address bar. You should be able to see the website hosted successfully.



**Expected Outcome**

The outcome of creating a simple backup script for your Git repository is a reliable, automated system that safeguards your project data. Specifically, you'll achieve the following:

* **Automated Daily Backups:** The script will run automatically (e.g., via cron) each day, creating a fresh backup of your Git repository without any manual intervention.
* **Local Copy of Your Repository:** A complete copy of your Git repository, including all files, directories, and Git history, will be stored in a designated local folder.
* **Protection Against Data Loss:** This local backup acts as a safeguard against accidental deletions, hard drive failures, remote repository issues (e.g., server outages, account problems), or other unforeseen events that could lead to data loss.
* **Improved Recovery Time:** In the event of data loss, restoring your repository from a local backup will be significantly faster and easier than cloning from a remote server, saving you valuable time and effort.
* **Increased Peace of Mind:** Knowing that your project is backed up daily will give you peace of mind and allow you to focus on your work without worrying about data loss.
* **Enhanced Data Security:** Having a local backup adds an extra layer of security, protecting your project from potential vulnerabilities or issues with your remote Git hosting provider.
* **Practical Scripting Skills:** You'll gain valuable experience in shell scripting, automation, and working with Git commands, enhancing your technical skills.